Form Methods Syst Des (2017) 50:140–167
DOI 10.1007/s10703-016-0258-3

Model checking parameterized asynchronous shared-memory systems

Antoine Durand-Gasselin[1] · **Javier Esparza**[1] · **Pierre Ganty**[2] · **Rupak Majumdar**[3]

Published online: 24 October 2016

Abstract We characterize the complexity of liveness verification for parameterized systems consisting of a leader process and arbitrarily many anonymous and identical contributor processes. Processes communicate through a shared, bounded-value register. While each operation on the register is atomic, there is no synchronization primitive to execute a sequence of operations atomically. We analyze the case in which processes are modeled by finite-state machines or pushdown machines and the property is given by a Büchi automaton over the alphabet of read and write actions of the leader. We show that the problem is decidable, and has a surprisingly low complexity: it is NP-complete when all processes are finite-state machines, and is in NEXPTIME (and PSPACE-hard) when they are pushdown machines. This complexity is lower than for the non-parameterized case: liveness verification of finitely many finite-state machines is PSPACE-complete, and undecidable for two pushdown machines. For finite-state machines, our proofs characterize infinite behaviors using existential abstraction and semilinear constraints. For pushdown machines, we show how contributor computations of high stack height can be simulated by computations of many contributors, each with low stack height. Together, our results characterize the complexity of verification for parameterized systems under the assumptions of anonymity and asynchrony.

Keywords Model checking · Shared-memory systems · Parametrized verification

1 Introduction

We study the verification problem for *parameterized asynchronous shared-memory systems* [9,10,13]. These systems consist of a *leader* process and arbitrarily many identical *contrib-*

✉ Pierre Ganty
pierreganty@gmail.com

1 TU Munich, Munich, Germany

2 IMDEA Software Institute, Madrid, Spain

3 MPI-SWS, Kaiserslautern, Germany

utors, processes with no identity, running at arbitrary relative speeds. The shared-memory consists of a read/write register that all processes can access. Processes can perform either a read operation or a write operation on the register. The register is bounded: the set of values that can be stored is finite. Read/write operations execute atomically but sequences of operations do not: no process can conduct an atomic sequence of reads and writes while excluding all other processes. In a previous paper [9,10], we have studied the complexity of safety verification, which asks to check if a safety property holds no matter how many contributors are present. In a nutshell, we showed that the problem is coNP-complete when both leader and contributors are finite-state automata and PSPACE-complete when they are pushdown automata.

In this paper we complete the study of this model by addressing the verification of liveness properties specified as ω-regular languages (which in particular encompasses LTL model-checking). Given a property like "every request is eventually granted" and a system with a fixed number of processes, one is often able to guess an upper bound on the maximal number of steps until the request is granted, and replace the property by the safety property "every request is granted after at most K steps." In parameterized systems this bound can depend on the (unbounded) number of processes, and so reducing liveness to safety, or to finitary reasoning, is not obvious. Indeed, for many parameterized models, liveness verification is undecidable even if safety is decidable [8,14].

Our results show that there is no large complexity gap between liveness and safety verification: liveness verification (existence of an infinite computation violating a property) is NP-complete in the finite-state case, and PSPACE-hard and in NEXPTIME in the pushdown case. In contrast, remember that liveness checking is already PSPACE-complete for a *finite* number of finite-state machines, and undecidable for a *fixed* number of pushdown systems. Thus, not only is liveness verification decidable in the parameterized setting but the complexity of the parameterized problem is *lower* than in the non-parameterized case, where all processes are part of the input. We interpret this as follows: in asynchronous shared-memory systems, the existence of arbitrarily many processes leads to a "noisy" environment, in which contributors may hinder progress by replying to past messages from the leader, long after the computation has moved forward to a new phase. It is known that imperfect communication can *reduce* the power of computation and the complexity of verification problems: the best known example are lossy channel systems, for which many verification problems are decidable, while they are undecidable for perfect channels (see e.g. [1,3]). Our results reveal another instance of the same phenomenon.

Technically, our proof methods are very different from those used for safety verification. Our previous results [10] relied on a fundamental Simulation Lemma, inspired by Hague's work [13], stating that the *finite* behaviors of an arbitrary number of contributors can be simulated by a finite number of *simulators*, one for each possible value of the register. Unfortunately, the Simulation Lemma does not extend to infinite behaviors, and so we have to develop new ideas. In the case in which both leader and contributors are finite-state machines, the NP-completeness result is obtained by means of a combination of an abstraction that overapproximates the set of possible infinite behaviors, and a semilinear constraint that allows us to regain precision. The case in which both leader and contributors are pushdown machines is very involved. In a nutshell, we show that pushdown runs in which a parameter called the *effective stack height* grows too much can be "distributed" into a number of runs with smaller effective stack height. We then prove that the behaviors of a pushdown machine with a bounded effective stack height can be simulated by an exponentially larger finite-state machine.

1.1 Related work

Parameterized verification has been studied extensively, both theoretically and practically. While very simple variants of the problem are already undecidable [6], many non-trivial parameterized models retain decidability. There is no clear "rule of thumb" that allows one to predict what model checking problems are decidable, nor their complexities, other than "liveness is generally harder than safety." For example, coverability for Petri nets—in which finite-state, identityless processes communicate via rendezvous or global shared state— is EXPSPACE-complete, higher than the PSPACE-completeness of the non-parameterized version. Depending on the formulation, verification of liveness properties can be equivalent to Petri net reachability, for which we only know non-primitive recursive upper bounds, or even undecidable. Safety verification for extensions to Petri nets with reset or transfer, or broadcast protocols, where arbitrarily many finite-state processes communicate through broadcast messages, is non-primitive recursive; liveness verification is undecidable in all cases [2,8,14]. Thus, our results, which show simultaneously lower complexity than non-parameterized problems, as well as similar complexity for liveness and safety, are unexpected.

German and Sistla [11] and Aminof *et al.* [4] have studied a parameterized model with rendezvous as communication primitive, where processes are finite-state machines. Model checking the fully symmetrical case—only contributors, no leaders—runs in polynomial time (other topologies have also been considered [4]), while the asymmetric case with a leader is EXPSPACE-complete. In this paper we study the same problems, but for a shared memory communication primitive.

1.2 Population protocols

Angluin et al. [5] are another well-studied model of identityless asynchronous finite-state systems communicating via rendezvous. The semantics of population protocols is given over fair runs, in which every potential interaction that is infinitely often enabled is infinitely often taken. With this semantics, population protocols compute exactly the semilinear predicates [5]. In this paper we do not study what our model can compute (in particular, we are agnostic with respect to which fairness assumptions are reasonable), but what we can compute or decide about the model.

Recently La Torre et al. [16] defined a general framework to show decidability of the safety checking problem for parameterized asynchronous shared-memory systems. A salient feature of the proposed framework is that it generalizes beyond the models of finite state and pushdown machines previously studied [10]. However, to the best of our knowledge generalizing their framework to liveness checking is a challenging research problem.

2 Formal model: non-atomic networks

In this paper, we identify systems with languages. System actions are modeled as symbols in an alphabet, executions are modeled as infinite words, and the system itself is modeled as the language of its executions. Composition operations that combine systems into larger ones are modeled as operations on languages.

2.1 Systems as languages

An *alphabet* Σ is a finite, non-empty, set of *symbols*. A *word* over Σ is a finite sequence over Σ including the empty sequence denoted ε, and a *language* is a set of words. An *ω-word*

over Σ is an infinite sequence of symbols of Σ, and an *ω-language* is a set of ω-words. We use Σ^* (resp. Σ^ω) to denote the language of all words (resp. ω-words) over Σ. When there is no ambiguity, we use "words" to refer to words or ω-words. We do similarly for languages. Let w be a sequence over some alphabet, define $\mathrm{dom}(w) = \{1, \ldots, n\}$ if $w = a_1 a_2 \ldots a_n$ is a word; else (w is an ω-word) $\mathrm{dom}(w)$ denotes the set $\mathbb{N}\backslash\{0\}$. Elements of $\mathrm{dom}(w)$ are called *positions*. The *length* of a sequence w is defined to be $\sup \mathrm{dom}(w)$ and is denoted $|w|$. We denote by $(w)_i$ the symbol of w at position i if $i \in \mathrm{dom}(w)$, ε otherwise. Moreover, let $(w)_{i..j}$ with $i, j \in \mathbb{N}$ denote $(w)_i (w)_{i+1} \ldots (w)_j$ if $i \leq j$ and ε otherwise. Also $(w)_{i..\infty}$ denotes $(w)_i (w)_{i+1} \ldots$ For words $u, v \in (\Sigma^\omega \cup \Sigma^*)$, we say u is a *prefix* of v if either $u = v$ or $u \in \Sigma^*$ and there is a $w \in (\Sigma^\omega \cup \Sigma^*)$ such that $v = uw$.

2.1.1 Combining systems: shuffle

Intuitively, the shuffle of systems L_1 and L_2 is the system interleaving the executions of L_1 with those of L_2. Given two ω-languages $L_1 \subseteq \Sigma_1^\omega$ and $L_2 \subseteq \Sigma_2^\omega$, their *shuffle*, denoted by $L_1 \between L_2$, is the ω-language over $(\Sigma_1 \cup \Sigma_2)$ defined as follows. Given two ω-words $x \in \Sigma_1^\omega$, $y \in \Sigma_2^\omega$, we say that $z \in (\Sigma_1 \cup \Sigma_2)^\omega$ is an *interleaving* of x and y if there exist (possibly empty) words $x_1, x_2, \ldots, x_i, \ldots \in \Sigma_1^*$ and $y_1, y_2, \ldots, y_i, \ldots \in \Sigma_2^*$ such that each $x_1 x_2 \cdots x_i$ is a prefix of x, and each $y_1 y_2 \cdots y_i$ is a prefix of y, and $z = x_1 y_1 x_2 y_2 \cdots x_i y_i \cdots \in \Sigma^\omega$ is an ω-word. Then $L_1 \between L_2 = \bigcup_{x \in L_1, y \in L_2} x \between y$, where $x \between y$ denotes the set of all interleavings of x and y. For example, if $L_1 = ab^\omega$ and $L_2 = ab^\omega$, we get $L_1 \between L_2 = (a + ab^*a)b^\omega$. Shuffle is associative and commutative, and so we can write $L_1 \between \cdots \between L_n$ or $\between_{i=1}^n L_i$.

2.1.2 Combining systems: asynchronous product

The asynchronous product of $L_1 \subseteq \Sigma_1^\omega$ and $L_2 \subseteq \Sigma_2^\omega$ also interleaves the executions but, this time, the actions in the common alphabet must now be executed jointly. The ω-language of the resulting system, called the *asynchronous product* of L_1 and L_2, is denoted by $L_1 \parallel L_2$, and defined as follows. Let $Proj_\Sigma(w)$ be the word obtained by erasing from w all occurrences of symbols not in Σ. $L_1 \parallel L_2$ is the ω-language over the alphabet $\Sigma = \Sigma_1 \cup \Sigma_2$ such that $w \in L_1 \parallel L_2$ iff $Proj_{\Sigma_1}(w)$ and $Proj_{\Sigma_2}(w)$ are prefixes of words in L_1 and L_2, respectively. We abuse notation and write $w_1 \parallel L_2$ instead of $\{w_1\} \parallel L_2$ when $L_1 = \{w_1\}$. For example, let $\Sigma_1 = \{a, c\}$ and $\Sigma_2 = \{b, c\}$. For $L_1 = (ac)^\omega$ and $L_2 = (bc)^\omega$ we get $L_1 \parallel L_2 = ((ab + ba)c)^\omega$. Observe that the language $L_1 \parallel L_2$ depends on L_1, L_2 and also on Σ_1 and Σ_2. For example, if $\Sigma_1 = \{a\}$ and $\Sigma_2 = \{b\}$, then $\{a^\omega\} \parallel \{b^\omega\} = (a + b)^\omega$, but if $\Sigma_1 = \{a, b\} = \Sigma_2$, then $\{a^\omega\} \parallel \{b^\omega\} = \emptyset$. So we should more properly write $L_1 \parallel_{\Sigma_1, \Sigma_2} L_2$. However, since the alphabets Σ_1 and Σ_2 will be clear from the context, we will omit them. Like shuffle, asynchronous product is also associative and commutative, and so we write $L_1 \parallel \cdots \parallel L_n$. Notice finally that shuffle and asynchronous product coincide if $\Sigma_1 \cap \Sigma_2 = \emptyset$, but usually differ otherwise. For instance, if $L_1 = ab^\omega$ and $L_2 = ab^\omega$, we get $L_1 \parallel L_2 = ab^\omega$.

We describe systems as combinations of shuffles and asynchronous products, for instance we write $L_1 \parallel (L_2 \between L_3)$. In these expressions we assume that $\between$ binds tighter than $\parallel$, and so $L_1 \between L_2 \parallel L_3$ is the language $(L_1 \between L_2) \parallel L_3$, and not $L_1 \between (L_2 \parallel L_3)$.

2.2 Non-atomic networks

A non-atomic network is an infinite family of systems parameterized by a number k. The kth element of the family has $k + 1$ components communicating through a global store by

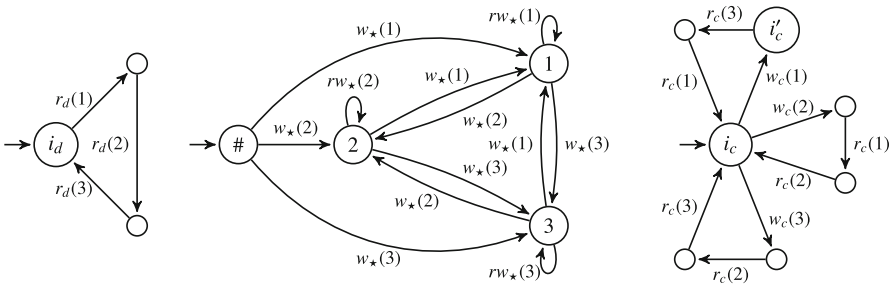

Fig. 1 Transition systems describing languages $\mathcal{D}$, $\mathcal{S}$, and $\mathcal{C}$. We write $rw_\star(g) = r_\star(g) \cup w_\star(g) = \{r_c(g), r_d(g)\} \cup \{w_c(g), w_d(g)\}$. The transition system for $\mathcal{S}$ is in state $i \in \{1, 2, 3\}$ when the current value of the store is i

means of read and write actions. The store is modeled as an atomic register whose set of possible values is finite. One of the $k + 1$ components is the leader, while the other k are the contributors. All contributors have exactly the same possible behaviors (they are copies of the same ω-language), while the leader may behave differently. The network is called non-atomic because components cannot atomically execute sequences of actions, only one single read or write.

Formally, we fix a finite set $\mathcal{G}$ of *global values*. A *read-write alphabet* is any set of the form $\mathcal{A} \times \mathcal{G}$, where $\mathcal{A}$ is a set of *read* and *write (actions)*. We denote a symbol $(a, g) \in \mathcal{A} \times \mathcal{G}$ by $a(g)$ and define $\mathcal{G}(a_1, \ldots, a_n) = \{a_i(g) | 1 \leq i \leq n,\ g \in \mathcal{G}\}$.

We fix two languages $\mathcal{D} \subseteq \Sigma_{\mathcal{D}}^\omega$ and $\mathcal{C} \subseteq \Sigma_{\mathcal{C}}^\omega$, called the *leader* and the *contributor*, with alphabets $\Sigma_{\mathcal{D}} = \mathcal{G}(r_d, w_d)$ and $\Sigma_{\mathcal{C}} = \mathcal{G}(r_c, w_c)$, respectively, where r_d, r_c are called *reads* and w_c, w_d are called *writes*. We write $w_\star$ (respectively, $r_\star$) to stand for either w_c or w_d (respectively, r_c or r_d). We further assume that $Proj_{\{r_\star(g), w_\star(g)\}}(\mathcal{D} \cup \mathcal{C}) \neq \emptyset$ holds for every $g \in \mathcal{G}$, else the value g is never used and can be removed from $\mathcal{G}$.

