Reachability in Networks of Register Protocols
under Stochastic Schedulers*†

Patricia Bouyer¹, Nicolas Markey¹, Mickael Randour‡,
Arnaud Sangnier³, and Daniel Stan¹

1 LSV – CNRS, ENS Cachan & University Paris-Saclay – France
2 Computer Science Department – Université Libre de Bruxelles (ULB) – Belgium
3 IRIF – University Paris Diderot & CNRS – France

Abstract
We study the almost-sure reachability problem in a distributed system obtained as the asynchronous composition of \( N \) copies (called processes) of the same automaton (called protocol), that can communicate via a shared register with finite domain. The automaton has two types of transitions: write-transitions update the value of the register, while read-transitions move to a new state depending on the content of the register. Non-determinism is resolved by a stochastic scheduler. Given a protocol, we focus on almost-sure reachability of a target state by one of the processes. The answer to this problem naturally depends on the number \( N \) of processes. However, we prove that our setting has a cut-off property: the answer to the almost-sure reachability problem is constant when \( N \) is large enough; we then develop an EXPSPACE algorithm deciding whether this constant answer is positive or negative.

1 Introduction

Verification of systems with many identical processes. It is a classical pattern in distributed systems to have a large number of identical components running concurrently (a.k.a. networks of processes). In order to verify the correctness of such systems, a naive option consists in fixing an upper bound on the number of processes, and applying classical verification techniques on the resulting system. This has several drawbacks, and in particular it gives no information whatsoever about larger systems. Another option is to use parameterized-verification techniques, taking as a parameter the number of copies of the protocol in the system being considered. In such a setting, the natural question is to find and characterize the set of parameter values for which the system is correct. Not only the latter approach is more general, but it might also turn out to be easier and more efficient, since it involves orthogonal techniques.

Different means of communication lead to different models. A seminal paper on parameterized verification of such distributed systems is the work of German and Sistla [18]. In this work, the authors consider networks of processes all following the same finite-state automaton; the communication between processes is performed thanks to rendez-vous communication. Various related settings have been proposed and studied since then, which mainly differ by the way the processes communicate. Among those, let us mention broadcast
Reachability in Networks of Register Protocols under Stochastic Schedulers

communication \cite{16,10}, token-passing \cite{8,2}, message passing \cite{6}, shared register with ring topologies \cite{1}, or shared memory \cite{17}. In his nice survey on such parameterized models \cite{15}, Esparza shows that minor changes in the setting, such as the presence of a controller in the system, might drastically change the complexity of the verification problems. The relative expressiveness of some of those models has been studied recently in \cite{3}, yielding several reductions of the verification problems for some of those classes of models.

Asynchronous shared-memory systems. We consider a communication model where the processes asynchronously access a shared register, and where read and write operations on this register are performed non-atomically. A similar model has been proposed by Hague in \cite{19}, where the behavior of processes is defined by a pushdown automaton. The complexity of some reachability and liveness problems for shared-memory models have then been established in \cite{17} and \cite{12}, respectively. These works consider networks in which a specific process, called the leader, runs a different program, and address the problem whether, for some number of processes, the leader can satisfy a given reachability or liveness property. In the case where there is no leader, and where processes are finite-state, the parameterized control-state reachability problem (asking whether one of the processes can reach a given control state) can be solved in polynomial time, by adapting the approach of \cite{9} for lossy broadcast protocols.

Fairness and cut-off properties. In this work, we further insert fairness assumptions in the model of parameterized networks with asynchronous shared memory, and consider reachability problems in this setting. There are different ways to include fairness in parameterized models. One approach is to enforce fairness expressed as a temporal-logic properties on the executions (e.g., any action that is available infinitely often must be performed infinitely often); this is the option chosen for parameterized networks with rendez-vous \cite{18} and for systems with disjunctive guards (where processes can query the states of other processes) in \cite{4}. We follow another choice, by equipping our networks with a stochastic scheduler that, at each step of the execution, assigns the same probability to the available actions of all the processes. From a high-level perspective, both forms of fairness are similar. However, expressing fairness via temporal logic allows for very regular patterns (e.g., round-robin execution of the processes), whereas the stochastic approach leads to consider all possible interleavings with probability 1. Under this stochastic scheduler assumption, we focus on almost-sure reachability of a given control state by any of the processes of the system. More specifically, as in \cite{4}, we are interested in determining the existence of a cut-off, i.e., an integer $k$ such that networks with more than $k$ processes almost-surely reach the target state. Deciding the existence and computing such cut-offs is important for at least two aspects: first, it ensures that the system is correct for arbitrarily large networks; second, if we are able to derive a bound on the cut-off, then using classical verification techniques we can find the exact value of the cut-off and exactly characterize the sizes of the networks for which the behavior is correct.

Our contributions. We prove that for finite-state asynchronous shared-memory protocols with a stochastic scheduler, and for almost-sure reachability of some control state by some process of the network, there always exists a positive or negative cut-off: positive cut-offs are those above which the target state is reached with probability 1, while negative cut-offs are those above which the target state is reached with probability strictly less than 1. Notice that both cut-offs are not complement of one another, so that our result is not trivial.

We then prove that the “sign” (positive or negative) of a cut-off can be decided in \textit{EXPSPACE}, and that this problem is \textit{PSPACE}-hard. Finally, we provide lower and upper
bounds on the values of the cut-offs, exhibiting in particular protocols with exponential (negative) cut-off. Notice how these results contrast with classical results in related areas: in the absence of fairness, reachability can be decided in polynomial time, and in most settings, when cut-offs exist, they generally have polynomial size [4, 14, 13].

2 Presentation of the model and of the considered problem

2.1 Preliminaries.

Let \( S \) be a finite set. A multiset over \( S \) is a mapping \( \mu: S \to \mathbb{N} \). The cardinality of a multiset \( \mu \) is \( |\mu| = \sum_{s \in S} \mu(s) \). The support \( \mathcal{P} \) of \( \mu \) is the subset \( \nu \subseteq S \) s.t. for all \( s \in S \), it holds \( s \in \nu \) if, and only if, \( \mu(s) > 0 \). For \( k \in \mathbb{N} \), we write \( \mathbb{N}^S_k \) for the set of multisets of cardinality \( k \) over \( S \), and \( \mathbb{N}^S \) for the set of all multisets over \( S \). For any \( s \in S \) and \( k \in \mathbb{N} \), we write \( s^k \) for the multiset where \( s^k(s) = k \) and \( s^k(s') = 0 \) for all \( s' \neq s \). We may write \( s \) instead of \( s^1 \) when no ambiguity may arise. A multiset \( \mu \) is included in a multiset \( \mu' \), written \( \mu \subseteq \mu' \), if \( \mu(s) \leq \mu'(s) \) for all \( s \in S \). Given two multisets \( \mu \) and \( \mu' \), their union \( \mu \lor \mu' \) is still a multiset s.t. \( (\mu \lor \mu')(s) = \mu(s) + \mu'(s) \) for all \( s \in S \). Assuming \( \mu \subseteq \mu' \), the difference \( \mu' \ominus \mu \) is still a multiset s.t. \( (\mu' \ominus \mu)(s) = \mu'(s) - \mu(s) \).

A quasi-order \( (A, \preceq) \) is a well quasi-order (wqo for short) if for every infinite sequence of elements \( a_1, a_2, \ldots \) in \( A \), there exist two indices \( i < j \) such that \( a_i \preceq a_j \). For instance, for \( n > 0 \), \( (\mathbb{N}^n, \preceq) \) (with lexicographic order) is a wqo. Given a set \( A \) with an ordering \( \preceq \) and a subset \( B \subseteq A \), the set \( B \) is said to be upward closed in \( A \) if for all \( a_1 \subseteq B \) and \( a_2 \subseteq A \), in case \( a_1 \preceq a_2 \), then \( a_2 \subseteq B \). The upward-closure of a set \( B \) for the ordering \( \preceq \), denoted by \( \uparrow \preceq(B) \) (or sometimes \( \uparrow(B) \) when the ordering is clear from the context), is the set \( \{ a \in A \mid \exists b \in B \text{ s.t. } b \preceq a \} \). If \( (A, \preceq) \) is a wqo and \( B \) is an upward closed set in \( A \), there exists a finite set of minimal elements \( \{ b_1, \ldots, b_k \} \) such that \( B = \uparrow \{ b_1, \ldots, b_k \} \).

2.2 Register protocols and associated distributed system.

We focus on systems that are defined as the (asynchronous) product of several copies of the same protocol. Each copy communicates with the others through a single register that can store values from a finite alphabet.

Definition 1. A register protocol is given by \( \mathcal{P} = (Q, D, q_0, T) \), where \( Q \) is a finite set of control locations, \( D \) is a finite alphabet of data for the shared register, \( q_0 \in Q \) is an initial location, \( T \subseteq Q \times \{ R, W \} \times D \times Q \) is the set of transitions of the protocol. Here \( R \) means read the content of the shared register, while \( W \) means write in the register.

