Model Checking Parameterized Asynchronous Shared-Memory Systems

Antoine Durand-Gasselin[1], Javier Esparza[1], Pierre Ganty[2], and Rupak Majumdar[3]

[1]TU Munich [2]IMDEA Software Institute [3]MPI-SWS

Abstract. We characterize the complexity of liveness verification for parameterized systems consisting of a leader process and arbitrarily many anonymous and identical contributors. 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 PSPACE-hard and in NEXPTIME 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.
Our proofs use combinatorial characterizations of independent interest. For finite-state machines, we 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.

1 Introduction

We study the verification problem for *parameterized asynchronous shared-memory systems* [12,9]. These systems consist of a *leader* process and arbitrarily many identical *contributors*, processes with no identity, running at arbitrarily relative speeds and subject to faults (a process can crash). The shared-memory consists of a read/write register that all processes can access to perform either a read operation or a write operation. 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], 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 typically depends on the (unbounded) number of processes, and so reducing liveness to safety, or to finitary reasoning, is typically much harder. Indeed, for many parameterized models, liveness verification is undecidable even if safety is decidable [8,13].

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 for a *fixed* number of finite-state machines is already PSPACE-complete, and for a fixed number of pushdown systems is undecidable. So we can now confirm a surprising fact: 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 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. [3,1]). Our results reveal another instance of the same phenomenon.

Technically, our proof methods are very different from those used for safety verification [9]. All our previous results [9] relied on a fundamental Simulation Lemma, inspired by Hague's work [12], 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.

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. On the other hand, 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, safety verification 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, and liveness verification is equivalent to Petri net reachability, for which we only know non-primitive recursive upper bounds. On the other hand, safety verification for extensions to Petri nets with reset or transfer, broadcast protocols, where arbitrarily many finite-state processes communicate through broadcast messages, or depth-bounded π-calculus, where dynamic processes communicate through messages, are non-primitive recursive and liveness verification is undecidable in all cases [2,8,13]. In most cases, introducing a stack makes the problem undecidable. Thus, our results, which show simultaneously lower complexity than non-parameterized problems, as well as similar complexity for liveness and safety, are quite unexpected.

German and Sistla [10] 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]), where the asymmetric case with a leader is EXPSPACE-complete. In this paper we study the same problems, but for a shared memory communication primitive.

Population protocols [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.

2 Formal Model: Non-Atomic Networks

In this paper, we often identify systems with languages: system actions are modeled as symbols in an alphabet, executions are modeled as infinite words, i.e., infinite sequences of symbols, and the system itself is modeled as the language of its executions. 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) $\mathrm{dom}(w)$ denote the set $\mathbb{N} \setminus \{0\}$. 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 $i, j \in \mathbb{N}$ and $i < j$ denote $(w)_i(w)_{i+1} \ldots (w)_j$. 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 $v = uw$.

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. Shuffle is associative and commutative, and so we can write $L_1 \between \cdots \between L_n$ or $\between_{i=1}^n L_i$. For example, if $L_1 = a^\omega$ and $L_2 = b^\omega$, we get $L_1 \between L_2 = (a+b)^\omega$.

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 alphabet of both systems 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 $L_1 \parallel \cdots \parallel L_n$.

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 $L_1 \between (L_2 \parallel L_3)$. Previously, we used ω-regular expressions. We extend them by using the $\parallel$ and $\between$ operators with the expected semantics.

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 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 $\mathcal{G}(a_1, \ldots, a_n) = \{a_i(g) \mid 1 \leq i \leq n,\ g \in \mathcal{G}\}$.

We fix two languages $\mathcal{D} \subseteq \Sigma_\mathcal{D}^\omega$ and $C \subseteq \Sigma_C^\omega$, called the *leader* and the *contributor*, with alphabets $\Sigma_\mathcal{D} = \mathcal{G}(r_d, w_d)$ and $\Sigma_C = \mathcal{G}(r_c, w_c)$, 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 for each value $g \in \mathcal{G}$, $Proj_{\{r_\star(g), w_\star(g)\}}(\mathcal{D} \cup C) \neq \emptyset$, else the value g is never used and is thus removed from $\mathcal{G}$.

Additionally, we fix an ω-language $\mathcal{S}$, called the *store*, over $\Sigma_\mathcal{D} \cup \Sigma_C$. It models the sequences of read and write operations supported by an 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 $\left(\sum_{g\in\mathcal{G}} \left(w_\star(g)\,(r_\star(g))^* \right)\right)^\omega + \left(\sum_{g\in\mathcal{G}} \left(w_\star(g)\,(r_\star(g))^* \right)^* \ \sum_{g\in\mathcal{G}} \left(w_\star(g)\,(r_\star(g))^\omega \right)\right)$ and any finite prefix thereof. Observe that $\mathcal{S}$ is completely determined by $\Sigma_\mathcal{D}$ and Σ_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 $C \subseteq \Sigma_C^\omega$ be a leader and a contributor, and let $k \geq 1$. The* k-instance of the $(\mathcal{D}, C)$-network *is the ω-language $\mathcal{N}^{(k)} = (\mathcal{D} \parallel \mathcal{S} \parallel \ \between_k \ C)$ where $\between_k C$ stands for $\between_{i=1}^k C$. The $(\mathcal{D}, C)$-network $\mathcal{N}$ is the ω-language $\mathcal{N} = \bigcup_{k=1}^\infty \mathcal{N}^{(k)}$. We omit the prefix $(\mathcal{D}, C)$ when it is clear from the context. It follows easily from the properties of shuffle and asynchronous product that $\mathcal{N} = (\mathcal{D} \parallel \mathcal{S} \parallel \ \between_\infty C)$, where $\between_\infty C$ is an abbreviation of $\bigcup_{k=1}^\infty \between_k C$.*

Next we introduce a notion of *compatibility* between a word of the leader and a multiset of words of the contributor. 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 Σ_C^ω (possibly containing multiple copies of a word). We say that u is* compatible *with M iff the ω-language $(u \parallel \mathcal{S} \parallel \ \between_{i=1}^k v_i)$ is non-empty. When u and M are compatible, there exists a word $s \in \mathcal{S}$ such that $(u \parallel s \parallel \ \between_{i=1}^k v_i) \neq \emptyset$. 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 Figure 1. The only ω-word of $\mathcal{D}$ is $(r_d(1)r_d(2)r_d(3))^\omega$ and the ω-language of 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 4 ω-words.

Stuttering property. Let $u \in \Sigma_\mathcal{D}^\omega$ and $M = \{v_1, \ldots, v_k\}$ and $s \in \mathcal{S}$ as in the previous definition, namely s witnesses the compatibility of u and M. Pick a set I of positions (viz. $I \subseteq \mathrm{dom}(s)$) such that $(s)_i \in \Sigma_C$ for each $i \in I$ and let s' to be the result of simultaneously appending to each $(s)_i$ $\ell_i \geq 0$ copies of $(s)_i$. We have that $s' \in \mathcal{S}$. Now define v_s to be the concatenation of all these copies in increasing order, i.e. $(s)_{i_1}^{\ell_{i_1}} \cdot (s)_{i_2}^{\ell_{i_2}} \cdots$ where $i_1 = \min(I)$, $i_2 = \min(I \setminus \{i_1\})$, ... We have that $(u \parallel s' \parallel v_s \between \ \between_{i=1}^k v_i) \neq \emptyset$, that is u is compatible with $M \oplus \{v_s\}$, the multiset consisting of M and v_s, as witnessed by s'.

An easy consequence of the stuttering property is the *copycat lemma* [9].

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

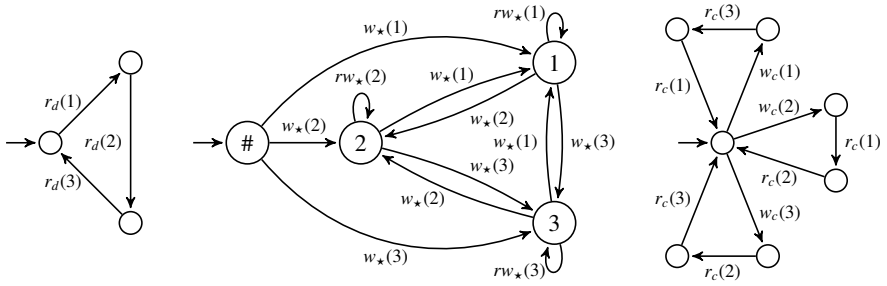

Fig. 1. Transition systems describing languages $\mathcal{D}$, $\mathcal{S}$, and C. We write $rw_\star(g) = r_\star(g) \cup w_\star(g) = \{r_c(g), r_d(g)\} \cup \{w_c(g), w_d(g)\}$. The transition system for $\mathcal{S}$ is in state $i \in \{1, 2, 3\}$ when the current value of the store is i.