Additionally, we fix an ω-language $\mathcal{S}$, called the *store*, over $\Sigma_{\mathcal{D}} \cup \Sigma_{\mathcal{C}}$. It models the sequences of read and write operations supported by a single atomic register: a write $w_\star(g)$ writes g to the register, while a read $r_\star(g)$ succeeds when the register's current value is g. Initially the store is only willing to execute a write. Formally $\mathcal{S}$ is defined as

$$\Big(\sum_{g\in\mathcal{G}} \big(w_\star(g)\,(r_\star(g))^* \big)\Big)^\omega + \Big(\sum_{g\in\mathcal{G}} \big(w_\star(g)\,(r_\star(g))^* \big)^* \sum_{g\in\mathcal{G}} \big(w_\star(g)\,(r_\star(g))^\omega \big)\Big)$$

and any finite prefix thereof. Observe that $\mathcal{S}$ is completely determined by $\Sigma_{\mathcal{D}}$ and $\Sigma_{\mathcal{C}}$. Figure 1 depicts a store with $\{1, 2, 3\}$ as possible values as the language of a transition system.

Definition 1 Let $\mathcal{D} \subseteq \Sigma_{\mathcal{D}}^\omega$ and $\mathcal{C} \subseteq \Sigma_{\mathcal{C}}^\omega$ be a leader and a contributor, and let $k \geq 1$. The *k-instance of the $(\mathcal{D}, \mathcal{C})$-network* is the ω-language $\mathcal{N}^{(k)} = (\mathcal{D} \parallel \mathcal{S} \parallel \between_k \mathcal{C})$ where $\between_k \mathcal{C}$ stands for $\between_{i=1}^k \mathcal{C}$. The *$(\mathcal{D}, \mathcal{C})$-network* $\mathcal{N}$ is the ω-language $\mathcal{N} = \bigcup_{k=1}^\infty \mathcal{N}^{(k)}$. We omit the prefix $(\mathcal{D}, \mathcal{C})$ when it is clear from the context. It follows easily from the properties of shuffle and asynchronous product that $\mathcal{N} = (\mathcal{D} \parallel \mathcal{S} \parallel \between_\infty \mathcal{C})$, where $\between_\infty \mathcal{C}$ is an abbreviation of $\bigcup_{k=1}^\infty \between_k \mathcal{C}$.

Next we introduce a notion of *compatibility* between a word of the leader and a multiset of words of the contributor (a multiset because several contributors may execute the same sequence of actions). Intuitively, compatibility means that all the words can be interleaved into a legal infinite sequence of reads and writes supported by an atomic register—that is, an infinite sequence belonging to $\mathcal{S}$. Formally:

Definition 2 Let $u \in \Sigma_{\mathcal{D}}^{\omega}$, and let $M = \{v_1, \ldots, v_k\}$ be a multiset of words over $\Sigma_{\mathcal{C}}^{\omega}$ (possibly containing multiple copies of a word). We say that u is *compatible* with M iff the ω-language $(u \parallel \mathcal{S} \parallel \Diamond_{i=1}^{k} v_i)$ is non-empty. When u and M are compatible, there exists a word $s \in \mathcal{S}$ such that $(u \parallel s \parallel \Diamond_{i=1}^{k} v_i) \neq \emptyset$. We call s a *witness* of compatibility.

Example 1 Consider the network with $\mathcal{G} = \{1, 2, 3\}$ where the leader, store, and contributor languages are given by the infinite paths of the transition systems from Fig. 1. The only ω-word of $\mathcal{D}$ is $(r_d(1)r_d(2)r_d(3))^{\omega}$ and the ω-language of $\mathcal{C}$ is $(w_c(1)r_c(3)r_c(1) + w_c(2)r_c(1)r_c(2) + w_c(3)r_c(2)r_c(3))^{\omega}$. For instance, $\mathcal{D} = (r_d(1)r_d(2)r_d(3))^{\omega}$ is compatible with the multiset M of 6 ω-words obtained by taking two copies of $(w(1)r(3)r(1))^{\omega}$, $(w(2)r(1)r(2))^{\omega}$ and $(w(3)r(2)r(3))^{\omega}$. The reader may be interested in finding another multiset compatible with $\mathcal{D}$ and containing only four ω-words.

2.3 Stuttering property

Intuitively, the stuttering property states that if we take an ω-word of a network $\mathcal{N}$ and "stutter" reads and writes of the contributors, e.g., going from $w_d(1)r_c(1)w_c(2)r_d(2)\ldots$ to $w_d(1)r_c(1)r_c(1)w_c(2)w_c(2)w_c(2)r_d(2)\ldots$, the result is again an ω-word of the network.

Let $s \in \mathcal{S}$ be a witness of compatibility of $u \in \Sigma_{\mathcal{D}}^{\omega}$ and $M = \{v_1, \ldots, v_k\}$. Pick a set I of positions (viz. $I \subseteq \mathrm{dom}(s)$) such that $(s)_i \in \Sigma_{\mathcal{C}}$ for each $i \in I$, and pick a number $\ell_i \geq 0$ for every $i \in I$. Let s' be the result of simultaneously replacing each $(s)_i$ by $(s)_i^{\ell_i+1}$ in s. We have that $s' \in \mathcal{S}$. Now let $v_s = (s)_{i_1}^{\ell_{i_1}} \cdot (s)_{i_2}^{\ell_{i_2}} \cdots$, where $i_1 = \min(I)$, $i_2 = \min(I \backslash \{i_1\})$, ... It is easy to see that $(u \parallel s' \parallel v_s \Diamond \Diamond_{i=1}^{k} v_i) \neq \emptyset$.

We assume the reader to be familiar with basic multiset operations including mutliset union and difference which we denote here by $\oplus$ and $\ominus$, respectively. Equipped with multiset notations, we proceed by stating the *Copycat Lemma* which follows easily from the stuttering property [10].

Lemma 1 (Copycat Lemma) *Let $u \in \Sigma_{\mathcal{D}}^{\omega}$ and let M be a multiset of words of $\Sigma_{\mathcal{C}}^{\omega}$. If u is compatible with M, then u is also compatible with $M \oplus \{v\}$ for every $v \in M$.*

2.4 The model-checking problem for linear-time properties

We consider the model checking problem for linear-time properties, that asks, given a network $\mathcal{N}$ and an ω-regular language L, decide whether $\mathcal{N} \parallel L$ is non-empty. We assume L is given as a Büchi automaton A over $\Sigma_{\mathcal{D}}$. Intuitively, A is a tester that observes the actions of the leader; we call this the *leader model checking problem*.

We study the complexity of leader model checking for networks in which the read-write ω-languages $\mathcal{D}$ and $\mathcal{C}$ of leader and contributor are generated by an abstract machine, like a finite-state machine (FSM) or a pushdown machine (PDM). (We give formal definitions later.) More precisely, given two classes of machines $\mathtt{D}$, $\mathtt{C}$, we study the model checking problem $\mathtt{MC}(\mathtt{D}, \mathtt{C})$ defined as follows:

Given: Machines $D \in \mathtt{D}$ and $C \in \mathtt{C}$, and a Büchi automaton A,
Decide: Is $\mathcal{N}_A = (L(A) \parallel L(D) \parallel \mathcal{S} \parallel \Diamond_{\infty} L(C))$ non-empty?

In the next sections we prove that $\mathtt{MC}(\mathtt{FSM}, \mathtt{FSM})$ and $\mathtt{MC}(\mathtt{PDM}, \mathtt{FSM})$ are NP-complete, while $\mathtt{MC}(\mathtt{PDM}, \mathtt{PDM})$ is in NEXPTIME and PSPACE-hard.

Example 2 Consider the instance of the model checking problem where $\mathcal{D}$ and $\mathcal{C}$ are as in Figure 1, and A is a Büchi automaton recognizing all words over $\Sigma_{\mathcal{D}}$ containing infinitely

many occurrences of $r_d(1)$. Since $\mathcal{D}$ is compatible with a multiset of words of the contributors, $\mathcal{N}_A$ is non-empty. In particular, $\mathcal{N}_A^{(4)} \neq \emptyset$.

Since $\Sigma_A = \Sigma_{\mathcal{D}}$, we can replace A and D by a machine $A \times D$ with a Büchi acceptance condition. The construction of $A \times D$ given A and D is standard. In what follows, we assume that D comes with a Büchi acceptance condition and forget about A.

There are two natural variants of the model checking problem, where $\Sigma_A = \Sigma_{\mathcal{C}}$, i.e., the alphabet of A contains the actions of all contributors, or $\Sigma_A = \Sigma_{\mathcal{D}} \cup \Sigma_{\mathcal{C}}$. In both these variants, the automaton A can be used to simulate atomic networks. Indeed, if the language of A consists of all sequences of the form $(w_d()r_c()w_c()r_d())^\omega$, and we design the contributors so that they alternate reads and writes, then the accepting executions are those in which the contributors read a value from the store and write a new value in an atomic step. So the complexity of the model-checking problem coincides with the complexity for atomic networks (undecidable for PDMs and EXPSPACE-complete for FSMs), and we do not study it further.

3 MC(FSM, FSM) is NP-complete

We fix some notations. A finite-state machine (FSM) (Q, δ, q_0) over Σ consists of a finite set of states Q containing an initial state q_0 and a transition relation $\delta \subseteq Q \times \Sigma \times Q$. A word $v \in \Sigma^\omega$ is *accepted* by an FSM if there exists a sequence $q_1 q_2 \cdots$ of states such that $(q_i, (v)_{i+1}, q_{i+1}) \in \delta$ for all $i \geq 0$. We denote by $q_0 \xrightarrow{(v)_1} q_1 \xrightarrow{(v)_2} \cdots$ the *run* accepting v. A Büchi automaton (Q, δ, q_0, F) is an FSM (Q, δ, q_0) together with a set $F \subseteq Q$ of accepting states. An ω-word $v \in \Sigma^\omega$ is accepted by a Büchi automaton if there is a run $q_0 \xrightarrow{(v)_1} q_1 \xrightarrow{(v)_2} \cdots$ such that $q_j \in F$ for infinitely many positions j. The ω-language of a FSM or Büchi automaton A, denoted by $L(A)$, is the set of ω-words accepted by A.

In the rest of the section we show that MC(FSM, FSM) is NP-complete. Section 3.1 defines the infinite transition system associated with a (FSM, FSM)-network. Section 3.2 introduces an associated finite abstract transition system. Section 3.3 states and proves a lemma (Lemma 3) characterizing the cycles of the abstract transition system that, loosely speaking, can be concretized into infinite executions of the concrete transition system. Membership in NP is then proved using this lemma. NP-hardness follows from NP-hardness of reachability [10].

3.1 (FSM, FSM)-networks: populations and transition system

We fix a Büchi automaton $D = (Q_D, \delta_D, q_{0D}, F)$ over $\Sigma_{\mathcal{D}}$ and an FSM $C = (Q_C, \delta_C, q_{0C})$ over $\Sigma_{\mathcal{C}}$. A *configuration* is a tuple $(q_D, g, \mathbf{p})$, where $q_D \in Q_D$, $g \in \mathcal{G} \cup \{\#\}$, and $\mathbf{p} : Q_C \to \mathbb{N}$ assigns to each state of C a natural number. Intuitively, q_D is the current state of D; g is a value or the special value #, modelling that the store has not been initialized yet, and no process reads before some process writes; finally, $\mathbf{p}(q)$ is the number of contributors currently at state $q \in Q_C$. We call $\mathbf{p}$ a *population* of Q_C, and write $|\mathbf{p}| = \sum_{q \in Q_C} \mathbf{p}(q)$ for the *size* of $\mathbf{p}$. Populations are part of a larger set of mappings into $\mathbb{Z}$. We thus can define linear combinations of populations as follows: let k_1 and k_2 be two integers, for every state $q \in Q_C$, we have $(k_1\mathbf{p_1} + k_2\mathbf{p_2})(q) := k_1\mathbf{p_1}(q) + k_2\mathbf{p_2}(q)$. Also we define the ordering $\leq$ as follows: $\mathbf{p_1} \leq \mathbf{p_2}$ iff $\mathbf{p_1}(q) \leq \mathbf{p_2}(q)$ for every q. Further, given $q \in Q_C$, we denote by $\mathbf{q}$ the population $\mathbf{q}(q') = 1$ if $q = q'$ and $\mathbf{q}(q') = 0$ otherwise, i.e., the population with one contributor in state q and no contributors6 elsewhere. We stick to the convention that $\mathbf{q}, \mathbf{q'}, \mathbf{q_1}, \ldots$ denote such population with one process while $\mathbf{p}, \mathbf{p'}, \mathbf{p_1}, \ldots$ denote general population with any

number of processes. A configuration is *accepting* if the state of D is accepting, that is whenever $q_D \in F$. Given a set of populations $\mathbf{P}$, we define $(q_D, g, \mathbf{P}) := \{(q_D, g, \mathbf{p}) | \mathbf{p} \in \mathbf{P}\}$.

The labelled transition system $TS = (X, T, X_0)$ associated to $\mathcal{N}_A$ is defined as follows:

- X is the set of all configurations, and $X_0 \subseteq X$ is the set of initial configurations, given by $(q_{0D}, \#, \mathbf{P_0})$, where $\mathbf{P_0} = \{k\mathbf{q_{0C}} | k \geq 1\}$ represents an arbitrary number of contributors in their initial state q_{0C};
- $T = T_D \cup T_C$, where
  - T_D is the set of triples $\big((q_D, g, \mathbf{p}),\ t,\ (q'_D, g', \mathbf{p})\big)$ such that t is a transition of D, viz. $t \in \delta_D$, and one of the following conditions holds: (i) $t = (q_D, w_d(g'), q'_D)$; or (ii) $t = (q_D, r_d(g), q'_D)$, $g = g'$.
  - T_C is the set of triples $\big((q_D, g, \mathbf{p}),\ t,\ (q_D, g', \mathbf{p}')\big)$ such that $t \in \delta_C$, and one of the following conditions holds: (iii) $t = (q_C, w_c(g'), q'_C)$, $\mathbf{p} \geq \mathbf{q_C}$, and $\mathbf{p}' = \mathbf{p} - \mathbf{q_C} + \mathbf{q'_C}$; or (iv) $t = (q_C, r_c(g), q'_C)$, $\mathbf{p} \geq \mathbf{q_C}$, $g = g'$, and $\mathbf{p}' = \mathbf{p} - \mathbf{q_C} + \mathbf{q'_C}$. In both cases, $\mathbf{p'}$ results from moving a contributor from $\mathbf{p}$ from state q_C to q'_C.

Observe that $|\mathbf{p}| = |\mathbf{p}'|$, because the total number of contributors of a population remains constant.

Given configurations c and c', we write $c \xrightarrow{t} c'$ if $(c, t, c') \in T$.

Example 3 Consider the $(\mathcal{D}, \mathcal{C})$-network of Fig. 1. In the corresponding labelled transition system, the transition $(i_d, \#, 2\mathbf{i'_c} + \mathbf{i_c}) \xrightarrow{t} (i_d, 1, 3\mathbf{i'_c})$ captures, in a 4-instance of the network (3 contributors and the leader), that one contributor moves from i_c to i'_c and overwrites the value # of the store with 1 while all the other processes remain unchanged.

A path (or *TS*-path when ambiguous) is a (possibly infinite) sequence of configurations $c_0, c_1, \ldots$ such that $c_{i-1} \xrightarrow{t_i} c_i$ holds for all i. A path is said to be accepting if at infinitely many positions along the run, the configuration is accepting. We introduce a notation important for Lemma 3 below. We define $\Delta(t) := \mathbf{p}' - \mathbf{p}$. Observe that $\Delta(t) = \mathbf{0}$ in cases (i) and (ii) above, and $\Delta(t) = -\mathbf{q_C} + \mathbf{q_C}'$ in cases (iii) and (iv). So $\Delta(t)$ depends only on the transition t, but not on $\mathbf{p}$.

