Checking Qualitative Liveness Properties of Replicated Systems with Stochastic Scheduling[⋆]

Michael Blondin[1], Javier Esparza[2], Martin Helfrich[2], Antonín Kučera[3], and Philipp J. Meyer[2]

[1] Université de Sherbrooke, Canada
michael.blondin@usherbrooke.ca
[2] Technical University of Munich, Germany
{esparza,helfrich,meyerphi}@in.tum.de
[3] Masaryk University, Czechia
tony@fi.muni.cz

Abstract. We present a sound and complete method for the verification of qualitative liveness properties of replicated systems under stochastic scheduling. These are systems consisting of a finite-state program, executed by an unknown number of indistinguishable agents, where the next agent to make a move is determined by the result of a random experiment. We show that if a property of such a system holds, then there is always a witness in the shape of a *Presburger stage graph*: a finite graph whose nodes are Presburger-definable sets of configurations. Due to the high complexity of the verification problem (non-elementary), we introduce an incomplete procedure for the construction of Presburger stage graphs, and implement it on top of an SMT solver. The procedure makes extensive use of the theory of well-quasi-orders, and of the structural theory of Petri nets and vector addition systems. We apply our results to a set of benchmarks, in particular to a large collection of population protocols, a model of distributed computation extensively studied by the distributed computing community.

Keywords: parameterized verification · liveness · stochastic systems.

1 Introduction

Replicated systems consist of a fully symmetric finite-state program executed by an unknown number of indistinguishable agents, communicating by rendez-vous or via shared variables [36,12,42,14]. Examples include distributed protocols and

⋆ Michael Blondin is supported by a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Fonds de recherche du Québec – Nature et technologies (FRQNT). Javier Esparza, Martin Helfrich and Philipp J. Meyer have received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 787367 (PaVeS). Antonín Kučera is supported by the Czech Science Foundation, grant No. 18-11193S.

multithreaded programs, or abstractions thereof. The communication graph of replicated systems is a clique. They are a special class of *parameterized systems*, i.e., infinite families of systems that admit a finite description in some suitable modeling language. In the case of replicated systems, the (only) parameter is the number of agents executing the program.

Verifying a replicated system amounts to proving that an infinite family of systems satisfies a given property. This is already a formidable challenge, but it is made even more difficult by the fact that correctness problems for distributed protocols require to check liveness properties against a class of schedulers. Indeed, liveness properties are known to be harder to verify than safety properties, and scheduling adds additional problems.

We address the verification of liveness properties of replicated systems with stochastic scheduling. Loosely speaking, stochastic schedulers select the set of agents that should execute the next action as the result of a random experiment. Stochastic scheduling often appears in distributed protocols, and in particular also in population protocols—a model much studied in distributed computing with applications in computational biology[4]—that supplies many of our case studies [8,54]. Under stochastic scheduling, the semantics of a replicated system is an infinite family of finite-state Markov chains. In this work, we study *qualitative* liveness properties, stating that the infinite runs starting at configurations satisfying a precondition almost surely reach and stay in configurations satisfying a postcondition. In this case, whether the property holds or not depends only on the topology of the Markov chains, and not on the concrete probabilities.

We present a formal model of replicated systems, based on multiset rewriting, that encompasses both shared variables and multiway synchronization, and introduce a sound and complete verification method, called *Presburger stage graphs*. These are directed acyclic graphs whose nodes are Presburger formulas representing possibly infinite inductive sets of configurations, i.e., sets of configurations closed under reachability. Such a graph has the property that successors of a node $\mathcal{S}$ represent sets of configurations that will eventually be visited from configurations of $\mathcal{S}$. A stage graph supplies a witness of this fact in the form of a *Presburger certificate*, a sort of ranking function expressible in Presburger arithmetic. Completeness, meaning that for every property that holds there is a stage graph proving that it holds, follows from deep results of the theory of vector addition systems (VASs) [47,48,49].

Unfortunately, the theory of VASs also shows that, while the verification problems we consider are decidable, they have non-elementary computational complexity [28]. As a consequence, verification techniques that systematically explore the space of possible stage graphs for a given property are bound to be very inefficient. For this reason, we design an incomplete but efficient algorithm for the computation of stage graphs. Inspired by theoretical results, the algorithm combines a solver for linear constraints with some elements of the theory of well-structured systems [34,2]. We report on the performance of this algorithm for a large number of case studies. In particular, it automatically verifies many

[4] Under the name of *chemical reaction networks*.

standard population protocols described in the literature [23,7,4,17,18,15,26], and liveness properties of distributed algorithms for leader election and mutual exclusion [45,57,60,39,37,35,55,3].

Related work. The parameterized verification of replicated systems was first studied in [36], where it was shown that they can be modeled as counter systems. This allows to apply many efficient techniques. Most are inherently designed for safety properties [43,32,10,19], and some can also handle fair termination properties [33], but none of these works can handle stochastic scheduling. To the best of our knowledge, the only works studying parameterized verification of liveness properties under our notion of stochastic scheduling are those dealing with the verification of population protocols. For *fixed* populations this problem can be tackled with standard probabilistic model checking [61,11], and early work on the automatic verification of population protocols follows this approach [23,26,56,59]. Subsequently, an algorithm and a tool for the *parameterized* verification of population protocols were described in [17,16], and a first version of stage graphs was introduced in [18] for analyzing the expected termination time of population protocols. In essence, we overhaul the framework of [18] for liveness verification, drawing inspiration from the safety verification technology of [17,16]. Compared to [17,16], our approach is not a priori limited to a specific subclass of protocols, and captures models beyond population protocols. Furthermore, our new techniques for computing Presburger certificates essentially subsume the procedure of [17]. In comparison to [18], we provide the first completeness and complexity results for stage graphs, which we redesigned for liveness verification.

There is also a large body of work on parameterized verification of systems via a cutoff property: a given system satisfies a specification φ for any number of agents iff it satisfies φ for any number of agents below some threshold called the cutoff (e.g., see [21,29,25,42,5], and [14] for a comprehensive survey). This approach can also be applied to systems with an array or ring communication structure, but it requires the existence and effectiveness of a cutoff, which is not the case in our setting. Other important parameterized verification techniques are regular model checking [20,1] and automata learning [6]. The classes of communication structures they can handle are orthogonal to ours: arrays and rings for regular model checking and automata learning, and cliques in our work. Regular model checking and learning have recently been employed to verify safety properties [24], liveness properties under arbitrary schedulers [50] and termination under finitary fairness [46]. The classes of schedulers considered in [50,46] are incomparable to ours: arbitrary schedulers in [50], and finitary-fair schedulers in [46]. Another significant difference is that these works are based on symbolic state-space exploration, while our techniques are based on automatic construction of invariants and ranking functions [14].

2 Preliminaries

Let $\mathbb{N}$ denote $\{0, 1, \ldots\}$ and let E be a finite set. A *unordered vector* over E is a mapping $V \colon E \to \mathbb{Z}$. In particular, a *multiset* over E is an unordered vector

$M \colon E \to \mathbb{N}$ where $M(e)$ denotes the number of occurrences of e in M. The sets of all unordered vectors and multisets over E are respectively denoted $\mathbb{Z}^E$ and $\mathbb{N}^E$. Vector addition, subtraction and comparison are defined componentwise. The *size* of a multiset M is denoted $|M| = \sum_{e \in E} M(e)$. We let $E^{\langle k \rangle}$ denote the set of all multisets over E of size k. We sometimes describe multisets using a set-like notation, e.g. $M = \llbracket f, g, g \rrbracket$ or equivalently $M = \llbracket f, 2 \cdot g \rrbracket$ is such that $M(f) = 1$, $M(g) = 2$ and $M(e) = 0$ for all $e \notin \{f, g\}$.

Presburger arithmetic. Let X be a set of variables. The set of formulas of *Presburger arithmetic* over X is the result of closing atomic formulas, as defined in the next sentence, under Boolean operations and first-order existential quantification. Atomic formulas are of the form $\sum_{i=1}^{k} a_i x_i \sim b$, where a_i and b are integers, x_i are variables and $\sim$ is either $<$ or $\equiv_m$, the latter denoting the congruence modulo m for any $m \geq 2$. Formulas over X are interpreted on $\mathbb{N}^X$. Given a formula ϕ of Presburger arithmetic, we let $\llbracket \phi \rrbracket$ denote the set of all multisets satisfying ϕ. A set $E \subseteq \mathbb{N}^X$ is a *Presburger set* if $E = \llbracket \phi \rrbracket$ for some formula ϕ.

2.1 Replicated systems

A *replicated system* over Q of arity n is a tuple $\mathcal{P} = (Q, T)$, where $T \subseteq \bigcup_{k=0}^{n} Q^{\langle k \rangle} \times Q^{\langle k \rangle}$ is a *transition relation* containing the set of *silent* transitions $\bigcup_{k=0}^{n} \{(\boldsymbol{x}, \boldsymbol{x}) \mid \boldsymbol{x} \in Q^{\langle k \rangle}\}$[5]. A *configuration* is a multiset C of states, which we interpret as a global state with $C(q)$ agents in each state $q \in Q$.

For every $t = (\boldsymbol{x}, \boldsymbol{y}) \in T$ with $\boldsymbol{x} = \llbracket X_1, X_2, \ldots, X_k \rrbracket$ and $\boldsymbol{y} = \llbracket Y_1, Y_2, \ldots, Y_k \rrbracket$, we write $X_1 X_2 \cdots X_k \mapsto Y_1 Y_2 \cdots Y_k$ and let ${}^\bullet t \overset{\text{def}}{=} \boldsymbol{x}$, $t^\bullet \overset{\text{def}}{=} \boldsymbol{y}$ and $\Delta(t) \overset{\text{def}}{=} t^\bullet - {}^\bullet t$. A transition t is *enabled* at a configuration C if $C \geq {}^\bullet t$ and, if so, can *occur*, leading to the configuration $C' = C + \Delta(t)$. If t is not enabled at C, then we say that it is *disabled*. We use the following reachability notation:

$$\begin{aligned}
C \xrightarrow{t} C' &\iff t \text{ is enabled at } C \text{ and its occurrence leads to } C', \\
C \rightarrow C' &\iff C \xrightarrow{t} C' \text{ for some } t \in T, \\
C \xrightarrow{w} C' &\iff C = C_0 \xrightarrow{w_1} C_1 \cdots \xrightarrow{w_n} C_n = C' \text{ for some } C_0, C_1, \ldots, C_n \in \mathbb{N}^Q, \\
C \xrightarrow{*} C' &\iff C \xrightarrow{w} C' \text{ for some } w \in T^*.
\end{aligned}$$

Observe that, by definition of transitions, $C \rightarrow C'$ implies $|C| = |C'|$, and likewise for $C \xrightarrow{*} C'$. Intuitively, transitions cannot create or destroy agents.

A *run* is an infinite sequence $C_0 t_1 C_1 t_2 C_2 \cdots$ such that $C_i \xrightarrow{t_{i+1}} C_{i+1}$ for every $i \geq 0$. Given $L \subseteq T^*$ and a set of configurations $\mathcal{C}$, we let

$$\begin{aligned}
\mathit{post}_L(\mathcal{C}) &\overset{\text{def}}{=} \{C' : C \in \mathcal{C}, w \in L, C \xrightarrow{w} C'\}, \\
\mathit{pre}_L(\mathcal{C}) &\overset{\text{def}}{=} \{C : C' \in \mathcal{C}, w \in L, C \xrightarrow{w} C'\}.
\end{aligned}$$

We write $\mathit{post}^*(\mathcal{C})$ and $\mathit{pre}^*(\mathcal{C})$ for $L = T^*$, and $\mathit{post}(\mathcal{C})$ and $\mathit{pre}(\mathcal{C})$ for $L = T$.

[5] In the paper, we will omit the silent transitions when giving replicated systems.

Stochastic scheduling. We assume that, given a configuration C, a probabilistic scheduler picks one of the transitions enabled at C. We only make the following two assumptions about the random experiment determining the transition: first, the probability of a transition depends only on C, and, second, every transition enabled at C has a nonzero probability of occurring. Since $C \xrightarrow{*} C'$ implies $|C| = |C'|$, the number of configurations reachable from any configuration C is finite. Thus, for every configuration C, the semantics of $\mathcal{P}$ from C is a finite-state Markov chain rooted at C.

Example 1. Consider the replicated system $\mathcal{P} = (Q, T)$ of arity 2 with states $Q = \{\mathrm{A_Y}, \mathrm{A_N}, \mathrm{P_Y}, \mathrm{P_N}\}$ and transitions $T = \{t_1, t_2, t_3, t_4\}$, where

$$\begin{aligned} t_1\colon \mathrm{A_Y}\,\mathrm{A_N} &\mapsto \mathrm{P_Y}\,\mathrm{P_N}, & \qquad t_2\colon \mathrm{A_Y}\,\mathrm{P_N} &\mapsto \mathrm{A_Y}\,\mathrm{P_Y}, \\ t_3\colon \mathrm{A_N}\,\mathrm{P_Y} &\mapsto \mathrm{A_N}\,\mathrm{P_N}, & \qquad t_4\colon \mathrm{P_Y}\,\mathrm{P_N} &\mapsto \mathrm{P_N}\,\mathrm{P_N}. \end{aligned}$$

Intuitively, at every moment in time, agents are either *Active* or *Passive*, and have output *Yes* or *No*, which corresponds to the four states of Q. This system is designed to satisfy the following property: for every configuration C in which all agents are initially active, i.e., C satisfies $C(\mathrm{P_Y}) = C(\mathrm{P_N}) = 0$, if $C(\mathrm{A_Y}) > C(\mathrm{A_N})$, then eventually all agents stay forever in the "yes" states $\{\mathrm{A_Y}, \mathrm{P_Y}\}$, and otherwise all agents eventually stay forever in the "no" states $\{\mathrm{A_N}, \mathrm{P_N}\}$. $\lhd$

2.2 Qualitative model checking

Let us fix a replicated system $\mathcal{P} = (Q, T)$. Formulas of *linear temporal logic (LTL)* on $\mathcal{P}$ are defined by the following grammar:

$$\varphi ::= \phi \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

where ϕ is a Presburger formula over Q. We look at ϕ as an atomic proposition over the set $\mathbb{N}^Q$ of configurations. Formulas of LTL are interpreted over runs of $\mathcal{P}$ in the standard way. We abbreviate $\Diamond\varphi \equiv \mathit{true}\,\mathbf{U}\,\varphi$ and $\Box\varphi \equiv \neg\Diamond\neg\varphi$.

Let us now introduce the probabilistic interpretation of LTL. A configuration C of $\mathcal{P}$ satisfies an LTL formula φ *with probability* p if $\Pr[C, \varphi] = p$, where $\Pr[C, \varphi]$ denotes the probability of the set of runs of $\mathcal{P}$ starting at C that satisfy φ in the finite-state Markov chain rooted at C. The measurability of this set of runs for every C and φ follows from well-known results [61]. The *qualitative model checking problem* consists of, given an LTL formula φ and a set of configurations $\mathcal{I}$, deciding whether $\Pr[C, \varphi] = 1$ for every $C \in \mathcal{I}$. We will often work with the complement problem, i.e., deciding whether $\Pr[C, \neg\varphi] > 0$ for some $C \in \mathcal{I}$.

In contrast to the action-based qualitative model checking problem of [30], our version of the problem is undecidable due to adding atomic propositions over configurations (see the appendix for a proof):

Theorem 1. *The qualitative model checking problem is not semi-decidable.*

It is known that qualitative model checking problems of finite-state probabilistic systems reduces to model checking of non-probabilistic systems under an adequate notion of fairness.

Definition 1. *A run of a replicated system $\mathcal{P}$ is* fair *if for every possible step $C \xrightarrow{t} C'$ of $\mathcal{P}$ the following holds: if the run contains infinitely many occurrences of C, then it also contains infinitely many occurrences of $C\,t\,C'$.*

So, intuitively, if a run can execute a step infinitely often, it eventually will. It is readily seen that a fair run of a finite-state transition system eventually gets "trapped" in one of its bottom strongly connected components, and visits each of its states infinitely often. Hence, fair runs of a finite-state Markov chain have probability one. The following proposition was proved in [30] for a model slightly less general than replicated systems; the proof can be generalized without effort:

Proposition 1 ([30, Prop. 7]). *Let $\mathcal{P}$ be a replicated system, let C be a configuration of $\mathcal{P}$, and let φ be an LTL formula. It is the case that* $\Pr[C, \varphi] = 1$ *iff every fair run of $\mathcal{P}$ starting at C satisfies φ.*

We implicitly use this proposition from now on. In particular, we define:

Definition 2. *A configuration C* satisfies φ with probability 1*, or just* satisfies *φ, if every fair run starting at C satisfies φ, denoted by $C \models \varphi$. We let $[\![\varphi]\!]$ denote the set of configurations satisfying φ. A set $\mathcal{C}$ of configurations* satisfies *φ if $\mathcal{C} \subseteq [\![\varphi]\!]$, i.e., if $C \models \varphi$ for every $C \in \mathcal{C}$.*

Liveness specifications for replicated systems. We focus on a specific class of temporal properties for which the qualitative model checking problem is decidable and which is large enough to formalize many important specifications. Using well-known automata-theoretic technology, the verification techniques we develop for this class can also be extended to all properties describable in action-based LTL, see e.g. [30].

