Reachability in Networks of Register Protocols under Stochastic Schedulers*†

Patricia Bouyer[1], Nicolas Markey[2], Mickael Randour‡[3], Arnaud Sangnier[4], and Daniel Stan[5]

1 **LSV, CNRS & ENS de Cachan, Cachan Cedex, France, and University Paris-Saclay, Paris, France**
2 **LSV, CNRS & ENS de Cachan, Cachan Cedex, France, and University Paris-Saclay, Paris, France**
3 **Computer Science Department, Université Libre de Bruxelles, Brussels, Belgium**
4 **IRIF, University Paris Diderot & CNRS, Paris, France**
5 **LSV, CNRS & ENS de Cachan, Cachan Cedex, France, and University Paris-Saclay, Paris, 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.

1998 ACM Subject Classification F.1.1 Models of Computation, F.3.1 Specifying and Verifying and Reasoning about Programs, C.2.2 Network Protocols

Keywords and phrases Networks of Processes, Parametrized Systems, Stochastic Scheduler, Almost-sure Reachability, Cut-Off Property

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.106

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

* A full version of the paper is available on Arxiv as [7].

† This work has been partly supported by ERC Starting grant EQualIS (FP7-308087), by European FET project Cassting (FP7-601148), and by the ANR research program PACS (ANR-14-CE28-0002).

‡ F.R.S.-FNRS Postdoctoral Researcher.

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 [17]. 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 communication [15, 10], token-passing [8, 2], message passing [6], shared register with ring topologies [1], or shared memory [16]. In his nice survey on such parameterized models [14], 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 [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 [18], 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 [16] and [11], 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 [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 [17] and for systems with disjunctive guards (where processes can query the states of other processes) in [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 [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 EXPSPACE, and that this problem is 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, 13, 12].

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 $\mu\colon S \to \mathbb{N}$. The cardinality of a multiset μ is $|\mu| = \sum_{s\in S} \mu(s)$. The support $\overline{\mu}$ of μ 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 $\mathbb{N}_k^S$ 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 μ is included in a multiset μ', written $\mu \sqsubseteq \mu'$, if $\mu(s) \leq \mu'(s)$ for all $s \in S$. Given two multisets μ and μ', their union $\mu \oplus \mu'$ is still a multiset s.t. $(\mu \oplus \mu')(s) = \mu(s) + \mu'(s)$ for all $s \in S$. Assuming $\mu \sqsubseteq \mu'$, the difference $\mu' \ominus \mu$ is still a multiset s.t. $(\mu' \ominus \mu)(s) = \mu'(s) - \mu(s)$.

A quasi-order $\langle A, \preceq\rangle$ 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 $n > 0$, $\langle \mathbb{N}^n, \leq\rangle$ (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 $a_1 \in B$ and $a_2 \in A$, in case $a_1 \preceq a_2$, then $a_2 \in B$. 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 $\langle A, \preceq\rangle$ 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 $\mathcal{P} = \langle Q, D, q_0, T\rangle$, 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.

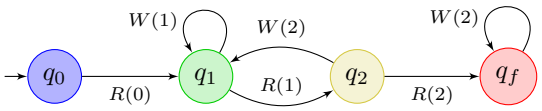

Figure 1 Example of a register protocol with $D = \{0, 1, 2\}$.

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 1 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*.*

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 in [18, 16, 11]. More precisely, when one process performs a transition, then all the processes that are in the same state are allowed to also perform the same transition just after, in fact write are always possible, and if a process performs a read of a specific value, since this read does not alter the value of the register, all processes in the same state can perform the same read (until one process performs a write). We will see later that dropping this hypothesis has a consequence on our results. We now give the formal definition of such a system.