2.3 The Model-checking Problem for Linear-time Properties

We consider the model checking problem for linear-time properties, formulated as follows. Given a network $\mathcal{N}$ and an ω-regular language L, decide whether $\mathcal{N} \parallel L$ is the empty language; equivalently, whether the projection of some ω-word of $\mathcal{N}$ onto the alphabet of L belongs to L. We assume L is given as a Büchi automaton A over the alphabet $\Sigma_{\mathcal{D}}$. Intuitively, the automaton 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 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 $\mathcal{N}_A = (L(A) \parallel L(D) \parallel \mathcal{S} \parallel \{\!\!\{}_\infty L(C))$ the empty ω-language?

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 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 by contributors, $\mathcal{N}_A$ is non-empty. In particular, $\mathcal{N}_A^{(4)} \neq \emptyset$.

Because $\Sigma_A = \Sigma_{\mathcal{D}}$, we replace A and D by a machine $A \times D$ with a Büchi acceptance condition. As the construction of $A \times D$ given A and D is quite standard, we omit the details to save space. Therefore, in what follows, we assume that D—the machine for the leader—comes with a Büchi acceptance condition and we simply forget about A.

There are two natural variants of the model checking problem, where the alphabet Σ_A of A is either the actions of all contributors $\Sigma_A = \Sigma_C$ or of all actions $\Sigma_A = \Sigma_{\mathcal{D}} \cup \Sigma_C$. In both these variants, the automaton A can be used to simulate *atomic* networks by picking specific contributors. Thus, the problem is undecidable for PDMs and has an EXPSPACE lower bound for FSMs and we do not study these problems further.

3 MC(FSM, FSM) is NP-complete

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. The language of an FSM A is the set of all accepted ω-words by A and is denoted $L(A)$. 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 Büchi automaton A, written $L(A)$, 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 to 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. Using this lemma we can then easily prove membership in NP. NP-hardness follows from NP-hardness of reachability [9].

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

We fix a Büchi automaton $D = (Q_D, \delta_D, q_{0D}, F)$ over $\Sigma_{\mathcal{D}}$, an FSM $C = (Q_C, \delta_C, q_{0C})$ over Σ_C. A *configuration* is a tuple $(q_D, g, \boldsymbol{p})$, where $q_D \in Q_D$, $g \in \mathcal{G} \cup \{\#\}$, and $\boldsymbol{p}\colon Q_C \to \mathbb{N}$ assigns to each state of C a natural number. Intuitively, q_D and g are the current states of D and the current value of the store including # the corrupted value nobody ever reads or writes, and for every state $q \in Q_C$, $\boldsymbol{p}(q)$ is the number of contributors currently at state q. We call $\boldsymbol{p}$ a *population* of Q_C, and write $|\boldsymbol{p}| = \sum_{q \in Q_C} \boldsymbol{p}(q)$ for the *size* of $\boldsymbol{p}$. Linear combinations of populations are defined componentwise: for every state $q \in Q_C$, we have $(k_1 \boldsymbol{p_1} + k_2 \boldsymbol{p_2})(q) := k_1 \boldsymbol{p_1}(q) + k_2 \boldsymbol{p_2}(q)$. Further, given $q \in Q_C$, we denote by $\boldsymbol{q}$ the population $\boldsymbol{q}(q') = 1$ if $q = q'$ and $\boldsymbol{q}(q') = 0$ otherwise, i.e., the population with one contributor in state q and no contributors elsewhere. A configuration is *accepting* if the state of D is accepting, that is whenever $q_D \in F$. Given a set of populations $\boldsymbol{P}$, we define $(q_D, g, \boldsymbol{P}) := \{(q_D, g, \boldsymbol{p}) \mid \boldsymbol{p} \in \boldsymbol{P}\}$.

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 including X_0, the set of initial configurations, given by $(q_{0D}, \#, \boldsymbol{P_0})$, where $\boldsymbol{P_0} = \{k\boldsymbol{q_{0C}} \mid k \geq 1\}$;
- $T = T_D \cup T_C$, where
  - T_D is the set of triples $\big((q_D, g, \boldsymbol{p})\,,\, t\,,\, (q'_D, g', \boldsymbol{p})\big)$ 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 $\big((q_D, g, \boldsymbol{p})\,,\, t\,,\, (q_D, g', \boldsymbol{p'})\big)$ such that $t \in \delta_C$, and one of the following conditions holds:
    (iii) $t = (q_C, w_c(g'), q'_C)$, $\boldsymbol{p} \geq \boldsymbol{q_C}$, and $\boldsymbol{p'} = \boldsymbol{p} - \boldsymbol{q_C} + \boldsymbol{q'_C}$; or
    (iv) $t = (q_C, r_c(g), q'_C)$, $\boldsymbol{p} \geq \boldsymbol{q_C}$, $g = g'$, and $\boldsymbol{p'} = \boldsymbol{p} - \boldsymbol{q_C} + \boldsymbol{q'_C}$.

We write $c \xrightarrow{t} c'$ if $(c, t, c') \in T$. We define $\Delta(t) := \boldsymbol{p}' - \boldsymbol{p}$. Observe that $\Delta(t) = \mathbf{0}$ in cases (i) and (ii), and $\Delta(t) = -\boldsymbol{q_C} + \boldsymbol{q_C}'$ in cases (iii) and (iv). So, indeed, $\Delta(t)$ depends only on the transition t, but not on $\boldsymbol{p}$. Furthermore, observe that in all cases we have $|\boldsymbol{p}| = |\boldsymbol{p}'|$, corresponding to the fact that the total number of contributors of a population remains constant.

3.2 The abstract transition system

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

$$\alpha(\boldsymbol{P}) = \{q \in Q_C \mid \boldsymbol{p}(q) \geq 1 \text{ for some } \boldsymbol{p} \in \boldsymbol{P}\}$$
$$\gamma(Q) = \{\boldsymbol{p} \mid \boldsymbol{p}(q) = 0 \text{ for every } q \in Q_C \setminus Q\} \ .$$

It is easy to see that α and γ satisfy $\gamma(\alpha(\boldsymbol{P})) \supseteq \boldsymbol{P}$ 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 including the initial ones $\alpha X_0 = \{(q_{0D}, \#, \alpha(\boldsymbol{P_0}))\} = \{(q_{0D}, \#, \{q_{0C}\})\}$;
- $(\ (q_D, g, Q),\ t,\ (q'_D, g', Q')\ ) \in \alpha T$ iff there is $\boldsymbol{p} \in \gamma(Q)$ and $\boldsymbol{p}'$ such that $(q_D, g, \boldsymbol{p}) \xrightarrow{t} (q'_D, g', \boldsymbol{p}')$ and $Q' = \alpha(\{\boldsymbol{p}' \mid \exists \boldsymbol{p} \in \gamma(Q) \colon (q_D, g, \boldsymbol{p}) \xrightarrow{t} (q'_D, g', \boldsymbol{p}')\})$.

Observe that the number of abstract configurations is bounded by $K = |Q_D| \cdot |\mathcal{G}| + 1 \cdot 2^{|Q_C|}$. We write $a \xrightarrow{t}_\alpha a'$ if $(a, t, a') \in \alpha T$. Notice that for every abstract transition $a \xrightarrow{t}_\alpha a'$ we have $t \in \delta_D \cup \delta_C$, hence the notation $\Delta(t)$ is well-defined. The abstraction satisfies the following properties:

(A) For each ω-path $c_0 \xrightarrow{t_1} c_1 \xrightarrow{t_2} c_2 \cdots$ of TS, there exists an ω-path $a_0 \xrightarrow{t_1}_\alpha a_1 \xrightarrow{t_2}_\alpha a_2 \cdots$ in αTS such that $c_i \in \gamma(a_i)$ for all $i \geq 0$.

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

- $t \in \delta_D$. Then $(q_D, g, \boldsymbol{p}) \xrightarrow{t} (q'_D, g', \boldsymbol{p})$ for every population $\boldsymbol{p}$ (because only the leader moves). So $(q_D, g, Q) \xrightarrow{t}_\alpha (q'_D, g', Q)$.
- $t \in \delta_C$. Consider the population $\boldsymbol{p} = 2 \sum_{q \in Q} \boldsymbol{q} \in \gamma(Q)$. Then $(q_D, g, \boldsymbol{p}) \xrightarrow{t} (q_D, g', \boldsymbol{p}')$, where $\boldsymbol{p}' = \boldsymbol{p} - \boldsymbol{q_C} + \boldsymbol{q_C}'$. But then $\boldsymbol{p}' \geq \sum_{q \in Q} \boldsymbol{q}$, and so $\alpha(\{\boldsymbol{p}'\}) \supseteq Q$, which implies $(q_D, g, Q) \xrightarrow{t}_\alpha (q_D, g', Q')$ for some $Q' \supseteq Q$.