A *stable termination property* is given by a pair $\Pi = (\varphi_{\text{pre}}, \Phi_{post})$, where $\Phi_{post} = \{\varphi^1_{\text{post}}, \ldots, \varphi^k_{\text{post}}\}$ and $\varphi_{\text{pre}}, \varphi^1_{\text{post}}, \ldots, \varphi^k_{\text{post}}$ are Presburger formulas over Q describing sets of configurations. Whenever $k = 1$, we sometimes simply write $\Pi = (\varphi_{\text{pre}}, \varphi_{\text{post}})$. The pair Π induces the LTL property

$$\varphi_{\Pi} \overset{\text{def}}{=} \Diamond \bigvee_{i=1}^{k} \Box \varphi^i_{\text{post}} \ .$$

Abusing language, we say that a replicated system $\mathcal{P}$ *satisfies* Π if $[\![\varphi_{\text{pre}}]\!] \subseteq [\![\varphi_{\Pi}]\!]$, that is, if every configuration C satisfying φ_{pre} satisfies φ_{Π} with probability 1. The *stable termination problem* is the qualitative model checking problem for $\mathcal{I} = [\![\varphi_{\text{pre}}]\!]$ and $\varphi = \varphi_{\Pi}$ given by a stable termination property $\Pi = (\varphi_{\text{pre}}, \Phi_{post})$.

Example 2. Let us reconsider the system from Example 1. We can formally specify that all agents will eventually agree on the majority output *Yes* or *No*. Let $\Pi^{\text{Y}} = (\varphi^{\text{Y}}_{\text{pre}}, \varphi^{\text{Y}}_{\text{post}})$ and $\Pi^{\text{N}} = (\varphi^{\text{N}}_{\text{pre}}, \varphi^{\text{N}}_{\text{post}})$ be defined by:

$$\varphi^{\text{Y}}_{\text{pre}} = (\text{A}_\text{Y} > \text{A}_\text{N} \land \text{P}_\text{Y} + \text{P}_\text{N} = 0), \qquad \varphi^{\text{Y}}_{\text{post}} = (\text{A}_\text{N} + \text{P}_\text{N} = 0),$$
$$\varphi^{\text{N}}_{\text{pre}} = (\text{A}_\text{Y} \leq \text{A}_\text{N} \land \text{P}_\text{Y} + \text{P}_\text{N} = 0), \qquad \varphi^{\text{N}}_{\text{post}} = (\text{A}_\text{Y} + \text{P}_\text{Y} = 0).$$

The system satisfies the property specified in Example 1 iff it satisfies Π^{Y} and Π^{N}. As an alternative (weaker) property, we could specify that the system always stabilizes to either output by $\Pi = (\varphi_{\mathrm{pre}}^{\mathrm{Y}} \lor \varphi_{\mathrm{pre}}^{\mathrm{N}}, \{\varphi_{\mathrm{post}}^{\mathrm{Y}}, \varphi_{\mathrm{post}}^{\mathrm{N}}\})$. $\lhd$

3 Stage graphs

In the rest of the paper, we fix a replicated system $\mathcal{P} = (Q, T)$ and a stable termination property $\Pi = (\varphi_{\mathrm{pre}}, \Phi_{post})$, where $\Phi_{post} = \{\varphi_{\mathrm{post}}^1, \ldots, \varphi_{\mathrm{post}}^k\}$, and address the problem of checking whether $\mathcal{P}$ satisfies Π. We start with some basic definitions on sets of configurations.

Definition 3 (inductive sets, leads to, certificates).

- *A set of configurations $\mathcal{C}$ is* inductive *if $C \in \mathcal{C}$ and $C \to C'$ implies $C' \in \mathcal{C}$.*
- *Let $\mathcal{C}, \mathcal{C}'$ be sets of configurations. We say that $\mathcal{C}$* leads to *$\mathcal{C}'$, denoted $\mathcal{C} \leadsto \mathcal{C}'$, if for all $C \in \mathcal{C}$, every fair run from C eventually visits a configuration of $\mathcal{C}'$.*
- *A* certificate *for $\mathcal{C} \leadsto \mathcal{C}'$ is a function $f \colon \mathcal{C} \to \mathbb{N}$ satisfying that for every $C \in \mathcal{C} \setminus \mathcal{C}'$, there exists an execution $C \xrightarrow{*} C'$ such that $f(C) > f(C')$.*

Note that certificates only require the existence of some executions decreasing f, not for all of them to to decrease it. Despite this, we have:

Proposition 2. *For all inductive sets $\mathcal{C}, \mathcal{C}'$ of configurations, it is the case that: $\mathcal{C}$ leads to $\mathcal{C}'$ iff there exists a certificate for $\mathcal{C} \leadsto \mathcal{C}'$.*

The proof, which can be found in the appendix, depends on two properties of replicated systems with stochastic scheduling. First, every configuration has only finitely many descendants. Second, for every fair run and for every finite execution $C \xrightarrow{w} C'$, if C appears infinitely often in the run, then the run contains infinitely many occurrences of $C \xrightarrow{w} C'$. We can now introduce stage graphs:

Definition 4 (stage graph). *A* stage graph *of $\mathcal{P}$ for the property Π is a directed acyclic graph whose nodes, called* stages*, are sets of configurations satisfying the following conditions:*

1. *every stage is an inductive set;*
2. *every configuration of $[\![\varphi_{\mathrm{pre}}]\!]$ belongs to some stage;*
3. *if $\mathcal{C}$ is a non-terminal stage with successors $\mathcal{C}_1, \ldots, \mathcal{C}_n$, then there exists a certificate for $\mathcal{C} \leadsto (\mathcal{C}_1 \cup \cdots \cup \mathcal{C}_n)$;*
4. *if $\mathcal{C}$ is a terminal stage, then $\mathcal{C} \models \varphi_{\mathrm{post}}^i$ for some i.*

The existence of a stage graph implies that $\mathcal{P}$ satisfies Π. Indeed, by conditions 2–3 and repeated application of Proposition 2, every run starting at a configuration of $[\![\varphi_{\mathrm{pre}}]\!]$ eventually reaches a terminal stage, say $\mathcal{C}$, and, by condition 1, stays in $\mathcal{C}$ forever. Since, by condition 4, all configurations of $\mathcal{C}$ satisfy some $\varphi_{\mathrm{post}}^i$, after its first visit to $\mathcal{C}$ every configuration satisfies $\varphi_{\mathrm{post}}^i$.

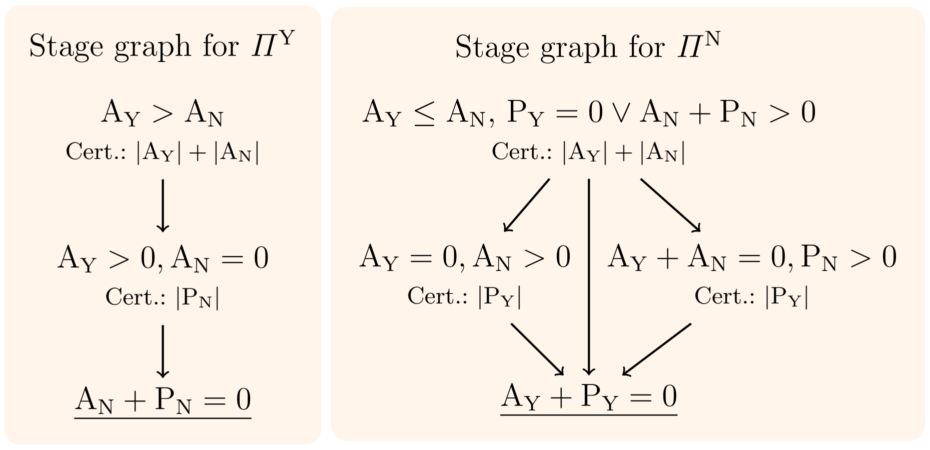
Stage graph for Π^{Y}
$\mathrm{A_Y} > \mathrm{A_N}$
Cert.: $|\mathrm{A_Y}| + |\mathrm{A_N}|$
$\mathrm{A_Y} > 0, \mathrm{A_N} = 0$
Cert.: $|\mathrm{P_N}|$
$\mathrm{A_N} + \mathrm{P_N} = 0$
Stage graph for Π^{N}
$\mathrm{A_Y} \leq \mathrm{A_N}$, $\mathrm{P_Y} = 0 \lor \mathrm{A_N} + \mathrm{P_N} > 0$
Cert.: $|\mathrm{A_Y}| + |\mathrm{A_N}|$
$\mathrm{A_Y} = 0, \mathrm{A_N} > 0$
Cert.: $|\mathrm{P_Y}|$
$\mathrm{A_Y} + \mathrm{A_N} = 0, \mathrm{P_N} > 0$
Cert.: $|\mathrm{P_Y}|$
$\mathrm{A_Y} + \mathrm{P_Y} = 0$

Fig. 1: Stage graphs for the system of Example 1.

Example 3. Figure 1 depicts stage graphs for the system of Example 1 and the properties defined in Example 2. The reader can easily show that every stage $\mathcal{C}$ is inductive by checking that for every $C \in \mathcal{C}$ and every transition $t \in \{t_1, \ldots, t_4\}$ enabled at C, the step $C \xrightarrow{t_i} C'$ satisfies $C' \in \mathcal{C}$. For example, if a configuration satisfies $\mathrm{A_Y} > \mathrm{A_N}$, so does any successor configuration. $\triangleleft$

The following proposition shows that stage graphs are a sound and complete technique for proving stable termination properties.

Proposition 3. *System $\mathcal{P}$ satisfies Π iff it has a stage graph for Π.*

Proposition 3 does not tell us anything about the decidability of the stable termination problem. To prove that the problem is decidable, we introduce Presburger stage graphs. Intuitively these are stage graphs whose stages and certificates can be expressed by formulas of Presburger arithmetic.

Definition 5 (Presburger stage graphs).

- *A stage $\mathcal{C}$ is* Presburger *if $\mathcal{C} = [\![\phi]\!]$ for some Presburger formula ϕ.*
- *A* bounded certificate *for $\mathcal{C} \rightsquigarrow \mathcal{C}'$ is a pair (f, k), where $f : \mathcal{C} \to \mathbb{N}$ and $k \in \mathbb{N}$, satisfying that for every $C \in \mathcal{C} \setminus \mathcal{C}'$, there exists an execution $C \xrightarrow{w} C'$ such that $f(C) > f(C')$ and $|w| \leq k$.*
- *A* Presburger certificate *is a bounded certificate (f, k) satisfying $f(C) = n \iff \varphi(C, n)$ for some Presburger formula $\varphi(\boldsymbol{x}, y)$.*
- *A* Presburger stage graph *is a stage graph whose stages and certificates are all Presburger.*

Using a powerful result from [31], we show that: (1) $\mathcal{P}$ satisfies Π iff it has a Presburger stage graph for Π (Theorem 2); (2) there exists a denumerable set of candidates for a Presburger stage graph for Π; and (3) there is an algorithm that decides whether a given candidate is a Presburger stage graph for Π (Theorem 3). Together, (1–3) show that the stable termination problem is semi-decidable. To obtain decidability, we observe that the complement of the stable termination problem is also semi-decidable. Indeed, it suffices to enumerate all initial configurations $C \models \varphi_{\text{pre}}$, build for each such C the (finite) graph G_C of configurations reachable from C, and check if some bottom strongly connected component $\mathcal{B}$ of G_C satisfies $\mathcal{B} \not\models \varphi^i_{\text{post}}$ for all i. This is the case iff some fair run starting at C visits and stays in $\mathcal{B}$, which in turn is the case iff $\mathcal{P}$ violates Π.

Theorem 2. *System $\mathcal{P}$ satisfies Π iff it has a Presburger stage graph for Π.*

We observe that testing whether a given graph is a Presburger stage graph reduces to Presburger arithmetic satisfiability, which is decidable [58] and whose complexity lies between 2-NEXP and 2-EXPSPACE [13]:

Theorem 3. *The problem of deciding whether an acyclic graph of Presburger sets and Presburger certificates is a Presburger stage graph, for a given stable termination property, is reducible in polynomial time to the satisfiability problem for Presburger arithmetic.*

4 Algorithmic construction of stage graphs

At the current state of our knowledge, the decision procedure derived from Theorem 3 has little practical relevance. From a theoretical point of view, the TOWER-hardness result of [28] implies that the stage graph may have non-elementary size in the system size. In practice, systems have relatively small stage graphs, but, even so, the enumeration of all candidates immediately leads to a prohibitive combinatorial explosion.

For this reason, we present a procedure to automatically *construct* (not guess) a Presburger stage graph G for a given replicated system $\mathcal{P}$ and a stable termination property $\Pi = (\varphi_{\text{pre}}, \Phi_{post})$. The procedure may *fail*, but, as shown in the experimental section, it succeeds for many systems from the literature.

The procedure is designed to be implemented on top of a solver for the existential fragment of Presburger arithmetic. While every formula of Presburger arithmetic has an equivalent formula within the existential fragment [58,27], quantifier-elimination may lead to a doubly-exponential blow-up in the size of the formula. Thus, it is important to emphasize that our procedure *never requires to eliminate quantifiers*: If the pre- and postconditions of Π are supplied as quantifier-free formulas, then all constraints of the procedure remain in the existential fragment.

We give a high-level view of the procedure (see Algorithm 1), which uses several functions, described in detail in the rest of the paper. The procedure maintains a workset WS of Presburger stages, represented by existential Presburger formulas. Initially, the only stage is an inductive Presburger overapproximation

Algorithm 1: procedure for the construction of stage graphs.

```
Input: replicated system $\mathcal{P} = (Q, T)$, stable term. property $\Pi = (\varphi_{\text{pre}}, \Phi_{post})$
Result: a stage graph of $\mathcal{P}$ for $\Pi$
1 $WS \leftarrow \{PotReach([\![\varphi_{\text{pre}}]\!])\}$
2 while $WS \neq \emptyset$ do
3     remove $\mathcal{S}$ from $WS$
4     if $\neg Terminal(\mathcal{S}, \Phi_{post})$ then
5         $U \leftarrow AsDead(\mathcal{S})$
6         if $U \neq \emptyset$ then
7             $WS \leftarrow WS \cup \{IndOverapprox(\mathcal{S}, U)\}$
8         else
9             $WS \leftarrow WS \cup Split(\mathcal{S})$
```

$PotReach([\![\varphi_{\text{pre}}]\!])$ of the configurations reachable from $[\![\varphi_{\text{pre}}]\!]$ ($PotReach$ is an abbreviation for "potentially reachable"). Notice that we must necessarily use an overapproximation, since $post^*([\![\varphi_{\text{pre}}]\!])$ is not always expressible in Presburger arithmetic[6]. We use a refinement of the overapproximation introduced in [32,17], equivalent to the overapproximation of [19].

In its main loop (lines 2–9), Algorithm 1 picks a Presburger stage $\mathcal{S}$ from the workset, and processes it. First, it calls $\text{Terminal}(\mathcal{S}, \Phi_{post})$ to check if $\mathcal{S}$ is terminal, i.e., whether $\mathcal{S} \models \varphi^i_{\text{post}}$ for some $\varphi^i_{\text{post}} \in \Phi_{post}$. This reduces to checking the unsatisfiability of the existential Presburger formula $\phi \wedge \neg\varphi^i_{\text{post}}$, where ϕ is the formula characterizing $\mathcal{S}$. If $\mathcal{S}$ is not terminal, then the procedure attempts to construct successor stages in lines 5–9, with the help of three further functions: *AsDead*, *IndOverapprox*, and *Split*. In the rest of this section, we present the intuition behind lines 5–9, and the specification of the three functions. Sections 5 to 7 present the implementations we use for these functions.