The configurations of the distributed system built on register protocol $\mathcal{P} = \langle Q, D, q_0, T\rangle$ belong to the set $\Gamma = \mathbb{N}^Q \times D$. The first component of a configuration is a multiset characterizing the number of processes in each state of Q, whereas the second component provides the content of the register. For a configuration $\gamma = \langle \mu, d\rangle$, we denote by $st(\gamma)$ the multiset μ in $\mathbb{N}^Q$ and by $data(\gamma)$ the data d in D. We overload the operators defined over multisets; in particular, for a multiset δ over Q, we write $\gamma \oplus \delta$ for the configuration $\langle \mu \oplus \delta, d\rangle$. Similarly, we write $\overline{\gamma}$ for the support of $st(\gamma)$.

A configuration $\gamma' = \langle \mu', d'\rangle$ is a *successor* of a configuration $\gamma = \langle \mu, d\rangle$ if, and only if, there is a transition $(q, \mathsf{op}, d'', q') \in T$ such that $\mu(q) > 0$, $\mu' = \mu \ominus q \oplus q'$ and either $\mathsf{op} = R$ and $d = d' = d''$, or $\mathsf{op} = W$ and $d' = d''$. In that case, we write $\gamma \to \gamma'$. Note that since $\mu(q) > 0$ and $\mu' = \mu \ominus q \oplus q'$, we have necessarily $|\mu| = |\mu'|$. In our system, we assume that there is no creation or deletion of processes during an execution, hence the size of

configurations (i.e., $|st(\gamma)|$) remains constant along transitions. We write Γ_k for the set of configurations of size k. For any configuration $\gamma \in \Gamma_k$, we denote by $\mathrm{Post}(\gamma) \subseteq \Gamma_k$ the set of successors of γ, and point out that such a set is finite and non-empty.

Now, the *distributed system* $\mathcal{S}_\mathcal{P}$ associated with a register protocol $\mathcal{P}$ is a discrete-time Markov chain $\langle \Gamma, Pr \rangle$ where $Pr \colon \Gamma \times \Gamma \to [0,1]$ is the transition probability matrix defined as follows: for all γ and $\gamma' \in \Gamma$, we have $Pr(\gamma, \gamma') = \frac{1}{|\mathrm{Post}(\gamma)|}$ if $\gamma \to \gamma'$, and $Pr(\gamma, \gamma') = 0$ otherwise. Note that Pr is well defined: by the restriction imposed on the transition relation T of the protocol, we have $0 < |\mathrm{Post}(\gamma)| < \infty$ for all configuration γ, and hence we also get $\Sigma_{\gamma' \in \Gamma} Pr(\gamma, \gamma') = 1$. 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 $\mathcal{S}^k_\mathcal{P} = \langle \Gamma_k, \Pr_k \rangle$, where $\Pr_k$ is the restriction of Pr 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 $\mathcal{S}_\mathcal{P}$ is a finite sequence of configurations $\gamma_0 \to \gamma_1 \ldots \to \gamma_k$. In such a case, we say that the path starts in γ_0 and ends in γ_k. We furthermore write $\gamma \to^* \gamma'$ if, and only if, there exists a path that starts in γ and ends in γ'. Given a location q_f, we denote by $[\![\Diamond q_f]\!]$ the set of paths of the form $\gamma_0 \to \gamma_1 \ldots \to \gamma_k$ for which there is $i \in [0; k]$ such that $st(\gamma_i)(q_f) > 0$. Given a configuration γ, we denote by $\mathbb{P}(\gamma, [\![\Diamond q_f]\!])$ the probability that some paths starting in γ belong to $[\![\Diamond q_f]\!]$ in $\mathcal{S}_\mathcal{P}$. This probability is well-defined since the set of such paths is measurable (see e.g., [5]). Given a register protocol $\mathcal{P} = \langle Q, D, q_0, T \rangle$, 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 $\mathbb{P}(\langle q_0^k, d_0 \rangle, [\![\Diamond q_f]\!]) = 1$.