So in every ω-path $a_0 \xrightarrow{t_1}_\alpha a_1 \xrightarrow{t_2}_\alpha a_2 \cdots$ of αTS, where $a_i = (q_{Di}, g_i, Q_i)$, there is an index $i \leq K$ after which the Q_i stabilize, that is, $Q_i = Q_{i+k}$ holds for every $k \geq 0$. However, the converse of (A) 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 $i \geq 0$. For

clarity, we show that this is true already in the case in which the leader consists of just one state and no transitions, and can therefore be omitted from the configurations. Consider a contributor machine C with two states q_0, q_1 and one single transition $t = (q_0, w_c(1), q_1)$. αTS contains the infinite path:

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

On the other hand, $TS = (X, T, X_0)$ is such that T contains a transition $((1, k_0\boldsymbol{q_0} + k_1\boldsymbol{q_1}),\ t,\ (1, (k_0{-}1)\boldsymbol{q_0} + (k_1{+}1)\boldsymbol{q_1}))$ for every $k_0 \geq 1$, and 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 certain lasso path in αTS. A lasso path consists of a stem and a cycle. Lemma 2 shows how every abstract finite path (like the stem) has a counterpart in TS. Lemma 3 characterizes precisely those cycles in αTS which have an infinite path counterpart in TS.

Lemma 2. *Let (q_D, g, Q) be an abstract configuration of αTS reachable from $(q_{0D}, \#, \alpha(\boldsymbol{P_0}))$ $(= \alpha X_0)$. For every $\boldsymbol{p} \in \gamma(Q)$, there exists $\hat{\boldsymbol{p}}$ such that $(q_D, g, \hat{\boldsymbol{p}})$ is reachable from $(q_{0D}, \#, \boldsymbol{P_0})$ and $\hat{\boldsymbol{p}} \geq \boldsymbol{p}$.*

A *cycle* of αTS is a path $a_0 \xrightarrow{t_1}_\alpha a_1 \xrightarrow{t_2}_\alpha a_2 \cdots \xrightarrow{t_{n-1}}_\alpha a_n$ such that $a_n = a_0$. A cycle is *realizable* if there is an infinite path $c_0 \xrightarrow{t'_1} c_1 \xrightarrow{t'_2} c_2 \cdots$ of TS such that for every $0 \leq k \leq n$ we have $c_k \in \gamma(a_k)$ and $t'_{k+1} = t_{k+1}$, and for every $k \geq n$ we have $c_k \in \gamma(a_{(k \mod n)})$ and $t'_{k+1} = t_{(k+1 \mod n)}$.

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 $\sum_{i=1}^n \Delta(t_i) = \mathbf{0}$.*

Theorem 1. $\mathtt{MC}(\mathit{FSM}, \mathit{FSM})$ *is NP-complete.*

Proof. NP-hardness, given in the appendix, follows from the NP-hardness of reachability [9]. We show membership in NP with the following high-level algorithm whose correctness, given in the appendix, relies on Lemmas 2 and 3:

- 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 $\ell \leq |Q_C|$.
- compute the set $\mathcal{Q}$ of abstract configurations $Q_D \times (\mathcal{G} \cup \{\#\}) \times \{\{q_{0C}\}, Q_1, \ldots, Q_\ell\}$ and compute the set Δ of abstract transitions $(q_D, g, Q) \xrightarrow{t}_\alpha (q'_D, g', Q')$ such that $(q_D, g, Q), (q'_D, g', Q') \in \mathcal{Q}$.
- guess an accepting abstract configuration of $a \in \mathcal{Q}$, that is $a = (q_D, g, Q)$ and q_D is accepting in D.
- check a is reachable from the initial abstract configurations $\alpha X_0 = \{(q_{0D}, \#, \{q_{0C}\})\}$.
- check there exists a path $a_0 \xrightarrow{t_1}_\alpha a_1 \ldots a_{n-1} \xrightarrow{t_n}_\alpha a_n$ where $n \geq 1$, $a_0 = a_n = a$ and $\sum_{i=1}^n \Delta(t_i) = \mathbf{0}$.

We show the algorithm belongs to NP. First, because the sequence guessed is no longer than $|Q_C|$, the guess can be done in polynomial time. Next, we give a NP algorithm to decide there exists a path $a_0 \xrightarrow{t_1}_\alpha a_1 \ldots a_{n-1} \xrightarrow{t_n}_\alpha a_n$ where $n \geq 1$, $a_0 = a_n = a$ and such that $\sum_{i=1}^n \Delta(t_i) = \mathbf{0}$. The algorithm performs the following steps.

– compute an FSA[1] $A_a^{\circlearrowleft}$ over the alphabet $\delta_D \cup \delta_C$ whose states and transitions are given by Q and Δ and such that its initial and final states are given by a and $\{a\}$, respectively;
– use the polynomial construction of Seidl *et al.* [14] to compute an (existential) Presburger formula Ω for the Parikh image of $L(A_a^{\circlearrowleft})$. The free variables of Ω are in one-to-one correspondence with the transitions of $\delta_D \cup \delta_C$. We thus adopt the convention that x_t denotes the variable corresponding to transition $t \in \delta_D \cup \delta_C$.
– compute Ω' by adding $|Q_C|$ variables and $|Q_C|$ constraints, one per state in $q_c \in Q_C$: $\sum_{tgt(t)=q_c} x_t = \sum_{src(t)=q_c} x_t$ 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.
– check satisfiability of Ω'. This step is in NP because satisfiability of an existential Presburger formula is in NP [11]. □

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 initial state q_0, a *stack alphabet* Γ including a bottom of 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 *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 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. A *run* $c_0 \xrightarrow{(v)_1} c_1 \xrightarrow{(v)_2} \dots$ on a word $v \in \Sigma^\omega$ is a sequence of configurations such that $c_0 = q_0\bot$ and $c_i \xrightarrow{(v)_{i+1}} c_{i+1}$ for all $i \geq 0$. We write $c \xrightarrow{*} c'$ if there is a run from c to c'. The language of a PDM P is denoted $L(P)$ and is given by the set of all words $v \in \Sigma^\omega$ such that the PDM has a run on v.

A Büchi PDM is a PDM with a set $F \subseteq Q$ of accepting states. A word is accepted by a Büchi PDM if there is a run on the word for which some state in F occurs infinitely often. The following lemma from Bouajjani *et al.* [7] characterizes accepting runs.

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

We now show MC(PDM, FSM) is decidable, generalizing the proof from Section 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 $(q_D, w, g, \boldsymbol{p})$, where $q_D \in Q_D$, $w \in \Gamma_D^*$ is the stack content, $g \in \mathcal{G} \cup \{\#\}$, and $\boldsymbol{p}$ is a population. Intuitively, $q_D w$ is the configuration of the leader. We extend the definitions from Section 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 $X_0 = (q_{0D}, \bot, \#, \boldsymbol{P_0})$ 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

[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.

$((q_D, w, g, \boldsymbol{p}), t, (q'_D, w', g', \boldsymbol{p}))$ 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) $t = (q_D w \xrightarrow{r_d(g)} q'_D w')$, $g = g'$. We define the abstraction αTS of TS as the obvious generalization of the abstraction in Section 3. An accepting path of the (abstract) transition system is an infinite path with infinitely many accepting (abstract) configurations. A detailed proof of the following theorem can be found in appendix.

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

5 MC(PDM, PDM) is in NEXPTIME

We show how to reduce MC(PDM, PDM) to MC(PDM, FSM) which, by the result of Section 4, shows that MC(PDM, PDM) is decidable. We first introduce the notion of *effective stack height* of a configuration in a run of a PDM, and define, given a PDM C, an FSM C_k that simulates all the runs of C of effective stack height k. Then we show that, for $k \in O(n^3)$, where n is the size of C, the language $(L(D) \parallel \mathcal{S} \parallel \langle\!\rangle_\infty L(C))$ is empty iff $(L(D) \parallel \mathcal{S} \parallel \langle\!\rangle_\infty L(C_k))$ is empty.

5.1 A FSM for runs of bounded effective stack height

Consider a run of a PDM that repeatedly pushes symbol 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 $k \geq 0$. The* active prefix *$ap(w_i)$ of w_i is the prefix satisfying $w_i = ap(w_i) \cdot ds(w_i)$. The* effective stack height *of c_i in ρ is $|ap(w_i)|$. We say that 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 (other runs for the same word may not be bounded).*

Intuitively, the effective stack height measures the actual memory required by the PDM to perform its run: the dark suffix can be safely erased. Given a position in the run, the elements of the stack never to be popped eventually are those in the longest common suffix of all subsequent stacks. Remark that the first element of that suffix may be read, therefore only the longest *proper* suffix is effectively useless. If follows that no configuration along an infinite run has effective stack height 0. Also observe that repeatedly pushing a symbol on the stack produces a run with effective stack height 1.