Lines 5–9 are inspired by the behavior of most replicated systems designed by humans, and are based on the notion of dead transitions:

Definition 6. *A transition of a replicated system $\mathcal{P}$ is* dead *at a configuration C if it is disabled at every configuration reachable from C (including C itself). A transition is* dead *at a stage $\mathcal{S}$ if it is dead at every configuration of $\mathcal{S}$. Given a stage $\mathcal{S}$ and a set U of transitions, we use the following notations:*

- *$Dead(\mathcal{S})$: the set of transitions dead at $\mathcal{S}$;*
- *$[\![dis(U)]\!]$: the set of configurations at which all transitions of U are disabled;*
- *$[\![dead(U)]\!]$: the set of configurations at which all transitions of U are dead.*

Observe that we can compute $Dead(\mathcal{S})$ by checking unsatisfiability of a sequence of existential Presburger formulas: as $\mathcal{S}$ is inductive, we have $Dead(\mathcal{S}) = \{t \mid \mathcal{S} \models \text{dis}(t)\}$, and $\mathcal{S} \models \text{dis}(t)$ holds iff the existential Presburger formula $\exists C \colon \phi(C) \wedge C \geq {}^{\bullet}t$ is unsatisfiable, where ϕ is the formula characterizing $\mathcal{S}$.

[6] This follows easily from the fact that $post^*(\psi)$ is not always expressible in Presburger arithmetic for vector addition systems, even if ψ denotes a single configuration [38].

Replicated systems are usually designed to run in *phases*. Initially, all transitions are alive, and the end of a phase is marked by the "death" of one or more transitions, i.e., by reaching a configuration at which these transitions are dead. The system keeps "killing transitions" until no transition that is still alive can lead to a configuration violating the postcondition. The procedure mimics this pattern. It constructs stage graphs in which if $\mathcal{S}'$ is a successor of $\mathcal{S}$, then the set of transitions dead at $\mathcal{S}'$ is a *proper superset* of the transitions dead at $\mathcal{S}$. For this, $\mathit{AsDead}(\mathcal{S})$ computes a set of transitions that are alive at some configuration of $\mathcal{S}$, but which will become dead in every fair run starting at $\mathcal{S}$ (line 5). Formally, $\mathit{AsDead}(\mathcal{S})$ returns a set $U \subseteq \overline{\mathit{Dead}(\mathcal{S})}$ such that $\mathcal{S} \models \Diamond \mathrm{dead}(U)$.

The following proposition, whose proof appears in the appendix, shows that determining whether a given transition will eventually become dead, while decidable, is PSPACE-hard. Therefore, Section 7 describes two implementations of this function, and a way to combine them, which exhibit a good trade-off between precision and computation time.

Proposition 4. *Given a replicated system $\mathcal{P}$, a stage $\mathcal{S}$ represented by an existential Presburger formula ϕ and a set of transitions U, determining whether $\mathcal{S} \models \Diamond \mathit{dead}(U)$ holds is decidable and* PSPACE*-hard.*

If the set U returned by $\mathit{AsDead}(\mathcal{S})$ is nonempty, then we know that every fair run starting at a configuration of $\mathcal{S}$ will eventually reach a configuration of $\mathcal{S} \cap [\![\mathrm{dead}(U)]\!]$. So, this set, or any inductive overapproximation of it, can be a legal successor of $\mathcal{S}$ in the stage graph. Function $\mathit{IndOverapprox}(\mathcal{S}, U)$ returns such an inductive overapproximation (line 7). To be precise, we show in Section 5 that $[\![\mathrm{dead}(U)]\!]$ is a Presburger set that can be computed exactly, albeit in doubly-exponential time in the worst case. The section also shows how to compute overapproximations more efficiently.

If the set U returned by $\mathit{AsDead}(\mathcal{S})$ is empty, then we cannot yet construct any successor of $\mathcal{S}$. Indeed, recall that we want to construct stage graphs in which if $\mathcal{S}'$ is a successor of $\mathcal{S}$, then $\mathit{Dead}(\mathcal{S}')$ is a *proper superset* of $\mathit{Dead}(\mathcal{S})$. In this case, we proceed differently and try to split $\mathcal{S}$:

Definition 7. *A* split *of some stage $\mathcal{S}$ is a set $\{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$ of (not necessarily disjoint) stages such that the following holds:*

- *$\mathit{Dead}(\mathcal{S}_i) \supset \mathit{Dead}(\mathcal{S})$ for every $1 \leq i \leq k$, and*
- *$\mathcal{S} = \bigcup_{i=1}^{k} \mathcal{S}_i$.*

If there exists a split $\{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$ of $\mathcal{S}$, then we can let $\mathcal{S}_1, \ldots, \mathcal{S}_k$ be the successors of $\mathcal{S}$ in the stage graph. Observe that a stage may indeed have a split. We have $\mathit{Dead}(\mathcal{C}_1 \cup \mathcal{C}_2) = \mathit{Dead}(\mathcal{C}_1) \cap \mathit{Dead}(\mathcal{C}_2)$, and hence $\mathit{Dead}(\mathcal{C}_1 \cup \mathcal{C}_2)$ may be a proper subset of both $\mathit{Dead}(\mathcal{C}_1)$ and $\mathit{Dead}(\mathcal{C}_2)$:

Example 4. Consider the system with states $\{q_1, q_2\}$ and transitions $t_i \colon q_i \mapsto q_i$ for $i \in \{1, 2\}$. Let $\mathcal{S} = \{C \mid C(q_1) = 0 \vee C(q_2) = 0\}$, i.e., $\mathcal{S}$ is the (inductive) stage of configurations disabling either t_1 or t_2. The set $\{\mathcal{S}_1, \mathcal{S}_2\}$, where $\mathcal{S}_i = \{C \in \mathcal{S} \mid C(q_i) = 0\}$, is a split of $\mathcal{S}$ satisfying $\mathit{Dead}(\mathcal{S}_i) = \{t_i\} \supset \emptyset = \mathit{Dead}(\mathcal{S})$. ◁

The canonical split of $\mathcal{S}$, if it exists, is the set $\{\mathcal{S} \cap [\![\text{dead}(t)]\!] \mid t \notin \mathit{Dead}(\mathcal{S})\}$. As mentioned above, Section 5 shows that $[\![\text{dead}(U)]\!]$ can be computed exactly for every U, but the computation can be expensive. Hence, the canonical split can be computed exactly at potentially high cost. Our implementation uses an underapproximation of $[\![\text{dead}(t)]\!]$, described in Section 6.

5 Computing and approximating $[\![\text{dead}(U)]\!]$

We show that, given a set U of transitions,

- we can effectively compute an existential Presburger formula describing the set $[\![\text{dead}(U)]\!]$, with high computational cost in the worst case, and
- we can effectively compute constraints that overapproximate or underapproximate $[\![\text{dead}(U)]\!]$, at a reduced computational cost.

Downward and upward closed sets. We enrich $\mathbb{N}$ with the limit element ω in the usual way. In particular, $n < \omega$ holds for every $n \in \mathbb{N}$. An *ω-configuration* is a mapping $C^\omega \colon Q \to \mathbb{N} \cup \{\omega\}$. The *upward closure* and *downward closure* of a set $\mathcal{C}^\omega$ of ω-configurations are the sets of configurations $\uparrow\mathcal{C}^\omega$ and $\downarrow\mathcal{C}^\omega$, respectively defined as:

$$\uparrow\mathcal{C}^\omega \overset{\text{def}}{=} \{C \in \mathbb{N}^Q \mid C \geq C^\omega \text{ for some } C^\omega \in \mathcal{C}^\omega\},$$
$$\downarrow\mathcal{C}^\omega \overset{\text{def}}{=} \{C \in \mathbb{N}^Q \mid C \leq C^\omega \text{ for some } C^\omega \in \mathcal{C}^\omega\}.$$

A set $\mathcal{C}$ of configurations is *upward closed* if $\mathcal{C} = \uparrow\mathcal{C}$, and *downward closed* if $\mathcal{C} = \downarrow\mathcal{C}$. These facts are well-known from the theory of well-quasi orderings:

Lemma 1. *For every set $\mathcal{C}$ of configurations, the following holds:*

1. *$\mathcal{C}$ is upward closed iff $\overline{\mathcal{C}}$ is downward closed (and vice versa);*
2. *if $\mathcal{C}$ is upward closed, then there is a unique minimal finite set of configurations $\mathit{inf}(\mathcal{C})$, called its* basis, *such that $\mathcal{C} = \uparrow \mathit{inf}(\mathcal{C})$;*
3. *if $\mathcal{C}$ is downward closed, then there is a unique minimal finite set of ω-configurations $\mathit{sup}(\mathcal{C})$, called its* decomposition, *such that $\mathcal{C} = \downarrow \mathit{sup}(\mathcal{C})$.*

Computing $[\![\text{dead}(U)]\!]$ exactly. It follows immediately from Definition 6 that both $[\![\text{dis}(U)]\!]$ and $[\![\text{dead}(U)]\!]$ are downward closed. Indeed, if all transitions of U are disabled at C, and $C' \leq C$, then they are also disabled at C', and clearly the same holds for transitions dead at C. Furthermore:

Proposition 5. *For every set U of transitions, the (downward) decomposition of both $\mathit{sup}([\![\mathit{dis}(U)]\!])$ and $\mathit{sup}([\![\mathit{dead}(U)]\!])$ is effectively computable.*

Proof. For every $t \in U$ and $q \in {}^\bullet t$, let $C^\omega_{t,q}$ be the ω-configuration such that $C^\omega_{t,q}(q) = {}^\bullet t(q) - 1$ and $C^\omega_{t,q}(p) = \omega$ for every $p \in Q \setminus \{q\}$. In other words, $C^\omega_{t,q}$ is the ω-configuration made only of ω's except for state q which falls short from

${}^\bullet t(q)$ by one. This ω-configurations captures all configurations disabled in t due to an insufficient amount of agents in state q. We have:

$$\sup(\llbracket \mathrm{dis}(U) \rrbracket) = \{C^\omega_{t,q} : t \in U, q \in {}^\bullet t\}.$$

The latter can be made minimal by removing superfluous ω-configurations.

For the case of $\sup(\llbracket \mathrm{dead}(U) \rrbracket)$, we invoke [40, Prop. 2] which gives a proof for the more general setting of (possibly unbounded) Petri nets. Their procedure is based on the well-known backwards reachability algorithm (see, e.g., [34,2]). $\square$

Since $\sup(\llbracket \mathrm{dead}(U) \rrbracket)$ is finite, its computation allows to describe $\llbracket \mathrm{dead}(U) \rrbracket$ by the following linear constraint[7]:

$$\bigvee_{C^\omega \in \sup(\llbracket \mathrm{dead}(U) \rrbracket)} \bigwedge_{q \in Q} [C(q) \leq C^\omega(q)] \,.$$

However, the cardinality of $\sup(\llbracket \mathrm{dead}(U) \rrbracket)$ can be exponential [40, Remark for Prop. 2] in the system size. For this reason, we are interested in constructing both under- and over-approximations.

Overapproximations of $\llbracket \mathbf{dead}(\boldsymbol{U}) \rrbracket$. For every $i \in \mathbb{N}$, define $\llbracket \mathrm{dead}(U) \rrbracket^i$ as:

$$\llbracket \mathrm{dead}(U) \rrbracket^0 \stackrel{\text{def}}{=} \llbracket \mathrm{dis}(U) \rrbracket \quad \text{and} \quad \llbracket \mathrm{dead}(U) \rrbracket^{i+1} \stackrel{\text{def}}{=} \mathit{post}(\llbracket \mathrm{dead}(U) \rrbracket^i) \cap \llbracket \mathrm{dis}(U) \rrbracket.$$

Loosely speaking, $\llbracket \mathrm{dead}(U) \rrbracket^i$ is the set of configurations C such that every configuration reachable in at most i steps from C disables U. We immediately have:

$$\llbracket \mathrm{dead}(U) \rrbracket = \bigcap_{i=0}^{\infty} \llbracket \mathrm{dead}(U) \rrbracket^i.$$

Using Proposition 5 and the following proposition, we obtain that $\llbracket \mathrm{dead}(U) \rrbracket^i$ is an effectively computable overapproximation of $\llbracket \mathrm{dead}(U) \rrbracket$.

Proposition 6. *For every Presburger set $\mathcal{C}$ and every set of transitions U, the set $\mathit{post}_U(\mathcal{C})$ is effectively Presburger.*

Recall that function $\mathit{IndOverapprox}(\mathcal{S}, U)$ of Algorithm 1 must return an *inductive* overapproximation of $\llbracket \mathrm{dead}(U) \rrbracket$. Since $\llbracket \mathrm{dead}(U) \rrbracket^i$ might not be inductive in general, our implementation uses either the inductive overapproximations $\mathit{IndOverapprox}^i(\mathcal{S}, U) \stackrel{\text{def}}{=} \mathit{PotReach}(\mathcal{S} \cap \llbracket \mathrm{dead}(U) \rrbracket^i)$, or the exact value $\mathit{IndOverapprox}^\infty(\mathcal{S}, U) \stackrel{\text{def}}{=} \mathcal{S} \cap \llbracket \mathrm{dead}(U) \rrbracket$. The table of results in the experimental section describes for each benchmark which overapproximation was used.

[7] Observe that if $C^\omega(q) = \omega$, then the term "$C(q) \leq \omega$" is equivalent to "**true**".

Underapproximations of $[\![\mathrm{dead}(U)]\!]$: Death certificates. A *death certificate* for U in $\mathcal{P}$ is a finite set $\mathcal{C}^\omega$ of ω-configurations such that:

1. $\downarrow\mathcal{C}^\omega \models \mathrm{dis}(U)$, i.e., every configuration of $\downarrow\mathcal{C}^\omega$ disables U, and
2. $\downarrow\mathcal{C}^\omega$ is inductive, i.e., $\mathit{post}(\downarrow\mathcal{C}^\omega) \subseteq \downarrow\mathcal{C}^\omega$.

If U is dead at a set $\mathcal{C}$ of configurations, then there is always a certificate that proves it, namely $\sup([\![\mathrm{dead}(U)]\!])$. In particular, if $\mathcal{C}^\omega$ is a death certificate for U then $\downarrow\mathcal{C}^\omega \subseteq [\![\mathrm{dead}(U)]\!]$, that is, $\downarrow\mathcal{C}^\omega$ is an underapproximation of $[\![\mathrm{dead}(U)]\!]$

Using Proposition 6, it is straightforward to express in Presburger arithmetic that a finite set $\mathcal{C}^\omega$ of ω-configurations is a death certificate for U:

Proposition 7. *For every $k \geq 1$ there is an existential Presburger formula $\mathit{DeathCert}_k(U, \mathcal{C}^\omega)$ that holds iff $\mathcal{C}^\omega$ is a death certificate of size k for U.*

6 Splitting a stage

Given a stage $\mathcal{S}$, we try to find a set $\mathcal{C}_1^\omega, \ldots, \mathcal{C}_\ell^\omega$ of death certificates for transitions $t_1, \ldots, t_\ell \in T \setminus \mathit{Dead}(\mathcal{S})$ such that $\mathcal{S} \subseteq \downarrow\mathcal{C}_1^\omega \cup \cdots \cup \downarrow\mathcal{C}_\ell^\omega$. This allows us to split $\mathcal{S}$ into $\mathcal{S}_1, \ldots, \mathcal{S}_\ell$, where $\mathcal{S}_i \overset{\text{def}}{=} \mathcal{S} \cap \downarrow\mathcal{C}_i^\omega$.

For any fixed size $k \geq 1$ and any fixed ℓ, we can find death certificates $\mathcal{C}_1^\omega, \ldots, \mathcal{C}_\ell^\omega$ of size at most k by solving a Presburger formula. However, the formula does not belong to the existential fragment, because the inclusion check $\mathcal{S} \subseteq \downarrow\mathcal{C}_1^\omega \cup \cdots \cup \downarrow\mathcal{C}_\ell^\omega$ requires universal quantification. For this reason, we proceed iteratively. For every $i \geq 0$, after having found $\mathcal{C}_1^\omega, \ldots, \mathcal{C}_i^\omega$ we search for a pair $(C_{i+1}, \mathcal{C}_{i+1}^\omega)$ such that

(i) $\mathcal{C}_{i+1}^\omega$ is a death certificate for some $t_{i+1} \in T \setminus \mathit{Dead}(\mathcal{S})$;
(ii) $C_{i+1} \in \mathcal{S} \cap \downarrow\mathcal{C}_{i+1}^\omega \setminus (\downarrow\mathcal{C}_1^\omega \cup \cdots \cup \downarrow\mathcal{C}_i^\omega)$.