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

- for all $h \geq k$, we have $\mathbb{P}(\langle q_0^h, d_0 \rangle, [\![\Diamond q_f]\!]) = 1$. In this case k is a *positive* cut-off;
- for all $h \geq k$, we have $\mathbb{P}(\langle q_0^h, d_0 \rangle, [\![\Diamond q_f]\!]) < 1$. 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 $\mathcal{P}$, 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 $\mathcal{P}$. Whether it is a positive or a negative cut-off can be decided in* EXPSPACE*, and is* PSPACE*-hard.*

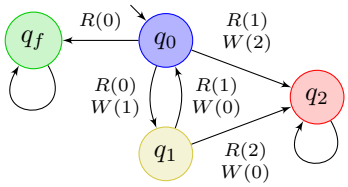

Figure 2 Example of a register protocol with atomic read/write operations.

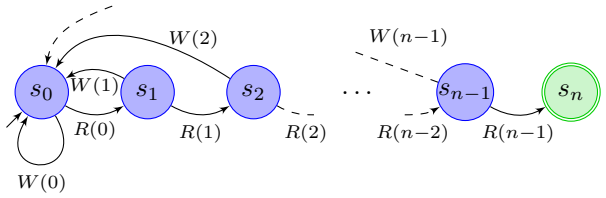

Figure 3 A "filter" protocol $\mathcal{F}_n$ for $n > 0$.

▶ 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. 3. 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.*

3.2 Basic results

In this section, we consider a register protocol $\mathcal{P} = \langle Q, D, q_0, T\rangle$, its associated distributed system $\mathcal{S}_\mathcal{P} = \langle \Gamma, Pr\rangle$, 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 Γ of configurations as follows: $\langle \mu, d\rangle \preceq \langle \mu', d'\rangle$ if, and only if, $d = d'$ and $\overline{\mu} = \overline{\mu'}$ and $\mu \sqsubseteq \mu'$. Note that with respect to the classical order over multisets, we require here that the supports of μ and μ' be the same (we add in fact a finite information to hold for the comparison). We know from Dickson's lemma that $\langle \mathbb{N}^Q, \sqsubseteq\rangle$ is a wqo and since Q, D and the supports of multisets in $\mathbb{N}^Q$ 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 $\mathrm{Pre}^*(\Delta)$ and $\mathrm{Post}^*(\Delta)$ as follows:

$$\mathrm{Pre}^*(\Delta) = \{\gamma \in \Gamma \mid \exists \gamma' \in \Delta . \gamma \to^* \gamma'\} \qquad \mathrm{Post}^*(\Delta) = \{\gamma' \in \Gamma \mid \exists \gamma \in \Delta . \gamma \to^* \gamma'\}$$

We also define the set $[\![q_f]\!]$ of configurations we aim to reach as $\{\gamma \in \Gamma \mid st(\gamma)(q_f) > 0\}$. It holds that $\gamma \in \mathrm{Pre}^*([\![q_f]\!])$ if, and only if, there exists a path in $[\![\Diamond q_f]\!]$ starting in γ.

As already mentioned, when $\langle \mu, d\rangle \to \langle \mu', d'\rangle$ in $\mathcal{S}_{\mathcal{P}}$, the multisets μ and μ' have the same cardinality. This implies that given $k > 0$, the set $\mathrm{Post}^*(\{\langle q_0^k, d_0\rangle\})$ is finite (remember that Q and D are finite). As a consequence, for a fixed k, checking whether $\mathbb{P}(\langle q_0^k, d_0\rangle, [\![\Diamond q_f]\!]) = 1$ can be easily achieved by analyzing the finite-state discrete-time Markov chain $\mathcal{S}_{\mathcal{P}}^k$ [5].

▶ **Lemma 6.** *Let $k \geq 1$. Then $\mathbb{P}(\langle q_0^k, d_0\rangle, [\![\Diamond q_f]\!]) = 1$ if, and only if, $\mathrm{Post}^*(\{\langle q_0^k, d_0\rangle\}) \subseteq \mathrm{Pre}^*([\![q_f]\!])$.*

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 $\uparrow\{\langle q_0, d_0\rangle\}$ (the upward closure of $\{\langle q_0, d_0\rangle\}$ 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 $\uparrow\{\langle q_0, d_0\rangle\} = \bigcup_{k\geq 1}\{\langle q_0^k, d_0\rangle\}$. 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 $\mathrm{Post}^*(\uparrow\{\langle q_0, d_0\rangle\})$ is upward-closed, we prove that register protocols enjoy the following monotonicity property. A similar property is given in [11] and derives from the non-atomicity of operations.