Proposition 1. *Every infinite run of a PDM contains infinitely many configurations of effective stack height 1.*

Proof. Let $p_0w_0 \to p_1w_1 \to p_2w_2 \to \cdots$ be any infinite run. Let X be the set of configurations inductively defined as follows: $p_0w_0 \in X$; if $p_iw_i \in X$, and $j > i$ is the smallest index satisfying $|w_j| \leq |w_k|$ for every $k > i$, then $p_jw_j \in X$. Clearly X is infinite, and it is easy to see that every configuration of X has effective stack height 1. □

We now construct a finite-state machine P_k recognizing the words of $L(P)$ accepted by k-bounded runs. This works because 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.

Definition 4. *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:*

- *The set of states Q_k is $Q \times \bigcup_{i=1}^{k} \Gamma^i$, that is states and stack contents no longer than k.*
- *The initial state q_{0k} is given by $(q_0, \bot)$.*
- *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.*

Theorem 3. *Given a PDM P, w admits a k-bounded run in P iff $w \in L(P_k)$.*

5.2 The Reduction Theorem

We fix a Büchi PDM D and a PDM C. By Theorem 3, in order to reduce $\mathtt{MC}(\mathtt{PDM}, \mathtt{PDM})$ to $\mathtt{MC}(\mathtt{PDM}, \mathtt{FSM})$ 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:*

$$(L(D) \parallel \mathcal{S} \parallel \mathbb{\}}_\infty L(C)) \neq \emptyset \quad \textit{iff} \quad (L(D) \parallel \mathcal{S} \parallel \mathbb{\}}_\infty L(C_N)) \neq \emptyset \ .$$

In the appendix we give an example where $\theta(N)$ is required (hence the bound is tight), and more instances of C_N are required (hence the notion of distribution is necessary).

This reduction yields the following upper bounds of complexity. The lower bound follows from the fact that the safety problem is PSPACE-hard [9], and can be easily reduced to the liveness problem as formulated in this paper using a reduction similar to that of Theorem 1. Remark that if the contributor is a one-counter machine (which can be seen as a special case of PDM), then its N-restriction is polynomial.

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

Given a run of D compatible with a finite multiset of runs of C, the proof of Theorem 4 constructs 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. 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 $(\sigma)_{1..i}((\sigma)_{i+1..j})^\omega$ 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.

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), we also sometimes write $\rho = r_1 r_2 r_3 \ldots$ thus enabling the use of $\mathrm{dom}(\rho)$, $(\rho)_k$, $(\rho)_{i..j}$ and $(\rho)_{i..\infty}$.

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.
  (That is, 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.)

Example 3. Let ρ be a run of a PDM P, below are two distributions of $\rho = r_a r_b r_b r_c r_c r_c$: on the left $R = \{\rho'_1, \rho'_2, \rho'_3\}$ and the embedding function ψ thereof; on the right $S = \{\sigma'_1, \sigma'_2, \sigma'_3\}$ and the function ψ' thereof.

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; let M_R the corresponding multiset of words, then $M \ominus \{v\} \oplus M_R$ is also compatible with u.*

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. We need some preliminaries. In Example 3, $(\rho)_{1..4}$ is distributed into $(\rho'_1)_{1..1}$, $(\rho'_2)_{1..2}$ and $(\rho'_3)_{1..3}$. If ρ was a run executed by some contributor in a non-atomic network, remark that we can replace it by 3 contributors with runs $\rho'_1, \rho'_2, \rho'_3$: instead of $(\rho)_1$, the three processes execute their first step $\{(\rho'_1)_1, (\rho'_2)_1, (\rho'_3)_1\}$ (in any order, but before any other action is performed by

the other processes of the network), then replace the execution of $(\rho)_{2..4}$ by $(\rho'_2)_2(\rho'_3)_{2..3}$. The rest of the network will not notice any difference.

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 largest position $k \in \mathrm{dom}(\rho')$ such that $\psi(\rho', k) \leq i$ (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 4. Let ρ, R, and ψ as in Example 3. 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 $c_\psi(\rho'_3, 5) = p\alpha\bot$. (Hint: see Example 5 in appendix for details.)

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. Further, we say that R is *synchronized* if for every configuration $c(\rho, i)$ with effective stack height 1 and for every $\rho' \in R$, $c_\psi(\rho', i) = c(\rho, i)$ (same control state and same stack content), and also has effective stack height 1. [2]

The Boundedness Lemma states 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 and synchronized distribution R_Z of ρ. The key of the proof is the following lemma.

Lemma 6. *Let $N = 2|Q_C|^2|\Gamma_C| + 1$, and let ρ be a run of C. Let $Z \in dom(\rho)$ 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 sketch of Lemma 6 (a detailed proof can be found in the Appendix). Because it is convenient, when we want to denote that, say, in a run ρ the configurations reached after $(\rho)_{1..i}$ and $(\rho)_{1..j}$ are c and c', we write it as follows $\rho = (\rho)_{1..i}\,[c]\,(\rho)_{i+1..j}\,[c']\,(\rho)_{j+1..\infty}$.

We construct a (Z, N)-bounded and synchronized distribution $\{\rho_a, \rho_b\}$ of ρ. Let $\alpha_{N+1}\alpha_N \cdots \alpha_1 w_0$ be the stack content of $c(\rho, Z)$. Define $\{\acute{p}_1, \grave{p}_1, \acute{p}_2, \grave{p}_2, \ldots, \acute{p}_N, \grave{p}_N\} \subseteq \mathrm{dom}(\rho)$ such that for each i, $1 \leq i \leq N$ we have $c(\rho, \acute{p}_i)$ and $c(\rho, \grave{p}_i)$ are the configurations immediately after the symbol α_i in $c(\rho, Z)$ is pushed, respectively popped and such that the stack content of each configuration between $\acute{p}_i$ (included) and $\grave{p}_i$ (excluded) equals $w_p\alpha_i\alpha_{i-1}\cdots\alpha_1 w_0$ for some $w_p \in \Gamma_C^*$. We get $c(\rho, \acute{p}_i) = q_i\alpha_i\alpha_{i-1}\ldots\alpha_0 w_0$ and $c(\rho, \grave{p}_i) = q'_i\alpha_{i-1}\ldots\alpha_0 w_0$ for some $q_i, q'_i \in Q_C$. Observe that the following holds: $\acute{p}_1 < \cdots < \acute{p}_{N-1} < \acute{p}_N < Z < \grave{p}_N < \grave{p}_{N-1} < \cdots < \grave{p}_1$.

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 $w_1 = \alpha_{j_1-1}\cdots\alpha_1$, $w_2 = \alpha_{j_2-1}\cdots\alpha_{j_1}$ and $w_3 = \alpha_{j_3-1}\cdots\alpha_{j_2}$, we have:

$$\rho = (\rho)_{1..\acute{p}_{j_1}}\,[q\alpha w_1]\,(\rho)_{\acute{p}_{j_1+1}..\acute{p}_{j_2}}\,[q\alpha w_2 w_1]\,(\rho)_{\acute{p}_{j_2+1}..\acute{p}_{j_3}}\,[q\alpha w_3 w_2 w_1]$$
$$(\rho)_{\acute{p}_{j_3+1}..\grave{p}_{j_3}}\,[q' w_3 w_2 w_1]\,(\rho)_{\grave{p}_{j_3+1}..\grave{p}_{j_2}}\,[q' w_2 w_1]\,(\rho)_{\grave{p}_{j_2+1}..\grave{p}_{j_1}}\,[q' w_1]\,(\rho)_{\grave{p}_{j_1+1}..\infty}\ .$$

[2] 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.

Now define ρ_a from ρ by simultaneously deleting $(\rho)_{\acute{p}_{j_1+1}..\acute{p}_{j_2}}$ and $(\rho)_{\grave{p}_{j_2+1}..\grave{p}_{j_1}}$. We similarly define ρ_b by deleting $(\rho)_{\acute{p}_{j_2+1}..\acute{p}_{j_3}}$ and $(\rho)_{\grave{p}_{j_3+1}..\grave{p}_{j_2}}$. The following shows that ρ_a defines a legal run since it is given by

$$(\rho)_{1..\acute{p}_{j_1}}\ [q\alpha w_1]\ (\rho)_{\acute{p}_{j_2+1}..\acute{p}_{j_3}}\ [q\alpha w_3 w_1](\rho)_{\acute{p}_{j_3+1}..\grave{p}_{j_3}}\ [q' w_3 w_1]\ (\rho)_{\grave{p}_{j_3+1}..\grave{p}_{j_2}}\ [q' w_1]\ (\rho)_{\grave{p}_{j_1+1}..\infty}\ .$$

A similar reasoning holds for ρ_b. Finally, one can show that $\{\rho_a, \rho_b\}$ is a (Z, N)-bounded and synchronized distribution of ρ.