An efficient implementation requires to guide the search for $(C_{i+1}, \mathcal{C}_{i+1}^\omega)$, because otherwise the search procedure might not even terminate, or might split $\mathcal{S}$ into too many parts, blowing up the size of the stage graph. Our search procedure employs the following heuristic, which works well in practice. We only consider the case $k = 1$, and search for a pair $(C_{i+1}, C_{i+1}^\omega)$ satisfying (i) and (ii) above, and additionally:

(iii) all components of C_{i+1}^ω are either ω or between 0 and $\max_{t \in T, q \in Q} {}^\bullet t(q) - 1$;
(iv) for every ω-configuration C^ω, if (C_{i+1}, C^ω) satisfies (i)–(iii), then $C_{i+1}^\omega \leq C^\omega$;
(v) for every pair (C, C^ω), if (C, C^ω) satisfies (i)–(iv), then $C^\omega \leq C_{i+1}^\omega$.

Condition (iii) guarantees termination. Intuitively, Condition (iv) leads to certificates valid for sets $U \subseteq T \setminus \mathit{Dead}(\mathcal{S})$ as large as possible. So it allows us to avoid splits that, loosely speaking, do not make as much progress as they could. Condition (v) allows us to avoid splits with many elements because each element of the split has a small intersection with $\mathcal{S}$.

Example 5. Let $\mathcal{P} = (Q, T)$ be the replicated system where $Q = \{a_1, \ldots, a_n\} \cup \{b_1, \ldots, b_n\} \cup \{c\}$ and $T = U \cup \{t_c \colon c \mapsto c\}$ with $U = \{t_i \colon a_i\, b_i \mapsto a_{i+1}\, b_{i+1} \mid 1 \leq i < n\} \cup \{t_n \colon a_n\, b_n \mapsto a_1\, b_1\}$. Let $\mathcal{S}$ be the set of all configurations C where either $C(c) = 0$ or $C(a_i) = C(b_i) = 0$ for all i. It is easy to see that no transition is dead at *every* configuration of $\mathcal{S}$, i.e., $\mathit{Dead}(\mathcal{S}) = \emptyset$, but every configuration of $\mathcal{S}$ has at least one dead transition: either $C(c) = 0$ and t_c is dead, or $C(c) > 0$ and all $t_i \in U$ are dead.

Consider the ω-configurations C^ω and D^ω defined as follows:

$$C^\omega(q) \overset{\text{def}}{=} \begin{cases} \omega & \text{if } q = c, \\ 0 & \text{otherwise,} \end{cases} \qquad D^\omega(q) \overset{\text{def}}{=} \begin{cases} \omega & \text{if } q \neq c, \\ 0 & \text{otherwise.} \end{cases}$$

C^ω is a death certificate for U, and D^ω is a death certificate for $\{t_c\}$. So the pairs $(\wr c \wr, C^\omega)$ and $(\wr a_1, \ldots, a_n, b_1, \ldots, b_n \wr, D^\omega)$ satisfy (i)–(iii). It is easy to see that they also satisfy (iv) and (v), and that the only split that can be returned by the procedure is $\{\mathcal{S} \cap \downarrow C^\omega, \mathcal{S} \cap \downarrow D^\omega\}$. So $\mathcal{S}$ is split into only two parts.

We now show that, if condition (iv) or (v) is dropped, then the splitting procedure might return splits of cardinality $2^n + 1$.

Let $\mathcal{M} \overset{\text{def}}{=} \{C \in \mathbb{N}^Q \mid C(c) = 0 \wedge \forall\, 1 \leq i \leq n \colon C(a_i) + C(b_i) = 1\} \subseteq \mathcal{S}$ and, for each $X \in \mathcal{M}$, define the ω-configurations C^ω_X, D^ω_X as follows:

$$C^\omega_X(q) \overset{\text{def}}{=} \begin{cases} \omega & \text{if } q = c \text{ or } X(q) > 0, \\ 0 & \text{otherwise,} \end{cases} \qquad D^\omega_X(q) \overset{\text{def}}{=} \begin{cases} \omega & \text{if } X(q) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

C^ω_X is a death certificate for U, and D^ω_X is a death certificate for $\{t_c\} \cup U$. So for every $X \in \mathcal{M}$ the pairs (X, C^ω_X) and (X, D^ω_X) satisfy (i)–(iii). Since we have $C^\omega \leq C^\omega_X$, $D^\omega_X \leq C^\omega_X$ and $D^\omega_X \leq D^\omega$ for every $X \in \mathcal{M}$, and otherwise the death certificates are pairwise incomparable, condition (iv) is satisfied by all the pairs (X, D^ω_X), but it is not satisfied by any of the pairs (X, C^ω_X). It follows that if we drop condition (iv) (removing the reference to (iv) in (v)), the splitting procedure might find the split $\{S \cap \downarrow C^\omega_X \mid X \in \mathcal{M}\} \cup \{\mathcal{S} \cap \downarrow D^\omega\}$. Without condition (v), but with (iv), it might find the split $\{S \cap \downarrow D^\omega_X \mid X \in \mathcal{M}\} \cup \{\mathcal{S} \cap \downarrow D^\omega\}$. Both splits have $2^n + 1$ elements. $\lhd$

7 Computing eventually dead transitions

Recall that the function $\mathit{AsDead}(\mathcal{S})$ takes an inductive Presburger set $\mathcal{S}$ as input, and returns a (possibly empty) set $U \subseteq \overline{\mathit{Dead}(\mathcal{S})}$ of transitions such that $\mathcal{S} \models \Diamond\text{dead}(U)$. This guarantees $\mathcal{S} \rightsquigarrow [\![\text{dead}(U)]\!]$ and, since $\mathcal{S}$ is inductive, also $\mathcal{S} \rightsquigarrow \mathcal{S} \cap [\![\text{dead}(U)]\!]$.

By Proposition 4, deciding if there exists a non-empty set U of transitions such that $\mathcal{S} \models \Diamond\text{dead}(U)$ holds is PSPACE-hard, which makes a polynomial reduction to satisfiability of existential Presburger formulas unlikely. So we design incomplete implementations of $\mathit{AsDead}(\mathcal{S})$ with lower complexity. Combining these implementations, the lack of completeness essentially vanishes in practice.

The implementations are inspired by Proposition 2, which shows that $\mathcal{S} \rightsquigarrow [\![\mathrm{dead}(U)]\!]$ holds iff there exists a certificate f such that:

$$\forall C \in \mathcal{S} \setminus [\![\mathrm{dead}(U)]\!] : \exists C \xrightarrow{*} C' : f(C) > f(C'). \tag{Cert}$$

To find such certificates efficiently, we only search for *linear* functions $f(C) = \sum_{q \in Q} \boldsymbol{a}(q) \cdot C(q)$ with coefficients $\boldsymbol{a}(q) \in \mathbb{N}$ for each $q \in Q$.

7.1 First implementation: Linear ranking functions

Our first procedure computes the existence of a linear *ranking function.*

Definition 8. *A function $r\colon \mathcal{S} \to \mathbb{N}$ is a ranking function for $\mathcal{S}$ and U if for every $C \in \mathcal{S}$ and every step $C \xrightarrow{t} C'$ the following holds:*

1. *if $t \in U$, then $r(C) > r(C')$; and*
2. *if $t \notin U$, then $r(C) \geq r(C')$.*

Proposition 8. *If $r\colon \mathcal{S} \to \mathbb{N}$ is a ranking function for $\mathcal{S}$ and U, then there exists $k \in \mathbb{N}$ such that (r, k) is a bounded certificate for $\mathcal{S} \rightsquigarrow [\![dead(U)]\!]$.*

Proof. Let M be the minimal finite basis of the upward closed set $\overline{[\![\mathrm{dead}(U)]\!]}$. For every configuration $D \in M$, let σ_D be a shortest sequence that enables some transition of $t_D \in U$ from D, i.e., such that $D \xrightarrow{\sigma_D} D' \xrightarrow{t_D} D''$ for some D', D''. Let $k \overset{\text{def}}{=} \max\{|\sigma_D t_D| : D \in M\}$.

Let $C \in \mathcal{S} \setminus [\![\mathrm{dead}(U)]\!]$. Since $C \in \overline{[\![\mathrm{dead}(U)]\!]}$, we have $C \geq D$ for some $D \in M$. By monotonicity, we have $C \xrightarrow{\sigma_D} C' \xrightarrow{t_D} C''$ for some configurations C' and C''. By Definition 8, we have $r(C) \geq r(C') > r(C'')$, and so condition (Cert) holds. As $|\sigma_D t_D| \leq k$, we have that (r, k) is a bounded certificate. $\square$

It follows immediately from Definition 8 that if r_1 and r_2 are ranking functions for sets U_1 and U_2 respectively, then r defined as $r(C) \overset{\text{def}}{=} r_1(C) + r_2(C)$ is a ranking function for $U_1 \cup U_2$. Therefore, there exists a unique maximal set of transitions U such that $\mathcal{S} \rightsquigarrow [\![\mathrm{dead}(U)]\!]$ can be proved by means of a ranking function. Further, U can be computed by collecting all transitions $t \in \overline{Dead(\mathcal{S})}$ such that there exists a ranking function r_t for $\{t\}$. The existence of a *linear* ranking function r_t can be decided in polynomial time via linear programming, as follows. Recall that for every step $C \xrightarrow{u} C'$, we have $C' = C + \Delta(u)$. So, by linearity, we have $r_t(C) \geq r_t(C') \iff r_t(C' - C) \leq 0 \iff r_t(\Delta(u)) \leq 0$. Thus, the constraints of Definition 8 can be specified as:

$$\boldsymbol{a} \cdot \Delta(t) < 0 \quad \wedge \bigwedge_{u \in \overline{Dead(\mathcal{S})}} \boldsymbol{a} \cdot \Delta(u) \leq 0,$$

where $\boldsymbol{a}\colon Q \to \mathbb{Q}_{\geq 0}$ gives the coefficients of r_t, that is, $r_t(C) = \boldsymbol{a} \cdot C$, and $\boldsymbol{a} \cdot \boldsymbol{x} \overset{\text{def}}{=} \sum_{q \in Q} \boldsymbol{a}(q) \cdot \boldsymbol{x}(q)$ for $\boldsymbol{x} \in \mathbb{N}^Q$. Observe that a solution may yield a function whose codomain differs from $\mathbb{N}$. However, this is not an issue since we can scale it with the least common denominator of each $\boldsymbol{a}(q)$.

7.2 Second implementation: Layers

Transitions layers were introduced in [17] as a technique to find transitions that will eventually become dead. Intuitively, a set U of transitions is a layer if (1) no run can contain only transitions of U, and (2) U becomes dead once disabled; the first condition guarantees that U eventually becomes disabled, and the second that it eventually becomes dead. We formalize layers in terms of *layer functions*.

Definition 9. *A function $\ell\colon \mathcal{S} \to \mathbb{N}$ is a* layer function *for $\mathcal{S}$ and U if:*

***C1**. $\ell(C) > \ell(C')$ for every $C \in \mathcal{S}$ and every step $C \xrightarrow{t} C'$ with $t \in U$; and*
***C2**. $[\![\mathit{dis}(U)]\!] = [\![\mathit{dead}(U)]\!]$.*

Proposition 9. *If $\ell\colon \mathcal{S} \to \mathbb{N}$ is a layer function for $\mathcal{S}$ and U, then $(\ell, 1)$ is a bounded certificate for $\mathcal{S} \rightsquigarrow [\![\mathit{dead}(U)]\!]$.*

Proof. Let $C \in \mathcal{S} \setminus [\![\mathrm{dead}(U)]\!]$. By condition **C2**, we have $C \notin [\![\mathrm{dis}(U)]\!]$. So there exists a step $C \xrightarrow{u} C'$ where $u \in U$. By condition **C1**, we have $\ell(C) > \ell(C')$, so condition (Cert) holds and $(\ell, 1)$ is a bounded certificate.

Let $\mathcal{S}$ be a stage. For every set of transitions $U \subseteq \overline{\mathit{Dead}(\mathcal{S})}$ we can construct a Presburger formula $\mathit{lin\text{-}layer}(U, \boldsymbol{a})$ that holds iff there there exists a *linear* layer function for U, i.e., a layer function of the form $\ell(C) = \boldsymbol{a} \cdot C$ for a vector of coefficients $\boldsymbol{a}\colon Q \to \mathbb{Q}_{\geq 0}$. Condition **C1**, for a linear function $\ell(C)$, is expressed by the existential Presburger formula

$$\mathit{lin\text{-}layer\text{-}fun}(U, \boldsymbol{a}) \overset{\text{def}}{=} \bigwedge_{u \in U} \boldsymbol{a} \cdot \Delta(u) < 0.$$

Condition **C2** is expressible in Presburger arithmetic because of Proposition 5. However, instead of computing $[\![\mathrm{dead}(U)]\!]$ explicitly, there is a more efficient way to express this constraint. Intuitively, $[\![\mathrm{dis}(U)]\!] = [\![\mathrm{dead}(U)]\!]$ is the case if enabling a transition $u \in U$ requires to have previously enabled some transition $u' \in U$. This observation leads to:

Proposition 10. *A set U of transitions satisfies $[\![\mathit{dis}(U)]\!] = [\![\mathit{dead}(U)]\!]$ iff it satisfies the existential Presburger formula*

$$\mathit{dis\text{-}eq\text{-}dead}(U) \overset{\mathit{def}}{=} \bigwedge_{t \in T} \bigwedge_{u \in U} \bigvee_{u' \in U} {}^{\bullet}t + ({}^{\bullet}u \ominus t^{\bullet}) \geq {}^{\bullet}u'$$

where $\boldsymbol{x} \ominus \boldsymbol{y} \in \mathbb{N}^Q$ is defined by $(\boldsymbol{x} \ominus \boldsymbol{y})(q) \overset{\mathit{def}}{=} \max(\boldsymbol{x}(q) - \boldsymbol{y}(q), 0)$ for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{N}^Q$.

This allows us to give the constraint $\mathit{lin\text{-}layer}(U, \boldsymbol{a})$, which is of polynomial size:

$$\mathit{lin\text{-}layer}(U, \boldsymbol{a}) \overset{\text{def}}{=} \mathit{lin\text{-}layer\text{-}fun}(U, \boldsymbol{a}) \wedge \mathit{dis\text{-}eq\text{-}dead}(U).$$

7.3 Comparing ranking and layer functions

The ranking and layer functions of Sections 7.1 and 7.2 are incomparable in power, that is, there are sets of transitions for which a ranking function but no layer function exists, and vice versa. This is shown by the following two systems:

$$\begin{aligned}
\mathcal{P}_1 &= (\ \{\ \mathrm{A}, \mathrm{B}, \mathrm{C}\ \}, \{\ t_1 \colon \mathrm{A}\,\mathrm{B} \mapsto \mathrm{C}\,\mathrm{C},\ t_2 \colon \mathrm{A} \mapsto \mathrm{B},\ t_3 \colon \mathrm{B} \mapsto \mathrm{A}\ \}\ ),\\
\mathcal{P}_2 &= (\ \{\ \mathrm{A}, \mathrm{B}\ \}, \quad \{\ t_4 \colon \mathrm{A}\,\mathrm{B} \mapsto \mathrm{A}\,\mathrm{A},\ t_5 \colon \mathrm{A} \mapsto \mathrm{B}\ \} \qquad\qquad).
\end{aligned}$$

Consider the system $\mathcal{P}_1$, and let $\mathcal{S} = \mathbb{N}^Q$, i.e., $\mathcal{S}$ contains all configurations. Transitions t_2 and t_3 never become dead at $\wr \mathrm{A} \int$ and can thus never be included in any U. Transition t_1 eventually becomes dead, as shown by the linear ranking function $r(C) = C(\mathrm{A}) + C(\mathrm{B})$ for $U = \{t_1\}$. But for this U, the condition **C2** for layer functions is not satisfied, as $[\![\mathrm{dis}(U)]\!] \ni \wr \mathrm{A}, \mathrm{A} \int \xrightarrow{t_2} \wr \mathrm{A}, \mathrm{B} \int \notin [\![\mathrm{dis}(U)]\!]$, so $[\![\mathrm{dis}(U)]\!] \neq [\![\mathrm{dead}(U)]\!]$. Therefore no layer function exists for this U.