Proposition 1 *Every accepting TS-path $c_0 \xrightarrow{t_1} c_1 \xrightarrow{t_2} c_2 \ldots$ is such that the same accepting configuration is visited infinitely often.*

Proof Note that because the total number of contributors remains constant and because Q_D and $\mathcal{G}$ are finite sets, the pigeonhole principle shows that along the accepting path the same configuration repeats infinitely often. □

We conclude by observing a property of accepting *TS*-path which hints at the existence of finite witnesses of their existence. Given an infinite path $c_0 \xrightarrow{t_1} c_1 \xrightarrow{t_2} c_2 \ldots$, it follows immediately from the previous result that, given any two positions $j_1 < j_2$ where the same accepting configuration repeats, the equality

$$\sum_{i=j_1+1}^{j_2} \Delta(t_i) = \mathbf{0}$$

holds.

On the other hand, if one can establish $c_0 \xrightarrow{*} c_h$ and $c_{\ell_1} \xrightarrow{t_{\ell_1+1}} c_{\ell_1+1} \ldots \xrightarrow{t_{\ell_2}} c_{\ell_2}$ such that *(i)* $c_h = (q, g, \mathbf{p}_h)$, $c_{\ell_1} = (q, g, \mathbf{p}_{\ell_1})$ with $\mathbf{p}_h \geq \mathbf{p}_{\ell_1}$, *(ii)* $c_{\ell_1} = c_{\ell_2}$ (which implies

$\Sigma_{i=\ell_1+1}^{\ell_2}\Delta(t_i) = \mathbf{0}$), then there exists an infinite path from c_0 by first visiting c_h and then firing the sequence of transitions $t_{\ell_1+1}\ldots t_{\ell_2}$ ad infinitum. Because $\mathbf{p}_h$ is such that $\mathbf{p}_h \geq \mathbf{p}_{\ell_1}$, it enables more transitions and does not disable any. Thus the sequence $(t_{\ell_1}\ldots t_{\ell_2})$ can also be executed from c_h.

3.2 The abstract transition system

We introduce an *abstraction function* α that assigns to a set $\mathbf{P}$ of populations the set of states of Q_C populated by $\mathbf{P}$. We also introduce a *concretization function* γ that assigns to a set $Q \subseteq Q_C$ the set of all populations $\mathbf{p}$ that only populate states of Q. Formally:

$$\alpha(\mathbf{P}) = \{q \in Q_C | \mathbf{p}(q) \geq 1 \quad \text{for some } \mathbf{p} \in \mathbf{P}\}$$
$$\gamma(Q) = \{\mathbf{p} | \mathbf{p}(q) = 0 \quad \text{for every } q \in Q_C \backslash Q\}.$$

It is easy to see that α and γ satisfy $\gamma(\alpha(\mathbf{P})) \supseteq \mathbf{P}$ and $\alpha(\gamma(Q)) = Q$, and so α and γ form a Galois connection (actually, a Galois insertion). An *abstract configuration* is a tuple (q_D, g, Q), where $q_D \in Q_D$, $g \in \mathcal{G} \cup \{\#\}$, and $Q \subseteq Q_C$. We extend α and γ to (abstract) configurations in the obvious way. An abstract configuration is *accepting* when the state of D is accepting, that is whenever $q_D \in F$.

Given $TS = (X, T, X_0)$, we define its *abstraction* $\alpha TS = (\alpha X, \alpha T, \alpha X_0)$ as follows:

- $\alpha X = Q_D \times (\mathcal{G} \cup \{\#\}) \times 2^{Q_C}$ is the set of all abstract configurations.
- $\alpha X_0 = \{(q_{0D}, \#, \alpha(\mathbf{P_0}))\} = \{(q_{0D}, \#, \{q_{0C}\})\}$ is the initial configuration.
- $((q_D, g, Q),\ t,\ (q'_D, g', Q')) \in \alpha T$ iff there is $\mathbf{p} \in \gamma(Q)$ and $\mathbf{p}'$ such that $(q_D, g, \mathbf{p}) \xrightarrow{t} (q'_D, g', \mathbf{p}')$ and $Q' = \alpha(\{\mathbf{p}' | \exists\, \mathbf{p} \in \gamma(Q) \colon (q_D, g, \mathbf{p}) \xrightarrow{t} (q'_D, g', \mathbf{p}')\})$.

Example 4 Consider the transition of labelled transition system given at Example 3. In the corresponding, abstract labelled transition system $(i_d, \#, \{i'_c, i_c\}) \xrightarrow{t}_\alpha (i_d, 1, \{i'_c, i_c\})$ is the abstract transition corresponding to t. Notice the presence of i_c on both sides of the abstract transition. To understand why, it suffices to consider $(i_d, \#, 2\mathbf{i'_c} + 2\mathbf{i_c}) \xrightarrow{t} (i_d, 1, 3\mathbf{i'_c} + \mathbf{i} + \mathbf{c})$ in the labelled transition system corresponding to a 5-instance of the network.

Observe that the number of abstract configurations is bounded by $K = |Q_D| \cdot (|\mathcal{G}| + 1) \cdot 2^{|Q_C|}$. Let us point out that our abstract transition system resembles but is different from that of Pnueli et al. [15]. We write $a \xrightarrow{t}_\alpha a'$ if $(a, t, a') \in \alpha T$. The notion of accepting configuration and path are defined as expected. The abstraction satisfies the following properties:

(A) $\gamma((q, g, Q)) \subseteq \gamma((q, g, Q'))$ for every q, g and $Q \subseteq Q'$.

(B) For each (possibly infinite) TS-path $c_0 \xrightarrow{t_1} c_1 \cdots$, there exists an equally long αTS-path $a_0 \xrightarrow{t_1}_\alpha a_1 \cdots$ such that $c_i \in \gamma(a_i)$ for all $i \geq 0$. Moreover, if $c_0 \in X_0$ then there exists an αTS-path starting from $a_0 \in \alpha X_0$.

(C) If $(q_D, g, Q) \xrightarrow{t}_\alpha (q'_D, g', Q')$, then $Q \subseteq Q'$.
To prove this claim, consider two cases:

(a) $t \in \delta_D$. Then $(q_D, g, \mathbf{p}) \xrightarrow{t} (q'_D, g', \mathbf{p})$ for every population $\mathbf{p}$ (because only the leader moves). So $(q_D, g, Q) \xrightarrow{t}_\alpha (q'_D, g', Q)$.

(b) $t \in \delta_C$. Consider the population $\mathbf{p} = 2\sum_{q \in Q} \mathbf{q} \in \gamma(Q)$. Then $(q_D, g, \mathbf{p}) \xrightarrow{t} (q_D, g', \mathbf{p}')$, where $\mathbf{p}' = \mathbf{p} - \mathbf{q_C} + \mathbf{q_{C'}}$. But then $\mathbf{p}' \geq \sum_{q \in Q} \mathbf{q}$, and so $\alpha(\{\mathbf{p}'\}) \supseteq Q$, which implies $(q_D, g, Q) \xrightarrow{t}_\alpha (q_D, g', Q')$ for some $Q' \supseteq Q$.

So in every infinite αTS-path $a_0 \xrightarrow{t_1}_\alpha a_1 \xrightarrow{t_2}_\alpha a_2 \cdots$, where $a_i = (q_{Di}, g_i, Q_i)$, there is an index i at which the Q_i stabilize, that is, $Q_i = Q_{i+k}$ holds for every $k \geq 0$. However, the converse of (B) does not hold: given a path $a_0 \xrightarrow{t_1}_\alpha a_1 \xrightarrow{t_2}_\alpha a_2 \cdots$ of αTS, there may be no path $c_0 \xrightarrow{t_1} c_1 \xrightarrow{t_2} c_2 \cdots$ in TS such that $c_i \in \gamma(a_i)$ for every $i \geq 0$. Consider a contributor machine C with two states q_0, q_1 and one single transition $t = (q_0, w_c(1), q_1)$. Then αTS contains the infinite path (omitting the state of the leader, which plays no role):

$$(\#, \{q_0\}) \xrightarrow{t}_\alpha (1, \{q_0, q_1\}) \xrightarrow{t}_\alpha (1, \{q_0, q_1\}) \xrightarrow{t}_\alpha (1, \{q_0, q_1\}) \cdots$$

However, the transitions of TS are of the form $(1, k_0\mathbf{q_0}+k_1\mathbf{q_1}) \xrightarrow{t} (1, (k_0-1)\mathbf{q_0}+(k_1+1)\mathbf{q_1})$, and so TS has no infinite paths.

3.3 Realizable cycles of the abstract transition system

We show that the existence of an infinite accepting path in TS reduces to the existence of a finite certificate in αTS. It is similar to what is called a lasso path elsewhere in the literature we prefer to use the term finite certificate to avoid confusion. Lemma 2 shows how every finite αTS-path has a counterpart in TS. Lemma 3 characterizes precisely those cycles in αTS which have a counterpart in TS.

Lemma 2 *Let (q_D, g, Q) be an abstract configuration of αTS reachable from $(q_{0D}, \#, \alpha(\mathbf{P_0}))$ (that is, αX_0). For every $\mathbf{p} \in \gamma(Q)$, there exists $\hat{\mathbf{p}}$ such that $(q_D, g, \hat{\mathbf{p}})$ is reachable from $(q_{0D}, \#, \mathbf{P_0})$ and $\hat{\mathbf{p}} \geq \mathbf{p}$.*

Proof The proof is by induction on the length of the abstract path of αTS from $\alpha X_0 = \{(q_{0D}, \#, \alpha(\mathbf{P_0}))\} = \{(q_{0D}, \#, \{q_{0C}\})\}$ to the configuration (q_D, g, Q).

The base case corresponds to $(q_D, g, Q) = (q_{0D}, \#, \{q_{0C}\})$. For every $\mathbf{p} \in \gamma(\{q_{0C}\})$ we can just take $\hat{\mathbf{p}} = \mathbf{p}$.

For the inductive case (in which the αTS-path has $n > 0$ transitions), let $(q_{1D}, g_1, Q_1) \xrightarrow{t}_\alpha (q_D, g, Q)$ be the last transition of the path, and assume $t \in T_C$ (in the case $t \in T_D$ we have $Q_1 = Q$ and the result follows immediately from the induction hypothesis). Let q_C and $q_C{}'$ be the source and target state of t, respectively. It follows from the definition of $\rightarrow_\alpha$ that:

$$\begin{aligned} Q &= \alpha\big(\{\mathbf{p'} | \exists \mathbf{p} \in \gamma(Q_1) \colon (q_{1D}, g_1, \mathbf{p}) \xrightarrow{t} (q_D, g, \mathbf{p'})\}\big) \\ &= \alpha\left(\left\{\mathbf{p'} | \exists \mathbf{p} \in \gamma(Q_1) \colon \mathbf{p} \geq \mathbf{q_C} \wedge \mathbf{p'} = \mathbf{p} - \mathbf{q_C} + \mathbf{q_C}{}'\right\}\right). \end{aligned} \tag{1}$$

By (C) and (1) we have $Q = Q_1 \cup \{q_C{}'\}$. If $q'_C \in Q_1$ then we again get $Q_1 = Q$, and the result follows from the induction hypothesis. So assume $q'_C \notin Q_1$. Given an arbitrary population $\mathbf{p} \in \gamma(Q)$, let $d = \mathbf{p}(q_C{}')$ and consider the population $\mathbf{p_1} = \mathbf{p} - d\mathbf{q_C}{}' + d\mathbf{q_C}$. We have $\mathbf{p_1}(q'_C) = d - d = 0$, and so $\mathbf{p_1} \in \gamma(Q_1)$. By induction hypothesis, there exists $\hat{\mathbf{p}_1} \in \gamma(Q_1)$ such that $\hat{\mathbf{p}_1} \geq \mathbf{p_1}$ and $(q_{1D}, g_1, \hat{\mathbf{p}_1})$ is reachable from X_0 in TS. Since $\hat{\mathbf{p}_1} \geq \mathbf{p_1} \geq d\mathbf{q_C}$, the mapping $\hat{\mathbf{p}} = \hat{\mathbf{p}_1} - d\mathbf{q_C} + d\mathbf{q_C}{}'$ is non-negative, and therefore a population. Now let d contributors of $\hat{\mathbf{p}_1}$ execute $t \in \delta_C$ (which is possible in a non-atomic network, thanks to the stuttering property and, in particular, the Copycat Lemma). We then have

$$(q_{1D}, g_1, \hat{\mathbf{p}_1}) \xrightarrow{\overbrace{t \ldots t}^{d \text{ times}}} (q_D, g, \hat{\mathbf{p}_1} - d\mathbf{q_C} + d\mathbf{q_C}{}') = (q_D, g, \hat{\mathbf{p}})$$

and we are done. □

Lemma 2 does not hold for atomic networks mainly because the stuttering property does not hold anymore. Indeed, consider a contributor with transitions $q_0 \xrightarrow{w_c(1)} q_1 \xrightarrow{r_c(1):w_c(2)} q_2 \xrightarrow{r_c(2):w_c(3)} q_3$, where $r_c(i) : w_c(j)$ denotes that the read and the write happen in one single atomic step. Then we have (omitting the state of the leader, which does not play any role here):

$$(\#, \{q_0\}) \xrightarrow{w_c(1)}_\alpha (1, \{q_0, q_1\}) \xrightarrow{r_c(1):w_c(2)}_\alpha (2, \{q_0, q_1, q_2\}) \xrightarrow{r_c(2):w_c(3)}_\alpha (3, \{q_0, \ldots, q_3\}).$$

Let $\mathbf{p}$ be the population putting one contributor in each of $q_0, \ldots, q_3$. This population belongs to $\gamma(\{q_0, \ldots, q_3\})$ but no configuration $(3, \hat{\mathbf{p}})$ with $\hat{\mathbf{p}} > \mathbf{p}$ is reachable from any population that only puts contributors in q_0, no matter how many. Indeed, after the first contributor moves to q_2, no further contributor can follow, and so we cannot have contributors simultaneously in both q_2 and q_3. On the contrary, in non-atomic networks the Copycat Lemma states that what the move by one contributor can always be replicated by arbitrarily many.

We proceed to characterize the cycles of αTS that can be "concretized."

A *cycle* of αTS (αTS-cycle) is a path $a_0 \xrightarrow{t_1}_\alpha a_1 \xrightarrow{t_2}_\alpha a_2 \cdots \xrightarrow{t_n}_\alpha a_n$ such that $a_n = a_0$. A αTS-cycle $a_0 \xrightarrow{t_1}_\alpha a_1 \xrightarrow{t_2}_\alpha a_2 \cdots \xrightarrow{t_n}_\alpha a_n$ is *realizable* if there is a TS-cycle $c_0 \xrightarrow{t_1} c_1 \xrightarrow{t_2} c_2 \cdots \xrightarrow{t_n} c_n$ where $c_i \in \gamma(a_i)$ for every i and $c_0 = c_n$.