▶ **Lemma 7.** *Let γ_1, γ_2, and γ_2' be configurations in Γ. If $\gamma_1 \to^* \gamma_2$ and $\gamma_2 \preceq \gamma_2'$, then there exists $\gamma_1' \in \Gamma$ such that $\gamma_1' \to^* \gamma_2'$ and $\gamma_1 \preceq \gamma_1'$.*

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

▶ **Lemma 8.** $\mathrm{Post}^*(\uparrow\{\langle q_0, d_0\rangle\})$ *and* $\mathrm{Pre}^*([\![q_f]\!])$ *are upward-closed sets in* $\langle \Gamma, \preceq\rangle$.

3.3 Existence of a cut-off

From Lemma 8, and from the fact that $\langle \Gamma, \preceq\rangle$ 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 $\mathrm{Post}^*(\uparrow\{\langle q_0, d_0\rangle\}) = \uparrow\{\theta_1, \ldots, \theta_n\}$ and $\mathrm{Pre}^*([\![q_f]\!]) = \uparrow\{\eta_1, \ldots, \eta_m\}$. 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 $\preceq$). We say that Δ *is included in* Δ' *modulo single-state incrementation* whenever for every $\gamma \in \Delta$, for every $q \in \overline{\gamma}$, 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 Δ and Δ'. In particular, we have the following lemma.

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

Using the previous characterization of inclusion modulo single-state incrementation for $\mathrm{Post}^*(\uparrow\{\langle q_0, d_0\rangle\})$ and $\mathrm{Pre}^*([\![q_f]\!])$ together with the result of Lemma 6, we are able to provide a first characterization of the existence of a negative cut-off.

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

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 $d_{i,q} = \max\{(|st(\eta_j)(q) - st(\theta_i)(q)|) \mid 1 \leq j \leq m \text{ and } \overline{\theta_i} = \overline{\eta_j}\}$.

▶ **Lemma 11.** *If* $\mathrm{Post}^*(\uparrow\{\langle q_0, d_0\rangle\})$ *is included in* $\mathrm{Pre}^*([\![q_f]\!])$ *modulo single-state incrementation, then* $\max_{1\leq i\leq n}(|st(\theta_i)| + \sum_{q\in\overline{\theta_i}} d_{i,q})$ *is a positive cut-off.*

The last two lemmas entail our first result:

▶ **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 $\mathcal{P} = \langle Q, D, q_0, T\rangle$, its associated distributed system $\mathcal{S}_\mathcal{P} = \langle \Gamma, Pr\rangle$, 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 $\mathcal{G} = \langle V, v_0, E\rangle$ where

- $V = \mathbb{N}_k^Q \times 2^Q \times D$ 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;
- $v_0 = \langle q_0^k, \{q_0\}, \{d_0\}\rangle$;
- 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 $\langle \mu, S, d\rangle \to \langle \mu', S', d'\rangle$ whenever there is a transition $(q, O, d'', q') \in T$ 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 $q \sqsubseteq \mu$ (that is, $\mu(q) > 0$) and $\mu' = \mu \ominus q \oplus q'$;
  - or $\mu = \mu'$ and $q \in S$ and $S' \in \{S \setminus \{q\} \cup \{q'\}, S \cup \{q'\}\}$.

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.

