Verification of Population Protocols with Unordered Data

Steffen van Bergerem
Humboldt-Universität zu Berlin, Germany

Roland Guttenberg
Technische Universität München, Germany

Sandra Kiefer
University of Oxford, UK

Corto Mascle
LaBRI, Université de Bordeaux, France

Nicolas Waldburger
IRISA, Université de Rennes, France

Chana Weil-Kennedy
IMDEA Software Institute, Madrid, Spain

Abstract

Population protocols are a well-studied model of distributed computation in which a group of anonymous finite-state agents communicates via pairwise interactions. Together they decide whether their initial configuration, i. e., the initial distribution of agents in the states, satisfies a property. As an extension in order to express properties of multisets over an infinite data domain, Blondin and Ladouceur (ICALP'23) introduced population protocols with unordered data (PPUD). In PPUD, each agent carries a fixed data value, and the interactions between agents depend on whether their data are equal or not. Blondin and Ladouceur also identified the interesting subclass of immediate observation PPUD (IOPPUD), where in every transition one of the two agents remains passive and does not move, and they characterised its expressive power.

We study the decidability and complexity of formally verifying these protocols. The main verification problem for population protocols is well-specification, that is, checking whether the given PPUD computes some function. We show that well-specification is undecidable in general. By contrast, for IOPPUD, we exhibit a large yet natural class of problems, which includes well-specification among other classic problems, and establish that these problems are in EXPSPACE. We also provide a lower complexity bound, namely coNEXPTIME-hardness.

2012 ACM Subject Classification Theory of computation → Verification by model checking; Theory of computation → Distributed computing models

Keywords and phrases Population protocols, Parameterized verification, Distributed computing, Well-specification

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.156

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2405.00921` [7]

Funding *Steffen van Bergerem*: This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 431183758 (gefördert durch die Deutsche Forschungsgemeinschaft (DFG) – Projektnummer 431183758).
Sandra Kiefer: This research was supported by the Glasstone Benefaction, University of Oxford [Violette and Samuel Glasstone Research Fellowships in Science 2022] as well as Jesus College in Oxford, UK.
Chana Weil-Kennedy: This work was supported by the grant PID2022-138072OB-I00, funded by MCIN, FEDER, UE and partially supported by PRODIGY Project (TED2021-132464B-I00) funded by MCIN and the European Union NextGeneration.

Acknowledgements This project started at and has benefitted substantially from the research camp Autobóz 2023 in Kassel, Germany. We would like to thank the host, sponsors, and organisers of the research camp for bringing us together.

1 Introduction

Population protocols (PP) model distributed computation and have received a lot of attention [1, 2, 9, 12, 14, 18] since their introduction in 2004 [3]. In a PP, a collection of indistinguishable mobile agents with constant-size memory communicate via pairwise interactions. When two agents meet, they exchange information about their states and update their states accordingly. The agents collectively compute whether their input configuration, i. e., the initial distribution of agents in each state, satisfies a certain predicate. For a PP to compute a predicate, the protocol must be *well-specified*, i. e., for every initial configuration, all fair runs starting in this configuration must converge to the same answer. It was shown that PP compute exactly the predicates of Presburger arithmetic [4]. Moreover, well-specification is known to be decidable but as hard as the reachability problem for Petri nets [17]. Note that deciding well-specification is a problem that concerns *parameterised verification* in the sense of [8, 15], i. e., one must decide that something holds with respect to every value of the parameter. Here the parameter is the number of agents that are present – the PP must converge to one answer for every initial configuration, no matter the number of agents.

Population protocols with unordered data (PPUD) were introduced by Blondin and Ladouceur as a means to compute predicates over arbitrarily large domains [10]. In this setting, each agent holds a read-only datum from an infinite set $\mathbb{D}$. When interacting, agents may check (dis)equality of their data. While PP can compute properties like "there are more than 5 agents in state q_1", PPUD can express, e. g., "there are more than 2 data with 5 agents each in state q_1". In [10], the authors construct a PPUD computing the absolute majority predicate, i. e., whether a datum is held by more than half of the agents. They also characterise the expressive power of *immediate observation PPUD* (IOPPUD), a subclass of interest in which interactions are restricted to observations. That is, in every interaction, one of the two agents is passive and does not change its state. The decidability and complexity of the main verification question for PPUD, namely well-specification, is left open in Blondin's and Ladouceur's article [10]. It is the subject of this paper.