Lemma 3 *A cycle $a_0 \xrightarrow{t_1}_\alpha a_1 \xrightarrow{t_2}_\alpha a_2 \cdots \xrightarrow{t_n}_\alpha a_n$ of αTS is realizable iff $\sum_{i=1}^n \Delta(t_i) = \mathbf{0}$.*

Proof ($\Rightarrow$) This direction is trivial since $c_0 = c_n$, hence $\sum_{i=1}^n \Delta(t_i) = \mathbf{0}$
($\Leftarrow$) Let $a_i = (q_{Di}, g_i, Q_i)$ for every $0 \leq i \leq n$. By (B) we have $Q_0 \subseteq Q_1 \subseteq \cdots \subseteq Q_n = Q_0$, and so all the Q_i are equal to Q_0. Let $\mathbf{p_0} = \sum_{q \in Q_0} n\mathbf{q}$ and let $\mathbf{p_i} = \mathbf{p_0} + \sum_{k=1}^i \Delta(t_k)$ for every $1 \leq i \leq n$. Then $\mathbf{p}_i$ is a population for every $0 \leq i \leq n$, since $\sum_{k=1}^i \Delta(t_k) \leq \mathbf{p_0}$. Also $\mathbf{p_0} = \mathbf{p_n}$ since $\sum_{i=1}^n \Delta(t_i) = \mathbf{0}$. Moreover setting $c_i = (q_{Di}, g_i, \mathbf{p_i})$ we find that $c_i \in \gamma(a_i)$ for all i. Furthermore, we find that $c_0 \xrightarrow{t_1} c_1 \xrightarrow{t_2} c_2 \cdots \xrightarrow{t_n} c_n$ holds, that it is a TS-cycle since $\mathbf{p_n} = \mathbf{p_0}$ and finally $c_n = c_0$. □

Theorem 1 MC(*FSM*, *FSM*) *is NP-complete.*

Proof *NP-membership* We show membership in NP with the following high-level nondeterministic algorithm whose correctness relies on Lemmas 2 and 3:

1. Guess a sequence $Q_1, \ldots, Q_\ell$ of subsets of Q_C such that $Q_i \subsetneq Q_{i+1}$ for all $i, 0 < i < \ell$. Note that $\ell \leq |Q_C|$.
2. Compute the set $\mathcal{Q} = Q_D \times (\mathcal{G} \cup \{\#\}) \times \{\{q_{0C}\}, Q_1, \ldots, Q_\ell\}$ of abstract configurations and the set $\mathcal{T}$ of abstract transitions between configurations of $\mathcal{Q}$.
3. Guess an accepting abstract configuration $a \in \mathcal{Q}$, that is, an $a = (q_D, g, Q)$ such that q_D is accepting in D.
4. Check that a is reachable from the initial abstract configuration $\alpha X_0 = \{(q_{0D}, \#, \{q_{0C}\})\}$ by means of abstract transitions of $\mathcal{T}$.
5. Check that the transition system with $\mathcal{Q}$ and $\mathcal{T}$ as states and transitions contains a cycle $a_0 \xrightarrow{t_1}_\alpha a_1 \cdots a_{n-1} \xrightarrow{t_n}_\alpha a_n$ such that $n \geq 1$, $a_0 = a_n = a$ and $\sum_{i=1}^n \Delta(t_i) = \mathbf{0}$.

We show that the algorithm runs in polynomial time. First, because the sequence guessed is no longer than $|Q_C|$, the guess can be done in polynomial time. Next, we give a polynomial algorithm for step (5):

– Compute an FSA[1] $A_a^{\circlearrowleft}$ over the alphabet $\delta_D \cup \delta_C$ with $\mathcal{Q}$ as set of states, $\mathcal{T}$ as set of transitions, a as initial state, and $\{a\}$ as set of final states.
– Use the polynomial construction of Seidl *et al.* [17] to compute an (existential) Presburger formula Ω for the Parikh image of $L(A_a^{\circlearrowleft})$.[2] The free variables of Ω are in one-to-one correspondence with the transitions of $\delta_D \cup \delta_C$. Denote by x_t the variable corresponding to transition $t \in \delta_D \cup \delta_C$.
– Compute the formula

$$\Omega' = \Omega \ \wedge \ \bigwedge_{q_c \in Q_c} \left(\sum_{tgt(t)=q_c} x_t = \sum_{src(t)=q_c} x_t \right) \ \wedge \ \sum_{t \in \delta_D \cup \delta_C} x_t > 0$$

where *tgt* and *src* returns the target and source states of the transition passed in argument. Ω' adds to Ω the realizability condition of Lemma 3.
– Check satisfiability of Ω'. This step requires nondeterministic polynomial time because satisfiability of an existential Presburger formula is in NP [12]. □

We now prove that the previous algorithm is sound and complete.

Soundness The algorithm computes a αTS-cycle $a_0 \xrightarrow{t_1}_\alpha a_1 \cdots a_{n-1} \xrightarrow{t_n}_\alpha a_n$ such that $a = a_0 = a_n$ is accepting, $\sum_{i=1}^{n} \Delta(t_i) = \mathbf{0}$ and a_0 is reachable from αX_0. Hence, Lemma 3 shows the cycle has a realization $c_0 \xrightarrow{t_1} c_1 \cdots c_{n-1} \xrightarrow{t_n} c_n$ with $c_0 = c_n$. Also, let $c_0 = (q_0, g_0, \mathbf{p}_0)$. Since $a = a_0$ is reachable from αX_0, Lemma 2 shows there exists a configuration $\hat{c}_0 = (q_0, g_0, \hat{\mathbf{p}_0})$ reachable from $(q_{0D}, \#, \mathbf{P_0})$ such that $\hat{\mathbf{p}_0} \geq \mathbf{p_0}$. Because $\hat{\mathbf{p}_0} \geq \mathbf{p_0}$ enables more transitions and does not disable any, the sequence $(t_1 \ldots t_n)$ can also be executed from $\hat{c}_0$ and reaches again $\hat{c}_0$ at the end since $\sum_{i=1}^{n} \Delta(t_i) = \mathbf{0}$. Therefore it can be repeated infinitely often. Finally, because $c_0 \in \gamma(a_0)$ and $a_0 = a$ is accepting, we found some accepting TS-path.

Completeness By Proposition 1 there exists a TS-path $c_0 \xrightarrow{t_1} c_1 \ldots$ where the same accepting configuration is visited infinitely often. It follows that, given any two positions $j_1 < j_2$ along that path where the same accepting configuration repeats, $\sum_{i=j_1+1}^{j_2} \Delta(t_i) = \mathbf{0}$.

If follows from (B) that the TS-path $c_{j_1} \xrightarrow{t_{j_1+1}} c_{j_1+1} \cdots c_{j_2-1} \xrightarrow{t_{j_2}} c_{j_2}$ with $c_{j_1} = c_{j_2}$ has a counterpart αTS-path $a_{j_1} \xrightarrow{t_{j_1+1}}_\alpha a_{j_1+1} \cdots a_{j_2-1} \xrightarrow{t_{j_2}}_\alpha a_{j_2}$ with $c_i \in \gamma(a_i)$ for every $i = j_1, \ldots, j_2$. Note that both a_{j_1} and a_{j_2} can be chosen so as to be reachable from αX_0 following (B). We further notice that although $c_{j_1} = c_{j_2}$, the equality $a_{j_1} = a_{j_2}$ does not necesarily holds. All we can state at this point is that $a_{j_1} = (q, g, Q_1)$, $a_{j_2} = (q, g, Q_2)$ for some $q \in Q_D$ and $g \in \mathcal{G}$, and $Q_1 \subseteq Q_2$ following (C).

However, following (A) and the definition of $\rightarrow_\alpha$ we find that the sequence $t_1 \ldots t_n$ is fireable from a_{j_2}. Now firing $t_1 \ldots t_n$ from a_{j_2} results in an abstract state a' such that $a' = a_{j_2}$. This is because firing $t_1 \ldots t_n$ from a_{j_2} does not add new Q_C states to the abstract configuration (they were already added between j_1 and j_2).

Let us denote this αTS-path by $a'_0 \xrightarrow{t_1}_\alpha a'_1 \cdots a'_{n-1} \xrightarrow{t_n}_\alpha a'_0$ where $a'_i = (q_{Di}, g_i, Q_i)$ for every i and $a' = a'_0$. We find that $Q_0 = Q_1 = \cdots = Q_{n-1}$. To show that every abstract configuration of the path belongs to the set $\mathcal{Q}$, and every transition to the set $\mathcal{T}$, it suffices to guess the right increasing sequence of states at step 1 that contains Q_0.

[1] A finite-state automaton (FSA) is an FSM which decides languages of finite words. Therefore an FSA is an FSM with a set F of accepting states.

[2] In a nutshell, the Parikh image of a language L over a n-sized alphabet is a set of n-dimensional vectors of natural number. Each vector is the (Parikh) image of a word of L obtained by counting how many times each symbol occur and forgetting about the ordering, e.g. let $L = abbb$ over alphabet $\Sigma = \{a, b\}$ the vector (1,3) is the image of $abbb$ by counting 1 occurrence of a and 3 of b.

To conclude, we show that a' is reachable from $\{(q_{0D}, \#, \{q_{0C}\})\} = \alpha X_0$ which follows from $a' = a_{j_2}$ and the previously shown fact that a_{j_2} is reachable form from αX_0.

So the transition system with $\mathcal{Q}$ as states and $\mathcal{T}$ as transitions contains a configuration a reachable from the initial abstract configuration, and a cycle starting and ending at a.

NP-hardness NP-hardness follows from the NP-hardness of the safety problem [10], which asks given a finite-state machine D for the leader $\mathcal{D}$ and C for the contributor $\mathcal{C}$—both of which are languages of finite words—whether there exists a word of the $(\mathcal{D}, \mathcal{C})$-network $\mathcal{N}$ that ends with an occurrence of $w_d(\$)$. We say that $\mathcal{N}$ is *safe* iff it contains no such word. Remark that, for the safety problem, $\mathcal{C}$ is assumed to be prefix-closed, hence every state of C is accepting. Also, we assume without loss of generality that every word of $\mathcal{D}$ ends with $w_d(\$)$. The reduction goes as follows, given an instance of the safety problem turn $\mathcal{D}$ into a ω-language by appending it $r_d(\$)^\omega$. We also turn $\mathcal{C}$ into a ω-language by appending it $w_c(\#)^\omega$ where $\# \notin \mathcal{G}$ is the corrupted value nobody else can read. At the machine level this is done by adding to each accepting state of D a selfloop labeled with action $r_d(\$)$ and interpreting $\mathcal{D}$ as a Büchi automaton. On the other hand since $\mathcal{C}$ is prefix closed we have that all its states are accepting. We turn C into a FSM by dropping F—the set of accepting states—and by adding a self-loop labelled $w_c(\#)$ to each state. This concludes the hardness proof. □

4 MC(PDM, FSM) is NP-complete

A pushdown system (PDM) $P = (Q, \Gamma, \delta, q_0)$ over Σ consists of a finite set Q of states including the initial state q_0, a *stack alphabet* Γ including the bottom stack symbol $\bot$, and a set of *rules* $\delta \subseteq Q \times \Sigma \times \Gamma \times Q \times (\Gamma \backslash \{\bot\} \cup \{\mathsf{pop}\})$ which either push or pop as explained below. A *PDM-configuration* qw consists of a state $q \in Q$ and a word $w \in \Gamma^*$ (denoting the stack content). For $q, q' \in Q$, $a \in \Sigma$, $\gamma, \gamma' \in \Gamma$, $w, w' \in \Gamma^*$, we say a PDM-configuration $q'w$ (resp. $q'\gamma'\gamma w$) a-follows $q\gamma w$ if $(q, a, \gamma, q', \mathsf{pop}) \in \delta$, (resp. $(q, a, \gamma, q', \gamma') \in \delta$); we write $qw \xrightarrow{a} q'w'$ if $q'w'$ a-follows qw, and call it a *transition*. A *run* ρ from a PDM-configuration c on a word $v \in \Sigma^\omega \cup \Sigma^*$ is a sequence $\rho = c_0 \xrightarrow{(v)_1} c_1 \xrightarrow{(v)_2} c_2 \cdots$ with $|v| + 1$ PDM-configurations such that $c_i \xrightarrow{(v)_{i+1}} c_{i+1}$ for all $i \in \mathrm{dom}(v)$ and $c = c_0$. Unless defined otherwise, we assume that $c = q_0\bot$. Also we write $c \xrightarrow{*} c'$ if there is a finite run from c to c'.

A Büchi PDM is a PDM with a set $F \subseteq Q$ of accepting states. A run ρ of a Büchi PDM is accepting if ρ visits configurations with state in F infinitely often.[3] A word $w \in \Sigma^\omega$ is accepted by a Büchi PDM if there is an accepting run on w.The language $L(P)$ of a Büchi PDM P is the set of all words $v \in \Sigma^\omega$ accepted by P.

Now we introduce the notion of effective stack height of a configuration in a run of a PDM. Consider a run of a PDM that repeatedly pushes symbols on the stack. The stack height of the configurations is unbounded, but, intuitively, the PDM only uses the topmost stack symbol during the run. To account for this we define the notion of effective stack height.

Definition 3 Let $\rho = c_0 \xrightarrow{(v)_1} c_1 \xrightarrow{(v)_2} \cdots$ be an infinite run of a PDM on ω-word v, where $c_i = q_i w_i$. The *dark suffix* of c_i in ρ, denoted by $ds(w_i)$, is the longest suffix of w_i that is also a proper suffix of w_{i+k} for every $k \geq 0$. Note that because the definition allows for $k = 0$, we have that $ds(w_i)$ is at most $|w_i| - 1$ symbols long and can be ε (e.g. when $|w_i| = 1$). The *active prefix* $ap(w_i)$ of w_i is what remains: it is the prefix satisfying $w_i = ap(w_i) \cdot ds(w_i)$.

[3] For readability, we write "configuration" for "PDM-configuration."

The *effective stack height* of c_i in ρ is $|ap(w_i)|$. Removing from c_i its dark suffix results in a *pruned configuration* $q_i ap(w_i)$.

Intuitively, the effective stack height measures the actual memory required by the PDM to perform its run. For example, repeatedly pushing symbols on the stack produces a run with effective stack height 1. Given a position in the run, the elements of the stack that are never popped are those in the longest common suffix of all subsequent stacks. The first element of that suffix may be read, therefore only the longest *proper* suffix is effectively useless, so no configuration along an infinite run has effective stack height 0.

Next, we show a stack property of infinite runs of PDM.

Proposition 2 *Every infinite run of a PDM contains infinitely many positions at which the effective stack height is 1.*

Proof Let $q_0w_0 \to q_1w_1 \to q_2w_2 \to \cdots$ be any infinite run. Notice that $|w_i| \geq 1$ for every $i \geq 0$, because otherwise the run would not be infinite. Let X be the set of positions of the run defined as: $i \in X$ iff $|w_i| \leq |w_j|$ for every $j > i$. Observe that X is infinite, because the first configuration of minimal stack height, say $q_k w_k$, belongs to it, and so does the first configuration of minimal stack height of the suffix $q_{k+1}w_{k+1} \to \cdots$, etc. By construction, the configuration at every position in X has effective stack height 1. □

It follows from the previous property that every infinite run of a PDM has a "finite certificate."

Lemma 4 *Let c be a configuration of a Büchi PDM. There is an accepting run from c iff there are states $q \in Q, q_f \in F$ and stack symbol $\gamma \in \Gamma$ such that $c \xrightarrow{*} q\gamma w$ for some $w \in \Gamma^*$ and there is a non-empty run $q\gamma \xrightarrow{*} q_f u \xrightarrow{*} q\gamma w'$ for some $u, w' \in \Gamma^*$.*

Proof The "if" direction pumps the cycle (see also [7]); we prove the "only if" direction. Let ρ be an accepting run from c. Proposition 2 shows that at infinitely many positions ρ visits a configuration with effective stack height of 1. Next, by the pigeonhole principle, there is a state q and stack symbol γ such that ρ visits infinitely often a configuration of effective stack height 1, state q and topmost stack symbol γ. Moreover, because ρ is accepting, it also visits some state $q_f \in F$ infinitely often. Now fix three positions in the run where a configuration with effective stack height 1, state q and topmost stack symbol γ is visited twice and a configuration with state $q_f \in F$ is visited in between. We denote by c_1, c_2 and c_3 those three configurations for which there is a non-empty run $c_1 \xrightarrow{*} c_2 \xrightarrow{*} c_3$. Clearly, we have $c \xrightarrow{*} c_1$. Also since c_1 has an effective stack height of 1, it holds that $q\gamma \xrightarrow{*} q_f u \xrightarrow{*} q\gamma w$ for some $u, w \in \Gamma^*$ and we are done. □

We now show $\mathtt{MC}(\mathtt{PDM}, \mathtt{FSM})$ is decidable, generalizing the proof from Sect. 3. Fix a Büchi PDM $P = (Q_D, \Gamma_D, \delta_D, q_{0D}, F)$, and a FSM $C = (Q_C, \delta_C, q_{0C})$. A *configuration* is a tuple $(q_D, w, g, \mathbf{p})$, where $q_D \in Q_D$, $w \in \Gamma_D^*$ is the stack content, $g \in \mathcal{G} \cup \{\#\}$, and $\mathbf{p}$ is a population. Intuitively, $q_D w$ is the PDM-configuration of the leader. We extend the definitions from Sect. 3 like accepting configuration in the obvious way.