▶ **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. 1 does not have a positive cut-off (assuming initial register value* 0*): indeed, with*

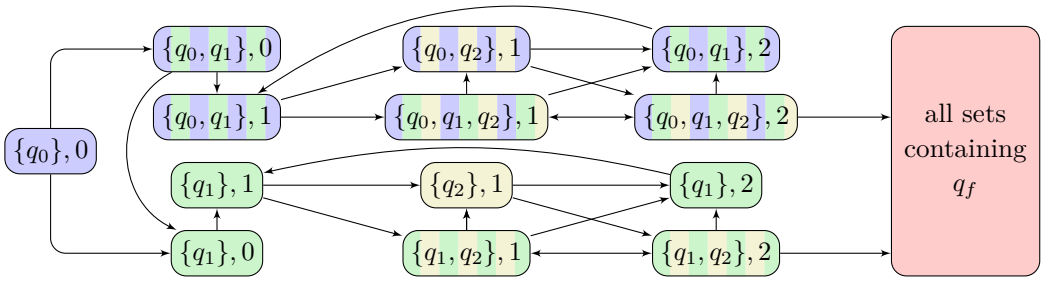

Figure 4 Symbolic graph (of index 0) of the protocol of Fig. 1 (self-loops omitted).

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^k 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 $\langle \mu, S, d\rangle$ and $\langle \mu', S', d'\rangle$, there is a transition from $\langle \mu, S, d\rangle$ to $\langle \mu', S', d'\rangle$ in the symbolic graph $\mathcal{G}$ of index k if, and only if, there exist multisets δ and δ' with respective supports S and S', and such that $\langle \mu \oplus \delta, d\rangle \to \langle \mu' \oplus \delta', d'\rangle$ in $\mathcal{S}_\mathcal{P}$.*

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 $\mathrm{Pre}^*(\llbracket q_f \rrbracket)$ is upward-closed in $\langle \Gamma, \preceq\rangle$, there is a finite set of configurations $\{\eta_i = \langle \mu_i, d_i\rangle \mid 1 \le i \le m\}$ such that $\mathrm{Pre}^*(\llbracket q_f \rrbracket) = \uparrow\{\eta_i \mid 1 \le i \le m\}$. We let $K = \max\{st(\eta_i)(q) \mid q \in Q,\ 1 \le i \le m\}$, 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 $\mathcal{P}$, 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 $\langle q_0^{K\cdot|Q|}, \{q_0\}, d_0\rangle$ but from which no configuration involving q_f is reachable.*

Proof. We begin with the converse implication, assuming that there is a state $\langle \mu, S, d\rangle$ 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 in $\llbracket q_f \rrbracket$ is reachable. Applying Lemma 14, there exist multisets $\delta_0 = q_0^N$ and δ, with respective supports $\{q_0\}$ and S, such that $\langle \mu \oplus \delta, d\rangle$ is reachable from $\langle q_0^{K\cdot|Q|} \oplus \delta_0, d_0\rangle$. If location q_f was reachable from $\langle \mu \oplus \delta, d\rangle$ in the distributed system, then there would exist a path from $\langle \mu, S, d\rangle$ to a state involving q_f in the symbolic graph, which contradicts our hypothesis. By Lemma 7, it follows that such a configuration $\langle \mu \oplus \delta', d\rangle$ — which cannot reach q_f — can be reached from $\langle q_0^{K\cdot|Q|} \oplus q_0^{N'}, d_0\rangle$ for any $N' \ge N$: hence it cannot be the case that q_f is reachable almost-surely for any $N' \ge N$. Therefore there cannot be a positive cut-off, which implies that there is a negative one (from Theorem 12).

Conversely, if there is a negative cut-off, then for some $N > K \cdot |Q|$, the distributed system $\mathcal{S}_\mathcal{P}^N$ 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 $\langle \mu, d \rangle \notin \mathrm{Pre}^*([\![q_f]\!])$, 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 $\langle q_0^N, d_0 \rangle$ to $\langle \mu, d \rangle$ while keeping track of the processes that end up in some q^i 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 δ with support S such that $[\![q_f]\!]$ is reachable from $\langle \kappa \oplus \delta, d \rangle$ in the distributed system. Then $\langle \kappa \oplus \delta, d \rangle \in \mathrm{Pre}^*([\![q_f]\!])$, so that for some $1 \leq i \leq m$, $(\kappa \oplus \delta)(q^i) \geq \mu_i(q^i)$, which is not possible as $\kappa(q^i) < \mu_i(q^i)$ and q^i is not in the support S of δ. 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 [19], 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 [20] on this doubly-exponential graph: this boils down to NEXPSPACE with regard to the protocol's size, hence EXPSPACE by Savitch's theorem [20].