In order to avoid deadlocks, it is required that each location has at least one outgoing transition. We also require that whenever some \( R \)-transition \( (q, R, d', q') \) appears in \( T \), then for all \( d \in D \), there exists at least one \( q_d \in Q \) such that \( (q, R, d, q_d) \in T \). The size of the protocol \( \mathcal{P} \) is given by \( |Q| + |T| \).

Example 1.a. Figure 4 displays a small register protocol with four locations, over an alphabet of data \( D = \{ 0, 1, 2 \} \). In this figure (and in the sequel), omitted \( R \)-transitions (e.g., transitions \( R(1) \) and \( R(2) \) from \( q_0 \)) are assumed to be self-loops. When the register contains 0, this protocol may move from initial location \( q_0 \) to location \( q_1 \). From there it can write 1 in the register, and then move to \( q_2 \). From \( q_2 \), as long as the register contains 1, the process can either stay in \( q_2 \) (with the omitted self-loop \( R(1) \)), or write 2 in the register and jump back to \( q_1 \). It is easily seen that if this process executes alone, it cannot reach state \( q_f \).
Reachability in Networks of Register Protocols under Stochastic Schedulers

We now present the semantics of distributed systems associated with our register protocols. We consider the asynchronous composition of several copies of the protocol (the number of copies is not fixed a priori and can be seen as a parameter). We are interested in the behavior of such a composition under a fair scheduler. Such distributed systems involve two sources of non-determinism: first, register protocols may be non-deterministic; second, in any configuration, all protocols have at least one available transition, and non-determinism arises from the asynchronous semantics. In the semantics associated with a register protocol, non-determinism will be solved by a randomized scheduler, whose role is to select at each step which process will perform a transition, and which transition it will perform among the available ones. Because we will consider qualitative objectives (almost-sure reachability), the exact probability distributions will not really matter, and we will pick the uniform one (arbitrary choice). Note that we assume non-atomic read/write operations on the register, as write are always possible, and if a process performs a read of a specific value, since this read also get T relation Pr as follows: for all \( q \in Q \), we have \( \Pr_{\text{after}}(\gamma) = \frac{1}{|\text{Post}(\gamma)|} \) if \( \gamma \rightarrow \gamma' \), and \( \Pr_{\text{after}}(\gamma, \gamma') = 0 \) otherwise. Note that \( \Pr \) is well defined: by the restriction imposed on the transition relation \( T \) of the protocol, we have \( 0 < |\text{Post}(\gamma)| < \infty \) for all configuration \( \gamma \), and hence we also get \( \sum_{\gamma' \in S} \Pr_{\text{after}}(\gamma, \gamma') = 1 \). For a fixed integer \( k \), we define the distributed system of size \( k \) associated with \( \mathcal{P} \) as the finite-state discrete-time Markov chain \( \mathcal{S}_k = (\Gamma_k, \Pr_k) \), where \( \Pr_k \) is the restriction of \( \Pr \) to \( \Gamma_k \times \Gamma_k \).

\[ \mathcal{S}_k = \langle \Gamma_k, \Pr_k \rangle \]

\[ \text{A configuration } \gamma' = \langle \mu', d' \rangle \text{ is a successor of a configuration } \gamma = \langle \mu, d \rangle \text{ if, and only if, there is a transition } (q, \text{op}, d', q') \in T \text{ such that } \mu(q) > 0, \mu' = \mu \ominus q \oplus q' \text{ and either } \text{op} = R \text{ and } d = d' = d'' \text{, or } \text{op} = W \text{ and } d' = d'' \text{. In that case, we write } \gamma \rightarrow \gamma'. \text{ Note that since } \mu(q) > 0 \text{ and } \mu' = \mu \ominus q \oplus q', \text{ we have necessarily } |\mu| = |\mu'|. \text{ In our system, we assume that there is no creation or deletion of processes during an execution, hence the size of configurations (i.e., } |\text{st}(\gamma)|) \text{ remains constant along transitions. We write } \Gamma_k \text{ for the set of configurations of size } k. \text{ For any configuration } \gamma \in \Gamma_k, \text{ we denote by } \text{Post}(\gamma) \subseteq \Gamma_k \text{ the set of successors of } \gamma, \text{ and point out that such a set is finite and non-empty.} \]

\[ \text{Now, the distributed system } \mathcal{S}_k \text{ associated with a register protocol } \mathcal{P} \text{ is a discrete-time Markov chain } (\Gamma, \Pr) \text{ where } \Pr: \Gamma \times \Gamma \rightarrow [0, 1] \text{ is the transition probability matrix defined as follows: for all } \gamma \text{ and } \gamma' \in \Gamma, \text{ we have } \Pr(\gamma, \gamma') = \frac{1}{|\text{Post}(\gamma)|} \text{ if } \gamma \rightarrow \gamma', \text{ and } \Pr(\gamma, \gamma') = 0 \text{ otherwise. Note that } \Pr \text{ is well defined: by the restriction imposed on the transition relation } T \text{ of the protocol, we have } 0 < |\text{Post}(\gamma)| < \infty \text{ for all configuration } \gamma, \text{ and hence we also get } \sum_{\gamma' \in S} \Pr_{\text{after}}(\gamma, \gamma') = 1. \text{ For a fixed integer } k \text{, we define the distributed system of size } k \text{ associated with } \mathcal{P} \text{ as the finite-state discrete-time Markov chain } \mathcal{S}_k = (\Gamma_k, \Pr_k), \text{ where } \Pr_k \text{ is the restriction of } \Pr \text{ to } \Gamma_k \times \Gamma_k. \]

\[ \text{We now present the semantics of distributed systems associated with our register protocols. We consider the asynchronous composition of several copies of the protocol (the number of copies is not fixed a priori and can be seen as a parameter). We are interested in the behavior of such a composition under a fair scheduler. Such distributed systems involve two sources of non-determinism: first, register protocols may be non-deterministic; second, in any configuration, all protocols have at least one available transition, and non-determinism arises from the asynchronous semantics. In the semantics associated with a register protocol, non-determinism will be solved by a randomized scheduler, whose role is to select at each step which process will perform a transition, and which transition it will perform among the available ones. Because we will consider qualitative objectives (almost-sure reachability), the exact probability distributions will not really matter, and we will pick the uniform one (arbitrary choice). Note that we assume non-atomic read/write operations on the register, as write are always possible, and if a process performs a read of a specific value, since this read also get T relation Pr as follows: for all } q \in Q \text{, we have } \Pr_{\text{after}}(\gamma) = \frac{1}{|\text{Post}(\gamma)|} \text{ if } \gamma \rightarrow \gamma', \text{ and } \Pr_{\text{after}}(\gamma, \gamma') = 0 \text{ otherwise. Note that } \Pr \text{ is well defined: by the restriction imposed on the transition relation } T \text{ of the protocol, we have } 0 < |\text{Post}(\gamma)| < \infty \text{ for all configuration } \gamma, \text{ and hence we also get } \sum_{\gamma' \in S} \Pr_{\text{after}}(\gamma, \gamma') = 1. \text{ For a fixed integer } k \text{, we define the distributed system of size } k \text{ associated with } \mathcal{P} \text{ as the finite-state discrete-time Markov chain } \mathcal{S}_k = (\Gamma_k, \Pr_k), \text{ where } \Pr_k \text{ is the restriction of } \Pr \text{ to } \Gamma_k \times \Gamma_k. \]
We are interested in analyzing the behavior of the distributed system for a large number of participants. More precisely, we are interested in determining whether almost-sure reachability of a specific control state holds when the number of processes involved is large. We are therefore seeking a cut-off property, which we formalize in the following.

A finite path in the system $S_P$ is a finite sequence of configurations $\gamma_0 \rightarrow \gamma_1 \ldots \rightarrow \gamma_k$. In such a case, we say that the path starts in $\gamma_0$ and ends in $\gamma_k$. We furthermore write $\gamma \rightarrow^* \gamma'$ if, and only if, there exists a path that starts in $\gamma$ and ends in $\gamma'$. Given a location $q_f$, we denote by $\mathbb{P}(q_f)$ the set of paths of the form $\gamma_0 \rightarrow \gamma_1 \ldots \rightarrow \gamma_k$ for which there is $i \in [0; k]$ such that $st(\gamma_i)(q_f) > 0$. Given a configuration $\gamma$, we denote by $\mathbb{P}(\gamma, \mathbb{Q}(q_f))$ the probability that some paths starting in $\gamma$ belong to $\mathbb{Q}(q_f)$ in $S_P$. This probability is well-defined since the set of such paths is measurable (see e.g., [5]). Given a register protocol $P = (Q, D, q_0, T)$, an initial register value $d_0$, and a target location $q_f \in Q$, we say that $q_f$ is almost-surely reachable for $k$ processes if $\mathbb{P}((q^h_0, d_0), \mathbb{Q}(q_f)) = 1$.

**Example 1.b.** Consider again the protocol depicted in Fig. 1 with initial register content $0$. As we explained already, for $k = 1$, the final state is not reachable at all, for any scheduler (here as $k = 1$, the scheduler only has to solve non-determinism in the protocol).

When $k = 2$, one easily sees that the final state is reachable: it suffices that both processes go to $q_2$ together, from where one process may write value $2$ in the register, which the other process can read and go to $q_f$. Notice that this does not ensure that $q_f$ is reachable almost-surely for this $k$ (and actually, it is not; see Example 1.c).