We define a labeled transition system $TS = (X, T, X_0)$, where X is the set of configurations including the set $X_0 = (q_{0D}, \bot, \#, \mathbf{P_0})$ of initial configurations, and the transition relation $T = T_D \cup T_C$, where T_C is as before and T_D is the set of triples $\big((q_D, w, g, \mathbf{p}), t, (q'_D, w', g', \mathbf{p})\big)$ such that t is a transition (not a rule) of D, and one of the following conditions holds: (i) $t = (q_D w \xrightarrow{w_d(g')} q'_D w')$; or (ii) $t = (q_D w \xrightarrow{r_d(g)} q'_D w')$

and $g = g'$. We define the abstraction αTS of TS as the obvious generalization of the abstraction in Sect. 3. An accepting path of the (abstract) transition system is an infinite path with infinitely many accepting (abstract) configurations. As for MC(FSM, FSM), not every accepting path of the abstract admits a concretization, but we find a realizability condition in terms of linear constraints. Here we use again the polynomial construction of Seidl *et al.* [17] mentioned in the proof of Theorem 1, this time to compute an (existential) Presburger formula for the Parikh image of a pushdown automaton.

Theorem 2 MC(*PDM*, *FSM*) *is NP-complete.*

Proof Hardness follows from the NP-hardness for MC(FSM, FSM). The non-deterministic polynomial time algorithm is essentially the same as that of Theorem 1, except that we have pushdown systems instead of finite-state systems. As before, we guess a sequence $Q_1, \ldots, Q_\ell$ of subsets of Q_C such that $Q_i \subsetneq Q_{i+1}$ for all i, $0 < i < \ell \leq |Q_C|$. We construct a Büchi PDM whose states $\mathcal{Q}$ are abstract configurations $Q_D \times (\mathcal{G} \cup \{\#\}) \times \{\{q_{0C}\}, Q_1, \ldots, Q_\ell\}$, whose stack alphabet is Γ_D, whose initial state is $q_0 = (q_{0D}, \bot, \#, \{q_{0C}\})$, whose accepting states are accepting abstract configurations (i.e. where A is accepting), and whose transitions are defined to mimic αTS.

Next, we guess an abstract configuration q and a stack symbol γ and look for a "finite certificate" as given in Lemma 4. To this end, we check if there is a non-empty run on some word u given by $q_0 \bot \xrightarrow{*} q\gamma w$ for some $w \in \Gamma_D^*$. This check is equivalent to pushdown reachability and can be performed in polynomial time [7]. We construct a PDA[4] $P_{q\gamma}^{\circlearrowleft}$ over finite words that accepts a word $u \in \Sigma^*$ if there is a non-empty run on u from $q\gamma$ to a configuration $q\gamma w'$ for some $w' \in \Gamma_D^*$ that passes through an accepting abstract configuration. The PDA $P_{q\gamma}^{\circlearrowleft}$ can be computed in polynomial time.

Finally, we check if there is a word accepted by the pushdown automaton whose "weight" is $\mathbf{0}$. For this check, as before, we compute an (existential) Presburger formula Ω for the Parikh image of $L(P_{q\gamma}^{\circlearrowleft})$. The free variables of Ω are in one-to-one correspondence with the transitions of the automaton. We thus adopt the convention that x_t denotes the variable corresponding to transition $t \in \delta_D \cup \delta_C$. We compute Ω' by adding $|Q_C|$ variables and $|Q_C|$ constraints, one per state in $q_c \in Q_C$: $\sum_{\mathrm{tgt}(t)=q_c} x_t = \sum_{\mathrm{src}(t)=q_c} x_t$ where tgt and src returns the target and source states of the transition passed in argument. Add also the constraints $\sum_{t \in \delta_D \cup \delta_C} x_t > 0$ to prevent $\mathbf{0}$ to be returned as a trivial solution. Finally, we check satisfiability of Ω' and accept if Ω' is satisfiable. This step is in NP because satisfiability of an existential Presburger formula is in NP [12].

To see that the algorithm is sound, notice that the algorithm accepts if there is a "finite certificate" such that the cyclic part has $\mathbf{0}$ weight. What Lemma 4 shows is that for an initial population that is large enough (essentially, cubic in the size of the PDM), we can execute the operations on the path to the cycle and then execute the cycle to come back to the same configuration as the starting point of the finite certificate. This witness can be pumped infinitely often to produce an accepting run of the Büchi PDM.

For completeness, we use the "only if" direction of Lemma 4 to deduce that from an accepting run of the Büchi PDM, we can find a finite certificate. By a similar pigeonhole argument as that of Lemma 3, we conclude that we can find a cyclic path whose weight is $\mathbf{0}$. □

[4] A pushdown automaton (PDA) is a PDM which decides languages of finite words. We define a PDA as a PDM with a set F of accepting states.

5 MC(PDM, PDM) is in NEXPTIME

5.1 Outline

For this case, we can use the same general proof idea as the two previous cases, in fact we will just show how to reduce MC(PDM, PDM) to MC(PDM, FSM). Notice that in the previous cases, to build a witness from an infinite run, we heavily relied on the fact that (1) there is a fixed finite number of contributors and (2) contributors have a finite number of configurations: it sufficed then to apply a pigeonhole principle on a sufficiently long prefix of the infinite run.

In this case, however, as each contributor is equipped with an unbounded stack, we do not have (2).

To restore (2), we show, intuitively, how the work of one contributor can be done by a set of contributor each using less resources. The meaning of "less resources" will be defined precisely later but, intuitively, the substitute contributors have an upper bound on the their effective stack height, hence a bounded amount of memory suffices to describe them.

The substitution of one run by many is the key insight to restore (2). In what follows, this roughly corresponds the boundedness lemma (Lemma 6) that builds upon the distribution lemma (Lemma 5). However, if applied naively, the boundedness lemma will introduce infinitely many contributors along infinite runs, therefore breaking (1). The challenge is thus to show that there is substitution that works for infinite runs without introducing infinitely many contributors. This challenge is tackled in the proof of the reduction theorem, Theorem 4 below. A key ingredient to show finitely many substitute contributors suffices is the definition of synchronized distributions (Definition 8) and the fact that the boundedness lemma (Lemma 6) is synchronization aware. Intuitively, synchronization allows to dispatch work among substitute contributors so that finitely many of them suffice.

With the reduction theorem, the reduction to MC(PDM, FSM) is as follows: given a contributor PDM C, we build an FSM C_k that simulates all the runs of C of effective stack height k. We show that, for $k \in O(n^3)$, where n is the size of C, the language $(L(D) \parallel \mathcal{S} \parallel \lozenge_\infty L(C))$ is empty iff $(L(D) \parallel \mathcal{S} \parallel \lozenge_\infty L(C_k))$ is empty.

5.2 An FSM for runs of bounded effective stack height

Definition 4 We say that a run ρ is *effectively k-bounded* (or simply k-bounded for the sake of readability) if every configuration of ρ has an effective stack height of at most k. Further, we say that ρ is *bounded* if it is k-bounded for some $k \in \mathbb{N}$. Finally, an ω-word of the PDM is *k-bounded*, respectively bounded, if it is the word generated by some k-bounded, respectively bounded, run. It suffices that there exists a k-bounded run; all runs for the same word need not be bounded.

In a k-bounded run, whenever the stack height exceeds k, the $k+1$-th stack symbol will never become the top symbol again, and so it becomes useless. So, we can construct a finite-state machine for the words accepted by k-bounded runs.

Definition 5 Given a PDM $P = (Q, \Gamma, \delta, q_0)$, the FSM $P_k = (Q_k, \delta_k, q_{0k})$, called the *k-restriction of* P, is defined as follows: (a) $Q_k = Q \times \bigcup_{i=1}^{k} \Gamma^i$ (a state of P_k consists of a state of P and a stack content no longer than k); (b) $q_{0k} = (q_0, \bot)$; (c) δ_k contains a transition $(q, (w)_{1..k}) \xrightarrow{a} (q', (w')_{1..k})$ iff $qw \xrightarrow{a} q'w'$ is a transition (not a rule) of P. As expected the alphabets of P and P_k coincide.

Theorem 3 *Given a PDM P and a word $v \in \Sigma^\omega \cup \Sigma^*$, v admits a k-bounded run in P iff v admits a run in P_k.*

Proof We first prove that if v admits an effectively k-bounded run ρ in P then w also admits a run in P_k. Let $\rho = q_0 w_0 \xrightarrow{(v)_1} q_1 w_1 \xrightarrow{(v)_2} \cdots$, and let $ap(i)$, resp. $ds(i)$, denote the active prefix, resp. dark suffix, of w_i. Recall that a state of P_k is a pair (q, s), where q is a state of P and s is a non-empty stack content of length at most k.

For every $i \geq 0$, we inductively define $(q_i, ap(i)u_i)$ where u_i is a possibly empty prefix of $ds(i)$ and we show that $(q_i, ap(i)u_i) \xrightarrow{(v)_{i+1}} (q_{i+1}, ap(i+1)u_{i+1})$ is a transition of P_k.

We define $u_0 = \varepsilon$. Observe that $ap(0) = \bot$ and $ds(0) = \varepsilon$, therefore we find that the initial state q_{0k} of P_k is such that $q_{0k} = (q_0, \bot) = (q_0, ap(0)u_0)$ and we are done. For the definition of u_{i+1}, assuming that u_i is already defined, we consider three cases:

- The transition $q_i w_i \xrightarrow{(v)_{i+1}} q_{i+1} w_{i+1}$ pops a symbol γ.
  Then $q_i\, \gamma v \xrightarrow{(v)_{i+1}} q_{i+1} v$ is a transition of P for every v, and so, in particular, $q_i\, \gamma ap(i+1)u_i \xrightarrow{(v)_{i+1}} q_{i+1}\, ap(i+1)u_i$ is a transition of P. Moreover, by the definition of an active prefix, we have $ap(i) = \gamma ap(i+1)$ and thus $ds(i) = ds(i+1)$ therefore u_i is also a prefix of $ds(i+1)$. By induction hypothesis, $|ap(i)u_i| \leq k$, which implies $|ap(i+1)u_i| < k$. Setting u_{i+1} to be u_i we thus obtain that $|ap(i+1)u_{i+1}| \leq k$ and finally that $(q_i, ap(i)u_i) \xrightarrow{(v)_{i+1}} (q_{i+1}, ap(i+1)u_{i+1})$ is a transition of P_k.
- The transition $q_i w_i \xrightarrow{(v)_{i+1}} q_{i+1} w_{i+1}$ pushes a symbol γ, and $|ap(i)u_i| < k$.
  Then $q_i ap(i)u_i \xrightarrow{(v)_{i+1}} q_{i+1}\gamma ap(i)u_i$ is a transition of P. Since $|ap(i)u_i| < k$, we have $|\gamma ap(i)u_i| \leq k$, hence $(q_i, ap(i)u_i) \xrightarrow{(v)_{i+1}} (q_{i+1}, \gamma ap(i)u_i)$ is also a transition of P_k. If γ is popped later on, then $ap(i+1) = \gamma\, ap(i)$; so $q_i\, ap(i)u_i \xrightarrow{(v)_{i+1}} q_{i+1}\, ap(i+1)u_i$ is a transition of P, and we set u_{i+1} to u_i. If γ is never popped, then $ap(i+1) = \gamma$, and we let u_{i+1} to be $ap(i)u_i$. In both cases, we find that $|ap(i+1)u_{i+1}| \leq k$ and hence that $(q_i, ap(i)u_i) \xrightarrow{(v)_{i+1}} (q_{i+1}, ap(i+1)u_{i+1})$ is a transition of P_k.
- The transition $q_i w_i \xrightarrow{(v)_{i+1}} q_{i+1} w_{i+1}$ pushes a symbol γ, and $|ap(i)u_i| = k$.
  Then $q_i ap(i)u_i \xrightarrow{(v)_{i+1}} q_{i+1}\gamma ap(i)u_i$ is a transition of P. Observe that since $|ap(i)u_i| = k$ we have $|\gamma ap(i)u_i| = k+1$. First we show that $|u_i| > 0$ by contradiction. Assume $|u_i| = 0$, we find that $|ap(i)| = k$. On the other hand, since $ds(i)$ is the longest proper suffix of all the $(w_j)_{j \geq i}$, and because $w_{i+1} = \gamma w_i$, we find that $ds(i)$ is also the longest proper suffix of all the $(w_j)_{j \geq i+1}$, hence $\gamma ap(i) = ap(i+1)$, and finally $|ap(i+1)| = k+1$ contradicting the hypothesis that the run is effectively k-bounded.
  Since $|\gamma ap(i)u_i| = k+1$, $(q_i, ap(i)u_i) \xrightarrow{(v)_{i+1}} (q_{i+1}, (\gamma ap(i)u_i)_{1..k})$ is a transition of P_k. If γ is popped later on, then $ap(i+1) = \gamma\, ap(i)$ and $u_{i+1} = (u_i)_{1..|u_i|-1}$. If γ is never popped, then $ap(i+1) = \gamma$, and $u_{i+1} = (ap(i)u_i)_{1..k-1}$. In both cases we conclude that $|ap(i+1)u_{i+1}| \leq k$ (in the former, $|u_i| > 0$ is crucial), hence that $(q_i, ap(i)u_i) \xrightarrow{(v)_{i+1}} (q_{i+1}, ap(i+1)u_{i+1})$ is a transition of P_k.

Now we show that if v admits a run in P_k, then it admits an effectively k-bounded run ρ' in P. Let $\rho = (q_0, w_0) \xrightarrow{(v)_1} (q_1, w_1) \xrightarrow{(v)_2} \cdots$ be a run of P_k for v. The definition of P_k shows that $|w_i| \leq k$ holds for every $i \geq 0$. We inductively construct $w'_0, w'_1, \ldots$ such that $\rho' = q_0 w_0 w'_0 \xrightarrow{(v)_1} q_1 w_1 w'_1 \xrightarrow{(v)_2} \cdots$ is a run of P satisfying the following invariant:

$$|w_{i+1} w'_{i+1}| - |w_i w'_i| \geq |w_{i+1}| - |w_i|, \quad \text{for all } i \geq 0\ . \tag{2}$$

We start by defining $w'_0 = \varepsilon$, which trivially satisfies (2). Assume $q_0 w_0 w'_0 \xrightarrow{(v)_1} \cdots \xrightarrow{(v)_{i+1}} q_i w_i w'_i$ is a run of P satisfying (2), and consider the transition $(q_i, w_i) \xrightarrow{(v)_{i+1}} (q_{i+1}, w_{i+1})$ of P_k. By the definition of the transitions of P_k, there are two possible cases:

- $q_i w_i \xrightarrow{(v)_{i+1}} q_{i+1} w_{i+1}$ is a transition of P.
  Then $q_i w_i w'_i \xrightarrow{(v)_{i+1}} q_{i+1} w_{i+1} w'_i$ is also a transition of P, and we can take w'_{i+1} to be w'_i, and (2) is satisfied as $|w_{i+1} w'_{i+1}| - |w_i w'_i| = |w_{i+1}| - |w_i|$
- $q_i w_i \xrightarrow{(v)_{i+1}} q_{i+1} w_{i+1} \gamma$ is a transition of P.
  Then $|w_i| = |w_{i+1}| = k$, and $q_i w_i w'_i \xrightarrow{(v)_{i+1}} q_{i+1} w_{i+1} \gamma w'_i$ is a transition of P. So setting w'_{i+1} to $\gamma w'_i$ satisfies (2) as $|w_{i+1} w'_{i+1}| - |w_i w'_i| = |w_{i+1}| - |w_i| + 1$

The induction is concluded, now we explain the meaning of equation (2). First remark that performing a telescope sum, we obtain that for any $i, j > 0$, $|w_{i+j} w'_{i+j}| - |w_i w'_i| \geq |w_{i+j}| - |w_i|$. Since $|w_{i+j}| \leq k$ and $|w_i| \geq 1$, we obtain $|w_{i+j} w'_{i+j}| - |w_i w'_i| \geq 1 - k$. Informally it means that the number of symbols in the stack at any position after i can't be much smaller (much meaning k) than at position i. Thus, at every position i, we never eventually pop the k top symbols of the stack at that position, as this would yield a configuration after i whose stack would be too small and contradict the inequality. Therefore the run ρ' is effectively k-bounded. □

5.3 The Reduction Theorem

We fix a Büchi PDM D and a PDM C. In order to reduce MC(PDM, PDM) to MC(PDM, FSM) it suffices to prove the following Reduction Theorem:

Theorem 4 (Reduction Theorem) *Let $N = 2|Q_C|^2|\Gamma_C| + 1$, where Q_C and Γ_C are the states and stack alphabet of C, respectively. Let C_N be the N-restriction of C. We have:*

$$\big(L(D) \parallel \mathcal{S} \parallel \langle\!\rangle_\infty L(C)\big) \neq \emptyset \quad \textit{iff} \quad \big(L(D) \parallel \mathcal{S} \parallel \langle\!\rangle_\infty L(C_N)\big) \neq \emptyset \ . \tag{†}$$

There are PDMs D, C for which (†) holds only for $N \in \Omega(|Q_C|^2|\Gamma_C|)$.