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

4.4 PSPACE-hardness for deciding cut-offs

▶ **Theorem 17.** *Deciding the existence of a negative cut-off is* PSPACE*-hard.*

Our proof is based on the encoding of a linear-bounded Turing machine [20]: 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. 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: 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 $\mathcal{F}_{n+1}$ (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 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

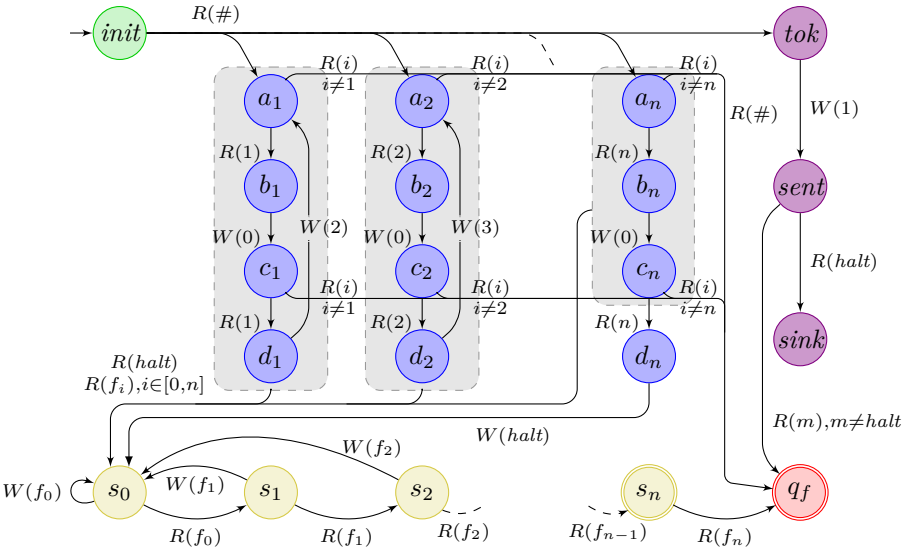

Figure 5 Simulating an exponential counter: grey boxes contain the nodes used to encode the bits of the counter; yellow nodes at the bottom correspond to the filter module from Fig. 3; purple nodes *tok*, *sent* and *sink* correspond to the second part of the protocol, and are used to produce tokens. Missing *read* edges are assumed to be self-loops.

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.

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 5 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.

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 γ_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 $st(\gamma_0)(tok) \geq 2^i$. Hence d_n, and thus also s_0, can be reached when $st(\gamma_0)(tok) \geq 2^n$. 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, $st(\gamma_0)(tok) \geq 2^n$. 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 4, location q_f cannot be reached from this configuration, which concludes our proof.

▶ **Theorem 18.** *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.*

▶ 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.

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.

▶ **Theorem 19.** *For a protocol $\mathcal{P} = \langle Q, D, q_0, T\rangle$ 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 $|\mathcal{P}|$.*

6 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 EXPSPACE whether that cut-off is positive, or negative, and we proved this decision problem to be 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 PSPACE lower bound and the 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

1 C. Aiswarya, Benedikt Bollig, and Paul Gastin. An automata-theoretic approach to the verification of distributed algorithms. In CONCUR'15, LIPIcs 42, pp. 340–353. LZI, 2015. DOI: 10.4230/LIPIcs.CONCUR.2015.340

2 Benjamin Aminof, Swen Jacobs, Ayrat Khalimov, and Sasha Rubin. Parametrized model checking of token-passing systems. In VMCAI'14, LNCS 8318, pp. 262–281. Springer, 2014. DOI: 10.1007/978-3-642-54013-4_15