We aim here at finding cut-offs for almost-sure reachability, i.e., we seek the existence of a threshold such that almost-sure reachability (or its negation) holds for all larger values.

**Definition 2.** Fix a protocol $P = (Q, D, q_0, T)$, $d_0 \in D$, and $q_f \in Q$. An integer $k \in \mathbb{N}$ is a cut-off for almost-sure reachability (shortly a cut-off) for $P$, $d_0$ and $q_f$ if one of the following two properties holds:

- for all $h \geq k$, we have $\mathbb{P}((q^h_0, d_0), \mathbb{Q}(q_f)) = 1$. In this case $k$ is a positive cut-off;
- for all $h \geq k$, we have $\mathbb{P}((q^h_0, d_0), \mathbb{Q}(q_f)) < 1$. Then $k$ is a negative cut-off.

An integer $k$ is a tight cut-off if it is a cut-off and $k - 1$ is not.

Notice that from the definition, cut-offs need not exist for a given distributed system. Our main result precisely states that cut-offs always exist, and that we can decide their nature.

**Theorem 3.** For any protocol $P$, any initial register value $d_0$ and any target location $q_f$, there always exists a cut-off for almost-sure reachability, whose value is at most doubly-exponential in the size of $P$. Whether it is a positive or a negative cut-off can be decided in EXPSPACE, and is PSPACE-hard.

**Remark.** When dropping the condition on non-atomic read/write operations, and allowing transitions with atomic read/write operations (i.e., one process is ensured to perform a read and a write operation without to be interrupted by another process), the existence of a cut-off (Theorem 3) is not ensured. This is demonstrated with the protocol of Fig. 2: one easily checks (e.g., inductively on the number of processes, since processes that end up in $q_2$ play no role anymore) that state $q_f$ is reached with probability 1 if, and only if, the number of processes is odd.
3 Properties of register protocols

3.1 Example of a register protocol

We illustrate our model with a family of register protocols \( (\mathcal{F}_n)_{n>0} \), depicted in Fig. 2. For a fixed \( n \), protocol \( \mathcal{F}_n \) has \( n+1 \) states and \( n \) different data; intuitively, in order to move from \( s_i \) to \( s_{i+1} \), two processes are needed: one writes \( i \) in the register and goes back to \( s_0 \), and the second process can proceed to \( s_{i+1} \) by reading \( i \). Since backward transitions to \( s_0 \) are always possible and since states can always exit \( s_0 \) by writing a 0 and reading it afterwards, no deadlock can ever occur so the main question remains to determine if \( s_n \) is reachable by one of the processes as we increase the number of initial processes. As shown in Lemma 4, the answer is positive: \( \mathcal{F}_n \) has a tight linear positive cut-off; it actually behaves like a “filter”, that can test if at least \( n \) processes are running together. We exploit this property later in Section 4.4.

**Lemma 4.** Fix \( n \in \mathbb{N} \). The “filter” protocol \( \mathcal{F}_n \), depicted in Fig. 3 with initial register value 0 and target location \( s_n \), has a tight positive cut-off equal to \( n \).

**Proof.** We consider the system \( \mathcal{S}^m_{\mathcal{F}_n} \), made of \( m \) copies of \( \mathcal{F}_n \), with initial register value 0. We first prove that any reachable configuration \( \gamma \) satisfies:

\[
\forall j \leq m \sum_{k=0}^{j} st(\gamma)(s_k) \geq j + \mathbb{1}_{data(\gamma)=j+1}
\]

The proof is by induction: the invariant is satisfied by the initial configuration \( \gamma_0 = (s_0^m, 0) \). Let us now consider the run \( \gamma_0 \rightarrow^* \gamma \rightarrow \gamma' \), in which \( \gamma \) satisfies the invariant, and with last transition \((q, op, d, q') \in T\).

- If \((op, d) = (R, 0)\), then \( q = s_0 \) and \( q' = s_1 \). Along that transition, the right-hand-side term is unchanged; so is the left-hand-side term as soon as \( j > 0 \), so that the inequality is preserved for those cases. The case \( j = 0 \) is trivial.

- If \((op, d) = (R, i)\) with \( i > 0 \), then \( q = s_i \) and \( q' = s_{i+1} \). We have \( st(\gamma') = st(\gamma) \oplus s_i \oplus s_{i+1} \) and \( data(\gamma') = data(\gamma) = i \). Again, along this read-transition, the right-hand side term is unchanged, while the left-hand side term is unchanged for all \( j \neq i \).

It remains to prove the inequality for \( j = i \). We apply the induction hypothesis in \( \gamma \) for \( j = i-1 \): since the transition \((q, R, i, q') \) is available, it must hold that \( st(\gamma)(s_i) \geq 1 \) and \( data(\gamma) = i = j + 1 \). Hence \( \sum_{k=0}^{i-1} st(\gamma)(s_k) \geq i - 1 + 1 = i \), and \( \sum_{k=0}^{i} st(\gamma)(s_k) \geq i + 1 \). This implies \( \sum_{k=0}^{i} st(\gamma')(s_k) \geq i \).

- If \((op, d) = (W, i)\), then \( q = s_i \) and \( q' = s_0 \). Thus \( st(\gamma') = st(\gamma) \ominus s_i \oplus s_0 \). For \( j = i - 1 \), the left-hand-side term of the inequality is increased by 1, while the right-hand-side one is either unchanged or also increased by 1. The property is preserved in both cases. For \( j \neq i - 1 \), the left-hand-side term cannot decrease, while the right-hand-side term cannot increase. Hence the invariant is preserved.
As a consequence, if \( m < n \), we have (for \( j = m \)) \( \sum_{k=0}^{m} st(\gamma)(s_k) = m \) for any reachable configuration \( \gamma \), so that \( st(\gamma)(s_n) = 0 \).

Conversely, if \( m \geq n \), from any configuration \( \gamma \), it is possible to reach \( \gamma_i = (s_0^m \oplus s_{i+1}^{m-i}, i) \) for any \( 0 \leq i < n \):

- for \( i = 0 \): all processes can go to \( s_0 \), then write 0 in the register, and all move to \( s_1 \): \( \gamma \rightarrow (s_0^m, d) \xrightarrow{(W,0)} (s_0^m, 0) \xrightarrow{R,0} m(s_0^m, 0); \)
- for \( 1 \leq i < n - 1 \), assuming \( (s_0^m \oplus s_{i+1}^{m-i}, i) \) can be reached, one of the processes in \( s_{i+1} \) can write \( i + 1 \) (going back to \( s_0 \)), and the remaining \( m - i - 1 \) processes in \( s_{i+1} \) can go to \( s_{i+2} \):

\[
\langle s_0^m, i \rangle \xrightarrow{(W,i+1)} \langle s_0^m_{i+1}, i+1 \rangle \xrightarrow{(R,i+1)} m-i-1(s_0^m_{i+2}, i+1)
\]

Thus from any \( \gamma \in \Gamma \), configuration \( \gamma_{n-1} = (s_0^n_{n-1}, s_{n+m}^{m-n+1}, n-1) \) is reachable. Furthermore, \( \gamma_{n-1} \) contains the final state \( s_n \) since \( m \geq n \).

Hence, we deduce that there is a unique bottom strongly-connected component in \( S_P \), and that \( \gamma_{n-1} \) belongs to it: this configuration is reached with probability 1 from \( (s_0^n, 0) \). It follows that \( P((s_0^n, 0), [\emptyset s_f]) = 1 \).

### 3.2 Basic results

In this section, we consider a register protocol \( \mathcal{P} = (Q, D, q_0, T) \), its associated distributed system \( \mathcal{S}_P = (\Gamma, Pr) \), an initial register value \( d_0 \in D \) and a target state \( q_f \in Q \). We define a partial order \( \preceq \) over the set \( \Gamma \) of configurations as follows: \( \langle \mu, d \rangle \preceq \langle \mu', d' \rangle \) if, and only if, \( d = d' \) and \( \mu = \overline{\mu} \) and \( \mu \subseteq \mu' \). Note that with respect to the classical order over multisets, we require here that the supports of \( \mu \) and \( \mu' \) be the same (we add in fact a finite information to hold for the comparison). We know from Dickson’s lemma that \( \langle N^Q \rangle, \sqsubset \) is a wqo and since \( Q, D \) and the supports of multisets in \( N^Q \) are finite, we can deduce the following lemma.

**Lemma 5.** \( \langle \Gamma, \preceq \rangle \) is a wqo.

We will give some properties of register protocols, but first we introduce some further notations. Given a set of configuration \( \Delta \subseteq \Gamma \), we define \( \text{Pre}^*(\Delta) \) and \( \text{Post}^*(\Delta) \) as follows:

\[
\text{Pre}^*(\Delta) = \{ \gamma \in \Gamma \mid \exists \gamma' \in \Delta. \gamma \rightarrow^* \gamma' \} \quad \text{Post}^*(\Delta) = \{ \gamma' \in \Gamma \mid \exists \gamma \in \Delta. \gamma \rightarrow^* \gamma' \}
\]

We also define the set \([q_f]|\mathcal{S}_P\) of configurations we aim to reach as \( \{ \gamma \in \Gamma \mid st(\gamma)(q_f) > 0 \} \). It holds that \( \gamma \in \text{Pre}^*([q_f]|\mathcal{S}_P) \) if, and only if, there exists a path in \([\emptyset q_f]|\mathcal{S}_P\) starting in \( \gamma \).