Theorems 4 and 2 provide an upper bound for MC(PDM, PDM). PSPACE-hardness of the reachability problem [10] gives a lower bound.

Theorem 5 MC(*PDM*, *PDM*) *is in NEXPTIME and PSPACE-hard. If the contributor is a one counter machine (with zero-test), it is NP-complete.*

Proof The NP algorithm from the previous section, together with Theorem 4 and the exponential upper bound on the size of C_N, show membership in NEXPTIME. Note that if C has only one stack symbol, the size of C_N is polynomial in the size of C: only the stack height needs to be stored. In the general case, C_N may store up to N stack symbols: the size of C_N is then exponential. □

The proof of Theorem 4 is very involved. Given a run of D compatible with a finite multiset of runs of C, we construct another run of D compatible with a finite multiset of N-bounded runs of C_N. (Here we extend compatibility to runs: runs are compatible if the words they accept are compatible).

The proof starts with the Distributing lemma, which, loosely speaking, shows how to replace a run of C by a multiset of "smaller" runs of C without the leader "noticing". After

this preliminary result, the first key proof element is the Boundedness Lemma. Let σ be an infinite run of D compatible with a finite multiset R of runs of C. The Boundedness Lemma states that, for any number Z, the first Z steps of σ are compatible with a (possibly larger) multiset R_Z of runs of C_N.[5] Since the size of R_Z may grow with Z, this lemma does not yet prove Theorem 4: it only shows that σ is compatible with an *infinite* multiset of runs of C_N. This obstacle is overcome in the final step of the proof. We show that, for a sufficiently large Z, there are indices $i < j$ such that, not σ itself, but the run $(\sigma)_{1..i}\big((\sigma)_{i+1..j}\big)^{\omega}$ for adequate i and j is compatible with a *finite* multiset of runs of C_N. Loosely speaking, this requires to prove not only that the leader can repeat $(\sigma)_{i+1..j}$ infinitely often, but also that the runs executed by the instances of C_N while the leader executes $(\sigma)_{i+1..j}$ can be repeated infinitely often.

5.3.1 The Distributing Lemma

Let $\rho = c_0 \xrightarrow{a_1} c_1 \xrightarrow{a_2} c_2 \xrightarrow{a_3} \cdots$ be a (finite or infinite) run of C. Let r_i be the PDM-rule of C generating the transition $c_{i-1} \xrightarrow{a_i} c_i$. Then ρ is completely determined by c_0 and the sequence $r_1 r_2 r_3 \ldots$Since c_0 is also fixed (for fixed C), in the rest of the paper we also sometimes write $\rho = r_1 r_2 r_3 \ldots$ This notation allows us to speak of $\mathrm{dom}(\rho)$, $(\rho)_k$, $(\rho)_{i..j}$ and $(\rho)_{i..\infty}$.

Definition 6 (Distribution) Let $\rho = c_0 \xrightarrow{a_1} c_1 \xrightarrow{a_2} c_2 \xrightarrow{a_3} \cdots$ be a (finite or infinite) run of C. We say that ρ *distributes to a multiset R of runs of C* if there exists an *embedding function* ψ that assigns to each run $\rho' \in R$ and to each position $i \in \mathrm{dom}(\rho')$ a position $\psi(\rho', i) \in \mathrm{dom}(\rho)$, and satisfies the following properties:

- $(\rho')_i = (\rho)_{\psi(\rho',i)}$. (A rule occurrence in ρ' is matched to another occurrence of the same rule in ρ).
- ψ is surjective. (For every position $k \in \mathrm{dom}(\rho)$ there is at least one $\rho' \in R$ and a position $i \in \mathrm{dom}(\rho')$ such that $\psi(\rho', i) = k$, or, informally, R "covers" ρ).
- If $i < j$, then $\psi(\rho', i) < \psi(\rho', j)$. (So $\psi(\rho', 1)\psi(\rho', 2)\cdots$ is a scattered subword of ρ).

When all the above properties are satisfied, we call the pair R, ψ a *distribution* of ρ or simply distribution when ρ is clear from the context.

Example 5 Let ρ be a run of a PDM P. Below are two distributions R, ψ and S, ψ' of $\rho = r_a r_b r_b r_c r_c r_c$. On the left we have the runs of P given by $R = \{\rho'_1, \rho'_2, \rho'_3\}$, and its embedding function ψ; on the right the runs $S = \{\sigma'_1, \sigma'_2, \sigma'_3\}$, and its function ψ'.

ψ	1	2	3
ρ'_1	1	6	
ρ'_2	1	2	5
ρ'_3	1	3	4

	1	2	3	4	5	6
$\rho =$	r_a	r_b	r_b	r_c	r_c	r_c
$\rho'_1 =$	r_a					r_c
$\rho'_2 =$	r_a	r_b			r_c	
$\rho'_3 =$	r_a		r_b	r_c		

ψ'	1	2	3	4
σ'_1	1	4		
σ'_2	1	2	4	5
σ'_3	1	3	5	6

	1	2	3	4	5	6
$\rho =$	r_a	r_b	r_b	r_c	r_c	r_c
$\sigma'_1 =$	r_a			r_c		
$\sigma'_2 =$	r_a	r_b			r_c	r_c
$\sigma'_3 =$	r_a		r_b		r_c	r_c

Lemma 5 (Distributing lemma) *Let $u \in L(D)$, and let M be a multiset of words of $L(C)$ compatible with u. Let $v \in M$ and let ρ an accepting run of v in C that distributes to a multiset R of runs of C, and let M_R the corresponding multiset of words. Then $M \ominus \{v\} \oplus M_R$ is also compatible with u.*

[5] See Theorem 4 for a definition of N.

Proof Since u is compatible with M, there exists a witness $s \in L(\mathcal{S})$ such that $s \in (u \parallel \lozenge_{w \in M} w)$. Since $\Sigma_{\mathcal{C}} \cap \Sigma_{\mathcal{D}} = \emptyset$, we have $(u \parallel \lozenge_{w \in M} w) = (u \between \lozenge_{w \in M} w)$, and so $s \in (u \between \lozenge_{w \in M} w)$. Therefore there exists an interleaving function, i.e. a bijection $\mathcal{I} \colon \coprod_{w \in \{u\} \oplus M} \mathrm{dom}(w) \to \mathrm{dom}(s)$, that assigns to each position in each word w in $\{u\} \oplus M$ a corresponding position in s with the same action. Further, the interleaving function satisfies $i < j \in \mathrm{dom}(w)$ iff $\mathcal{I}(w, i) < \mathcal{I}(w, j)$ for each $w \in \{u\} \oplus M$.

For example, if $u = w_d(1)$ and $M = \{w_1, w_2\}$, where $w_1 = r_c(1)w_c(2)r_c(1)$ and $w_2 = r_c(2)w_c(1)$, then we can take $s = w_d(1)r_c(1)w_c(2)r_c(2)w_c(1)r_c(1)$, with $\mathcal{I}(w_1, 1) = 2$, $\mathcal{I}(w_1, 2) = 3, \mathcal{I}(w_1, 3) = 6, \mathcal{I}(w_2, 1) = 4, \mathcal{I}(w_2, 2) = 5$ and $\mathcal{I}(u, 1) = 1$.

We have to show that, given $v \in M$, a run ρ of C accepting a word v, and a distribution R of ρ accepting a multiset M_R of words, then $M \ominus \{v\} \oplus M_R$ is compatible with u.

Let R, ψ be the distribution R of ρ. We construct a word s' witnessing that u and $M \ominus \{v\} \oplus M_R$ are compatible. The word s' is a stuttering of s, that is, it is obtained from s by repeating some letters of s; since, by definition of the store, $\mathcal{S}$ is closed under stuttering, we have $s' \in \mathcal{S}$. Let $s = a_1 a_2 \ldots$, and let i be such that $\mathcal{I}(v, j) = i$ for some $j \in \mathrm{dom}(v)$. Further, let k be the number of runs in R such that some position in them is mapped to position j by the embedding function ψ (intuitively, k is the number of runs in R executing the action at position j. Then we replace a_i by a_i^k (that is, by the word $a_i \ldots a_i$ of length k).

For example, if we distribute w_1 above to $\{r_c(1)w_c(2), w_c(2)r_c(1), w_c(2)\}$, then we get $s' = w_d(1)r_c(1)(w_c(2)w_c(2)w_c(2))r_c(2)w_c(1)r_c(1)$.

We clearly have a one-to-one correspondence between positions in s' and positions in $u \oplus (M \ominus \{v\}) \oplus M_R$. □

5.3.2 The boundedness lemma

We are interested in distributing a multiset of runs of C into another multiset with, loosely speaking, "better" effective stack height.

Fix a run ρ of C and a distribution R of ρ with embedding function ψ. In Example 5, $(\rho)_{1..4}$ is distributed into $(\rho'_1)_{1..1}$, $(\rho'_2)_{1..2}$ and $(\rho'_3)_{1..3}$. Assume ρ is executed by one contributor. We can replace it by 3 contributors executing $\rho'_1, \rho'_2, \rho'_3$, without the rest of the network noticing any difference. Indeed, the three processes can execute r_a immediately after each other, which for the rest of the network is equivalent to the old contributor executing one r_a. Then we replace the execution of $(\rho)_{2..4}$ by $(\rho'_2)_2(\rho'_3)_{2..3}$.

We introduce some definitions allowing us to formally describe such facts. Given $k \in \mathrm{dom}(\rho)$, we denote by $c(\rho, k)$ the configuration reached by ρ after k steps. We naturally extend this notation to define $c(\rho, 0)$ as the initial configuration. We denote by $last_\psi(\rho', i)$ the greatest position $p \in \mathrm{dom}(\rho')$ such that $\psi(\rho', p) \leq i$ (similarly if none exists, we fix $last_\psi(\rho', i) = 0$). Further, we denote by $c_\psi(\rho', k)$ the configuration reached by ρ' after k steps of ρ, that is, the configuration reached by ρ' after the execution of $last_\psi(\rho', k)$ transitions; formally, $c_\psi(\rho', k) = c(\rho', last_\psi(\rho', k))$.

Example 6 Let ρ, R, and ψ as in Example 5. Assuming that the PDM P has one single state p, stack symbols $\{\bot, \alpha\}$ such that the three rules r_a, r_b and r_c are given by $r_a \colon p\bot \to p\alpha\bot$, $r_b \colon p\alpha \to p\alpha\alpha$, and $r_c \colon p\alpha \to p$, then we have $c(\rho, 5) = p\alpha\bot$. Further, $last_\psi(\rho'_1, 5) = 1$, $last_\psi(\rho'_2, 5) = 3$, and $last_\psi(\rho'_3, 5) = 3$. Finally, $c_\psi(\rho'_1, 5) = p\alpha\bot$, $c_\psi(\rho'_2, 5) = p\alpha\bot$, and $c_\psi(\rho'_3, 5) = p\alpha\bot$.

Definition 7 ((Z, K)-bounded distribution) Given $Z \in \mathrm{dom}(\rho)$ and $K \in \mathbb{N}$, we say that a distribution R of ρ is (Z, K)-*bounded* if for every $\rho' \in R$ and for every $i \leq Z$, the effective stack height of $c_\psi(\rho', i)$ is bounded by K.

	0	1	2	3	4	5	6
ρ:	$\perp$	$\alpha\perp$	$\alpha\alpha\perp$	$\alpha\alpha\alpha\perp$	$\alpha\alpha\perp$	$\alpha\perp$	$\perp$
e.s.h.	1	2	3	4	3	2	1
ρ'_1:	$\perp$	$\alpha\perp$					$\perp$
e.s.h.	1	2					1
ρ'_2:	$\perp$	$\alpha\perp$	$\alpha\alpha\perp$			$\alpha\perp$	
e.s.h.	1	1	2			1	
ρ'_3:	$\perp$	$\alpha\perp$		$\alpha\alpha\perp$	$\alpha\perp$		
e.s.h.	1	1		2	1		

	0	1	2	3	4	5	6
ρ:	$\perp$	$\alpha\perp$	$\alpha\alpha\perp$	$\alpha\alpha\alpha\perp$	$\alpha\alpha\perp$	$\alpha\perp$	$\perp$
e.s.h.	1	2	3	4	3	2	1
σ'_1:	$\perp$	$\alpha\perp$			$\perp$		
e.s.h.	1	2			1		
σ'_2:	$\perp$	$\alpha\perp$	$\alpha\alpha\perp$		$\alpha\perp$	$\perp$	
e.s.h.	1	2	3		2	1	
σ'_3:	$\perp$	$\alpha\perp$		$\alpha\alpha\perp$		$\alpha\perp$	$\perp$
e.s.h.	1	2		3		2	1

Fig. 2 Configurations and effective stack heights (e.s.h.) of the distributions R and S of Example 5. S is on the right

We want to show a boundedness lemma that allows us to distribute a single run into bounded runs of many contributors. An initial attempt at the lemma would show that there is a constant N, depending only on C, such that for every run ρ of C and for every $Z \in \mathrm{dom}(\rho)$ there is a (Z, N)-bounded R_Z of ρ.

However, this statement alone is not enough. To see the problem, imagine we have a way to build such a distribution R_Z but with the property that the greater the value of Z the more contributors must be involved. In turn, this would imply that infinitely many contributors would have to be involved to simulate an infinite run. This is a problem since we want finitely many contributors. Thus, our boundedness lemma below requires a further property, called synchronization, on R_Z. Synchronization will allow us escape the use of unboundedly many contributors.

Definition 8 (Synchronized distribution) A distribution R is *synchronized* if for every configuration $c(\rho, i)$ with effective stack height 1 and for every $\rho' \in R$, we have $c_\psi(\rho', i) = c(\rho, i)$ (same control state and same stack content) and, moreover, $c_\psi(\rho', i)$ also has effective stack height 1.[6]

Example 7 We now give an example of two distributions of a finite run that decrease the effective stack height, one of them being synchronized. Consider the two distributions R and S of $\rho = r_a r_b r_b r_c r_c r_c$ in Example 5. Further assume that the PDM P has one single state p, stack symbols $\{\perp, \alpha\}$ such that the three rules r_a, r_b and r_c are given by $r_a\colon p\perp \to p\alpha\perp$, $r_b\colon p\alpha \to p\alpha\alpha$, and $r_c\colon p\alpha \to p$. Figure 2 graphically depicts the stack contents of the configurations of the runs (the control state is always p), and their respective effective stack heights.

We observe that ρ is effectively 4-bounded. The distribution R is $(Z, 2)$-bounded for every $1 \le Z \le 6$, because the configurations $c_\psi(\rho'_j, i)$ have effective stack height at most 2 for every $1 \le j \le 3$ and every $1 \le i \le 6$. The distribution is not synchronized. Indeed, the configuration $c(\rho, 6) = p\perp$ has effective stack height 1, but $c_\psi(\rho'_2, 6) = p\alpha\perp \neq c(\rho, 6)$. The distribution S is $(Z, 3)$-bounded for every $1 \le Z \le 6$ and synchronized. Note that in each of $\{\sigma'_i\}_{i=1,2,3}$ at positions $last_\psi(\sigma'_i, 0)$ and $last_\psi(\sigma'_i, 6)$ (the only two positions at which ρ has effective stack height 1), the stack content is $\perp$, and thus the effective stack height is also 1.