Lemma 7 (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.*

The proof is by induction on Z. The distribution ψ_{Z+1}, R_{Z+1} is obtained from ψ_Z, R_Z by distributing each run ρ' of R_Z to a $(\psi_Z(\rho', Z) + 1, N)$-bounded run (applying Lemma 6).

Proof sketch of the Reduction Theorem (Theorem 4). Given a run σ of D compatible with a finite multiset M of runs of C, we construct another run τ of D, and a multiset R of N-bounded runs of C_N such that τ and R are compatible as well. We consider only the special case in which M has one single element ρ (and one single copy of it). Since σ is compatible with ρ, we fix a witness $\pi \in \mathcal{S}$ such that $\pi \in \sigma \between \rho$. We construct a "lasso run" out of π of the form $\lambda_1[\lambda_2]^\omega$. It suffices to find two positions in π where the content of the store is the same, the corresponding configurations of the leader are the same, and similarly for each contributor; the fragment between these two positions can be repeated (is "pumpable").

Given a position i of π, let i_ρ and i_σ denote the corresponding positions in ρ and σ.[3] Further, for every Z let R_Z be a (Z, N)-bounded and synchronized distribution of ρ with embedding function ψ (which exists by the Boundedness Lemma). Let $R_Z(i_\rho) = \{c_\psi(\eta, i_\rho) \mid \eta \in R_Z\}$ denote the multiset of configurations reached by the runs of R_Z after i steps of π. Proposition 1 shows that every infinite run of a PDM contains infinitely many positions of effective stack height 1. Using that (i) the store has a finite number of values, (ii) R_Z is (Z, N)-bounded, and (iii) there are only finitely many active prefixes of length at most N, we can apply the pigeonhole principle to find a sufficiently large number Z and three positions $i < j < k \leq Z$ in π satisfying the following properties:

(1) The contents of the store at positions i and k of π coincide.
(2) The configurations $c(\sigma, i_\sigma)$ and $c(\sigma, k_\sigma)$ of the leader have effective stack height 1, same topmost stack symbol and same control state. Further, σ enters and leaves some accepting state between i_σ and k_σ.
(3) The configuration $c(\rho, j_\rho)$ has effective stack height 1.
(4) For every configuration of $R_Z(i_\rho)$ there is a configuration of $R_Z(k_\rho)$ with the same control state and active prefix, and vice versa.

Condition (4) means that, after removing the dark suffixes, $R_Z(i_\rho)$ and $R_Z(k_\rho)$ contain the same pruned configurations, although possibly a different number of times (same set, different multisets). If we obtain the same multiset, then the fragment of π between positions i and k is pumpable by (1) and (2), and we are done. Otherwise, we use (3) and the fact that R_Z is synchronized (which had not been used so far) to obtain a new distribution in which the multisets coincide. This is achieved by adding new runs to R_Z.

[3] Position p in π defines position p_σ in σ such that $(\sigma)_{1..p_\sigma} = Proj_{\Sigma_\mathcal{D}}((\pi)_{1..p})$, similarly p_ρ is defined as satisfying $(\rho)_{1..p_\rho} = Proj_{\Sigma_C}((\pi)_{1..p})$.

References

1. Abdulla, P.A., Bertrand, N., Rabinovich, A., Schnoebelen, P.: Verification of probabilistic systems with faulty communication. Information and Computation 202(2), 105–228 (2005)
2. Abdulla, P.A., Cerans, K., Jonsson, B., Tsay, Y.K.: General decidability theorems for infinite-state systems. In: LICS '96. pp. 313–321. IEEE Computer Society (1996)
3. Abdulla, P.A., Jonsson, B.: Verifying programs with unreliable channels. Information and Computation 127(2), 91–101 (1996)
4. Aminof, B., Kotek, T., Rubin, S., Spegni, F., Veith, H.: Parameterized model checking of rendezvous systems. In: CONCUR '14: Proc. 25th Int. Conf. on Concurrency Theory. LNCS, vol. 8704, pp. 109–124. Springer (2014)
5. Angluin, D., Aspnes, J., Eisenstat, D., Ruppert, E.: The computational power of population protocols. Distributed Computing 20(4), 279–304 (2007)
6. Apt, K.R., Kozen, D.C.: Limits for automatic verification of finite-state concurrent systems. Information Processing Letters 22(6), 307 – 309 (1986)
7. Bouajjani, A., Esparza, J., Maler, O.: Reachability analysis of pushdown automata: Application to model-checking. In: CONCUR '97: Proc. 8th Int. Conf. on Concurrency Theory. LNCS, vol. 1243, pp. 135–150. Springer (1997)
8. Esparza, J., Finkel, A., Mayr, R.: On the verification of broadcast protocols. In: LICS '99. pp. 352–359. IEEE Computer Society (1999)
9. Esparza, J., Ganty, P., Majumdar, R.: Parameterized verification of asynchronous shared-memory systems. In: CAV '13: Proc. 23rd Int. Conf. on Computer Aided Verification. LNCS, vol. 8044, pp. 124–140. Springer (2013)
10. German, S.M., Sistla, A.P.: Reasoning about systems with many processes. Journal of ACM 39(3), 675–735 (1992)
11. Grädel, E.: Subclasses of presburger arithmetic and the polynomial-time hierarchy. Theor. Comput. Sci. 56, 289–301 (1988)
12. Hague, M.: Parameterised pushdown systems with non-atomic writes. In: Proc. of FSTTCS'11. LIPIcs, vol. 13, pp. 457–468. Schloss Dagstuhl (2011)
13. Meyer, R.: On boundedness in depth in the pi-calculus. In: In Procedings of IFIP TCS 2008. IFIP, vol. 273, pp. 477–489. Springer (2008)
14. Verma, K.N., Seidl, H., Schwentick, T.: On the complexity of equational horn clauses. In: CADE'05: 20th Int. Conf. on Automated Deduction. LNCS, vol. 1831, pp. 337–352. Springer (2005)

6 Appendix

6.1 Proofs of Section 3

Proof (of Lemma 2). The proof is by induction on the length of the abstract path of $\alpha_\rho TS$ from $\alpha_\rho X_0$ to configuration (q_D, g, Q).

The base case is such that $(q_D, g, Q) = (q_{0D}, \#, \{q_{0C}\})$ where $\alpha_\rho X_0 = \{(q_{0D}, \#, \alpha_\rho(\boldsymbol{P_0}))\} = \{(q_{0D}, \#, \{q_{0C}\})\}$. We conclude this case by observing that $\gamma(\alpha_\rho(\boldsymbol{P_0})) = \boldsymbol{P_0}$.

For the inductive case (the abstract path is $n > 0$ steps), let $(q_{1D}, g_1, Q_1) \xrightarrow{t}_\alpha (q_D, g, Q)$ be the last step of that path where $t \in T_C$ (the case $t \in T_D$ yields $Q_1 = Q$ and is easy). Moreover we assume t has q_C and $q_C{}'$ as source and target state, respectively. It follows from the definitions of $\rightarrow_\alpha$ that:

$$Q = \alpha_\rho\left(\{\boldsymbol{p'} \mid \exists \boldsymbol{p} \in \gamma(Q_1) \colon \boldsymbol{p} \geq \boldsymbol{q_C} \wedge \boldsymbol{p'} = \boldsymbol{p} - \boldsymbol{q_C} + \boldsymbol{q_C}{}'\}\right) \quad . \tag{1}$$

Because the component of Q_D and $\mathcal{G} \cup \{\#\}$ pose no difficulty, the rest of the proof focuses on populations. From (B) and (1), it follows that $Q = Q_1 \cup \{q_C{}'\}$. Next, given $\boldsymbol{p} \in \gamma(Q)$, define population $\boldsymbol{pp}$ to be such that $\boldsymbol{pp} \geq \boldsymbol{p}$, $\boldsymbol{pp} \in \gamma(Q)$, and $\boldsymbol{pp} \geq \boldsymbol{q_C}{}'$. Such $\boldsymbol{pp}$ exists since $q_C{}' \in Q$. In particular, we set $\boldsymbol{pp}(q_C{}') = 1$ if $q_C{}' \notin Q_1$. Now define $\boldsymbol{p_1}$ as $\boldsymbol{pp} - \boldsymbol{q_C}{}' + \boldsymbol{q_C}$. We find that $\boldsymbol{p_1}$ is well-defined and $\boldsymbol{p_1} \in \gamma(Q_1)$. By induction hypothesis, it follows that there exists $\hat{\boldsymbol{p}}_\mathbf{1} \in \gamma(Q_1)$ such that $\hat{\boldsymbol{p}}_\mathbf{1} \geq \boldsymbol{p_1}$ and $\hat{\boldsymbol{p}}_\mathbf{1}$ is reachable from X_0 in TS. We then conclude that $\hat{\boldsymbol{p}}_\mathbf{1} - \boldsymbol{q_C} + \boldsymbol{q_C}{}'$ is defined since $\hat{\boldsymbol{p}}_\mathbf{1} \geq \boldsymbol{p_1} \geq \boldsymbol{q_C}$, hence that $(q_{1D}, g_1, \hat{\boldsymbol{p}}_\mathbf{1}) \xrightarrow{t} (q_D, g, \hat{\boldsymbol{p}}_\mathbf{1} - \boldsymbol{q_C} + \boldsymbol{q_C}{}')$ and we are done. □