Contributions. We start by showing that well-specification is *undecidable* for PPUD. This follows from a reduction from 2-counter machines; in fact, the presence of data allows us to encode zero-tests. Contrasting this, we show that deciding well-specification is in ExpSpace for IOPPUD. To this end, we define *generalised reachability expressions* (GRE) and establish that, for IOPPUD, deciding whether the set of configurations that satisfy a given GRE is empty is in ExpSpace. This decidability result is powerful; indeed, this emptiness problem subsumes classic verification problems like reachability and coverability, as well as parameterised verification problems such as well-specification and correctness, where the latter asks whether a given protocol computes a given predicate. Lastly, we exhibit a coNExpTime lower bound for deciding emptiness of GRE for IOPPUD.

Related work. For a recent survey of the research on verification of PP (without data), see [16]. In particular, the well-specification problem for PP is known to be decidable, but as hard as Petri net reachability [17] and therefore Ackermann-complete [13, 24, 25]. In their seminal paper on the computational power of PP, Angluin, Aspnes, Eisenstat, and Ruppert also introduced five subclasses of PP that model one-way communication [4]. One of these is immediate observation population protocols (IOPP), which correspond to IOPPUD without data. The complexity of well-specification for all five subclasses is determined in [18]. In particular, the paper shows that well-specification for IOPP is PSPACE-complete. IOPP were modelled by immediate observation Petri nets, where classic parameterised problems can be decided in polynomial space. The notion of generalised reachability expression was first phrased in this setting, and one of the consequences is that the emptiness problem of GRE for IOPP is PSPACE-complete [29]. Our result shows that adding data to the model as in [9] (and extending GRE naturally) pushes the emptiness problem between coNEXPTIME and EXPSPACE.

While the introduction of data in the PP model happened recently [10], a similar approach has been studied in the related model of Petri nets, under the name of *data nets*. In this setting, the classic problem of coverability (or control-state reachability) is decidable but non-primitive recursive [23] and in fact $\mathbf{F}_{\omega^\omega}$-complete [27]. While PPUD can be encoded into data nets, our results show that the problems that we study cannot be reduced to coverability. Another related model is formed by broadcast networks of register automata (BNRA) [20], an extension of reconfigurable broadcast networks (RBN) with data. RBN subsume IOPP [6], and consequently BNRA subsume IOPPUD. However, the complexity of coverability in BNRA is known to be $\mathbf{F}_{\omega^\omega}$-complete, hence non-primitive recursive, and more complex problems quickly become undecidable [20]. These hardness results contrast with the EXPSPACE membership.

Organisation. In Section 2, we introduce the models of PPUD and IOPPUD, the notion of GRE, and we state our main results. We prove undecidability of well-specification for PPUD in Section 3. The next sections are dedicated to the study of IOPPUD. In Section 4, we establish bounds on the number of observed agents. In Section 5, we introduce the technical notions of boxes and containers and use the bounds from the previous section to translate GRE into containers. We present the complexity bounds for emptiness of GRE in Section 6.

2 Population Protocols and Main Results

We use the notation $[m,n] \coloneqq \{\ell \in \mathbb{N} \mid m \leq \ell \leq n\}$ for $m,n \in \mathbb{N}$ and $[m,+\infty) \coloneqq \{\ell \in \mathbb{N} \mid m \leq \ell\}$.

2.1 Population Protocols with Unordered Data

We fix an infinite *data* domain $\mathbb{D}$, an infinite set of *agents* $\mathbb{A}$, and a function $\mathbf{dat}\colon \mathbb{A} \to \mathbb{D}$ such that $\mathbf{dat}^{-1}(d)$ is infinite for all $d \in \mathbb{D}$. For $d \in \mathbb{D}$, a *d-agent* is an agent $a \in \mathbb{A}$ with $\mathbf{dat}(a) = d$.