Consider now the system $\mathcal{P}_2$, again with $\mathcal{S} = \mathbb{N}^Q$, and let $U = \{t_5\}$. Once t_5 is disabled, there is no agent in A, so both t_4 and t_5 are dead. So $[\![\mathrm{dis}(U)]\!] = [\![\mathrm{dead}(U)]\!]$. The linear layer function $\ell(C) = C(\mathrm{A})$ satisfies *lin-layer-fun*$(U, \boldsymbol{a})$, showing that U eventually becomes dead. As $C \xrightarrow{t_4 t_5} C$ for $C = \wr \mathrm{A}, \mathrm{B} \int$, there is no ranking function r for this U, which would need to satisfy $r(C) < r(C)$.

For our implementation of $\mathit{AsDead}(\mathcal{S})$, we therefore combine both approaches. We first compute (in polynomial time) the unique maximal set U for which there is a linear ranking function. If this U is non-empty, we return it, and otherwise compute a set U of maximal size for which there is a linear layer function.

8 Experimental results

We implemented the procedure of Section 4 on top of the SMT solver Z3 [53]. The resulting tool automatically constructs stage graphs that verify stable termination properties for replicated systems. We evaluated it on two sets of benchmarks, described below. The first set contains population protocols, and the second leader election and mutex exclusion algorithms. All tests where performed on a machine with an Intel Xeon CPU E5-2630 v4 @ 2.20GHz and 8GB of RAM. The results are depicted in Figure 2. For parametric families of replicated systems, we always report the largest instance that we were able to verify with a timeout of one hour. For *IndOverapprox*, from the approaches in Section 5, we use *IndOverapprox*^{0} in the examples marked with * and *IndOverapprox*$^\infty$ otherwise.

Population protocols. Population protocols [7,8] are replicated systems that compute Presburger predicates following the computation-as-consensus paradigm [9]. Depending on whether the initial configuration of agents satisfies the predicate or not, the agents of a correct protocol eventually agree on the output "yes" or "no", almost surely. Example 1 can be interpreted as a population protocol for the majority predicate $\mathrm{A_Y} > \mathrm{A_N}$, and the two stable termination properties that verify its correctness are described in Example 2. To show that a population

Population protocols (correctness)			
Parameters	$\|Q\|$	$\|T\|$	**Time**
Broadcast [26,17] *			
	2	1	$< 1s$
Majority (Example 1)[17] *			
	4	4	$< 1s$
Majority [18, Ex. 3] *			
	5	6	$< 1s$
Majority [4] ("fast & exact")			
m=13, d=1	16	136	$4s$
m=21, d=1 (TO: 23,1)	24	300	$466s$
m=21, d=20 (TO: 23,22)	62	1953	$3301s$
Flock-of-birds [23,17] *: $x \geq c$			
$c = 20$	21	210	$5s$
$c = 40$	41	820	$45s$
$c = 60$	61	1830	$341s$
$c = 80$ (TO: $c = 90$)	81	3240	$1217s$
Flock-of-birds [15, Sect. 3]: $x \geq c$			
$c = 60$	8	18	$15s$
$c = 90$	9	21	$271s$
$c = 120$ (TO: $c = 127$)	9	21	$2551s$
Flock-of-birds [26,17, *threshold-n*] *: $x \geq c$			
$c = 10$	11	19	$< 1s$
$c = 15$	16	29	$1s$
$c = 20$ (TO: $c = 25$)	21	39	$18s$
Threshold [7][17, $v_{\max}$=$c+1$] *: $\boldsymbol{a} \cdot \boldsymbol{x} \geq c$			
$c = 2$	28	288	$7s$
$c = 4$	44	716	$26s$
$c = 6$	60	1336	$107s$
$c = 8$ (TO: $c = 10$)	76	2148	$1089s$
Threshold [15] ("succinct"): $\boldsymbol{a} \cdot \boldsymbol{x} \geq c$			
$c = 7$	13	37	$2s$
$c = 31$	17	55	$11s$
$c = 127$	21	73	$158s$
$c = 511$ (TO: $c = 1023$)	25	91	$2659s$
Remainder [17] *: $\boldsymbol{a} \cdot \boldsymbol{x} \equiv_m c$			
$m = 5$	7	20	$< 1s$
$m = 15$	17	135	$34s$
$m = 20$ (TO: $m = 25$)	22	230	$1646s$

Population protocols (stable cons.)			
Parameters	$\|Q\|$	$\|T\|$	**Time**
Approx. majority [22] (Cell cycle sw.) *			
	3	4	$< 1s$
Approx. majority [46] (Coin game) *			
$k = 3$	2	4	$< 1s$
Approx. majority [52] (Moran proc.) *			
	2	2	$< 1s$

Leader election/Mutex algorithms			
Processes	$\|Q\|$	$\|T\|$	**Time**
Leader election [39] (Israeli-Jalfon)			
20	40	80	$7s$
60	120	240	$1493s$
70 (TO: 80)	140	280	$3295s$
Leader election [37] (Herman)			
21	42	42	$9s$
51	102	102	$300s$
81 (TO: 91)	162	162	$2800s$
Mutex [35] (Array)			
2	15	95	$2s$
5	33	239	$5s$
10 (TO: 11)	63	479	$938s$
Mutex [55] (Burns)			
2	11	75	$1s$
4	19	199	$119s$
5 (TO: 6)	23	279	$2232s$
Mutex [3] (Dijkstra)			
2	19	196	$66s$
3 (TO: 4)	27	488	$3468s$
Mutex [45] (Lehmann Rabin)			
2	19	135	$3s$
5	43	339	$115s$
9 (TO: 10)	75	611	$2470s$
Mutex [57] (Peterson)			
2	13	86	$2s$
Mutex [60] (Szymanski)			
2	17	211	$10s$
3 (TO: 4)	24	895	$667s$

Fig. 2: Columns $|Q|$, $|T|$, and **Time** give the number of states and non-silent transitions, and the time for verification. Population protocols are verified for an infinite set of configurations. For parametric families, the smallest instance that could not be verified within one hour is shown in brackets, e.g. (TO: $c = 90$). Leader election and mutex algorithms are verified for one configuration. The number of processes leading to a timeout is given in brackets, e.g. (TO: 10).

protocol correctly computes a given predicate, we thus construct two Presburger stage graphs for the two corresponding stable termination properties. In all these examples, correctness is proved for an infinite set of initial configurations.

Our set of benchmarks contains a broadcast protocol [26], three majority protocols (Example 1, [18, Ex. 3], [4]), and multiple instances of parameterized families of protocols, where each protocol computes a different instance of a parameterized family of predicates[8]. These include various *flock-of-birds* protocol families ([23], [15, Sect. 3], [26, *threshold-n*]) for the family of predicates $x \geq c$ for some constant $c \geq 0$; two families for threshold predicates of the form $\boldsymbol{a} \cdot \boldsymbol{x} \geq c$ [7,15]; and one family for remainder protocols of the form $\boldsymbol{a} \cdot \boldsymbol{x} \equiv_m c$ [17]. Further, we check approximate majority protocols ([22], [46, *coin game*], [52]). As these protocols only compute the predicate with large probability but not almost surely, we only verify that they always converge to a stable consensus.

Comparison with [17]. The approach of [17] can only be applied to so-called *strongly-silent* protocols. We are able to verify all six protocols reported in [17]. Further, we are also able to verify the protocols Majority [4], Flock-of-birds [15, Sect. 3] and Threshold [15], which are not strongly-silent. Although our approach is more general and complete, the time to verify many strongly-silent protocol does not differ significantly between the two approaches. Exceptions are the Flock-of-birds [23] protocols where we are faster ([17] reaches the timeout at $c = 55$) as well as the Remainder and the Flock-of-birds-threshold-n protocols where we are substantially slower ([17] reaches the timeout at $m = 80$ and $c = 350$, respectively). Loosely speaking, the approach of [17] can be faster because they compute inductive overapproximations using an iterative procedure instead of *PotReach*. In some instances already a very weak overapproximation, much less precise than *PotReach*, suffices to verify the result. Our procedure can be adapted to accommodate this (it essentially amounts to first running the procedure of [17], and if it is inconclusive then run ours).

Other distributed algorithms. We have also used our approach to verify arbitrary LTL liveness properties of non-parameterized systems with arbitrary communication structure. To verify arbitrary LTL properties we apply standard automata-theoretic techniques. We construct a product of the system and a *limit-deterministic Büchi automaton* that accepts the negation of the property. Checking that the runs of the product accepted by the automaton have positive probability reduces to checking a stable termination property.

Since we only check correctness of one single finite-state system, we can also apply a probabilistic model checker based on state-space exploration. However, our technique delivers a stage graph, which plays two roles. First, it gives an explanation of why the property holds in terms of invariants and ranking functions, and second, it is a certificate of correctness that can be efficiently checked by independent means.

[8] Notice that for each protocol we check correctness for all inputs; we cannot yet automatically verify that infinitely many protocols are correct, each of them for all possible inputs.

We verify liveness properties for several leader election and mutex algorithms from the literature [39,37,35,55,3,45,57,60] under the assumption of a probabilistic scheduler. For the leader election algorithms, we check that a leader is eventually chosen; for the mutex algorithms, we check that the first process enters its critical section infinitely often.

Comparison with PRISM [44]. We compared execution times for verification by our technique and by PRISM on the same models. While PRISM only needs a few seconds to verify a single instance of the mutex algorithms [35,55,3,45,57,60] up to the point where we reach the timeout, it reaches the memory limit for the two leader election algorithms [39,37] already for 70 and 71 processes, respectively, which we can still verify.

9 Conclusion and further work

We have presented stage graphs, a sound and complete technique for the verification of stable termination properties of replicated systems, an important class of parameterized systems. Using deep results of the theory of Petri nets, we have shown that Presburger stage graphs, a class of stage graphs whose correctness can be reduced to the satisfiability problem of Presburger arithmetic, are also sound and complete. This provides a decision procedure for the verification of termination properties, which is of theoretical nature since it involves a blind enumeration of candidates for Presburger stage graphs. For this reason, we have presented a technique for the algorithmic construction of Presburger stage graphs, designed to exploit the strengths of SMT-solvers for existential Presburger formulas, i.e., integer linear constraints. Loosely speaking, the technique searches for *linear* functions certifying the progress between stages, even though only the much larger class of Presburger functions guarantees completeness.

We have conducted extensive experiments on a large set of benchmarks. In particular, our approach is able to prove correctness of nearly all the standard protocols described in the literature, including several protocols that could not be proved by the technique of [17], which only worked for so-called strongly-silent protocols. We have also successfully applied the technique to some self-stabilization algorithms, leader election and mutual exclusion algorithms.

Our technique is based on the mechanized search for invariants and ranking functions. It avoids the use of state-space exploration as much as possible. For this reason, it also makes sense as a technique for the verification of liveness properties of non-parameterized systems with a finite but very large state space.

References

1. Abdulla, P.A.: Regular model checking. International Journal on Software Tools for Technology Transfer **14**(2), 109–118 (2012). https://doi.org/10.1007/s10009-011-0216-8

2. Abdulla, P.A., Cerans, K., Jonsson, B., Tsay, Y.: General decidability theorems for infinite-state systems. In: Proc. 11th Annual IEEE Symposium on Logic in Computer Science (LICS). pp. 313–321 (1996). https://doi.org/10.1109/LICS.1996.561359
3. Abdulla, P.A., Delzanno, G., Henda, N.B., Rezine, A.: Regular model checking without transducers (on efficient verification of parameterized systems). In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 721–736. Springer (2007). https://doi.org/10.1007/978-3-540-71209-1_56
4. Alistarh, D., Gelashvili, R., Vojnović, M.: Fast and exact majority in population protocols. In: Proc. ACM Symposium on Principles of Distributed Computing (PODC). pp. 47–56 (2015). https://doi.org/10.1145/2767386.2767429
5. Aminof, B., Rubin, S., Zuleger, F., Spegni, F.: Liveness of parameterized timed networks. In: Proc. 42nd International Colloquium on Automata, Languages, and Programming (ICALP). pp. 375–387 (2015). https://doi.org/10.1007/978-3-662-47666-6_30
6. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. **75**(2), 87–106 (1987). https://doi.org/10.1016/0890-5401(87)90052-6
7. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. In: Proc. 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC). pp. 290–299 (2004). https://doi.org/10.1145/1011767.1011810
8. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. Distributed Computing **18**(4), 235–253 (2006). https://doi.org/10.1007/s00446-005-0138-3
9. Angluin, D., Aspnes, J., Eisenstat, D., Ruppert, E.: The computational power of population protocols. Distributed Computing **20**(4), 279–304 (2007). https://doi.org/10.1007/s00446-007-0040-2
10. Athanasiou, K., Liu, P., Wahl, T.: Unbounded-thread program verification using thread-state equations. In: Proc. 8th International Joint Conference on Automated Reasoning (IJCAR). pp. 516–531 (2016). https://doi.org/10.1007/978-3-319-40229-1_35
11. Baier, C., Katoen, J.: Principles of model checking. MIT Press (2008)
12. Basler, G., Mazzucchi, M., Wahl, T., Kroening, D.: Symbolic counter abstraction for concurrent software. In: CAV. Lecture Notes in Computer Science, vol. 5643, pp. 64–78. Springer (2009). https://doi.org/10.1007/978-3-642-02658-4_9
13. Berman, L.: The complexitiy of logical theories. Theoretical Computer Science **11**, 71–77 (1980). https://doi.org/10.1016/0304-3975(80)90037-7
14. Bloem, R., Jacobs, S., Khalimov, A., Konnov, I., Rubin, S., Veith, H., Widder, J.: Decidability of Parameterized Verification. Synthesis Lectures on Distributed Computing Theory, Morgan & Claypool Publishers (2015). https://doi.org/10.2200/S00658ED1V01Y201508DCT013
15. Blondin, M., Esparza, J., Jaax, S.: Large flocks of small birds: on the minimal size of population protocols. In: Proc. 35th Symposium on Theoretical Aspects of Computer Science (STACS). pp. 16:1–16:14 (2018). https://doi.org/10.4230/LIPIcs.STACS.2018.16
16. Blondin, M., Esparza, J., Jaax, S.: Peregrine: A tool for the analysis of population protocols. In: Proc. 30th International Conference on Computer Aided Verification (CAV). pp. 604–611 (2018). https://doi.org/10.1007/978-3-319-96145-3_34

17. Blondin, M., Esparza, J., Jaax, S., Meyer, P.J.: Towards efficient verification of population protocols. In: Proc. 36th ACM Symposium on Principles of Distributed Computing (PODC). pp. 423–430 (2017). https://doi.org/10.1145/3087801.3087816
18. Blondin, M., Esparza, J., Kučera, A.: Automatic analysis of expected termination time for population protocols. In: Proc. 29th International Conference on Concurrency Theory (CONCUR). pp. 33:1–33:16 (2018). https://doi.org/10.4230/LIPIcs.CONCUR.2018.33
19. Blondin, M., Finkel, A., Haase, C., Haddad, S.: The logical view on continuous Petri nets. ACM Transactions on Computational Logic (TOCL) **18**(3), 24:1–24:28 (2017). https://doi.org/10.1145/3105908
20. Bouajjani, A., Jonsson, B., Nilsson, M., Touili, T.: Regular model checking. In: Proc. 12th International Conference Computer Aided Verification (CAV). pp. 403–418 (2000). https://doi.org/10.1007/10722167_31
21. Browne, M.C., Clarke, E.M., Grumberg, O.: Reasoning about networks with many identical finite state processes. Information and Computation **81**(1), 13–31 (1989). https://doi.org/10.1016/0890-5401(89)90026-6
22. Cardelli, L., Csikász-Nagy, A.: The cell cycle switch computes approximate majority. Scientific Reports **2**(1), 656 (2012). https://doi.org/10.1038/srep00656
23. Chatzigiannakis, I., Michail, O., Spirakis, P.G.: Algorithmic verification of population protocols. In: Proc. 12th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS). pp. 221–235. Springer (2010). https://doi.org/10.1007/978-3-642-16023-3_19
24. Chen, Y., Hong, C., Lin, A.W., Rümmer, P.: Learning to prove safety over parameterised concurrent systems. In: FMCAD. pp. 76–83. IEEE (2017). https://doi.org/10.23919/FMCAD.2017.8102244
25. Clarke, E.M., Talupur, M., Touili, T., Veith, H.: Verification by network decomposition. In: 15th International Conference on Concurrency Theory (CONCUR). pp. 276–291 (2004). https://doi.org/10.1007/978-3-540-28644-8_18
26. Clément, J., Delporte-Gallet, C., Fauconnier, H., Sighireanu, M.: Guidelines for the verification of population protocols. In: Proc. 31st International Conference on Distributed Computing Systems (ICDCS). pp. 215–224 (2011). https://doi.org/10.1109/ICDCS.2011.36
27. Cooper, D.C.: Theorem proving in arithmetic without multiplication. Machine Intelligence **7**, 91–99 (1972)
28. Czerwiński, W., Lasota, S., Lazić, R., Leroux, J., Mazowiecki, F.: The reachability problem for Petri nets is not elementary. In: Proc. 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC). pp. 24–33 (2019). https://doi.org/10.1145/3313276.3316369
29. Emerson, E.A., Namjoshi, K.S.: On reasoning about rings. International Journal of Foundations of Computer Science **14**(4), 527–550 (2003). https://doi.org/10.1142/S0129054103001881
30. Esparza, J., Ganty, P., Leroux, J., Majumdar, R.: Model checking population protocols. In: Proc. 36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS). vol. 65, pp. 27:1–27:14 (2016). https://doi.org/10.4230/LIPIcs.FSTTCS.2016.27
31. Esparza, J., Ganty, P., Leroux, J., Majumdar, R.: Verification of population protocols. Acta Informatica **54**(2), 191–215 (2017). https://doi.org/10.1007/s00236-016-0272-3