As already mentioned, when \( \langle \mu, d \rangle \rightarrow \langle \mu', d' \rangle \) in \( \mathcal{S}_P \), the multisets \( \mu \) and \( \mu' \) have the same cardinality. This implies that given \( k > 0 \), the set \( \text{Post}^*([q_f]|\mathcal{S}_P) \) is finite (remember that \( Q \) and \( D \) are finite). As a consequence, for a fixed \( k \), checking whether \( P((q_0^k, d_0), [\emptyset q_f]) = 1 \) can be easily achieved by analyzing the finite-state discrete-time Markov chain \( \mathcal{S}_P^{(k)} \).

**Lemma 6.** Let \( k \geq 1 \). Then \( P((q_0^k, d_0), [\emptyset q_f]) = 1 \) if, and only if, \( \text{Post}^*([q_f]|\mathcal{S}_P) \subseteq \text{Pre}^*([q_f]|\mathcal{S}_P) \).

The difficulty here precisely lies in finding such a \( k \) and in proving that, once we have found one correct value for \( k \), all larger values are correct as well (to get the cut-off property). Characteristics of register protocols provide us with some tools to solve this problem. We base our analysis on reasoning on the set of configurations reachable from initial configurations in \( \uparrow\{(q_0, d_0)\} \) (the upward closure of \( \{(q_0, d_0)\} \) w.r.t. \( \preceq \)), remember that since the order \( \langle \Gamma, \preceq \rangle \) requires equality of support for elements to be comparable, we have that \( \uparrow\{(q_0, d_0)\} = \)
\[ \bigcup_{k \geq 1} \{ q^k_0, d_0 \} \]. We begin by showing that this set of reachable configurations and the set of configurations from which \([q_f]\) is reachable are both upward-closed. Thanks to Lemma 5, they can be represented as upward closures of finite sets. To show that \(\text{Post}^*([\{q_0, d_0\}])\) is upward-closed, we prove that register protocols enjoy the following monotonicity property. A similar property is given in [12] and derives from the non-atomicity of operations.

- **Lemma 7.** Let \(\gamma_1, \gamma_2, \text{ and } \gamma_2'\) be configurations in \(\Gamma\). If \(\gamma_1 \rightarrow^* \gamma_2 \text{ and } \gamma_2 \leq \gamma_2'\), then there exists \(\gamma_1' \in \Gamma\) such that \(\gamma_1' \rightarrow^* \gamma_2'\) and \(\gamma_1 \leq \gamma_1'\).

**Proof.** Assume \(\gamma_1 \rightarrow \gamma_2\) with transition \((q_1, op, d, q_2) \in T\) and \(\gamma_2 \leq \gamma_2'\). Let \(k = st(\gamma_2')(q_2) - st(\gamma_2)(q_2) \geq 0\). Then \(st(\gamma_2')(q_2) = k + st(\gamma_2)(q_2) \geq k + 1\) so we define \(\gamma_1' = \langle st(\gamma_2') \oplus q_2^{k+1} \oplus q_1^{k+1}, \text{data}(\gamma_1) \rangle\). Then:

- if \(op = W\), the path \(\gamma_1' \rightarrow \gamma_2'\), obtained by performing \(k + 1\) times the transition \((q_1, W, d, q_2)\), is a valid path since write operations can always be performed, independently of the content of the register;

- if \(op = R\), the path \(\gamma_1' \rightarrow \gamma_2'\), defined by applying \(k + 1\) times the transition \((q_1, R, d, q_2)\), is also a valid path, since the data \(d\) in the register is unchanged.

By construction, we have \(st(\gamma_1')(q_2) = st(\gamma_1')(q_2)\), and \(1 \leq st(\gamma_1')(q_1) \leq st(\gamma_1')(q_1)\), and \(st(\gamma_1')(q) = st(\gamma_1')(q)\) for all \(q \neq q_2\). Hence \(\gamma_1 \leq \gamma_1'\). The result is then generalized to arbitrary path \(\gamma_1 \rightarrow \gamma_2\) by induction.

\(\text{Pre}^*([q_f])\) is also upward-closed, since \([q_f]\) can be reached from some configuration \(\gamma\), it can also be reached by a larger configuration by keeping the extra copies idle. Thus:

- **Lemma 8.** \(\text{Post}^*([\{q_0, d_0\}])\) and \(\text{Pre}^*([q_f])\) are upward-closed sets in \((\Gamma, \leq)\).

### 3.3 Existence of a cut-off

From Lemma 8 and from the fact that \((\Gamma, \leq)\) is a wqo, there must exist two finite sequences of configurations \((\theta_i)_{1 \leq i \leq n}\) and \((\eta_i)_{1 \leq i \leq m}\) such that \(\text{Post}^*([\{q_0, d_0\}]) = [\theta_1, \ldots, \theta_n]\) and \(\text{Pre}^*([q_f]) = [\eta_1, \ldots, \eta_m]\). By analyzing these two sequences, we now prove that any register protocol has a cut-off (for any initial register value and any target location).

We let \(\Delta, \Delta' \subseteq \Gamma\) be two upward-closed sets (for \(\leq\)). We say that \(\Delta\) is included in \(\Delta'\) modulo single-state incrementation whenever for every \(\gamma \in \Delta\), for every \(q \in \gamma\), there is some \(k \in \mathbb{N}\) such that \(\gamma \oplus q^k \in \Delta'\). Note that this condition can be checked using only comparisons between minimal elements of \(\Delta\) and \(\Delta'\). In particular, we have the following lemma.

- **Lemma 9.** \(\text{Post}^*([\{q_0, d_0\}])\) is included in \(\text{Pre}^*([q_f])\) modulo single-state incrementation if, and only if, for all \(i \in [1; n]\), and for all \(q \in \gamma\), there exists \(j \in [1; m]\) such that \(\text{data}(\theta_i) = \text{data}(\eta_j)\) and \(\overline{\theta_i} = \overline{\eta_j}\) and \(st(\eta_j)(q^k) \leq st(\theta_i)(q')\) for all \(q' \in Q \setminus \{q\}\).

**Proof.** Suppose that \(\text{Post}^*([\{q_0, d_0\}])\) is included in \(\text{Pre}^*([q_f])\) modulo single-state incrementation. Let \(i \in [1; n]\) and \(q \in \gamma\). By definition, there exists some \(k \in \mathbb{N}\) such that \(\theta_i \oplus q^k \in \text{Pre}^*([q_f])\). Hence there is \(j \in [1; m]\) such that \(\eta_j \leq \theta_i \oplus q^k\). Hence we have \(\text{data}(\theta_i) = \text{data}(\eta_j)\) and \(\overline{\theta_i} = \overline{\eta_j}\) and \(st(\eta_j)(q^k) \leq st(\theta_i)(q')\) for all \(q' \in Q \setminus \{q\}\).

Now assume that for all \(i \in [1; n]\), and for all \(q \in \gamma\), there exists \(j \in [1; m]\) such that \(\text{data}(\theta_i) = \text{data}(\eta_j)\) and \(\overline{\theta_i} = \overline{\eta_j}\) and \(st(\eta_j)(q^k) \leq st(\theta_i)(q')\) for all \(q' \in Q \setminus \{q\}\). Let \(\gamma \in \text{Post}^*([\{q_0, d_0\}])\). Hence there exists \(i \in [1; n]\) such that \(\theta_i \leq \gamma\) (note that hence \(\overline{\theta_i} = \gamma\)). Let \(q \in \gamma\). Then there exists \(j \in [1; m]\) such that \(\text{data}(\theta_i) = \text{data}(\eta_j)\) and \(\overline{\theta_i} = \overline{\eta_j}\) and \(st(\eta_j)(q^k) \leq st(\theta_i)(q')\) for all \(q' \in Q \setminus \{q\}\). Let \(k = |st(\eta_j)(q) - st(\theta_i)(q)|\). We consider the configuration \(\gamma' = \gamma \oplus q^k\). For all \(q' \in Q \setminus \{q\}\), we have \(st(\eta_j)(q') \leq st(\theta_i)(q') \leq st(\gamma')(q')\).
And we have \(\text{st}(\eta_j)(q) \leq \text{st}(\theta_j)(q) + k \leq \text{st}(\gamma')(q)\). This allows us to deduce that \(\eta_j \preceq \gamma \oplus q^k\) and consequently \(\gamma \oplus q^k \in \text{Pre}^*([q_f])\). Consequently \(\text{Post}^*([q_0, d_0])\) is included in \(\text{Pre}^*([q_f])\) modulo single-state incrementation.

Using the previous characterization of inclusion modulo single-state incrementation for \(\text{Post}^*([q_0, d_0])\) and \(\text{Pre}^*([q_f])\) together with the result of Lemma 5, we are able to provide a first characterization of the existence of a negative cut-off.