We can now state the boundedness lemma. Notice that the constant N in the lemma depends only on C.

[6] Notice that the effective stack height of a configuration depends on the run it belongs to, and so $c(\rho, i) = c_\psi(\rho', i)$ does not necessarily imply that they have the same effective stack height.

Lemma 6 (Boundedness lemma) *Let $N = 2|Q_C|^2|\Gamma_C| + 1$, and let ρ be a run of C. For every $Z \in dom(\rho)$ there is an (Z, N)-bounded and synchronized distribution R_Z of ρ.*

We will prove the lemma by induction on $Z \in \mathrm{dom}(\rho)$. In order to do this, we need two ingredients. The first ingredient (Lemma 7 below) allows us to distribute the first position whose stack height goes beyond N. The second ingredient (Definition 9 and Lemma 8 will allow us to compose distributions for the induction step.

Lemma 7 *Let $N = 2|Q_C|^2|\Gamma_C|+1$. Let ρ be a run of C and $Z \in dom(\rho)$ be the first position of ρ such that $c(\rho, Z)$ is not N-bounded. Then there is a (Z, N)-bounded and synchronized distribution of ρ.*

Proof For convenience, when we want to denote that, in a run ρ, the configurations reached after $(\rho)_{1..i}$ and $(\rho)_{1..j}$ are c and c', we write

$$\rho = (\rho)_{1..i}\ [c]\ (\rho)_{i+1..j}\ [c']\ (\rho)_{j+1..\infty}\ .$$

We construct a (Z, N)-bounded and synchronized distribution $\{\rho_a, \rho_b\}$ of ρ. We denote by

$$\alpha_{N+1}\alpha_N \cdots \alpha_1 w_0,$$

the stack content of $c(\rho, Z)$ where the α_i are stack symbols and w_0 is a stack word. Because $c(\rho, Z)$ is not N-bounded in ρ, the active prefix of $c(\rho, Z)$ starts with $\alpha_{N+1}\alpha_N \cdots \alpha_1$. Define $\{\overrightarrow{p}_1, \overleftarrow{p}_1, \overrightarrow{p}_2, \overleftarrow{p}_2, \ldots, \overrightarrow{p}_N, \overleftarrow{p}_N\} \subseteq \mathrm{dom}(\rho)$ such that for each i, $1 \leq i \leq N$ we have $c(\rho, \overrightarrow{p}_i)$ and $c(\rho, \overleftarrow{p}_i)$ are the configurations immediately after the symbol α_i in $c(\rho, Z)$ is pushed, respectively popped and such that each configuration between $\overrightarrow{p}_i$ (included) and $\overleftarrow{p}_i$ (excluded) has $\alpha_i\alpha_{i-1}\cdots\alpha_1 w_0$ as a suffix. We get $c(\rho, \overrightarrow{p}_i) = q_i\alpha_i\alpha_{i-1}\ldots\alpha_0 w_0$ and $c(\rho, \overleftarrow{p}_i) = q_i'\alpha_{i-1}\ldots\alpha_0 w_0$ for some $q_i, q_i' \in Q_C$. Observe that the following holds: $\overrightarrow{p}_1 < \cdots < \overrightarrow{p}_{N-1} < \overrightarrow{p}_N < Z < \overleftarrow{p}_N < \overleftarrow{p}_{N-1} < \cdots < \overleftarrow{p}_1$.

Since $N = 2|Q_C|^2|\Gamma_C|+1$, by the pigeonhole principle we find q, α, q' and three indices $1 \leq j_1 < j_2 < j_3 \leq N$ such that by letting

$$\underbrace{\alpha_{j_3-1}\cdots\alpha_{j_2}}_{=w_3}\underbrace{\alpha_{j_2-1}\cdots\alpha_{j_1}}_{=w_2}\underbrace{\alpha_{j_1-1}\cdots\alpha_1}_{=w_1}$$

we have

$$\begin{aligned}\rho = {} & (\rho)_{1..\overrightarrow{p}_{j_1}}\ [q\alpha w_1 w_0]\ (\rho)_{\overrightarrow{p}_{j_1+1}..\overrightarrow{p}_{j_2}}\ [q\alpha w_2 w_1 w_0]\ (\rho)_{\overrightarrow{p}_{j_2+1}..\overrightarrow{p}_{j_3}}\ [q\alpha w_3 w_2 w_1 w_0] \\ & (\rho)_{\overrightarrow{p}_{j_3+1}..\overleftarrow{p}_{j_3}}\ [q' w_3 w_2 w_1 w_0]\ (\rho)_{\overleftarrow{p}_{j_3+1}..\overleftarrow{p}_{j_2}}\ [q' w_2 w_1 w_0]\ (\rho)_{\overleftarrow{p}_{j_2+1}..\overleftarrow{p}_{j_1}}\ [q' w_1 w_0]\ (\rho)_{\overleftarrow{p}_{j_1+1}..\infty}\ .\end{aligned}$$

Now define ρ_a from ρ by simultaneously deleting w_2 (actually, the two following parts of the run $(\rho)_{\overrightarrow{p}_{j_1+1}..\overrightarrow{p}_{j_2}}$ and $(\rho)_{\overleftarrow{p}_{j_2+1}..\overleftarrow{p}_{j_1}}$). We similarly define ρ_b by deleting w_3 (actually, $(\rho)_{\overrightarrow{p}_{j_2+1}..\overrightarrow{p}_{j_3}}$ and $(\rho)_{\overleftarrow{p}_{j_3+1}..\overleftarrow{p}_{j_2}}$). The following shows that ρ_a defines a legal run since it is given by

$$\begin{aligned}& (\rho)_{1..\overrightarrow{p}_{j_1}}\ [q\alpha w_1 w_0]\ (\rho)_{\overrightarrow{p}_{j_2+1}..\overrightarrow{p}_{j_3}}\ [q\alpha w_3 w_1 w_0] \\ & (\rho)_{\overrightarrow{p}_{j_3+1}..\overleftarrow{p}_{j_3}}\ [q' w_3 w_1 w_0]\ (\rho)_{\overleftarrow{p}_{j_3+1}..\overleftarrow{p}_{j_2}}\ [q' w_1 w_0]\ (\rho)_{\overleftarrow{p}_{j_1+1}..\infty}\ .\end{aligned}$$

A similar reasoning holds for ρ_b. We conclude by proving two claims.

ρ distributes to $\{\rho_a, \rho_b\}$. The embedding function ψ for ρ_a (again, the case of ρ_b is analogous) is given by

$$\psi(\rho_a, i) = \begin{cases} i & \text{for } 1 \leq i \leq \overrightarrow{p}_{j_1} \\ i + (\overrightarrow{p}_{j_2} - \overrightarrow{p}_{j_1}) & \text{for } \overrightarrow{p}_{j_1} + 1 \leq i \leq \overleftarrow{p}_{j_2} - (\overrightarrow{p}_{j_2} - \overrightarrow{p}_{j_1}) \\ i + (\overleftarrow{p}_{j_1} - \overleftarrow{p}_{j_2}) + (\overrightarrow{p}_{j_2} - \overrightarrow{p}_{j_1}) & \text{for } \overleftarrow{p}_{j_2} - (\overrightarrow{p}_{j_2} - \overrightarrow{p}_{j_1}) + 1 \leq i \end{cases}$$

Hence, we have a disribution $\{\rho_a, \rho_b\}, \psi$ of ρ.
$\{\rho_a, \rho_b\}, \psi$ *is a* (Z, N)*-bounded and synchronized distribution of* ρ. Since the effective stack height of every configuration of ρ_a (resp. ρ_b) up to position $last_\psi(\rho_a, Z)$ (resp. $last_\psi(\rho_b, Z)$) is at most N, the distribution is (Z, N)-bounded. Finally, observe that we have $c(\rho, i) = c_\psi(\rho_a, i) = c_\psi(\rho_b, i)$ for every $i \leq \overrightarrow{p}_{j_1}$ and every $i \geq \overleftarrow{p}_{j_1}$. Since all configurations of ρ of effective stack height 1 are in these two areas, the distribution is synchronized. □

We now introduce the second ingredient for the proof of the Boundedness Lemma: a notion of composition of distributions that allows us to "nest" distributions (that is, to distribute a run into several runs, and then distribute one of these runs again into several runs), while preserving the properties of synchronization and boundedness. We then show, in Lemma 8, that the composition of distributions is well-defined, that is, that the composition of distributions is indeed a distribution of ρ.

Definition 9 Let R, ψ be a distribution of ρ. Let $\rho' \in R$, and let R', ψ' be a distribution of ρ'. The *composition* of R, ψ and R', ψ' is given by $(R \ominus \{\rho'\}) \oplus R', \psi''$, where the embedding function ψ'' is defined as follows:

- $\psi''(r, i) = \psi(r, i)$ for every $r \in R \ominus \{\rho'\}$, and
- $\psi''(r, i) = \psi(\rho', \psi'(r, i))$ for every $r \in R'$.

Lemma 8 *Let R, ψ be a distribution of ρ. Let $\rho' \in R$ and let R', ψ' be a distribution of ρ'. The composition $(R \ominus \{\rho'\}) \oplus R', \psi''$ is a distribution of ρ.*

Proof We need to show that ψ'' satisfies the three properties of an embedding function.

- $(\rho'')_i = (\rho)_{\psi''(\rho'', i)}$ *for every* $\rho'' \in R \ominus \{\rho'\} \oplus R'$ If $\rho'' \in R \ominus \{\rho'\}$, then, as ψ is a distribution of ρ, we have $(\rho'')_i = (\rho)_{\psi(\rho'', i)}$. By definition of ψ'', we get $\psi''(\rho'', i) = \psi(\rho'', i)$. If $\rho'' \in R'$, then, since R', ψ' is a distribution of ρ', $(\rho'')_i = (\rho')_{\psi'(\rho'', i)}$. Since R, ψ is a distribution and $\rho' \in R$, $(\rho')_j = (\rho)_{\psi(\rho', j)}$. Taking $j = \psi'(\rho'', i)$, we get $(\rho'')_i = (\rho')_{\psi'(\rho'', i)} = (\rho)_{\psi(\rho', \psi'(\rho'', i))} = (\rho)_{\psi''(\rho'', i)}$. So, for every $\rho'' \in R \ominus \{\rho'\} \oplus R'$, we finally obtain $(\rho'')_i = (\rho)_{\psi''(\rho'', i)}$.
- *Surjectivity* If $k \in \mathrm{dom}(\rho)$, we first exploit the surjectivity of ψ: either there exists $\rho'' \in R \ominus \{\rho'\}$, and some $i \in \mathrm{dom}(\rho'')$ such that $\psi(\rho'', i) = k$ (which means that $\psi''(\rho'', i) = k$) or there is some $j \in \mathrm{dom}(\rho')$ such that $\psi(\rho', j) = k$. In the latter case, we then exploit the fact that R', ψ' is a distribution of ρ', and deduce that there exists $\rho'' \in R'$ and $i \in \mathrm{dom}(\rho'')$ such that $\psi'(\rho'', i) = j$; hence we have $\psi(\rho', \psi'(\rho'', i)) = k$, and so $\psi''(\rho'', i) = k$.
- *Monotonicity* For every $\rho'' \in R \ominus \{\rho'\}$, from the monotonicity of ψ we obtain that $\psi''(\rho'', i) < \psi''(\rho'', j)$ for every $i < j$. If $\rho'' \in R'$, first we derive from the monotonicity of ψ' that $\psi'(\rho'', i) < \psi'(\rho'', j)$ holds for every $i < j$. Then, by monotonicity of ψ, we obtain $\psi(\rho', \psi'(\rho'', i)) < \psi(\rho', \psi'(\rho'', j))$, and so $\psi''(\rho'', i) < \psi''(\rho'', j)$. □

Proof (*of boundedness lemma, Lemma 6*) The proof is by induction on Z. If $Z = 1$, then we claim $R_1 = \{\rho\}$. This is because the first configuration of a run has effective stack height at

most 2 (if the first rule was a push, and that symbol will be later popped). Since by definition $N \geq 3$, we get that ρ is $(1, N)$-bounded and trivially synchronized.

For the induction step, assume that some distribution R_{Z-1}, ψ of ρ is $(Z-1, N)$-bounded and synchronized. If R_{Z-1} is also (Z, N)-bounded, we take $R_Z = R_{Z-1}$, and we are done. Otherwise, there is $\rho' \in R_{Z-1}$ such that $c_\psi(\rho', 0), \ldots, c_\psi(\rho', Z-1)$ are effectively N-bounded, but $c_\psi(\rho', Z)$ is not.

Informally, this means that the Z-th transition of ρ was distributed to ρ'. Let $Z_{\rho'}$ be that position in ρ'; formally $Z = \psi(\rho', Z_{\rho'})$ (if no such $Z_{\rho'}$ exists, $c_\psi(\rho', Z-1) = c_\psi(\rho, Z)$). Since $Z_{\rho'}$ is the first position of ρ' whose configuration is not N-bounded, we have that $c(\rho', 0), \ldots, c(\rho', Z_{\rho'} - 1)$ are N-bounded, but $c(\rho', Z_{\rho'})$ is not. We apply Lemma 7 to each such ρ' and $Z_{\rho'}$, and get $(Z_{\rho'}, N)$-bounded and synchronized distributions for those ρ': $R_{\rho'} = \{\rho'_a, \rho'_b\}$. Let R_Z be the distribution obtained by replacing in R_{Z-1} every bad run ρ' by $R_{\rho'}$. Then R_Z is an (Z, N)-bounded and synchronized distribution of ρ. □

Example 8 With this example, we make the following two claims: the first claim is about the bound $(2|Q|^2|\Gamma| + 1)$. We claim this bound is tight as the minimal value of N required for the boundedness lemma (Lemma 6) and also the reduction theorem (Theorem 4) to hold is $\Theta(|Q|^2|\Gamma|)$. We do not prove $(2|Q|^2|\Gamma| + 1)$ to be exact, we prove it to be (linearly) tight.

The second claim is that if we consider a network admitting an infinite run, we may need more instances of the $(2|Q|^2|\Gamma| + 1)$-restriction of the PDM contributor to produce an infinite run. Intuitively, this means bounding memory for contributors comes at the price of witnesses involving more contributors. Below we give an example where with unbounded memory there is a witness with one contributor whereas with bounded memory a witness requires at least two contributors.

We build a PDM with $k_1 + k_2 + 2$ states and with stack alphabet $\{\bot\} \cup [1, k_3]$, where k_1, k_2, k_3 are distinct prime numbers. Conceptually, the PDM has two cycles. The first cycle has k_1 states, and each traversal of the cycle pushes the word $(1 \ldots k_3)^{k_1}$ on to the stack. The cycle can be left non-deterministically. We use an additional state, the only initial one, to ensure that the cycle is traversed at least once.

On leaving the first cycle, the PDM enters a second cycle. The second cycle has k_2 states and each state pops a stack symbol. Thus, each traversal of the cycle pops k_2 symbols from the stack. The PDM can only leave this cycle from its first state, and only when $\bot$ is the topmost stack symbol. On leaving the second cycle, the PDM moves to the last state, from where it writes victory in the store.

In order to reach the last state, the stack of the PDM must reach a height of at least $(1 + k_1k_2k_3)$ symbols. This is because, at the end of the first cycle, the stack has some multiple of k_1k_3 symbols which is also a multiple of k_2. By relative primality of k_1, k_2, and k_3, the stack must have at least $k_1k_2k_3$ symbols in addition to $\bot$. Therefore, no run reaching the last state can be distributed into runs exhibiting effective stack height smaller than $(1 + k_1k_2k_3) = \Theta(|Q|^2|\Gamma|)$, which proves claim 1.