32. Esparza, J., Ledesma-Garza, R., Majumdar, R., Meyer, P.J., Nikšić, F.: An SMT-based approach to coverability analysis. In: Proc. 26th International Conference on Computer Aided Verification (CAV). pp. 603–619 (2014). https://doi.org/10.1007/978-3-319-08867-9_40
33. Esparza, J., Meyer, P.J.: An SMT-based approach to fair termination analysis. In: Formal Methods in Computer-Aided Design (FMCAD). pp. 49–56 (2015). https://doi.org/10.1109/FMCAD.2015.7542252
34. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! Theoretical Computer Science **256**(1-2), 63–92 (2001). https://doi.org/10.1016/S0304-3975(00)00102-X
35. Fribourg, L., Olsén, H.: Reachability sets of parameterized rings as regular languages. Electronic Notes in Theoretical Computer Science **9**, 40 (1997). https://doi.org/https://doi.org/10.1016/S1571-0661(05)80427-X
36. German, S.M., Sistla, A.P.: Reasoning about systems with many processes. Journal of the ACM **39**(3), 675–735 (1992). https://doi.org/10.1145/146637.146681
37. Herman, T.: Probabilistic self-stabilization. Inf. Process. Lett. **35**(2), 63–67 (1990). https://doi.org/10.1016/0020-0190(90)90107-9
38. Hopcroft, J.E., Pansiot, J.: On the reachability problem for 5-dimensional vector addition systems. Theoretical Computer Science **8**, 135–159 (1979). https://doi.org/10.1016/0304-3975(79)90041-0
39. Israeli, A., Jalfon, M.: Token management schemes and random walks yield self-stabilizing mutual exclusion. In: Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing, Quebec City, Quebec, Canada, August 22-24, 1990. pp. 119–131 (1990). https://doi.org/10.1145/93385.93409
40. Jančar, P., Purser, D.: Structural liveness of Petri nets is ExpSpace-hard and decidable. Acta Informatica **56**(6), 537–552 (2019). https://doi.org/10.1007/s00236-019-00338-6
41. Jones, N.D., Landweber, L.H., Lien, Y.E.: Complexity of some problems in petri nets. Theoretical Computer Science **4**(3), 277–299 (1977). https://doi.org/10.1016/0304-3975(77)90014-7
42. Kaiser, A., Kroening, D., Wahl, T.: Dynamic cutoff detection in parameterized concurrent programs. In: Proc. 22nd International Conference on Computer Aided Verification (CAV). pp. 645–659 (2010). https://doi.org/10.1007/978-3-642-14295-6_55
43. Kaiser, A., Kroening, D., Wahl, T.: A widening approach to multithreaded program verification. ACM Transactions on Programming Languages and Systems **36**(4), 14:1–14:29 (2014). https://doi.org/10.1145/2629608
44. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) Proc. 23rd International Conference on Computer Aided Verification (CAV'11). LNCS, vol. 6806, pp. 585–591. Springer (2011). https://doi.org/10.1007/978-3-642-22110-1_47
45. Lehmann, D., Rabin, M.O.: On the advantages of free choice: A symmetric and fully distributed solution to the dining philosophers problem. In: Conference Record of the Eighth Annual ACM Symposium on Principles of Programming Languages, Williamsburg, Virginia, USA, January 1981. pp. 133–138 (1981). https://doi.org/10.1145/567532.567547
46. Lengál, O., Lin, A.W., Majumdar, R., Rümmer, P.: Fair termination for parameterized probabilistic concurrent systems. In: Proc. 23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). pp. 499–517 (2017). https://doi.org/10.1007/978-3-662-54577-5_29

47. Leroux, J.: Vector addition systems reachability problem (A simpler solution). In: Turing-100 - The Alan Turing Centenary, Manchester, UK, June 22-25, 2012. pp. 214–228 (2012). https://doi.org/10.29007/bnx2
48. Leroux, J.: Presburger vector addition systems. In: Proc. 28th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). pp. 23–32 (2013). https://doi.org/10.1109/LICS.2013.7
49. Leroux, J.: Vector addition system reversible reachability problem. Logical Methods in Computer Science **9**(1) (2013). https://doi.org/10.2168/LMCS-9(1:5)2013
50. Lin, A.W., Rümmer, P.: Liveness of randomised parameterised systems under arbitrary schedulers. In: Proc. 28th International Conference on Computer Aided Verification (CAV). pp. 112–133 (2016). https://doi.org/10.1007/978-3-319-41540-6_7
51. Minsky, M.: Computation: Finite and Infinite Machines. Prentice-Hall, Englewood Cliffs, N. J. (1967)
52. Moran, P.A.P.: Random processes in genetics. Mathematical Proceedings of the Cambridge Philosophical Society **54**(1), 60–71 (1958). https://doi.org/10.1017/S0305004100033193
53. de Moura, L.M., Bjørner, N.: Z3: An efficient SMT solver. In: Proc. 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). pp. 337–340 (2008). https://doi.org/10.1007/978-3-540-78800-3_24
54. Navlakha, S., Bar-Joseph, Z.: Distributed information processing in biological and computational systems. Communications of the ACM **58**(1), 94–102 (2015). https://doi.org/10.1145/2678280
55. Nilsson, M.: Regular model checking. Ph.D. thesis, Uppsala University (2000)
56. Pang, J., Luo, Z., Deng, Y.: On automatic verification of self-stabilizing population protocols. In: Proc. Second IEEE/IFIP International Symposium on Theoretical Aspects of Software Engineering (TASE). pp. 185–192. IEEE Computer Society (2008). https://doi.org/10.1109/TASE.2008.8
57. Peterson, G.L.: Myths about the mutual exclusion problem. Inf. Process. Lett. **12**(3), 115–116 (1981). https://doi.org/10.1016/0020-0190(81)90106-X
58. Presburger, M.: Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. Comptes Rendus du Ier Congrès des mathématiciens des pays slaves pp. 192–201 (1929)
59. Sun, J., Liu, Y., Dong, J.S., Pang, J.: PAT: towards flexible verification under fairness. In: Proc. 21st International Conference on Computer Aided Verification (CAV). pp. 709–714. Springer (2009). https://doi.org/10.1007/978-3-642-02658-4_59
60. Szymanski, B.K.: A simple solution to Lamport's concurrent programming problem with linear wait. In: Proceedings of the 2nd international conference on Supercomputing, ICS 1988, Saint Malo, France, July 4-8, 1988. pp. 621–626 (1988). https://doi.org/10.1145/55364.55425
61. Vardi, M.Y.: Automatic verification of probabilistic concurrent finite-state programs. In: Proc. 26th Annual Symposium on Foundations of Computer Science (FOCS). pp. 327–338 (1985). https://doi.org/10.1109/SFCS.1985.12

A Appendix

A.1 Missing proofs for Section 2

We show that the qualitative model checking problem is not semi-decidable. The result holds even for the subclass of replicated systems of arity 2 (i.e., for population protocols) and when $\mathcal{I} = [\![\varphi_1]\!]$ and the LTL formula is of the form $\varphi = \Box\varphi_2 \vee \Diamond\varphi_3$, where φ_1, φ_2 and φ_3 are quantifier-free Presburger predicates with atomic formulas of the form $q{=}0$, $q{=}1$, or $q{\geq}1$ for $q \in Q$.

Theorem 1. *The qualitative model checking problem is not semi-decidable.*

Proof. A two-counter *Minsky machine* $\mathcal{M}$ is a finite sequence of labeled instructions

$$\ell_1 : ins_1, \quad \ldots \quad , \ell_m : ins_m, \ \ell_{m+1} : \textbf{halt}$$

where every ins_i is either a *Type I* instruction of the form

$$\textbf{inc } c_j; \ \textbf{goto } \ell_k$$

where $j \in \{1, 2\}$ and $1 \leq k \leq m + 1$, or a *Type II* instruction of the form

$$\textbf{if } c_j{=}0 \textbf{ then goto } \ell_k \textbf{ else dec } c_j; \ \textbf{goto } \ell_n$$

where $j \in \{1, 2\}$ and $1 \leq k, n \leq m + 1$.

A computation of $\mathcal{M}$ starts by executing the first instruction with both counters c_1 and c_2 initialized to zero. The problem of determining whether $\mathcal{M}$ *halts*, i.e., eventually executes the **halt** instruction, is undecidable [51]. Consequently, the problem of whether $\mathcal{M}$ does *not* halt is not even semi-decidable.

We prove our theorem by reducing the non-halting problem for two-counter Minsky machines to the qualitative model checking problem. For a given Minsky machine $\mathcal{M}$, let L_I and L_{II} be the sets of all indices $i \in \{1, \ldots, m\}$ such that ins_i is a Type I and Type II instruction, respectively. We construct a replicated system $\mathcal{P} = (Q, T)$ where

$$Q \overset{\text{def}}{=} \{q_1, \ldots, q_{m+1}, Z_1, O_1, Z_2, O_2\} \cup \{\hat{q}_i \mid i \in L_{II}\},$$

and T is the (least) set of transitions satisfying the following:

- For every Type I instruction of the form

  $$\ell_i : \ \textbf{inc } c_j; \ \textbf{goto } \ell_k$$

  there is a transition $q_i\, Z_j \mapsto q_k\, O_j$.
- For every Type II instruction of the form

  $$\ell_i : \ \textbf{if } c_j{=}0 \textbf{ then goto } \ell_k \textbf{ else dec } c_j; \ \textbf{goto } \ell_n$$

  there are transitions $q_i\, Z_j \mapsto \hat{q}_i\, Z_j$, $\hat{q}_i\, Z_j \mapsto q_k\, Z_j$, and $q_i\, O_j \mapsto q_n\, Z_j$.

Consider the following Presburger formulas:

$$\begin{aligned}
\mathit{Init} &\overset{\text{def}}{=} q_1{=}1 \ \wedge\ Z_1{\geq}1 \ \wedge\ Z_2{\geq}1 \ \wedge \bigwedge_{q\in Q\setminus\{q_1,Z_1,Z_2\}} q{=}0\\
\mathit{Overflow} &\overset{\text{def}}{=} \bigvee_{i\in L_I} (q_i{=}1 \wedge Z_{j_i}{=}0)\\
\mathit{Cheat} &\overset{\text{def}}{=} \bigvee_{i\in L_{II}} (\hat{q}_i{=}1 \wedge O_{j_i}{\geq}1)
\end{aligned}$$

In the above formulas, we use $j_i \in \{1,2\}$ to denote the counter used by instruction ins_i. Furthermore, let $\mathcal{I} \overset{\text{def}}{=} [\![\mathit{Init}]\!]$ and

$$\varphi \overset{\text{def}}{=} \Box(q_{m+1}{=}0) \ \vee\ \Diamond(\mathit{Overflow} \vee \mathit{Cheat}).$$

We claim that $\mathcal{M}$ does not halt iff $\Pr[C,\varphi] = 1$ for every configuration $C \in \mathcal{I}$.

Suppose $\mathcal{M}$ does not halt. Let $C \in \mathcal{I}$. As C satisfies the formula *Init*, it has precisely one agent in state q_1, at least one agent in each state Z_1 and Z_2, and no agents elsewhere. The transitions of $\mathcal{P}$ are constructed so that they allow for simulating $\mathcal{M}$ from C. In every configuration C' reachable from C, there is precisely one agent in a state of $\{q_1,\ldots,q_{m+1}\} \cup \{\hat{q}_i \mid i \in L_{II}\}$, and the values of c_1 and c_2 are represented by $C'(O_1)$ and $C'(O_2)$, respectively. Since $C'(O_1)$ and $C'(O_2)$ are bounded, the simulation may fail due to a *counter overflow* when some Type I instruction tries to increase a counter c_j (i.e., rewrite Z_j into O_j) but no agent in Z_j is available. This is captured by the formula *Overflow*. Furthermore, the simulation of a Type II instruction is not necessarily faithful, because the transition $q_i\, Z_j \mapsto \hat{q}_i\, Z_j$ can be executed even if there is an agent in state O_j in the current configuration (i.e., the corresponding counter value is positive). This is detected by the formula *Cheat*. Hence, if $\mathcal{M}$ does not halt, then every run initiated in C either does *not* correspond to a faithful simulation of $\mathcal{M}$, i.e., visits a configuration satisfying *Overflow* or *Cheat*, or simulates $\mathcal{M}$ faithfully, i.e., the state q_{m+1} does not occur in any configuration visited by the run. Hence, all runs initiated in C satisfy the formula φ.

If $\mathcal{M}$ halts, then the instruction **halt** is executed after a *finite* computation along which the counters are increased only to some finite values. Hence, for all sufficiently large n, there exist a configuration $C \in \mathcal{I}$ with $C(Z_1) = C(Z_2) = n$ and a finite path initiated in C corresponding to a faithful simulation of $\mathcal{M}$. Note that the last configuration C' of this path (where $C'(q_{m+1}) = 1$) has only one successor C', i.e., the self-loop $C' \to C'$ is inevitably selected with probability 1. The probability of executing this path (and the associated run) is positive, and the run does not satisfy the formula φ. Hence, $\Pr[C,\varphi] < 1$.

A.2 Missing proofs for Section 3

Proposition 2. *For all inductive sets $\mathcal{C}, \mathcal{C}'$ of configurations, it is the case that: $\mathcal{C}$ leads to $\mathcal{C}'$ iff there exists a certificate for $\mathcal{C} \rightsquigarrow \mathcal{C}'$.*

Proof. ($\Rightarrow$): Assume $\mathcal{C}$ leads to $\mathcal{C}'$. By definition of the "leads to" relation, for every $C \in \mathcal{C}$, there exists $C' \in \mathcal{C}'$ such that $C \xrightarrow{*} C'$. Hence, the function defined by $f(C) = 1$ if $C \in \mathcal{C} \setminus \mathcal{C}'$, and $f(C) = 0$ otherwise is a certificate for $\mathcal{C} \rightsquigarrow \mathcal{C}'$.

($\Leftarrow$): Assume there is a certificate for $\mathcal{C} \rightsquigarrow \mathcal{C}'$. We claim that for every $C \in \mathcal{C}$, there exists $C' \in \mathcal{C}'$ such that $C \xrightarrow{*} C'$. Assume the contrary. By assumption, the definition of certificates and as $\mathcal{C}$ is inductive, there are configurations $C_0, C_1, \ldots \in \mathcal{C} \setminus \mathcal{C}'$ such that $C = C_0 \xrightarrow{*} C_1 \xrightarrow{*} \cdots$ and $f(C_i) > f(C_{i+1})$ for every $i \geq 0$. This is impossible as the codomain of f is $\mathbb{N}$, which proves the claim.