\bf{Lemma 10.} If \(\text{Post}^*([q_0, d_0])\) is not included in \(\text{Pre}^*([q_f])\) modulo single-state incrementation, then \(\max_{1 \leq i \leq n}(\text{st}(\theta_i))\) is a negative cut-off.

**Proof.** Applying the previous lemma, there is \(i \in [1;n]\) and \(q \in \overline{\theta}_i\) such that for every \(j \in [1;m]\), either \(\text{data}(\theta_i) \neq \text{data}(\eta_j)\), or \(\overline{\theta}_i \neq \gamma_j\), or there is \(q_j \neq q\) such that \(\text{st}(\eta_j)(q_j) > \text{st}(\theta_i)(q_j)\).

Let \(k_i = |\text{st}(\theta_i)|\), and fix \(k \geq k_i\). We define \(\gamma_{i,k} = \theta_i \oplus q^{k-k_i}\). Clearly \(\theta_i \preceq \gamma_{i,k} \in \text{Post}^*([q_0, d_0])\). On the opposite, for every \(j \in [1;m]\), \(\eta_j \not\preceq \gamma_{i,k}\); hence we conclude that \(\gamma_{i,k} \notin \text{Pre}^*([q_f])\).

Applying Lemma 6, we get that \(\mathcal{P}([q^k_0, d_0], [\emptyset q_f]) < 1\) for every \(k \geq k_i\).

We now prove that if the condition of Lemma 10 fails to hold, then there is a positive cut-off. In order to make our claim precise, for every \(i \in [1;n]\) and for any \(q \in \overline{\theta}_i\), we let \(d_{i,q} = \max\{|\text{st}(\eta_j)(q) - \text{st}(\theta_i)(q)| | 1 \leq j \leq m\} \text{ and } \overline{\theta}_i = \gamma_\theta\).

\bf{Lemma 11.} If \(\text{Post}^*([q_0, d_0])\) is included in \(\text{Pre}^*([q_f])\) modulo single-state incrementation, then \(\max_{1 \leq i \leq n}(\text{st}(\theta_i)) + \sum_{q \in \overline{\theta}_i} d_{i,q}\) is a positive cut-off.

**Proof.** Let \(k_0 = \max_{1 \leq i \leq n}(\text{st}(\theta_i)) + \sum_{q \in \overline{\theta}_i} d_{i,q}\), and \(k \geq k_0\). Consider a configuration \(\gamma \in \text{Post}^*([q_0, d_0])\). There exists \(i \in [1;n]\) with \(\theta_i \preceq \gamma\).

Choose \(q \in \overline{\theta}_i = \gamma\) such that \(\text{st}(\gamma)(q) \geq \text{st}(\theta_i)(q) + d_{i,q}\) (this \(q\) should exist since \(\text{st}(\gamma) \geq k_0\)). We apply Lemma 3 and exhibit \(j \in [1;m]\) such that \(\text{data}(\theta_i) = \text{data}(\eta_j)\), \(\overline{\theta}_i = \gamma_j\) and for every \(q' \neq q\), \(\text{st}(\eta_j)(q') \leq \text{st}(\theta_i)(q')\). Now, \(\text{st}(\eta_j)(q) \leq \text{st}(\theta_i)(q) + d_{i,q} \leq \text{st}(\gamma)(q)\). We conclude that \(\eta_j \preceq \gamma\), and therefore that \(\text{Post}^*([q^k_0, d_0]) \subseteq \text{Pre}^*([q_f])\). By Lemma 6, we conclude that \(k_0\) is a positive cut-off.

The last two lemmas entail our first result:

\bf{Theorem 12.} Any register protocol admits a cut-off (for any given initial register value and target state).

### 4 Detecting negative cut-offs

We develop an algorithm for deciding whether a distributed system associated with a register protocol has a negative cut-off. Thanks to Theorem 12, this can also be used to detect the existence of a positive cut-off. Our algorithm relies on the construction and study of a symbolic graph, as we define below: for any given protocol \(\mathcal{P}\), the symbolic graph has bounded size, but can be used to reason about arbitrarily large distributed systems built from \(\mathcal{P}\). It will store sufficient information to decide the existence of a negative cut-off.
4.1 \( k \)-bounded symbolic graph

In this section, we consider a register protocol \( \mathcal{P} = (Q, D, q_0, T) \), its associated distributed system \( \mathcal{S}_\mathcal{P} = (\Gamma, \mathcal{P}_\mathcal{R}) \), an initial register value \( d_0 \in D \), and a target location \( q_f \in Q \) of \( \mathcal{P} \). With \( \mathcal{P} \), we associate a finite-state graph, called symbolic graph of index \( k \), which for \( k \) large enough contains enough information to decide the existence of a negative cut-off.

Definition 13. Let \( k \) be an integer. The symbolic graph of index \( k \) associated with \( \mathcal{P} \) and \( d_0 \) is the transition system \( \mathcal{G} = (\mathcal{V}, \mathcal{V}_0, \mathcal{E}) \) where

- \( \mathcal{V} = \mathcal{K}_k^Q \times 2^Q \times D \) contains triples made of a multiset of states of \( Q \) of size \( k \), a subset of \( Q \), and the content of the register; the multiset (called concrete part) is used to exactly keep track of a fixed set of \( k \) processes, while the subset of \( Q \) (the abstract part) encodes the support of the arbitrarily many remaining processes;
- \( \mathcal{V}_0 = \{q_0^0, \{q_0\}, \{d_0\}\} \);
- transitions are of two types, depending whether they involve a process in the concrete part or a process in the abstract part. Formally, there is a transition \( \langle \mu, S, d \rangle \rightarrow \langle \mu', S', d' \rangle \) whenever there is a transition \( \langle q, O, d'', q' \rangle \in T \) such that \( d = d' = d'' \) if \( O = R \) and \( d' = d'' \) if \( O = W \), and one of the following two conditions holds:
  - either \( S' = S \) and \( q \subseteq \mu \) (that is, \( \mu(q) > 0 \)) and \( \mu' = \mu \ominus q \ominus q' \);
  - or \( \mu = \mu' \) and \( q \in S \) and \( S' \in \{S \setminus \{q\} \cup \{q'\} \cup \{q'\}\} \).

The symbolic graph of index \( k \) can be used as an abstraction of distributed systems made of at least \( k + 1 \) copies of \( \mathcal{P} \): it keeps full information of the states of \( k \) processes, and only gives the support of the states of the other processes. In particular, the symbolic graph of index \( 0 \) provides only the states appearing in each configuration of the system.

![Figure 4 Symbolic graph (of index 0) of the protocol of Fig. 1](self-loops omitted).

Example 1.c. Consider the protocol depicted in Fig. 1. Its symbolic graph of index \( 0 \) is depicted in Fig. 4. Notice that the final state (representing all configurations containing \( q_f \)) is reachable from any state of this symbolic graph. However, our original protocol \( \mathcal{P} \) of Fig. 4 does not have a positive cut-off (assuming initial register value 0): indeed, with positive probability, a single process will go to \( q_1 \) and immediately write 1 in the register, thus preventing any other process to leave \( q_0 \); then one may check that the process in \( q_1 \) alone cannot reach \( q_f \), so that the probability of reaching \( q_f \) from \( q_0 \) is strictly less than 1, for any \( k > 0 \). This livelock is not taken into account in the symbolic graph of index \( 0 \), because from any configuration with support \( \{q_0, q_1\} \) and register data equal to 1, the symbolic graph has a transition to the configuration with support \( \{q_0, q_1, q_2\} \), which only exists in the concrete system when there are at least two processes in \( q_1 \). As we prove in the following, analyzing the symbolic graph for a sufficiently large index guarantees to detect such a situation.
For any index \( k \), the symbolic graph achieves the following correspondence:

**Lemma 14.** Given two states \( (\mu, S, d) \) and \( (\mu', S', d') \), there is a transition from \( (\mu, S, d) \) to \( (\mu', S', d') \) in the symbolic graph \( \mathcal{G} \) of index \( k \) if, and only if, there exist multisets \( \delta \) and \( \delta' \) with respective supports \( S \) and \( S' \), and such that \( (\mu \oplus \delta, d) \rightarrow (\mu' \oplus \delta', d') \) in \( S_P \).

**Proof.** We begin with the reverse implication: if there is a transition from \( (\mu \oplus \delta, d) \) to \( (\mu' \oplus \delta', d') \) (assuming it is a write transition, the other case being similar) in the distributed system, then this transition originates from a transition \( (q, W, d', q') \) in \( P \), and either this transition affects a process from the set of \( k \) processes that are monitored exactly by the symbolic graph, or it affects a process in the abstract part, in which only the support is monitored. In the former case, \( \delta = \delta' \), hence also their supports are equal, and the transition \( (q, W, d', q') \) is applied to a location in \( \mu \), which entails \( q \subseteq \mu \) and \( \mu' = \mu \oplus q \oplus q' \) and \( d' = d'' \). In the latter case, we get \( \mu = \mu' \), and the transition \( (q, W, d', q') \) is applied to a state in the support, so that \( q \in S \) and \( S' \) is either \( S \cup \{q'\} \) (in case \( \delta(q) > 1 \)), or it is \( S \setminus \{q\} \cup \{q'\} \) (in case \( \delta(q) = 1 \)).