Proof (of Lemma 3). ($\Rightarrow$) Assume the cycle is realizable. Then there is an infinite path $c_0 \xrightarrow{t_1} c_1 \xrightarrow{t_2} c_2 \cdots$ of TS such that $c_0 \in \gamma(a_0), \ldots, c_{n-1} \in \gamma(a_{n-1})$ and $c_k \in \gamma(a_{(k \mod n)})$ for every $k \geq n$, and the transitions match. Let $\boldsymbol{p}_i$ be the population of c_i. Since the number of contributors of a population remains constant across transitions, we have $|\boldsymbol{p}_i| = |\boldsymbol{p}_0|$ for every $i \geq 0$. Since there are only finitely many different populations of a given size, by the pigeonhole principle there exist $k_1 < k_2$ such that $\boldsymbol{p}_{k_1 n} = \boldsymbol{p}_{k_2 n}$. Since

$$\boldsymbol{p}_{k_2 n} = \boldsymbol{p}_{k_1 n} + \sum_{i=(k_1 n)+1}^{k_2 n} \Delta(t_i) \quad ,$$

the sum on the right is equal to $\mathbf{0}$. Since

$$\sum_{i=(k_1 n)+1}^{k_2 n} \Delta(t_i) = (k_2 - k_1) \sum_{i=1}^{n} \Delta(t_i)$$

the lemma follows.

($\Leftarrow$) Let $a_i = (q_{Di}, g_i, Q_i)$. Let $c_0 = (q_{0D}, g_0, n\boldsymbol{q_0})$, and let $c_i = (q_{Di}, g_i, \boldsymbol{p}_i)$ for every $1 \leq i \leq n$, where $\boldsymbol{p}_i = n\boldsymbol{q_0} + \sum_{k=1}^{i-1} \Delta(t_k)$. Then $\boldsymbol{p}_i$ is a population for every $0 \leq i \leq n$, since $|\sum_{k=1}^{i-1} \Delta(t_k)| \leq n$. Moreover $c_0 \xrightarrow{t_1} c_1 \xrightarrow{t_2} c_2 \cdots \xrightarrow{t_n} c_n$ is a path of TS and, since $\sum_{i=1}^{n} \Delta(t_i) = \mathbf{0}$, we have $c_n = c_0$. So the TS cycle $c_0 \xrightarrow{t_0} c_1 \cdots \xrightarrow{t_n} c_n$ can be iterated arbitrarily often, and so the cycle $a_0 \xrightarrow{t_1}_\alpha a_1 \xrightarrow{t_2}_\alpha a_2 \cdots \xrightarrow{t_n}_\alpha a_n$ is realizable. □

Proof (of Theorem 1). We start with the upper bound and describe a high-level algorithm. After showing its soundness and completeness, we establish its time complexity looking at a more detailed version. The algorithm goes as follows:

- 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 $\ell \leq |Q_C|$.
- compute the set $\mathcal{Q}$ of abstract configurations $Q_D \times \mathcal{G} \cup \{\#\} \times \{\{q_{0C}\}, Q_1, \ldots, Q_\ell\}$ and compute the set Δ of abstract transitions $(q_D, g, Q) \xrightarrow{t} (q'_D, g', Q') \in \alpha_\rho TS$ such that $(q_D, g, Q), (q'_D, g', Q') \in \mathcal{Q}$.
- guess an accepting abstract configuration of $a \in \mathcal{Q}$, that is $a = (q_D, g, Q)$ and q_D is accepting in D.
- check a is reachable from the initial abstract configuration $(q_{0D}, \#, \{q_{0C}\})$.
- check there exists a path $a_0 \xrightarrow{t_1}_\alpha a_1 \ldots a_{n-1} \xrightarrow{t_n}_\alpha a_n$ where $n \geq 1$, $a_0 = a_n = a$ and $\sum_{i=1}^{n} \Delta(t_i) = \mathbf{0}$.

Soundness. It is easy to see that this algorithm computes a cycle of $\alpha_\rho TS$ which satisfies the assumptions of Lemma 3. So it is realizable. Further, it visits some accepting state Q_D infinitely often and it is reachable. Lemma 2 then concludes the case.

Completeness. Let $c_0 \xrightarrow{t_1} c_1 \xrightarrow{t_2} c_2 \cdots$ be an ω-path of TS on which the Büchi automaton accepts. Because the number of contributors in the population along that path stays constant, there exist, by the pigeonhole principle, two positions $i_1 < i_2$ such that $c_{i_1} = c_{i_2}$, and they are accepting. Take two such positions. Clearly, we have $\sum_{k=i_1}^{i_2} \Delta(t_i) = \mathbf{0}$, hence $c_0 \xrightarrow{t_1} c_1 \xrightarrow{t_2} c_2 \cdots (c_{i_1} \xrightarrow{t_{i_1+1}} c_{i_1+1} \cdots \xrightarrow{t_{i_2}})^\omega$ is also an ω-path of TS that is accepting. We know from (A) that every ω-path $c_0 \xrightarrow{t_1} c_1 \xrightarrow{t_2} c_2 \cdots$ of TS has a counterpart in $\alpha_\rho TS$, that is, there exists an ω-path $a_0 \xrightarrow{t_1}_\alpha a_1 \xrightarrow{t_2}_\alpha \cdots$ in $\alpha_\rho TS$ such that $c_i \in \gamma(a_i)$ for all $i \geq 0$. Note also that (i) the sequence of transitions fired along these two paths is the same, and (ii) the state of D and A coincide in c_i and a_i for all i, hence the concrete path is accepting iff the abstract is. Because the number of abstract configurations of $\alpha_\rho TS$ is finite, $a_0 \xrightarrow{t_1}_\alpha a_1 \xrightarrow{t_2}_\alpha a_2 \cdots$ is necessarily a lasso. Next, for each $i \geq 0$, let $a_i = (q_{Di}, g_i, Q_i)$. We know from (B) that $Q_i \subseteq Q_{i+1}$ and that there is an index $i \leq K$ after which the Q_i stabilize, that is, $Q_i = Q_{i+k}$ holds for every $k \geq 0$. (Recall K is an upper bound on the number of abstract configurations.) So to find this lasso, it is sufficient to guess correctly the finite sequence of Q_i sets. Because no two sets repeat and the sequence is strictly increasing, the length of the sequence is bounded by $|Q_C|$. Therefore if such finite sequence exists, the algorithm guesses it. The construction of the abstract configurations $\mathcal{Q}$ and abstract transitions of $\alpha_\rho TS$ with both ends in $\mathcal{Q}$ guarantees that the lasso is captured. Hence completeness follows from the search of a cycle of $\mathbf{0}$ weight that is reachable.

Polynomial time. We show the algorithm belongs to NP. First, because the sequence guessed is no longer than $|Q_C|$, the guess can be done in polynomial time. Next, because this is the only challenging step, we give a NP algorithm to decide there exists a path $a_0 \xrightarrow{t_1}_\alpha a_1 \ldots a_{n-1} \xrightarrow{t_n}_\alpha a_n$ where $n \geq 1$ $a_0 = a_n = a$ and such that $\sum_{i=1}^{n} \Delta(t_i) = \mathbf{0}$. The algorithm performs the following steps.

– compute a FSA $A_a^{\circlearrowleft}$ over the alphabet $\delta_D \cup \delta_C$ whose states and transitions are given by Q and Δ and such that its initial and final states are given by a and $\{a\}$, respectively;
– use the polynomial construction of Seidl *et al.* [14] to compute an (existential) Presburger formula Ω for the Parikh image of $L(A_a^{\circlearrowleft})$. The free variables of Ω are in one-to-one correspondence with the transitions of $\delta_D \cup \delta_C$. We thus adopt the convention that x_t denotes the variable corresponding to transition $t \in \delta_D \cup \delta_C$.
– 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.
– check satisfiability of Ω'. This step is in NP because satisfiability of an existential Presburger formula is in NP [11].