We may now prove that $\mathcal{C}$ leads to $\mathcal{C}'$. Let ρ be a fair run starting at some configuration of $\mathcal{C}$. Since $\mathcal{C}$ is inductive, ρ only visits configurations of $\mathcal{C}$. Further, since all configurations visited by ρ have the same size, some configuration $C \in \mathcal{C}$ is visited infinitely often. By the claim, there exists a sequence $C \xrightarrow{\sigma} C'$ such that $C' \in \mathcal{C}'$. We show that ρ visits C', by induction on $|\sigma|$. If $|\sigma| = 0$, then $C' = C$ and we are done. Assume $\sigma = t\tau$ and $C \xrightarrow{t} D \xrightarrow{\tau} C'$, where $t \in T$. By fairness, D occurs infinitely often in ρ. Since $|\tau| = |\sigma| - 1$, we can apply the induction hypothesis to τ and conclude that C' occurs infinitely often in ρ. $\square$

Proposition 3. *System $\mathcal{P}$ satisfies Π iff it has a stage graph for Π.*

Proof. ($\Leftarrow$): Assume $\mathcal{P}$ has a stage graph for Π. Let $\mathcal{B}$ be a terminal stage of the stage graph. By condition 4, $\mathcal{B} \models \varphi^i_{\text{post}}$ for some i, and by inductiveness $\mathcal{B} \models \Box\varphi^i_{\text{post}}$ Let $\mathcal{L}$ be the union of the terminal stages of the stage graph. We have $\mathcal{L} \models \bigvee_{i=1}^{k} \Box\varphi^i_{\text{post}}$. By Proposition 2 and conditions 1 and 3 of the definition of a stage graph, every stage $\mathcal{C}$ satisfies $\mathcal{C} \rightsquigarrow \mathcal{L}$. Therefore every stage $\mathcal{C}$ satisfies $\Diamond \bigvee_{i=1}^{k} \Box\varphi^i_{\text{post}}$. By condition 2, we have that $[\![\varphi_{\text{pre}}]\!] \models \Diamond \bigvee_{i=1}^{k} \Box\varphi^i_{\text{post}}$. Thus $C \models \Diamond \bigvee_{i=1}^{k} \Box\varphi^i_{\text{post}}$ for any configuration $C \in [\![\varphi_{\text{pre}}]\!]$, and hence $\mathcal{P} \models \varphi_\Pi$.

($\Rightarrow$): Assume $\mathcal{P}$ satisfies Π. Consider a stage graph with $k+1$ stages: an initial stage $\mathcal{C}_{in}$ containing the set of all configurations reachable from $[\![\varphi_{\text{pre}}]\!]$, and a terminal stage $\mathcal{C}_{f_i}$, for every $1 \leq i \leq k$, containing all configurations satisfying $\Box\varphi^i_{\text{post}}$. Conditions 1, 2, and 4 hold by definition. Since $\mathcal{P}$ satisfies Π, every fair run from a configuration of the initial stage $\mathcal{C}_{in}$ eventually visits a terminal stage $\mathcal{C}_{f_i}$, and therefore $\mathcal{C}_{in}$ leads to $(\mathcal{C}_{f_1} \cup \ldots \cup \mathcal{C}_{f_k})$. Consequently, Proposition 2 yields a certificate for $\mathcal{C}_{in} \rightsquigarrow (\mathcal{C}_{f_1} \cup \ldots \cup \mathcal{C}_{f_k})$. $\square$

Theorem 2. *System $\mathcal{P}$ satisfies Π iff it has a Presburger stage graph for Π.*

Proof. We say that a configuration C is *bottom* if $C \xrightarrow{*} D$ implies $D \xrightarrow{*} C$. Let $\mathcal{B}$ be the set of all bottom configurations of $\mathcal{P}$. Let $\overset{*}{\leftrightarrow}$ denote the *mutual reachability relation* defined by $C \overset{*}{\leftrightarrow} D \overset{\text{def}}{\iff} (C \xrightarrow{*} D \land D \xrightarrow{*} C)$. It is known from [31, Thm. 13 and Prop. 14] that both $\overset{*}{\leftrightarrow}$ and $\mathcal{B}$ are (effectively) Presburger. Let $\mathcal{S}_i \overset{\text{def}}{=} \mathcal{B} \cap [\![\Box\varphi^i_{\text{post}}]\!]$ for every $i \in [n]$. Note that $\mathcal{S}_i$ is Presburger since it can be written as

$$\mathcal{S}_i = \left\{ C \in \mathcal{B} : \forall D \left[(C \overset{*}{\leftrightarrow} D) \implies (D \models \varphi^i_{\text{post}}) \right] \right\}.$$

Let $\mathcal{S} \overset{\text{def}}{=} \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n$ and let $\mathcal{I} \overset{\text{def}}{=} [\![\varphi_{\text{pre}}]\!]$. Note that $\mathcal{S}$ and $\mathcal{I}$ are Presburger. Since $\mathcal{P}$ satisfies Π, we have $\mathit{post}^*(\mathcal{I}) \cap (\mathcal{B} \setminus \mathcal{S}) = \emptyset$. Therefore, by [47, Lem. 9.1], there exists an inductive Presburger set $\mathcal{I}' \supseteq \mathcal{I}$ such that $\mathit{post}^*(\mathcal{I}') \cap (\mathcal{B} \setminus \mathcal{S}) = \emptyset$. Since any set of configurations leads to $\mathcal{B}$, this implies $\mathcal{I}' \rightsquigarrow \mathcal{S}$.

The directed acyclic graph made of $\mathcal{I}'$ with $\mathcal{S}_1, \ldots, \mathcal{S}_n$ as its successors is a stage graph for Π. Indeed:

1. $\mathcal{I}'$ and $\mathcal{S}_i$ are inductive, where the latter follows by definition;
2. $[\![\varphi_{\text{pre}}]\!] = \mathcal{I}$ is a subset of stage $\mathcal{I}'$;
3. $\mathcal{I}' \rightsquigarrow \mathcal{S} = (\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n)$ holds, by the above; and
4. $\mathcal{S}_i \models \varphi^i_{\text{post}}$ by definition of $\mathcal{S}_i$.

It remains to exhibit a Presburger certificate. Since $\mathcal{I}'$ and $\mathcal{S}$ are both Presburger, [48, Cor. XI.3] yields a bounded language $L = w_1^* w_2^* \cdots w_k^* \subseteq T^*$ such that $\mathcal{I}' \subseteq \mathit{pre}_L(\mathcal{S})$. Let $\mathit{pre}_L(\mathcal{S})$ be the set of configurations C such that $C \xrightarrow{w} C'$ for some $C' \in \mathcal{S}$ and $w \in L$, and L' be the language made of all sequences of L and their suffixes. We have $\mathcal{I}' \subseteq \mathit{pre}_L(\mathcal{S}) \subseteq \mathit{pre}_{L'}(\mathcal{S})$. For every $C \in \mathcal{I}'$, let $f(C) \overset{\text{def}}{=} |\sigma_C|$ where $\sigma_C \in L'$ is a shortest sequence such that:

$$C \xrightarrow{\sigma_C} D \text{ for some } D \in \mathcal{S}.$$

Since $\mathcal{I}'$ is inductive and since L' is closed under suffixes, if $\sigma_C = t\tau$ and $C \xrightarrow{t} C'$, then we have $f(C) = f(C') + 1$. Hence, $(f, 1)$ is a bounded certificate for $\mathcal{I}' \rightsquigarrow \mathcal{S}$.

It remains to construct a Presburger formula $\varphi(C, \ell)$ that holds iff $\ell = f(C)$. We only consider the case where $L = w^*$ for some finite sequence w; the generalization to $L = w_1^* w_2^* \cdots w_k^*$ being straightforward.

A simple induction on the length of w shows that the set of configurations that enable w has a unique minimal configuration C_w. Further, also by induction on w there exists a vector $\Delta(w) \in \mathbb{Z}^Q$ such that $C \xrightarrow{w} C + \Delta(w)$ for every $C \geq C_w$. More precisely, $\Delta(w) = \sum_{i=1}^{|w|} \Delta(w_i)$. It follows that a configuration C enables sequence w^k iff $C + i \cdot \Delta(w) \geq C_w$ for every $0 \leq i < k$. Furthermore, if C enables w^k, then we have:

$$C \xrightarrow{w^k} C + k \cdot \Delta(w).$$

This condition is expressible by a Presburger formula $\mathit{enab}_w(C, k)$. Let $\mathit{suff}(w)$ denote the set of sufffixes of w, and let

$$\mathit{FS}(C, \ell) \overset{\text{def}}{=} \bigvee_{w' \in \mathit{suff}(w)} \mathit{enab}_{w'}(C, 1) \wedge \exists k \colon \begin{array}{l} \mathit{enab}_w(C + \Delta(w'), k) \quad \wedge \\ C + \Delta(w') + k \cdot \Delta(w) \in \mathcal{S} \wedge \\ \ell = |w'| + k. \end{array}$$

Formula $\mathit{FS}(C, \ell)$ holds iff there exists a sequence $\sigma \in L'$ of length ℓ such that $C \xrightarrow{\sigma} C'$ and $C' \in \mathcal{S}$. Let

$$\varphi(C, \ell) \overset{\text{def}}{=} \mathit{FS}(C, \ell) \wedge \forall \ell' \colon \mathit{FS}(C, \ell') \rightarrow (\ell' \geq \ell).$$

Formula $\varphi(C, \ell)$ holds iff ℓ is the length of a shortest sequence $\sigma \in L'$ such that $C \xrightarrow{\sigma} C'$ and $C' \in \mathcal{S}$, as desired. $\square$

Theorem 3. *The problem of deciding whether an acyclic graph of Presburger sets and Presburger certificates is a Presburger stage graph, for a given stable termination property, is reducible in polynomial time to the satisfiability problem for Presburger arithmetic.*

Proof. First we observe that for any two configurations C, C', we have $C \to C'$ iff there exists a transition t such that $C \geq {}^\bullet t$ and $C' = C + \Delta(t)$. With that, checking the inductiveness of a Presburger stage $\mathcal{S}$ reduces to checking satisfiability of this sentence:

$$\forall C\colon C \in \mathcal{S} \to \bigwedge_{t \in T} \left(C \geq {}^\bullet t \to (C + \Delta(t)) \in \mathcal{S}\right).$$

Checking whether $[\![\varphi_{\text{pre}}]\!]$ is included in the union of all stages reduces to checking satisfiability of this sentence:

$$\forall C\colon \varphi_{\text{pre}}(C) \to \bigvee_{\text{stage } \mathcal{S}} C \in \mathcal{S}.$$

Let $\mathcal{S}$ be a terminal stage of the graph. Checking that $\mathcal{S} \models \varphi^i_{\text{post}}$ for some i reduces to checking satisfiablity of this sentence:

$$\bigvee_{i=1}^{k} \forall C : C \in \mathcal{S} \to \varphi^i_{\text{post}}(C).$$

Let (f, k) be a Presburger certificate and $\varphi(\mathbf{x}, y)$ be the existential Presburger formula given for f. Checking that φ actually describes some function f reduces to checking satisfiability of this sentence:

$$(\forall C\colon \exists y\colon \varphi(C, y)) \wedge (\forall C, y, y'\colon\ (\varphi(C, y) \wedge \varphi(C, y')) \to y = y').$$

Since we have the silent transition $(\wr\wr, \wr\wr) \in T$ that is always enabled, it suffices to check (f, k) for sequences of length exactly k instead of at most k. We now obtain that (f, k) is a Presburger certificate for $\mathcal{S} \rightsquigarrow (\mathcal{S}_1 \cup \ldots \cup \mathcal{S}_n)$ iff the following sentence is satisfiable:

$$\forall C_0, y\colon \exists C_1, \ldots, C_k, y'\colon \left[\varphi(C, y) \wedge C \in \mathcal{S} \wedge \neg \bigvee_{i=1}^{n} C \in \mathcal{S}_i\right] \to$$
$$\left[\varphi(C_k, y') \wedge y > y' \wedge \bigwedge_{i=0}^{k-1} \bigvee_{t \in T} (C_i \geq {}^\bullet t \wedge C_{i+1} = C_i + \Delta(t))\right].$$

Determining if the given graph is a Presburger stage for Π now amounts to determining satisfiability of the conjunction of all the constructed Presburger sentences. We note that if all Presburger formulas for stages and certificates are quantifier-free and the bounds k on the certificates are given in unary, then the constructed Presburger sentences are of polynomial size and in the $\forall\exists$ fragment. $\square$

A.3 Missing proofs for Section 4

Proposition 4. *Given a replicated system $\mathcal{P}$, a stage $\mathcal{S}$ represented by an existential Presburger formula ϕ and a set of transitions U, determining whether $\mathcal{S} \models \Diamond dead(U)$ holds is decidable and* PSPACE*-hard.*

Proof. Let us first establish decidability. It suffices to show decidability of $\mathcal{S} \not\models \Diamond\text{dead}(U)$, i.e., whether $C \not\models \Diamond\text{dead}(U)$ for some $C \in \mathcal{S}$. Observe that

$$\begin{aligned} C \not\models \Diamond\text{dead}(U) &\iff C \not\models \Diamond \bigwedge_{t \in U} \text{dead}(t) \\ &\iff C \models \Box \bigvee_{t \in U} \neg\text{dead}(t). \end{aligned}$$

In other words, $C \not\models \Diamond\text{dead}(U)$ holds iff at every configuration C' reachable from C, some transition from U is enabled at C'. It has been observed in [40, Sect. 4] that this notion of liveness is equivalent to liveness of a *single* transition. Indeed, it suffices to introduce a new transition $t^\dagger$ and two new states $p^\dagger, q^\dagger$ such that:

- a single agent is initially in state $p^\dagger$, while none is in state $q^\dagger$;
- each transition from U is altered so that it takes an agent in state $p^\dagger$ and moves it to state $q^\dagger$;
- $t^\dagger$ moves an agent in state $q^\dagger$ to state $p^\dagger$.

This way, $C \models \Box \bigvee_{t \in U} \neg\text{dead}(t)$ holds in the original system iff $C \models \Box\neg\text{dead}(t^\dagger)$ holds in the altered system. Moreover, the alteration of $\mathcal{S}$ remains semilinear as one can simply add the conjunct $(p^\dagger = 1 \land q^\dagger = 0)$ to ϕ.

By [40, Thm. 2], the problem of determining whether $\mathcal{C} \models \Box\neg\text{dead}(t^\dagger)$ holds for some $\mathcal{C} \in \mathcal{S}$, is decidable. Although the statement of [40, Thm. 2] only covers the specific case of $\mathcal{S} = \mathbb{N}^Q$, its proof explicitly handles any effective semilinear set $\mathcal{S}$. Therefore, this establishes decidability of our problem.

Let us now show PSPACE-hardness. Observe that replicated systems are Petri nets where states correspond to places and where transitions correspond to transitions and arcs. In fact, replicated systems amount *precisely* to the class of *1-conservative* Petri nets, i.e., where every transition produces as many tokens as it consumes. Since the reachability problem for 1-conservative Petri nets is PSPACE-complete [41], the same holds for the reachability problem for replicated systems defined as:

INPUT: a replicated system $\mathcal{P} = (Q, T)$ and configurations $C, C' \in \mathbb{N}^Q$,
OUTPUT: does $C \xrightarrow{*} C'$ hold?

We give a (many-one) reduction from this problem to the following variant of the *partial structural liveness problem* for replicated systems:

INPUT: a replicated system $\mathcal{P} = (Q, T)$ and a transition $t \in T$,
OUTPUT: does $\mathbb{N}^Q \not\models \Diamond\text{dead}(t)$ hold?

Let us fix a replicated system $\mathcal{P} = (Q, T)$ and configurations $C_{\text{init}}, C_{\text{tgt}} \in \mathbb{N}^Q$. We design a replicated system $\mathcal{P}' = (Q', T')$ and a transition $t_{\text{tgt}} \in T$ such that:

$$C_{\text{init}} \xrightarrow{*} C_{\text{tgt}} \text{ in } \mathcal{P} \iff \mathbb{N}^{Q'} \not\models \Diamond\text{dead}(t_{\text{tgt}}) \text{ in } \mathcal{P}'.$$

The validity of this equivalence proves the proposition as it is a special case of the problem we wish to show PSPACE-hard. Note that we are implicitly using the fact that PSPACE = NPSPACE as we deal with "$\not\models$" instead of "$\models$".

Construction. Let us describe $\mathcal{P}'$. Its set of states is defined as

$$Q' \overset{\text{def}}{=} Q \cup \{q_{\text{free}}, q_{\text{out}}\},$$

where q_{free} indicates that a "retired" agent is "free" to move to a state of Q, and where q_{out} indicates that a "retired" agent is permanently retired.