Conversely, if there is a transition \( (\mu, S, d) \rightarrow (\mu', S', d') \) (assuming it originates from a \( W \)-transition \( (q, W, d', q') \) in \( P \), the other case being similar), we again have to consider two separate cases.

- The first case is when \( S' = S, q \subseteq \mu \) and \( \mu' = \mu \oplus q \oplus q' \), corresponding to the case where the transition is performed by one of the \( k \) processes tracked exactly by the symbolic graph. In that case, for any \( \delta \) with support \( S \), there is a transition from \( (\mu \oplus \delta, d) \) to \( (\mu' \oplus \delta, d') \) in the concrete distributed system.

- In the second case, \( \mu' = \mu, q \in S, \) and \( S' \) is either \( S \setminus \{q\} \cup \{q'\} \) or \( S \cup \{q'\} \). Consider any multiset \( \delta \) with support \( S \), and such that \( \delta(q) > 1 \) in case \( S' = S \cup \{q'\} \), and \( \delta(q) = 1 \) if \( S' = S \setminus \{q\} \cup \{q'\} \). Let \( \delta' = \delta \oplus q \oplus q' \); then the support of \( \delta' \) is \( S' \), and there is a transition from \( (\mu \oplus \delta, d) \) to \( (\mu' \oplus \delta', d') \), as required.

This concludes our proof.

### 4.2 Deciding the existence of a negative cut-off

We now explain how the symbolic graph can be used to decide the existence of a negative cut-off. Since \( \text{Pre}^*(\llbracket q \rrbracket) \) is upward-closed in \( (\mathcal{T}, \preceq) \), there is a finite set of configurations \( \{\eta_i = (\mu_i, d_i) \mid 1 \leq i \leq m\} \) such that \( \text{Pre}^*(\llbracket q \rrbracket) = \{\eta_i \mid 1 \leq i \leq m\} \). We let \( K = \max\{s\theta(\eta_i)(q) \mid q \in Q, 1 \leq i \leq m\} \), and show that for our purpose, it is enough to consider the symbolic graph of index \( K \cdot |Q| \); we provide a bound on \( K \) in the next section.

**Lemma 15.** There is a negative cut-off for \( P \), \( d_0 \) and \( q_f \) if, and only if, there is a node in the symbolic graph of index \( K \cdot |Q| \) that is reachable from \( \llbracket q_0^K \rrbracket \), \( \{q_0\}, d_0 \) but from which no configuration involving \( q_f \) is reachable.

**Proof.** We begin with the converse implication, assuming that there is a state \( (\mu, S, d) \) in the symbolic graph of index \( K \cdot |Q| \) that is reachable from \( \llbracket q_0^K \rrbracket \), \( \{q_0\}, d_0 \) and from which no configuration in \( \llbracket q_f \rrbracket \) is reachable. Applying Lemma 14, there exist multisets \( \delta_0 = q_0^N \) and \( \delta \), with respective supports \( \{q_0\} \) and \( S \), such that \( (\mu \oplus \delta, d) \) is reachable from \( \llbracket q_0^K \rrbracket \oplus \delta_0, d_0 \).

If location \( q_f \) was reachable from \( (\mu \oplus \delta, d) \) in the distributed system, then there would exist a path from \( (\mu, S, d) \) to a state involving \( q_f \) in the symbolic graph, which contradicts our hypothesis. By Lemma 7, it follows that such a configuration \( (\mu \oplus \delta', d) \) — which cannot reach \( q_f \) — can be reached from \( \llbracket q_0^K \rrbracket \oplus q_0^N, d_0 \) for any \( N' \geq N \); hence it cannot be the case that \( q_f \) is reachable almost-surely for any \( N' \geq N \). Therefore there cannot be a positive cut-off, which implies that there is a negative one (from Theorem 12).
12 Reachability in Networks of Register Protocols under Stochastic Schedulers

Conversely, if there is a negative cut-off, then for some \( N > K \cdot |Q| \), the distributed system \( S_P^N \) with \( N \) processes has probability less than 1 of reaching \( \llbracket q_f \rrbracket \) from \( q_0^N \). This system being finite, there must exist a reachable configuration \( \langle \mu, d \rangle \) from which \( q_f \) is not reachable [5]. Hence \( \langle \mu, d \rangle \notin \text{Pre}^*(\llbracket q_f \rrbracket) \), and for all \( i \leq m \), there is a location \( q^i \) such that \( \mu(q^i) < \mu_i(q^i) \leq K \). Then there must exist a reachable state \( \langle \kappa, S, d \rangle \) of the symbolic graph of index \( K \cdot |Q| \) for which \( \kappa(q^i) = \mu(q^i) \) and \( q^i \notin S \), for all \( 1 \leq i \leq m \): it indeed suffices to follow the path from \( \langle q_f^N, d_0 \rangle \) to \( \langle \mu, d \rangle \) while keeping track of the processes that end up in some \( q^j \) in the concrete part; this is possible because the concrete part has size at least \( K \cdot |Q| \).

It remains to be proved that no state involving \( q_f \) is reachable from \( \langle \kappa, S, d \rangle \) in the symbolic graph. If it were the case, then by Lemma 14, there would exist \( \delta \) with support \( S \) such that \( \llbracket q_f \rrbracket \) is reachable from \( \langle \kappa \oplus \delta, d \rangle \) in the distributed system. \( \langle \kappa \oplus \delta, d \rangle \in \text{Pre}^*(\llbracket q_f \rrbracket) \), so that for some \( 1 \leq j \leq m \), \( \langle \kappa \oplus \delta, q^j \rangle \geq \mu_i(q^j) \), which is not possible as \( \kappa(q^j) < \mu_i(q^j) \) and \( q^j \) is not in the support \( S \) of \( \delta \). This contradiction concludes the proof. □

**Remark.** Besides the existence of a negative cut-off, this proof also provides us with an upper bound on the tight cut-off, as we shall see in Section 5.

### 4.3 Complexity of the algorithm

We now consider the complexity of the algorithm that can be deduced from Lemma 15. Using results by Rackoff on the coverability problem in Vector Addition Systems [20], we can bound \( K \) and consequently the size of the needed symbolic graph—by a double-exponential in the size of the protocol. Therefore, it suffices to solve a reachability problem in NLOGSPACE [22] on this doubly-exponential graph: this boils down to NEXPSIZE with regard to the protocol’s size, hence EXPSPACE by Savitch’s theorem [22].

**Theorem 16.** Deciding the existence of a negative cut-off is in EXPSPACE.

**Proof.** Recall that \( \text{Pre}^*(\llbracket q_f \rrbracket) \) is exactly the set of configurations that can cover \( q_f \), i.e., configurations \( \gamma \) from which there exists a path \( \gamma \to^* \gamma' \) with \( \text{st}(\gamma')(q_f) > 0 \). Recall also that it can be written as an upward-closure of minimal elements: \( \text{Pre}^*(\llbracket q_f \rrbracket) = \uparrow \{ \eta_1, \ldots, \eta_m \} \). Now, consider the value \( K \) in Lemma 15; it is defined as \( K = \max \{ \text{st}(\eta_i)(q) \mid q \in Q, \, 1 \leq i \leq m \} \), i.e., the maximum number of states appearing in any multiset of any minimal configuration \( \eta_i \). The value of \( K \) can be bounded using classical results on the coverability problem in Vector Addition Systems (VAS) [20].

Intuitively, a \( b \)-dimensional VAS is a system composed of an initial \( b \)-dimensional vector \( v_0 \) of naturals (the axiom) and a finite set of \( b \)-dimensional integer vectors (the rules). An execution is built as follows: it starts from the axiom and, at each step, the next vector is derived from the current one by adding a rule, provided that this derivation is admissible, i.e., that the resulting vector only contains non-negative integers. An execution ends if no derivation is admissible. The coverability problem asks if a given target vector \( v = (v_1, \ldots, v_b) \) can be covered, i.e., does there exists a (possibly extendable) execution \( v_0 \leadsto v_1 \leadsto \ldots \leadsto v_n = v' \) such that, for all \( 1 \leq i \leq b \), it holds that \( v_i \leq v'_i \).

Our distributed system \( S_P \) can be seen as a \( |Q| \)-dimensional VAS where each transition is modeled by a rule vector modifying the multiset of the current configuration. Formally, one has to take into account that available rules depend on the data stored in the shared register. This can be achieved by either considering the expressively equivalent model of VAS with states (VASS, see e.g., [21]) or by adding \( O(|D|) \) dimensions to enforce this restriction. Over such a VAS(S), we are interested in the coverability of the vector corresponding to the multiset \( q_f \) (i.e., containing only one copy of \( q_f \) and no other state). In particular, we
want to bound the size of vectors needed to cover \( q_f \), as it will lead to a bound on minimal elements \( \eta_i \) of \( \text{Pre}^*(\|q_f\|) \), hence a bound on the value \( K \).