Hardness. NP-hardness follows from the NP-hardness of the safety problem [9], which asks given a finite-state machine D for the leader $\mathcal{D}$ and C for the contributor C—both of which are languages of finite words—whether there exists a word of the $(\mathcal{D}, C)$-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, 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 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 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. □

6.2 Proofs of Section 4

Proof (of Theorem 2). 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 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 $\alpha_\rho TS$.

Next, we guess an abstract configuration q and a stack symbol γ. We check if there is a word that takes $q_0\bot$ to $q\gamma w$ 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 run on u from the starting

[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.

configuration $q\gamma$ 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 [11].

To see that the algorithm is sound, notice that the algorithm accepts if there is a (pushdown) lasso such that the cyclic part has $\mathbf{0}$ weight. 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 lasso and then execute the cycle to come back to the same configuration as the starting point of the lasso. This lasso can be pumped infinitely often to produce an accepting run of the Büchi PDM.

For completeness, we use Lemma 4 to deduce that from an accepting run of the Büchi PDM, we can find a lasso-shaped path as defined above. By a similar pigeonhole argument as that of Lemma 3, we conclude that we can find a cyclic path whose weight is $\mathbf{0}$. □

6.3 Proofs of Section 5

Proof (of Theorem 3). 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 $\rho = q_0w_0 \xrightarrow{(v)_1} q_1w_1 \xrightarrow{(v)_2} \cdots$, 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 the initial state of P_k is in the desired form. For the definition of u_{i+1}, assuming that u_i is already defined, we consider three cases:

- The transition $q_iw_i \xrightarrow{(v)_{i+1}} q_{i+1}w_{i+1}$ 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 $q_iw_i \xrightarrow{(v)_{i+1}} q_{i+1}w_{i+1}$ pushes a symbol γ, and $|ap(i)u_i| < k$.
  Then $q_iap(i)u_i \xrightarrow{(v)_{i+1}} q_{i+1}\gamma ap(i)u_i$ 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 $|u_i| > 0$, if $|u_i| = 0$, then $|ap(i)| = k$, and more importantly $ds(i)$ is the largest proper suffix of all the $(w_j)_{j\geq i}$, and since w_i is a proper suffix of w_{i+1}, $ds(i)$ is also the largest proper suffix of all the $(w_j)_{j\geq i+1}$, therefore $\gamma ap(i) = ap(i+1)$, so $|ap(i+1)| = k+1$ contradicting the hypothesis that the run is effectively k-bounded.

  We can therefore write $u_i = u_i'\gamma'$. 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 $u_{i+1} = u_i'$. 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 $|ap(i+1)u_{i+1}| \leq k$, 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 such that $|w_i| \leq k$ 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. □

Proof (of Lemma 5, the Distributing Lemma). 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 $\Sigma_C \cap \Sigma_{\mathcal{D}} = \emptyset$, 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 exits an interleaving function, i.e. a bijection $\mathcal{I} : \coprod_{w\in u\oplus M} \mathrm{dom}(w) \to \mathrm{dom}(s)$, that assigns to each position in each word 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 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 $I(w_1, 1) = 2$, $I(w_1, 2) = 3$, $I(w_1, 3) = 5$, $I(w_2, 1) = 4$, $I(w_2, 2) = 5$.

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 ψ be the embedding function of the distribution R. 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 a position of s such that $\mathcal{I}(v, j) = i$ for some $j \in \mathrm{dom}(v)$ (so, loosely speaking, position j in the interleaving s comes from the word $v \in M$). 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 $u \oplus M \ominus v \oplus M_R$. □

Proof of the Boundedness Lemma. Before proving Lemma 6 and the Boundedness lemma we give an example of two distributions of a finite run that decrease the effective stack height, one of them being moreover synchronized.

Example 5. Consider the two distributions R and S of $\rho = r_a r_b r_b r_c r_c r_c$ in Example 3. Further assume that the PDM P has one single state p, stack symbols $\{\bot, \alpha_\rho\}$ such that the three rules r_a, r_b and r_c are given by $r_a \colon p\bot \to p\alpha_\rho\bot$, $r_b \colon p\alpha_\rho \to p\alpha_\rho\alpha_\rho$, and $r_c \colon p\alpha_\rho \to p$. 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 $c(\rho, 6) = p\bot$ has effective stack height 1, but $c_\psi(\rho'_2, 6) = p\alpha\bot \neq c(\rho, 6)$. The distribution S is $(Z, 3)$-bounded for every $1 \le Z \le 6$ and synchronized. Remark 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 $\bot$ thus effective stack height is 1.

Proof (of Lemma 6). For convenience, when we want to denote that, say, in a run ρ the configurations reached after $(\rho)_{1..i}$ and $(\rho)_{1..j}$ are c and c', we write $\rho = (\rho)_{1..i}\,[c]\,(\rho)_{i+1..j}\,[c']\,(\rho)_{j+1..\infty}$.

	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 of the distributions of Example 3.

We construct a (Z, N)-bounded and synchronized distribution $\{\rho_a, \rho_b\}$ of ρ. Let $\alpha_{N+1}\alpha_N \cdots \alpha_1 w_0$ be the stack content of $c(\rho, Z)$. Define $\{\acute{p}_1, \grave{p}_1, \acute{p}_2, \grave{p}_2, \ldots, \acute{p}_N, \grave{p}_N\} \subseteq \mathrm{dom}(\rho)$ such that for each i, $1 \leq i \leq N$ we have $c(\rho, \acute{p}_i)$ and $c(\rho, \grave{p}_i)$ are the configurations immediately after the symbol α_i in $c(\rho, Z)$ is pushed, respectively popped and such that the stack content of each configuration between $\acute{p}_i$ (included) and $\grave{p}_i$ (excluded) equals $w_p\alpha_i\alpha_{i-1}\cdots\alpha_1 w_0$ for some $w_p \in \Gamma_C^*$. We get $c(\rho, \acute{p}_i) = q_i\alpha_i\alpha_{i-1}\ldots\alpha_0 w_0$ and $c(\rho, \grave{p}_i) = q'_i\alpha_{i-1}\ldots\alpha_0 w_0$ for some $q_i, q'_i \in Q_C$. Observe that the following holds: $\acute{p}_1 < \cdots < \acute{p}_{N-1} < \acute{p}_N < Z < \grave{p}_N < \grave{p}_{N-1} < \cdots < \grave{p}_1$.

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 $w_1 = \alpha_{j_1-1}\cdots\alpha_1$, $w_2 = \alpha_{j_2-1}\cdots\alpha_{j_1}$ and $w_3 = \alpha_{j_3-1}\cdots\alpha_{j_2}$, we have:

$$\rho = (\rho)_{1..\acute{p}_{j_1}}\,[q\alpha w_1]\,(\rho)_{\acute{p}_{j_1+1}..\acute{p}_{j_2}}\,[q\alpha w_2 w_1]\,(\rho)_{\acute{p}_{j_2+1}..\acute{p}_{j_3}}\,[q\alpha w_3 w_2 w_1]$$
$$(\rho)_{\acute{p}_{j_3+1}..\grave{p}_{j_3}}\,[q' w_3 w_2 w_1]\,(\rho)_{\grave{p}_{j_3+1}..\grave{p}_{j_2}}\,[q' w_2 w_1]\,(\rho)_{\grave{p}_{j_2+1}..\grave{p}_{j_1}}\,[q' w_1]\,(\rho)_{\grave{p}_{j_1+1}..\infty}\ .$$

Now define ρ_a from ρ by simultaneously deleting $(\rho)_{\acute{p}_{j_1+1}..\acute{p}_{j_2}}$ and $(\rho)_{\grave{p}_{j_2+1}..\grave{p}_{j_1}}$. We similarly define ρ_b by deleting $(\rho)_{\acute{p}_{j_2+1}..\acute{p}_{j_3}}$ and $(\rho)_{\grave{p}_{j_3+1}..\grave{p}_{j_2}}$. The following shows that ρ_a defines a legal run since it is given by

$$(\rho)_{1..\acute{p}_{j_1}}\,[q\alpha w_1]\,(\rho)_{\acute{p}_{j_2+1}..\acute{p}_{j_3}}\,[q\alpha w_3 w_1](\rho)_{\acute{p}_{j_3+1}..\grave{p}_{j_3}}\,[q' w_3 w_1]\,(\rho)_{\grave{p}_{j_3+1}..\grave{p}_{j_2}}\,[q' w_1]\,(\rho)_{\grave{p}_{j_1+1}..\infty}\ .$$

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

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

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

$\{\rho_a, \rho_b\}$ is a (Z, N)-bounded and synchronized distributions 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 $i \leq \acute{p}_{j_1}$ and every $i \geq \grave{p}_{j_1}$. Since all configurations of ρ of effective stack height 1 are in these two areas, the distribution is synchronized. □

In order to prove the Boundedness Lemma (Lemma 7), we introduce a definition 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.