To address claim 2, we further improve this example so as to show that a single instance of the contributor run in parallel with a special leader may reach the last state, but at least two instances of its N-restriction are required, so that at least one of them may reach that state.

It is possible for the leader to be informed whenever a contributor takes a loop (once in each loop the contributor informs the leader through the store and pauses until it receives acknowledgment through the store). Then the contributor asks permission before entering in the last state. The leader only grants permission if he was informed exactly a multiple of k_4 times of the entrance of some contributor in some loop (the leader is simply a loop with $1+k_4$ states), where k_4 is a prime number different from the others. If there is only one contributor,

the contributor may reach the victory state by growing a $1 + k_1k_2k_3k_4$-sized stack, which is too large for its N-restriction. Therefore a single instance of the N-restricted contributor does not suffice. At least two are required for an accepting run.

5.3.3 The proof of the Reduction Theorem

Proof (*of Theorem* 4) Theorem 4 states the non-emptiness is equivalent to the non-emptiness using N-restricted contributors. By Theorem 3, the non-restricted contributors capture all behaviours of the restricted ones. Thus one implication is trivial. For the other direction, we show that from an infinite run (with unrestricted contributors), we can (Z, N)-distribute each contributor with Z sufficiently large, which allows us, using the pigeonhole principle, to deduce an infinite run using the N-restricted contributors.

Let w be a word of $L(D)$ and let M a be multiset of words of C compatible with w. Let s be a witness of compatibility, and let $\mathcal{I}$ be the corresponding interleaving function (as introduced in the proof of the Distributing lemma). Recall that s is an interleaving of w and M, that $\mathcal{I}(w, i)$ is the position of s at which we find the i-th letter of w, and that $\mathcal{I}(v, j)$ is the position of s at which we find the j-th letter of v, for every $v \in M$.

In the sequel of the proof we consider that M only contains infinite words. Remark that if M contains some finite words, we can apply the distribution lemma over those runs and deduce (a possibly larger) multiset of words that yield N-bounded runs while still preserving compatibility. Without further formalization, we only consider the case where M contains only infinite words.

Let σ be an accepting run of w, and let R be a multiset of runs accepting each element of the multiset M. The proof follows three steps:

(1) We find a sequence of positions of s corresponding to actions of the leader, such that both the run of the leader and s can be pumped between any two such positions.
(2) We take a position far enough in this sequence, say $\mathcal{Z}$, and distribute all the runs of R into a multiset $R_{\mathcal{Z}}$, such that every run of $R_{\mathcal{Z}}$ is N-bounded up to position $\mathcal{Z}$. We show the existence of two positions, say X, Y, both smaller than $\mathcal{Z}$, satisfying the following condition. Take the multisets of configurations of the runs of $R_{\mathcal{Z}}$ at positions X and Y, and "prune" them by removing their dark suffixes. Let C_X and C_Y be the resulting multisets of pruned configurations. Then C_X and C_Y have the same support (that is, they contain the same elements, although not necessarily the same number of times).
(3) We show that by adding more runs to $R_{\mathcal{Z}}$, we can obtain a new distribution for which the multisets C_X and C_Y not only have the same support, but are equal. We then show that the runs executed by the leader and by the contributors of this new distribution between positions X and Y can be pumped. This yields a word $w_1 w_2^\omega \in L(D)$ (where w_2 is the word executed by the leader between positions X and Y) compatible with a multiset of words of the form $\{v_{11} v_{21}^\omega, \ldots, v_{1n} v_{2n}^\omega\}$ (where $v_{21}, \ldots v_{2n}$ are the runs executed by the contributors between positions X and Y), and for which we can find a witness of compatibility of the form $s_1 s_2^\omega$, where s_1 is an interleaving of w_1 and $\{v_{11}, \ldots, v_{1n}\}$, and s_2 is an interleaving of w_2 and $\{v_{21}, \ldots, v_{2n}\}$.

Step (1) Since σ is an infinite accepting run of D, by Proposition 2 it contains infinitely many positions of effective stack height 1. By the pigeonhole principle, from this sequence of positions we can extract an infinite subsequence of configurations with the same control state and topmost stack symbol. Since σ is also an accepting run, we can further extract from this sequence an infinite subsequence such that between any two positions an accepting state

of D is visited. Let $(b_i)_{i\in\mathbb{N}}$ denote the image of this last infinite sequence by $\mathcal{I}$. That is, $(b_i)_{i\in\mathbb{N}}$ denotes the infinite sequence of positions of s obtained by the procedure above.

Now from $(b_i)_{i\in\mathbb{N}}$ we extract a subsequence $(c_i)_{i\in\mathbb{N}}$ such that between any two elements of it, every run of R reaches a configuration with effective stack height 1. More formally, for every i and for every $\rho \in R$, there exists $p_{i,\rho} \in \mathrm{dom}(\rho)$ such that $c(\rho, p_{\rho,i})$ has effective stack height 1 and $c_i < \mathcal{I}(\rho, p_{\rho,i}) < c_{i+1}$. Since, by Proposition 2, every run of R reaches infinitely often such configurations, $(c_i)_{i\in\mathbb{N}}$ exists. This gives us our sequence of positions in s.

Step (2) Let $t = |M|2^{|Q_C||\Gamma_C|^{|Q_C|^2|\Gamma_C|+1}} + 1$, and let $\mathcal{Z} = c_t$ (that is, position $\mathcal{Z}$ (in s) is the t-th element of the sequence $(c_i)_{i\in\mathbb{N}}$, and is likely much further than t-th position of s). For each run $\rho \in R$, let Z_ρ denote an element of $\mathrm{dom}(\rho)$ such that $\mathcal{I}(\rho, Z_\rho) \geq \mathcal{Z}$. By Lemma 6, we can distribute each run $\rho \in R$ into a $(Z_\rho, N)-$bounded multiset R_ρ (with embedding function ψ_ρ).

For every $i \geq 1$, let $q_{\rho,i}$ be the largest position of $\mathrm{dom}(\rho)$ such that $\mathcal{I}(\rho, q_{\rho,i}) \leq c_i$, and let $R_\rho(q_{\rho,i}) = \{c_{\psi_\rho}(\tau, q_{\rho,i}) | \tau \in R_\rho\}$ be the multiset of configurations of R_ρ at the position corresponding to $q_{\rho,i}$. We denote by $\alpha_\rho(i)$ the result of removing the dark suffixes of the configurations of $R_\rho(q_{\rho,i})$. We call the result pruned configurations.

If $i \leq t$, then, by the definition of R_ρ, all the active prefixes of $R_\rho(q_{\rho,i})$ are N-bounded. So the pruned configurations of $\alpha_\rho(i)$ consist of a control state and a stack content of length at most N, and therefore the number of possible pruned configurations is bounded by $|Q_C||\Gamma_C|^{|Q_C|^2|\Gamma_C|+1}$. It follows that the number of possible sets (not multisets!) of pruned configurations is strictly smaller than t. So by the pigeonhole principle we find two elements c_l and c_r of the sequence $(c_i)_{i\in\mathbb{N}}$, where $l < r \leq t$, such that $\alpha_\rho(l)$ and $\alpha_\rho(r)$ have the same support for every ρ, (i.e. the sets are equal though the multisets may not be).

Step (3) We show how to modify the distributions R_ρ so that the multisets $\alpha_\rho(l)$ and $\alpha_\rho(r)$ not only have same support, but are equal. Observe that, even though the multisets are not equal, they have the same cardinality. We introduce new runs in the distribution to "balance" these multisets. Denote by α the common support of $\alpha_\rho(l)$ and $\alpha_\rho(r)$. For every $a \in \alpha$, we find two runs ρ_a^l and ρ_a^r in R_ρ such that $c_{\psi_\rho}(\rho_a^l, q_{\rho,l})$ and $c_{\psi_\rho}(\rho_a^r, q_{\rho,r})$ have pruned configuration a.

Now we define a new distribution of ρ to a multiset $R_\rho \oplus \{\rho_{a,a'} | a, a' \in \alpha\}$ with embedding function ψ'_ρ. The run $\rho_{a,a'}$ is such that the pruned configuration $c_{\psi'_\rho}(\rho_{a,a'}, q_{\rho,l})$ is a and $c_{\psi'_\rho}(\rho_{a,a'}, q_{\rho,r})$ is a' : informally $\rho_{a,a'}$ does as ρ_a^l up to position $\psi_\rho(\rho_a^l, p_{\rho,l})$, and then as $\rho_{a'}^r$ from $\psi_\rho(\rho_{a'}^r, p_{\rho,l})$. (Remark that building these "multiset balancing runs" crucially relies on synchonization by requiring the existence of a position where the runs reach a position with effective stack height 1). Formally, ψ'_ρ is the same as ψ_ρ over each $\tau \in R_\rho$, and $\psi'_\rho(\rho_{a,a'}, i) = \psi_\rho(\rho_a^l, i)$ when $i \leq \psi_\rho(\rho_a^l, p_{\rho,l})$ and $\psi'_\rho(\rho_{a,a'}, i) = \psi_\rho(\rho_a^r, i - \psi_\rho(\rho_a^l, p_{\rho,l}) + \psi_\rho(\rho_{a'}^r, p_{\rho,l}))$ when $i > \psi_\rho(\rho_a^l, p_{\rho,l})$. Observe that since $c(\rho, p_{\rho,l})$ has effective stack height 1, it is exactly the same configuration as $c_{\psi_\rho}(\rho_a^l, p_{\rho,l})$ and $c_{\psi_\rho}(\rho_{a'}^r, p_{\rho,l})$. It is also the same configuration as $c_{\psi'_\rho}(\rho_{a,a'}, p_{\rho,l})$. So $\rho_{a,a'}$ is a run of C, and $R_\rho \oplus \{\rho_{a,a'} | a, a' \in \alpha\}$, ψ'_ρ is a synchronized (Z_ρ, N)-bounded distribution of ρ. By adding to R_ρ sufficiently many instances of the appropriate $\rho_{a,a'}$, we obtain a new distribution R'_ρ of ρ, such that the two multisets $\alpha_\rho(l)$ and $\alpha_\rho(r)$ are the same.

By the Distribution Lemma, the word w of $L(D)$ is compatible with the words of the runs $\bigoplus_{\rho\in R} R'_\rho$. Let π be a witness of compatibility. Consider the fragment of π between the positions corresponding to c_l and c_r in π. The content of the store is the same at these two positions. Also, recall that we chose the $(c_i)_{i\in\mathbb{N}}$ so that the projection of the fragment onto the actions of the leader can be repeated infinitely often. Denote by $w^\circlearrowleft$ the run of

the leader consisting of repeating the subrun between positions corresponding to c_l and c_r. Finally, for each R'_ρ, the multiset of pruned configurations of R'_ρ at positions c_l and c_r is the same, each run in R'_ρ is N-bounded between c_l and c_r, and has effective stack height 1 at some point on that fragment. This does not mean that for every run $\tau \in R_\rho$ the pruned configuration will be the same at those positions, but that there exists a permutation μ of R_ρ such that the pruned configuration of τ at position c_l is the same as $\mu(\tau)$ at position c_r. Let us briefly mention that we once again rely on synchonization for the correctness of the construction: it is crucial that there is a position with effective stack height 1 between c_l and c_r so that concatenating successive fragments of runs does not modify the effective stack height. Denoting $\ell_\tau = (\tau)_{c_{\psi'_\rho}(\tau,p_{\rho,l})+1..c_{\psi'_\rho}(\tau,p_{\rho,r})}$, we get that the multiset $\{(\tau)_{1..c_{\psi'_\rho}(\tau,p_{\rho,l})}\ell_\tau\ell_{\mu(\tau)}\ell_{\mu^2(\tau)}\ldots | \tau \in R'_\rho, \rho \in R\}$ is a multiset of N-bounded runs of C, that is compatible with $w^\circlearrowleft$. Finally Theorem 3 shows those N-bounded runs are captured by the N-restricted contributors which concludes the proof. □

6 Conclusion

Together with [10], our results complete the complexity picture for safety and liveness verification for parameterized systems consisting of a leader process and arbitrarily many anonymous and identical contributor processes communicating through a shared, bounded-value, register through reads and writes. When processes are modeled by finite-state machines, the complexity of both safety and liveness verification is NP-complete. Surprisingly, this complexity is lower than the non-parameterized case where a system consists of a leader and a fixed number of contributors. For pushdown processes, safety verification is PSPACE-complete, and our results show an NEXPTIME upper bound for liveness verification. Again, contrast the result with the undecidability of safety verification for a fixed number (two or more) of pushdown processes.

Acknowledgements Pierre Ganty has been supported by the Madrid Regional Government project S2013/ICE-2731, *N-Greens Software - Next-GeneRation Energy-EfficieNt Secure Software*, and the Spanish Ministry of Economy and Competitiveness project No. TIN2015-71819-P, *RISCO - RIgorous analysis of Sophisticated COncurrent and distributed systems*.

References

1. Abdulla PA, Bertrand N, Rabinovich A, Schnoebelen P (2005) Verification of probabilistic systems with faulty communication. Inf Comput 202(2):105–228
2. Abdulla PA, Cerans K, Jonsson B, Tsay Y-K (1996) General decidability theorems for infinite-state systems. In: LICS '96, IEEE Computer Society, Washington, DC, pp 313–321
3. Abdulla PA, Jonsson B (1996) Verifying programs with unreliable channels. Inf Comput 127(2):91–101
4. Aminof B, Kotek T, Rubin S, Spegni F, Veith H (2014) Parameterized model checking of rendezvous systems. In: CONCUR '14 Proceedings of the 25th International Conference on Concurrency Theory, vol 704 of LNCS. Springer, Heidelberg, pp 109–124
5. Angluin D, Aspnes J, Eisenstat D, Ruppert E (2007) The computational power of population protocols. Distrib Comput 20(4):279–304
6. Apt KR, Kozen DC (1986) Limits for automatic verification of finite-state concurrent systems. Inf Process Lett 22(6):307–309
7. Bouajjani A, Esparza J, Maler O (1997) Reachability analysis of pushdown automata: application to model-checking. In: CONCUR '97 Proceedings of the 8th International Conference on Concurrency Theory, vol 1243 of LNCS. Springer, Heidelberg, pp 135–150

8. Esparza J, Finkel A and Mayr R (1999) On the verification of broadcast protocols. In: LICS '99, IEEE Computer Society, Washington, DC, pp 352–359
9. Esparza J, Ganty P, Majumdar R (2013) Parameterized verification of asynchronous shared-memory systems. In: CAV '13 Proceedings of the 23rd International Conference on Computer Aided Verification, vol 8044 of LNCS. Springer, Heidelberg, pp 124–140
10. Esparza J, Ganty P, Majumdar R (2016) Parameterized verification of asynchronous shared-memory systems. J ACM 63(1):10
11. German SM, Sistla AP (1992) Reasoning about systems with many processes. J ACM 39(3):675–735
12. Grädel E (1988) Subclasses of presburger arithmetic and the polynomial-time hierarchy. Theor Comput Sci 56:289–301
13. Hague M (2011) Parameterised pushdown systems with non-atomic writes. In: Proceedings of FSTTCS '11, vol 13 of LIPIcs. Schloss Dagstuhl, Wadern, pp 457–468
14. Meyer R (2008) On boundedness in depth in the pi-calculus. In: Proceedings of IFIP TCS 2008, vol 273 of IFIP. Springer, Heidelberg, pp 477–489
15. Pnueli A, Xu J, Zuck LD (2002) Liveness with (0, 1, infty)-counter abstraction. In: CAV '02 Proceedings of 14th International Conference on Computer Aided Verification, vol 2404 of LNCS. Springer, Heidelberg, pp 107–122
16. Torre SL, Muscholl A, Walukiewicz I (2015) Safety of parametrized asynchronous shared-memory systems is almost always decidable. In: CONCUR '15 Proceedings of 26th International Conference on Concurrency Theory, vol 42 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Wadern, pp 72–84
17. Verma KN, Seidl H, Schwentick T (2005) On the complexity of equational horn clauses. In: CADE '05 20th International Conference on Automated Deduction, vol 1831 of LNCS. Springer, Heidelberg, pp 337–352