Let $k \overset{\text{def}}{=} |C_{\text{init}}|$. Let $t_{\text{init}} \overset{\text{def}}{=} \wr k \cdot q_{\text{free}} \wr \mapsto C_{\text{init}}$ and let $t_{\text{tgt}} \overset{\text{def}}{=} C_{\text{tgt}} \mapsto C_{\text{tgt}}$. The purpose of these transitions is respectively to generate the initial configuration C_{init}, and to check whether the target configuration C_{tgt} is present. Let

$$t_{\text{clean}} \overset{\text{def}}{=} \wr k \cdot q_{\text{free}}, q_{\text{free}} \wr \mapsto \wr k \cdot q_{\text{free}}, q_{\text{out}} \wr.$$

The purpose of transition t_{clean} is to permanently retire agents until there are at most k remaining. Let $S \overset{\text{def}}{=} \{s_q : q \in Q\}$ where $s_q \overset{\text{def}}{=} \wr q \wr \mapsto \wr q_{\text{free}} \wr$. Transitions S make the system "lossy" in the sense that agents can non deterministically retire from Q, either temporarily or eventually permanently. Overall, the set of transitions of $\mathcal{P}'$ is defined as

$$T' \overset{\text{def}}{=} T \cup S \cup \{t_{\text{clean}}, t_{\text{init}}, t_{\text{tgt}}\}.$$

General idea. Let us explain the idea behind the construction of $\mathcal{P}'$, which is illustrated in Figure 3. If C_{tgt} is reachable from C_{init} in $\mathcal{P}$, then this is also the case in $\mathcal{P}'$, as it can simulate the former. Moreover, if $\mathcal{P}'$ gets stuck by choosing the wrong transitions, then this does not yield a dead configuration, as lossy transitions S can temporarily retire agents so that t_{init} resets the system to C_{init}. This way, transition t_{tgt} can occur infinitely often from any reachable configuration.

Since $\mathcal{P}'$ could in principle start from a configuration that differs from C_{init}, there is a risk that t_{tgt} occurs infinitely often even though $C_{\text{init}} \xrightarrow{*} C_{\text{tgt}}$ does *not* hold in $\mathcal{P}$. Thus, the role of t_{clean} is to permanently retire agents until at most k can move to Q. This ensures that $\mathcal{P}'$ eventually either sets its non retired agents to C_{init}, or gets stuck. The latter only happens if there are *less* that k agents.

Proof of the reduction. Let us prove the claim formally.

$\Rightarrow$) Assume $C_{\text{init}} \xrightarrow{*} C_{\text{tgt}}$ holds in $\mathcal{P}$. Let us show that $C_{\text{init}} \not\models \Diamond\text{dead}(t_{\text{tgt}})$ in $\mathcal{P}'$. Let D be some configuration of $\mathcal{P}'$ such that $C_{\text{init}} \xrightarrow{*} D$. We must show that D

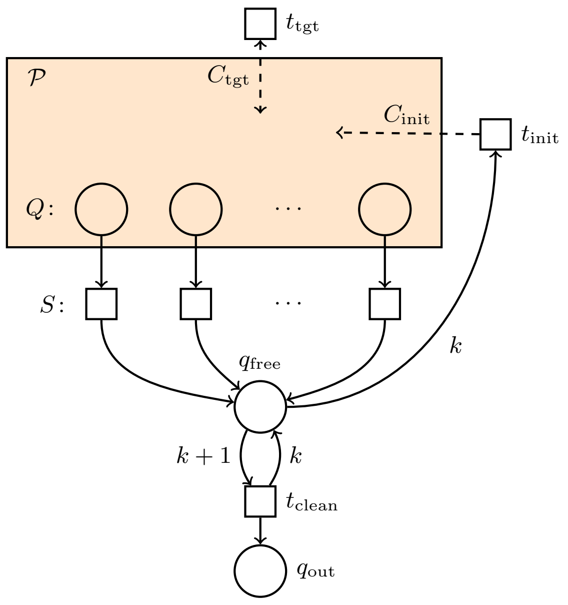

Fig. 3: Replicated system $\mathcal{P}'$ depicted as a (1-conservative) Petri net.

can reach C_{tgt}, which allows t_{tgt} to occur. Since D is arbitrary, the validity of this claim implies that t_{tgt} is not dead at any reachable configuration.

Observe that t_{clean} cannot occur at any reachable configuration as there are k agents, while t_{clean} requires $k+1$ agents. By definition of S and since $D(Q' \setminus \{q_{\text{out}}\}) = k$, we have $D \xrightarrow{*} \wr k \cdot q_{\text{free}} \int$. Since t_{init} can occur from the latter, we have $D \xrightarrow{*} C_{\text{init}}$. As $\mathcal{P}'$ contains all transitions from $\mathcal{P}$, this implies that $D \xrightarrow{*} C_{\text{tgt}}$ in $\mathcal{P}'$. Hence, t_{tgt} can occur from there.

$\Leftarrow$) Assume $C_{\text{init}} \xrightarrow{*} C_{\text{tgt}}$ does *not* hold in $\mathcal{P}$. Let us show that $\mathbb{N}^{Q'} \models \Diamond\text{dead}(t_{\text{tgt}})$ in $\mathcal{P}'$. Let $D_{\text{init}} \in \mathbb{N}^{Q'}$. We must argue "adversarially" that D_{init} can reach a configuration D at which t_{tgt} is dead.

By using "lossy" transitions S repeatedly, we can remove all agents from Q. Hence, $D_{\text{init}} \xrightarrow{*} \wr a \cdot q_{\text{free}}, b \cdot q_{\text{out}} \int$ for some $a, b \in \mathbb{N}$. If $a > k$, then using transition t_{clean} repeatedly, we obtain k agents in state q_{free} and $b + (a-k)$ agents in state q_{out}. In other words, we have

$$D_{\text{init}} \xrightarrow{*} \wr a' \cdot q_{\text{free}}, b' \cdot q_{\text{out}} \int \text{ where } a' \leq k \text{ and } b' \in \mathbb{N}.$$

If $a' < k$, then *all* transitions are dead and we are done. Hence, let us assume that $a' = k$. The only enabled transition at this point is t_{init}, which forces $\mathcal{P}'$ to move to configuration $D \overset{\text{def}}{=} C_{\text{init}} + \wr b' \cdot q_{\text{out}} \int$. Note that t_{clean} is dead at D. Since C_{tgt} is not reachable from C_{init} in $\mathcal{P}$, system $\mathcal{P}'$ cannot reach $C_{\text{tgt}} + \wr b' \cdot q_{\text{out}} \int$ either. Moreover, it cannot reach any configuration larger than $C_{\text{tgt}} + \wr b' \cdot q_{\text{out}} \int$ as the number of agents never changes. Thus, t_{tgt} is dead at D, which completes the proof. $\square$

A.4 Missing proofs for Section 5

Proposition 6. *For every Presburger set $\mathcal{C}$ and every set of transitions U, the set $post_U(\mathcal{C})$ is effectively Presburger.*

Proof. We use the fact that $C \xrightarrow{t} C'$ iff $C \in \overline{[\![\mathrm{dis}(t)]\!]}$ and $C' = C + \Delta(t)$, for every $C, C' \in \mathbb{N}^Q$ and $t \in T$. Then $C' \in post_U(\mathcal{C})$ holds iff the following holds:

$$\begin{aligned}
&\exists C \in \mathcal{C} : \bigvee_{t \in U} \left(C \in \overline{[\![\mathrm{dis}(t)]\!]} \wedge C' = C + \Delta(t) \right) \\
\equiv &\bigvee_{t \in U} \left((C' - \Delta(t)) \in \mathcal{C} \wedge (C' - \Delta(t)) \in \overline{[\![\mathrm{dis}(t)]\!]} \right).
\end{aligned}$$

Proposition 7. *For every $k \geq 1$ there is an existential Presburger formula $DeathCert_k(U, \mathcal{C}^\omega)$ that holds iff $\mathcal{C}^\omega$ is a death certificate of size k for U.*

Proof. Let $\mathcal{C}^\omega = \{C_1^\omega, \ldots, C_k^\omega\}$. By definition, we have that $\mathcal{C}^\omega$ is a death certificate for U iff $\downarrow\mathcal{C}^\omega \models \mathrm{dis}(U)$ and $post_T(\downarrow\mathcal{C}^\omega) \subseteq \downarrow\mathcal{C}^\omega$. We easily have

$$\downarrow\mathcal{C}^\omega \models \mathrm{dis}(U) \equiv \bigwedge_{i=1}^{k} \bigwedge_{u \in U} \neg \left(C_i^\omega \geq {}^\bullet u \right).$$

Using the constraint from the proof of Proposition 6, we rewrite the inductivity constraint as follows:

$$\begin{aligned}
&post_T(\downarrow\mathcal{C}^\omega) \subseteq \downarrow\mathcal{C}^\omega \\
\equiv\ &\forall C' : C' \in post_T(\downarrow\mathcal{C}^\omega) \Rightarrow C' \in \downarrow\mathcal{C}^\omega \\
\equiv\ &\forall C' : \left(\bigvee_{t \in T} \left((C' - \Delta(t)) \in \downarrow\mathcal{C}^\omega \wedge (C' - \Delta(t)) \in \overline{[\![\mathrm{dis}(t)]\!]} \right) \right) \Rightarrow C' \in \downarrow\mathcal{C}^\omega \\
\equiv\ &\forall C : \bigwedge_{t \in T} \left(\left(C \in \downarrow\mathcal{C}^\omega \wedge C \in \overline{[\![\mathrm{dis}(t)]\!]} \right) \Rightarrow (C + \Delta(t)) \in \downarrow\mathcal{C}^\omega \right)
\end{aligned}$$

As $\downarrow\mathcal{C}^\omega$ is downward closed, it suffices to check the constraint for all elements in the decomposition of $\downarrow\mathcal{C}^\omega$, i.e., C_1^ω to C_k^ω. This gives us the following formula:

$$\begin{aligned}
&\bigwedge_{i=1}^{k} \bigwedge_{t \in T} \left(C_i^\omega \in \overline{[\![\mathrm{dis}(t)]\!]} \Rightarrow (C_i^\omega + \Delta(t)) \in \downarrow\mathcal{C}^\omega \right) \\
\equiv &\bigwedge_{i=1}^{k} \bigwedge_{t \in T} \left(C_i^\omega \geq {}^\bullet t \Rightarrow \bigvee_{j=1}^{k} (C_i^\omega + \Delta(t)) \leq C_j^\omega \right).
\end{aligned}$$

Together we obtain the following Presburger formula for $DeathCert_k(U, \mathcal{C}^\omega)$:

$$\left(\bigwedge_{i=1}^{k} \bigwedge_{u \in U} \neg \left(C_i^\omega \geq {}^\bullet u \right) \right) \wedge \left(\bigwedge_{i=1}^{k} \bigwedge_{t \in T} \left(C_i^\omega \geq {}^\bullet t \Rightarrow \bigvee_{j=1}^{k} (C_i^\omega + \Delta(t)) \leq C_j^\omega \right) \right).$$

For a fixed k, the size of the formula is polynomial w.r.t. the size of the system.

A.5 Missing proofs for Section 7

Proposition 10. *A set U of transitions satisfies $[\![dis(U)]\!] = [\![dead(U)]\!]$ iff it satisfies the existential Presburger formula*

$$\textit{dis-eq-dead}(U) \overset{\textit{def}}{=} \bigwedge_{t \in T} \bigwedge_{u \in U} \bigvee_{u' \in U} {}^\bullet t + ({}^\bullet u \ominus t^\bullet) \geq {}^\bullet u'$$

where $\boldsymbol{x} \ominus \boldsymbol{y} \in \mathbb{N}^Q$ is defined by $(\boldsymbol{x} \ominus \boldsymbol{y})(q) \overset{\textit{def}}{=} \max(\boldsymbol{x}(q) - \boldsymbol{y}(q), 0)$ for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{N}^Q$.

Proof. We have that $[\![\mathrm{dis}(U)]\!] = [\![\mathrm{dead}(U)]\!]$ iff $[\![\mathrm{dis}(U)]\!]$ is inductive, that is $\mathit{post}_T([\![\mathrm{dis}(U)]\!]) \subseteq [\![\mathrm{dis}(U)]\!]$. We show that

$$\mathit{post}_T([\![\mathrm{dis}(U)]\!]) \subseteq [\![\mathrm{dis}(U)]\!] \equiv \bigwedge_{t \in T} \bigwedge_{u \in U} \bigvee_{u' \in U} {}^\bullet t + ({}^\bullet u \ominus t^\bullet) \geq {}^\bullet u'.$$

We start by rewriting the formula as follows:

$$\begin{aligned}
&\mathit{post}_T([\![\mathrm{dis}(U)]\!]) \subseteq [\![\mathrm{dis}(U)]\!] \\
\equiv\ &\forall C' : C' \in \mathit{post}_T([\![\mathrm{dis}(U)]\!]) \Rightarrow C' \in [\![\mathrm{dis}(U)]\!] \\
\equiv\ &\forall C' : \left(\bigvee_{t \in T} \left((C' - \Delta(t)) \in [\![\mathrm{dis}(U)]\!] \wedge (C' - \Delta(t)) \in \overline{[\![\mathrm{dis}(t)]\!]} \right) \right) \Rightarrow C' \in [\![\mathrm{dis}(U)]\!] \\
\equiv\ &\forall C : \bigwedge_{t \in T} \left(\left(C \in [\![\mathrm{dis}(U)]\!] \wedge C \in \overline{[\![\mathrm{dis}(t)]\!]} \right) \Rightarrow (C + \Delta(t)) \in [\![\mathrm{dis}(U)]\!] \right) \\
\equiv\ &\forall C : \bigwedge_{t \in T} \left(\left(C \in \overline{[\![\mathrm{dis}(t)]\!]} \wedge (C + \Delta(t)) \in \overline{[\![\mathrm{dis}(U)]\!]} \right) \Rightarrow C \in \overline{[\![\mathrm{dis}(U)]\!]} \right).
\end{aligned}$$

Let $\mathcal{Y}(U, t) \overset{\text{def}}{=} \{C \mid C \in \overline{[\![\mathrm{dis}(t)]\!]} \wedge (C + \Delta(t)) \in \overline{[\![\mathrm{dis}(U)]\!]}\}$. The above formula can be rewritten as:

$$\forall C : \bigwedge_{t \in T} \left(C \in \mathcal{Y}(U, t) \Rightarrow C \in \overline{[\![\mathrm{dis}(U)]\!]} \right) \ \equiv \bigwedge_{t \in T} \mathcal{Y}(U, t) \subseteq \overline{[\![\mathrm{dis}(U)]\!]}.$$

Observe that $\mathcal{Y}(U, t)$ is upward closed, as both $\overline{[\![\mathrm{dis}(t)]\!]}$ and $\overline{[\![\mathrm{dis}(U)]\!]}$ are upward closed. Therefore, the inclusion check is between two upward closed sets, which amounts to a comparison of their bases. We claim that $\uparrow \mathcal{X}(U, t) = \mathcal{Y}(U, t)$ where

$$\mathcal{X}(U, t) \overset{\text{def}}{=} \{{}^\bullet t + ({}^\bullet u \ominus t^\bullet) \mid u \in U\}.$$

Let us prove the claim. Let $C = {}^\bullet t + ({}^\bullet u \ominus t^\bullet) \in \mathcal{X}(U, t)$ for some $u \in U$. Clearly, $C \in \overline{[\![\mathrm{dis}(t)]\!]}$ since $C \geq {}^\bullet t$. Note that $C + \Delta(t) = t^\bullet + ({}^\bullet u \ominus t^\bullet) \geq {}^\bullet u$. Therefore, $C + \Delta(t) \in \overline{[\![\mathrm{dis}(U)]\!]}$ and consequently $C \in \mathcal{Y}(U, t)$. Since this also holds for any configuration $C' \geq C$, we obtain $\uparrow \mathcal{X}(U, t) \subseteq \mathcal{Y}(U, t)$.

For the other inclusion, let $C \in \mathcal{Y}(U, t)$. We have $C + \Delta(t) \geq {}^\bullet u$ for some $u \in U$ and hence follows $\mathcal{Y}(U, t) \subseteq \uparrow \mathcal{X}(U, t)$ by

$$C = C + \Delta(t) - \Delta(t) \geq {}^\bullet u + {}^\bullet t - t^\bullet \geq {}^\bullet t + ({}^\bullet u \ominus t^\bullet) \in \mathcal{X}(U, t).$$

We now get the following final formula:

$$\bigwedge_{t \in T} \bigwedge_{C \in \mathcal{X}(U,t)} C \in \overline{[\![\mathrm{dis}(U)]\!]} \equiv \bigwedge_{t \in T} \bigwedge_{u \in U} \bigvee_{u' \in U} {}^{\bullet}t + ({}^{\bullet}u \ominus t^{\bullet}) \geq {}^{\bullet}u'.$$