Definition 5. *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 the distribution $R \ominus \{\rho'\} \oplus R', \psi''$ of ρ, where the embedding function ψ'' is defined as follows:*

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

The following lemma proves that the composition of distributions is not ill-defined, that is, that the composition of distributions is indeed a distribution of ρ.

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.

- $(\rho'')_i = (\rho)_{\psi''(\rho'', i)}$.
  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 ψ' is a distribution, $(\rho'')_i = (\rho')_{\psi'(\rho'', i)}$. Since ψ is a distribution $(\rho')_j = (\rho)_{\psi(\rho', j)}$. 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 $\psi_{\rho'}$ 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 $\psi_{\rho'}(\rho'', i) < \psi_{\rho'}(\rho'', j)$ holds for every $i < j$. Then, by monotonicity of ψ, we obtain $\psi(\rho', \psi'(\rho'', i)) < \psi(\rho', \psi'(\rho'', j))$, and so $\psi''(\rho'', i) < \psi''(\rho'', j)$. □

Lemma 9. *Let σ be a run of D, and let $M \oplus \{\rho\}$ be a multiset of runs of C compatible with σ. Let R, ψ be a $(Z, N)-$bounded synchronized distribution of ρ. For every $\rho' \in R$, let $R_{\rho'}, \psi_{\rho'}$ be a $(\psi(\rho', Z) + 1, N)-$bounded synchronized distribution of ρ'. Then $\bigoplus_{\rho' \in R} R_{\rho'}$ is a $(Z + 1, N)-$bounded synchronized distribution of ρ.*

Proof. By repeated application of Lemma 8, ρ can be distributed to $\bigoplus_{\rho' \in R} R_{\rho'}$. Let Ψ be the corresponding embedding function, obtained also by repeated application of Lemma 8. We have to prove that $\bigoplus_{\rho' \in R} R_{\rho'}, \Psi$ is a synchronized and $(Z+1, N)$-bounded distribution.

We first show that $\bigoplus_{\rho' \in R} R_{\rho'}, \Psi$ is synchronized. Assume that the effective stack height of $c(\rho, i)$ is 1. Let $\rho'' \in \bigoplus_{\rho' \in R} R_{\rho'}$, and let ρ' be the element of R it corresponds to.

We have to show that $last_\Psi(\rho'', i) = last_{\psi_{\rho'}}(\rho'', last_\psi(\rho', i))$ (which we easily deduce from the fact that $last_\Psi(\rho'', i) = last_\Psi(\rho'', \psi(\rho', last_\psi(\rho', i)))$. Since ψ is synchronized, we deduce that $c_\Psi(\rho'', i)$ is the same configuration as $c_\psi(\rho', i) = c(\rho', last_\psi(\rho', i))$ and has effective stack height 1. Since $\psi_{\rho'}$ is synchronized, $c(\rho', last_\psi(\rho', i))$ is the same configuration as $c_{\psi_{\rho'}}(\rho'', last_\psi(\rho', i)) = c(\rho'', last_{\psi_{\rho'}}(\rho'', last_\psi(\rho', i)))$.

We now prove that $\bigoplus_{\rho' \in R} R_{\rho'}, \Psi$ is $(Z+1, N)$-bounded. Again, we pick $\rho'' \in \bigoplus_{\rho' \in R} R_{\rho'}$, and let ρ' be the corresponding element of R. We have to show that $c_\Psi(\rho'', i)$ has effective stack height at most N for every $0 \leq i \leq Z+1$. Since $last_\psi(\rho', Z+1) \leq last_\psi(\rho', Z) + 1$, by monotonicity we deduce that $last_\Psi(\rho'', Z+1) \leq last_{\psi_{\rho'}}(\rho'', last_\psi(\rho', Z) + 1)$ and we are done. □

Proof (of Lemma 7, the Boundedness Lemma). The proof is by induction on Z. If $Z = 1$ then $R_0 = \{\rho\}$, because the first configuration of a run has effective height at most 2 (if the first rule was a push, and that symbol will be later popped). Since by definition $N \geq 2$, we get that ρ is $(1, N)$-bounded.

For the induction step, assume that some distribution R_{Z-1} of ρ is $(Z-1, N)$-bounded and synchronized, and let ψ be the embedding function of the distribution. 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 6 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 6. We give an example showing that that the bound on the effective stack height used in the Boundedness Lemma is optimal: for any smaller bound, the lemma is no longer true.

We build a PDM with $k_1 + k_2 + 1$ states and with stack alphabet $\{\bot\} \cup [1, k_3]$, where k_1, k_2, k_3 are distinct prime numbers. With the k_1 first states, we build a circuit that pushes the word $(1 \ldots k_3)^{k_1}$ onto the stack. After that, the PDM leaves this circuit, and enters another one, consisting of k_2 states, that pops k_2 stack symbols. The PDM can only leave this circuit from its first state, and only when $\bot$ is the topmost stack symbol; if and when this condition holds, the PDM moves to the last state, from where it writes victory in the store. It should be clear that, in order to reach the last state, the stack of the PDM must reach a height of at least $(1 + k_1 k_2 k_3)$ symbols. Therefore, no run reaching the last state can be distributed into runs exhibiting a lower effective stack height.

We now show that we can 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, for at least one of them reaching 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. If 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, then if there is only one contributor, he 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. □

Proof of the Reduction Theorem. Finally, we give the proof of the Reduction Theorem.

Proof (of Theorem 4). 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 $I(w, i)$ is the position of s at which we find the i-th letter of w, and that $I(v, j)$ is the position of s at which we find the j-th letter of v, for every $v \in M$.

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 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 betwen positions X and Y can be pumped. This yields a word $w_1w_2^\omega \in L(D)$ (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 $s_1s_2^\omega$, 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 run of D, by Proposition 1 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 of the Büchi automaton A, we can further extract from this sequence an infinite subsequence such that between any two positions an accepting state of A is visited. Let (b_i) denote the image of this last infinite sequence by $\mathcal{I}$. That is, (b_i) denotes the infinite sequence of positions of s obtained by the procedure above.

Now from (b_i) we extract a subsequence (c_i) 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 1, every run of R reaches infinitely often such configurations, (c_i) 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, $\mathcal{Z}$ is the position of the t-th element of the sequence (c_i)). 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 7, 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 $R_\rho(q_{\rho,i}) = \{c_{\psi_\rho}(\tau, q_{\rho,i}) \mid \tau \in R_\rho\}$ 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 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), 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_ρ the common support of $\alpha_\rho(l)$ and $\alpha_\rho(r)$. For every $a \in \alpha_\rho$, we find two runs ρ^l_a and ρ^r_a in R_ρ such that $c_{\psi_\rho}(\rho^l_a, q_{\rho,l})$ and $c_{\psi_\rho}(\rho^r_a, q_{\rho,r})$ have pruned configuration a.

Now we define a new distribution of ρ to a multiset $R_\rho \oplus \{\rho_{a,a'} \mid a, a' \in \alpha_\rho\}$ 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^l_a up to position $\psi_\rho(\rho^l_a, p_{\rho,l})$, and then as $\rho^r_{a'}$ from $\psi_\rho(\rho^r_{a'}, p_{\rho,l})$. Formally, ψ'_ρ is the same as ψ_ρ over each $\tau \in R_\rho$, and $\psi'_\rho(\rho_{a,a'}, i) = \psi_\rho(\rho^l_a, i)$ when $i \leq \psi_\rho(\rho^l_a, p_{\rho,l})$ and $\psi'_\rho(\rho_{a,a'}, i) = \psi_\rho(\rho^r_a, i - \psi_\rho(\rho^l_a, p_{\rho,l}) + \psi_\rho(\rho^r_{a'}, p_{\rho,l}))$ when $i > \psi_\rho(\rho^l_a, p_{\rho,l})$. Observe that since $c(\rho, p_{\rho,l})$ has effective stack height 1, it is exactly the same configuration as $c_{\psi_\rho}(\rho^l_a, p_{\rho,l})$ and $c_{\psi_\rho}(\rho^r_{a'}, p_{\rho,l})$. 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 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 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 betweeen 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) 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 has effective stack height 1 at c_l and c_r, and is N-bounded 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. Denoting $\ell_\tau = (\tau)_{c_{\psi'_\rho}(\tau, p_{\rho,l})+1..c_{\psi'_\rho}(\tau, p_{\rho,r})}$, we get that the multiset $\{(\tau)_{1..c_{\psi'_\rho}(\tau,p_{\rho,l})}\ell_\tau \ell_{\mu(\tau)}\ell_{\mu^2(\tau)} \dots \mid \tau \in R'_\rho, \rho \in R\}$ is a multiset of N-bounded runs, that is compatible with $w^{\circlearrowleft}$. This concludes the proof. □