Results by Rackoff (hereby as reformulated by Demri et al. [11] Lemma 3) state that if a covering execution exists from an initial vector \( v_0 \), then there is one whose length may be doubly-exponential in the size of the input: singly-exponential in the size of the rule set and the target vector, and doubly-exponential in the dimension of the VAS. Hence, for our distributed system \( S_P \), seen as a VAS, this implies that if \( q_f \) can be covered from a configuration \( \gamma \), there is a covering execution whose length is bounded by some \( L \) in \( 2^{O(|Q|\cdot|D|)} \). This bound on the length of the execution obviously also implies a bound on the number of processes actively involved in the execution (because at each transition, only one process is active). Hence, we can deduce that if a configuration \( \gamma = (\mu, d) \) can cover \( q_f \) (i.e., there exists a path \( \gamma \rightarrow^\ast \gamma' \) with \( \text{st}(\gamma')(q_f) > 0 \)), then it is also the case of configuration \( \gamma'' = (\mu'', d) \), which we build as follows: \( \forall q \in Q, \mu''(q) = \min\{\mu(q), L\} \). That is, it also holds that there exists a path \( \gamma'' \rightarrow^\ast \gamma''' \) with \( \text{st}(\gamma''')(q_f) > 0 \).

By definition of \( K \) as \( K = \max\{\text{st}(\eta_i)(q) \mid q \in Q, 1 \leq i \leq m\} \) and configurations \( \eta_i \) as minimal elements for the upward-closure \( \text{Pre}^*(\|q_f\|) = \uparrow\{\eta_1, \ldots, \eta_m\} \), we have that \( K \leq L \) in any case. Hence, for our algorithm to be correct, it suffices to consider the symbolic graph of index \( L \cdot |Q| \), as presented in Lemma 15, and to solve a reachability problem over this graph. Let us study the size of this graph. Its state space is \( V = N^L_{\cdot |Q|} \times 2^Q \times D \). The multisets of \( N^L_{\cdot |Q|} \) are essentially mappings \( Q \rightarrow [0; L \cdot |Q|] \). Hence, we have that: \(|V| \leq (L \cdot |Q| + 1)^{|Q|} \cdot 2^{|Q|} \cdot |D|\), which is doubly-exponential in both the state space of the protocol and the size of the data alphabet (because \( L \) is). Since reachability over directed graphs lies in \text{NLOGSPACE} [22] with regard to the size of the graph, we obtain \text{NEXPSPACE}-membership with regard to the size of the protocol. Finally, by Savitch’s theorem [22], we know that \text{NEXPSPACE} = \text{EXPSPACE}, which concludes our proof.

4.4 PSPACE-hardness for deciding cut-offs

Our proof is based on the encoding of a linear-bounded Turing machine [22]: we build a register protocol for which there is a negative cut-off if, and only if, the machine reaches its final state \( q_{halt} \) with the tape head reading the last cell of the tape.

\[ \textbf{Theorem 17.} \text{Deciding the existence of a negative cut-off is PSPACE-hard.} \]

Write \( n \) for the size of the tape of the Turing machine. We assume (without loss of generality) that the machine is deterministic, and that it accepts only if it ends in its halting state \( q_{halt} \) while reading the last cell of the tape. Our reduction works as follows (see Fig. 5): some processes of our network will first be assigned an index \( i \) in \( [1; n] \) indicating the cell of the tape they shall encode during the simulation. The other processes are stuck in the initial location, and will play no role. The state \( q \) and position \( j \) of the head of the Turing machine are stored in the register. During the simulation phase, when a process is scheduled to play, it checks in the register whether the tape head is on the cell it encodes, and in that case it performs the transition of the Turing machine. If the tape head is not on the cell it encodes, the process moves to the target location (which we consider as the target for the almost-sure reachability problem). Finally, upon seeing \( (q_{halt}, n) \) in the register, all processes move to a \((n + 1)\)-filter protocol \( F_{n+1} \) (similar to that of Fig. 3) whose last location \( s_{n+1} \) is the aforementioned target location.

If the Turing machine halts, then the corresponding run can be mimicked with exactly one process per cell, thus giving rise to a finite run of the distributed system where \( n \) processes
Reachability in Networks of Register Protocols under Stochastic Schedulers

Figure 5 Distributed protocol $P_M$ encoding the linear-bounded Turing machine $M$.

end up in the $(n+1)$-filter (and the other processes are stuck in the initial location); from there $s_{n+1}$ cannot be reached. If the Turing machine does not halt, then assume that there is an infinite run of the distributed system never reaching the target location. This run cannot get stuck in the simulation phase forever, because it would end up in a strongly connected component from which the target location is reachable. Thus this run eventually reaches the $(n+1)$-filter, which requires that at least $n+1$ processes participate in the simulation (because with $n$ processes it would simulate the exact run of the machine, and would not reach $q_{halt}$, while with fewer processes the tape head could not go over cells that are not handled by a process). Thus at least $n+1$ processes would end up in the $(n+1)$-filter, and with probability 1 the target location should be reached.

We now formalize this construction, by describing the states and transitions of the protocol within these three phases. We fix a linear-bounded Turing machine $M = (Q, q_0, q_{halt}, \Sigma, \delta)$, where $Q$ is the set of states, $q_0, q_{halt} \in Q$ are the initial and halting states, $\Sigma$ is the alphabet, and $\delta \subseteq Q \times \Sigma \times Q \times \Sigma \times \{-1,+1\}$ is the set of transitions. We define the data alphabet $D = \{\#\} \cup Q \times \Sigma \cup \{f_i \mid 0 \leq i \leq n\}$, and the set of locations $P = \{p_{init}, p'_{init}, p_{sink}\} \cup \{(1:n) \times \Sigma \times (Q \cup \{\epsilon\})\} \cup \{(s_i \mid 0 \leq i \leq n+1\}$. The set of locations corresponds to three phases (see Fig. 5):

- The initialization phase contains $p_{init}, p'_{init}$ and $p_{sink}$. From the initial state $p_{init}$, upon reading # (the initial content of the register), the protocol has transitions to each state $(i, \sigma)$ for all $2 \leq i \leq n$, where $\sigma$ is the $i$-th letter of the initial content of the tape. If reading anything different from #, the protocol moves to the sink state $p_{sink}$. Finally, there are transitions $(p_{init}, R(\#), p'_{init})$ and $(p'_{init}, W(q_0,1), (1,c_1))$, where $q_0$ is the initial state of the Turing machine: this pair of transitions is used to initialize the computation, by setting the content of the first cell and modifying the register, so that the initialization phase is over (there are no transitions writing # in the register).

- The second phase, called simulation phase, uses register alphabet $Q \times [1,n)$, in order to encode the state and position of the head of the Turing machine. The state space for the simulation phase is $[1:n] \times \Sigma \times (Q \cup \{\epsilon\})$: state $(i, \sigma, \epsilon)$ (written $(i, \sigma)$ in the sequel)
encodes the fact that the content of the $i$-th cell is $\sigma$; the states of the form $(i, \sigma, q)$ are intermediary states used during the simulation of one transition: when in state $(i, \sigma)$ and reading $(q, i)$ in the register, the protocol moves to $(i, \sigma, q)$, from which it moves to $(i, \sigma')$ and writes $(q', j)$ in the register, provided that the machine has a transition $(q, \sigma) \rightarrow (q', \sigma', j - i)$. If the active process does not encode the position that the tape head is reading (i.e., the process is in state $(i, \sigma)$ and reads $(q, j)$ with $j \neq i$) then it moves to the final state $s_{n+1}$ of the third phase.

The role of the *counting phase* is to count the number of processes participating in the simulation. When seeing the halting state in the register, each protocol moves to a module whose role is to check whether at least $n + 1$ protocols are still “running”. This uses data \{ $f_i \mid 0 \leq i \leq n$ \} and states \{ $s_i \mid i \in [0, n + 1]$ \}, with transitions from any state of the simulation phase to $s_0$ if the register contains $(\#_{\text{halt}}, n)$ or any of \{ $f_i \mid 0 \leq i \leq n$ \}.

We now prove that our construction is correct:

**Lemma 18.** The register protocol $P_M$, with initial register content $\#$ and target location $s_{n+1}$, has a negative cut-off if, and only if, the Turing machine $M$ reaches $q_{\text{halt}}$ in the last cell of the tape.

**Proof.** First assume that there is a negative cut-off: there exists $N_0$ such that for any $N \geq N_0$, starting from the initial configuration $(P_{\text{init}}, \#)$ of the system $S_{P_M}^N$ made of $N$ copies of $P_M$, the probability that at least one process reaches $s_{n+1}$ is strictly less than 1. Since $S_{P_M}^N$ is a finite Markov chain, this implies that there is a cone of executions never visiting $s_{n+1}$, i.e., a finite execution $\rho$ whose continuations never visit $s_{n+1}$. Since the register initially contains $\#$, this finite execution (or a finite continuation of it) must contain at least one configuration where some process has entered the simulation part.