3 Benjamin Aminof, Sasha Rubin, and Florian Zuleger. On the expressive power of communication primitives in parameterised systems. In LPAR'15, LNCS 9450, p. 313–328. Springer, 2015. DOI: 10.1007/978-3-662-48899-7_22

4 Simon Außerlechner, Swen Jacobs, and Ayrat Khalimov. Tight cutoffs for guarded protocols with fairness. In VMCAI'16, LNCS 9583, pp. 476–494. Springer, 2016. DOI: 10.1007/978-3-662-49122-5_23

5 Christel Baier and Joost-Pieter Katoen. *Principles of Model-Checking*. MIT Press, 2008.

6 Benedikt Bollig, Paul Gastin, and Jana Schubert. Parameterized verification of communicating automata under context bounds. In RP'14, LNCS 8762, pp. 45–57. Springer, 2014. DOI: 10.1007/978-3-319-11439-2_4

7 Patricia Bouyer, Nicolas Markey, Mickael Randour, Arnaud Sangnier, and Daniel Stan. Reachability in networks of register protocols under stochastic schedulers. Technical Report abs/1602.05928, arXiv CoRR, 2016. URL: http://arxiv.org/abs/1602.05928

8 Edmund M. Clarke, Muralidhar Talupur, Tayssir Touili, and Helmut Veith. Verification by network decomposition. In CONCUR'04, LNCS 3170, pp. 276–291. Springer, 2004. DOI: 10.1007/978-3-540-28644-8_18

9 Giorgio Delzanno, Arnaud Sangnier, Riccardo Traverso, and Gianluigi Zavattaro. On the complexity of parameterized reachability in reconfigurable broadcast networks. In FSTTCS'12, LIPIcs 18, pp. 289–300. LZI, 2012. DOI: LIPIcs.FSTTCS.2012.289

10 Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In CONCUR'10, LNCS 6269, pp. 313–327. Springer, 2010. DOI: 10.1007/978-3-642-15375-4_22

11 Antoine Durand-Gasselin, Javier Esparza, Pierre Ganty, and Rupak Majumdar. Model checking parameterized asynchronous shared-memory systems. In CAV'15, LNCS 9206, pp. 67–84. Springer, 2015. DOI: 10.1007/978-3-319-21690-4_5

12 E. Allen Emerson and Vineet Kahlon. Reducing model checking of the many to the few. In CADE'00, LNAI 1831, pp. 236–254. Springer, 2000. DOI: 10.1007/10721959_19

13 E. Allen Emerson and Kedar Namjoshi. On reasoning about rings. *Int. J. Found. Comp. Sci.*, 14(4):527–550, 2003. DOI: 10.1142/S0129054103001881

14 Javier Esparza. Keeping a crowd safe: On the complexity of parameterized verification (invited talk). In STACS'14, LIPIcs 25, pp. 1–10. LZI, 2014. DOI: 10.4230/LIPIcs.STACS.2014.1

15 Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In LICS'99, pp. 352–359. IEEE Comp. Soc. Press, 1999. DOI: 10.1109/LICS.1999.782630

16 Javier Esparza, Pierre Ganty, and Rupak Majumdar. Parameterized verification of asynchronous shared-memory systems. In CAV'13, LNCS 8044, pp. 124–140. Springer, 2013. DOI: 10.1007/978-3-642-39799-8_8

17 Steven M. German and A. Prasad Sistla. Reasoning about systems with many processes. *J. of the ACM*, 39(3):675–735, 1992.

18 Matthew Hague. Parameterised pushdown systems with non-atomic writes. In FSTTCS'11, LIPIcs 13, pp. 457–468. LZI, 2011. DOI: 10.4230/LIPIcs.FSTTCS.2011.457

19 Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theor. Comp. Sci.*, 6:223–231, 1978. DOI: 10.1016/0304-3975(78)90036-1

20 Michael Sipser. *Introduction to the theory of computation.* PWS Publishing Co., 1997.