Now, in the simulation phase, we notice that, right after taking a transition $((i, \sigma, q), w(q', i \pm 1), (i, \sigma'))$, the transition $((i, \sigma'), r(\cdot, j), s_{n+1})$ is always enabled. It follows that at the end of the finite run $\rho$, no simulation transition should be enabled; hence all processes that had entered the simulation part should have left it. Hence some process must have visited $s_0$ along $\rho$ (because we assume that $\rho$ does not involve $s_{n+1}$). Moreover, by Lemma 17 for $s_{n+1}$ not to be reachable along any continuation of $\rho$, no more than $n$ processes must be able to reach $s_0$ along any continuation of $\rho$, hence at most $n$ processes may have entered the simulation phase. On the other hand, for $s_0$ to be visited, some process has to first write $(\#_{\text{halt}}, n)$ in the register; since the register initially contains $(\#_0, 1)$, and no process can write $(\cdot, i + 1)$ without first reading $(\cdot, i)$, then for each $i \in [1, n]$ there must be at least one process visiting some state $(i, \sigma_i)$, for some $\sigma_i$; It follows that at least $n$ processes must have entered the simulation phase.

In the end, along $\rho$, exactly one process visits $(i, c_i)$, for each $i \in [1, n]$, and encode the content of the $i$-th cell. As a consequence, along $\rho$, each cell of the tape of the Turing machine is encoded by exactly one process, and the execution mimics the exact computation of the Turing machine. Since the configuration $(\#_{\text{halt}}, n)$ is eventually reached, the Turing machine halts with the tape head on the last cell of the tape.

Conversely, assume the Turing machine halts, and consider the execution of $N \geq n$ processes where exactly one process goes in each of the $(i, c_i)$ and mimics the run of the Turing machine (the other processes going to $P_{\text{sink}}$). We get a finite execution ending up in a configuration where all processes are either in $P_{\text{init}}$ or in $P_{\text{sink}}$, except for $n$ processes that are in the counting phase. No continuation of this prefix ever reaches $s_{n+1}$, so that the probability that some process reaches $s_{n+1}$ is strictly less than 1. □
5 Bounds on cut-offs

5.1 Existence of exponential tight negative cut-offs

We exhibit a family of register protocols that admits negative cut-off exponential in the size of the protocol. The construction reuses ideas from the PSPACE-hardness proof. Our register protocol has two parts: one part simulates a counter over \( n \) bits, and requires a token (a special value in the register) to perform each step of the simulation. The second part is used to generate the tokens (i.e., writing 1 in the register). Figure 6 depicts our construction.

We claim that this protocol, with \(#\) as initial register value and \( q_f \) as target location, admits a negative tight cut-off larger than \( 2^n \): in other terms, there exists \( N > 2^n \) such that the final state will be reached with probability strictly less than 1 in the distributed system made of at least \( N \) processes (starting with \( # \) in the register), while the distributed system with \( 2^n \) processes will reach the final state almost-surely. In order to justify this claim, we explain now the intuition behind this protocol.

![Figure 6 Simulating an exponential counter: grey boxes contain the nodes used to encode the bits of the counter; yellow nodes at the bottom correspond to the filter module from Fig. 3; purple nodes \( tok, sent \) and \( sink \) correspond to the second part of the protocol, and are used to produce tokens. Missing read edges are assumed to be self-loops.](image)

We first focus on the first part of the protocol, containing nodes named \( a_i, b_i, c_i, d_i \) and \( s_i \). This part can be divided into three phases: the initialization phase lasts as long as the register contains \( # \); the counting phase starts when the register contains \( halt \) for the first time; the simulation phase is the intermediate phase.

During the initialization phase, processes move to locations \( a_i \) and \( tok \), until some process in \( tok \) writes 1 in the register (or until some process reaches \( q_f \), using a transition from \( a_i \) to \( q_f \) while reading \#). Write \( \gamma_0 \) for the configuration reached when entering the simulation phase (i.e., when 1 is written in the register for the first time). We assume that \( st(\gamma_0)(a_i) > 0 \) for some \( i \), as otherwise all the processes are in \( tok \), and they all will eventually reach \( q_f \). Now, we notice that if \( st(\gamma_0)(a_i) = 0 \) for some \( i \), then location \( d_n \) cannot be reached, so that no process can reach the counting phase. In that case, some process (and actually all of them) will eventually reach \( q_f \). We now consider the case where \( st(\gamma_0)(a_i) \geq 1 \) for all \( i \).
One can prove (inductively) that $d_i$ is reachable when $st(\gamma_0)(\text{tok}) \geq 2^i$. Hence $d_n$, and thus also $s_0$, can be reached when $st(\gamma_0)(\text{tok}) \geq 2^n$. Assuming $q_f$ is not reached, the counting phase must never contain more than $n$ processes, hence we actually have that $st(\gamma_0)(a_i) = 1$. With this new condition, $s_0$ is reached if, and only if, $st(\gamma_0)(\text{tok}) \geq 2^n$. When the latter condition is not true, $q_f$ will be reached almost-surely, which proves the second part of our claim: the final location is reached almost-surely in systems with strictly less than $n + 2^n$ copies of the protocol.

We now consider the case of systems with at least $n + 2^n$ processes. We exhibit a finite execution of those systems from which no continuation can reach $q_f$, thus proving that $q_f$ is reached with probability strictly less than 1 in those systems. The execution is as follows: during initialization, for each $i$, one process enters $a_i$; all other processes move to tok, and one of them write 1 in the register. The $n$ processes in the simulation phase then simulate the consecutive incrementations of the counter, consuming one token at each step, until reaching $d_n$. At that time, all the processes in tok move to sent, and the process in $d_n$ writes halt in the register and enters $s_0$. The processes in the simulation phase can then enter $s_0$, and those in sent can move to sink. We now have $n$ processes in $s_0$, and the other ones in sink. According to Lemma 14, location $q_f$ cannot be reached from this configuration, which concludes our proof.

\begin{theorem}
There exists a family of register protocols which, equipped with an initial register value and a target location, admit negative tight cut-offs whose size are exponential in the size of the protocol.
\end{theorem}

\begin{remark}
The question whether there exists protocols with exponential positive cut-offs remains open. The family of filter protocols described at Section 3.1 is an example of protocols with a linear positive cut-off.
\end{remark}

### 5.2 Upper bounds on tight cut-offs

The results (and proofs) of Section 4 can be used to derive upper bounds on tight cut-offs. We make this explicit in the following theorem.

\begin{theorem}
For a protocol $P = (Q, D, q_0, T)$ equipped with an initial register value $d_0 \in D$ and a target location $q_f \in Q$, the tight cut-off is at most doubly-exponential in $|P|$.
\end{theorem}

\begin{proof}
First assume that the cut-off is negative. From Lemma 15, there is a state $(\mu, S, d)$ in the symbolic graph of index $K \cdot |Q|$ that is reachable from $(q_0^{K \cdot |Q|}, \{q_0\}, d_0)$ and from which no configuration containing $q_f$ is reachable. Applying Lemma 14, there exist multisets $\delta_0 = q_0^N$ and $\delta$, with respective supports $\{q_0\}$ and $S$, such that $(\mu \oplus \delta, d)$ is reachable from $(q_0^{K \cdot |Q|} \oplus \delta_0, d_0)$. Hence $N + K \cdot |Q|$ is a negative cut-off.

Let us evaluate the size of $N$: this number is extracted from the symbolic path $(q_0^{K \cdot |Q|}, \{q_0\}, d_0) \rightarrow^* (\mu, S, d)$, which has length at most $|V| - 1$ (where $V$ is the set of states of the symbolic graph of index $K \cdot |Q|$). Applying Lemma 15 $|V| - 1$ times, increasing the size of the concrete representation of $S$ by one each time, we get $N \leq |V|$. Thus, both $K \cdot |Q|$ and $N$ are doubly-exponential in $|P|$, thanks to the proof of Theorem 16.

The proof of Lemma 15 also entails that if the distributed system with some $N > K \cdot |Q|$ processes does not almost-surely reach the target state, then there is a negative cut-off. Hence for there to be a positive cut-off, the target has to be almost-surely reachable for all $N > K \cdot |Q|$, which makes $K \cdot |Q|$ a (doubly-exponential) positive cut-off.
\end{proof}
Conclusions and future works

We have shown that in networks of identical finite-state automata communicating (non-atomically) through a single register and equipped with a fair stochastic scheduler, there always exists a cut-off on the number of processes which either witnesses almost-sure reachability of a specific control-state (positive cut-off) or its negation (negative cut-off). This cut-off determinacy essentially relies on the monotonicity induced by our model, which allows to use well-quasi order techniques. By analyzing a well-chosen symbolic graph, one can decide in \textsc{ExpSpace} whether that cut-off is positive, or negative, and we proved this decision problem to be \textsc{PSpace}-hard. This approach allows us to deduce some doubly-exponential bounds on the value of the cut-offs. Finally, we gave an example of a network in which there is a negative cut-off, which is exponential in the size of the underlying protocol. Note however that no such lower-bound is known yet for positive cut-offs.

We have several further directions of research. First, it would be nice to fill the gap between the \textsc{PSpace} lower bound and the \textsc{ExpSpace} upper bound for deciding the nature of the cut-off. We would like also to investigate further atomic read/write operations, which generate non-monotonic transition systems, but for which we would like to decide whether there is a cut-off or not. Finally, we believe that our techniques could be extended to more general classes of properties, for instance, universal reachability (all processes should enter a distinguished state), or liveness properties.

References