▶ **Definition 1.** *A* population protocol with unordered data *(PPUD) is a tuple* (Q, Δ, I, O) *where Q is a finite set of* states*,* $\Delta \subseteq Q^2 \times \{=,\neq\} \times Q^2$ *the set of* transitions*,* $I \subseteq Q$ *the set of* initial states*, and* $O\colon Q \to \{\top, \bot\}$ *the* output function*.*

The size of a PPUD $\mathcal{P}$, written $|\mathcal{P}|$, is its number of states. We fix a PPUD (Q, Δ, I, O).

A *configuration* is a function $\gamma\colon \mathbb{A} \to Q \cup \{*\}$ such that $\gamma(a) \in Q$ only holds for finitely many agents $a \in \mathbb{A}$ (the agents *appearing* in γ). We denote by Γ the set of all configurations, and by $\Gamma_{\mathsf{init}} \coloneqq \{\gamma \in \Gamma \mid \forall a \in \mathbb{A},\, \gamma(a) \notin Q \setminus I\}$ the set of *initial configurations*. Given $\gamma \in \Gamma$ and $d \in \mathbb{D}$, we let $\gamma_d^{\#}\colon Q \to \mathbb{N}$ be the function that maps each state q to the number of d-agents in q in γ.

Given $\gamma, \gamma' \in \Gamma$, we write $\gamma \to \gamma'$, and call it a *step* from γ to γ', when there are states $q_1, q_2, q_3, q_4 \in Q$ and two distinct agents $a_1, a_2 \in \mathbb{A}$ such that $((q_1, q_2),\ \bowtie,\ (q_3, q_4)) \in \Delta$, $\gamma(a_1) = q_1$, $\gamma(a_2) = q_2$, $\gamma'(a_1) = q_3$, $\gamma'(a_2) = q_4$, $\gamma(a) = \gamma'(a)$ for all $a \in \mathbb{A} \setminus \{a_1, a_2\}$, and, additionally, if $\bowtie$ is an equality (resp. disequality), then $\mathbf{dat}(a_1) = \mathbf{dat}(a_2)$ (resp. $\mathbf{dat}(a_1) \neq \mathbf{dat}(a_2)$). A *run* ρ is a (finite or infinite) sequence of consecutive steps $\rho\colon \gamma_1 \to \gamma_2 \to \gamma_3 \to \dots$. We write $\rho\colon \gamma \xrightarrow{*} \gamma'$ to denote that ρ is a finite run from γ to γ', and simply $\gamma \xrightarrow{*} \gamma'$ to denote the existence of such a run. For every $\gamma \in \Gamma$, let $\mathsf{Post}^*(\gamma) \coloneqq \{\gamma' \in \Gamma \mid \gamma \xrightarrow{*} \gamma'\}$ and $\mathsf{Pre}^*(\gamma) \coloneqq \{\gamma' \in \Gamma \mid \gamma' \xrightarrow{*} \gamma\}$. A run ρ *covers* a state $q \in Q$ if there is a configuration γ in ρ such that $\gamma(a) = q$ for some agent a.

In accordance with [3] and [10], we consider a run $\gamma_1 \to \gamma_2 \to \dots$ *fair* if it is infinite[1] and for every configuration γ with $\left|\{i \in \mathbb{N} \mid \gamma_i \xrightarrow{*} \gamma\}\right| = \infty$, it holds that $\left|\{i \in \mathbb{N} \mid \gamma_i = \gamma\}\right| = \infty$. That is, every infinitely often reachable configuration also occurs infinitely often along the run. For $b \in \{\top, \bot\}$, a *b-consensus* is a configuration γ in which, for all agents $a \in \mathbb{A}$ appearing in γ, it holds that $O(\gamma(a)) = b$. A fair run $\rho\colon \gamma_1 \to \gamma_2 \to \cdots$ *stabilises* to $b \in \{\top, \bot\}$ if there is an $n \in \mathbb{N}$ such that for every $i \geq n$, γ_i is a b-consensus. A protocol is *well-specified* if, for every initial configuration $\gamma_0 \in \Gamma_{\mathsf{init}}$, there is $b \in \{\top, \bot\}$ such that all fair runs starting in γ_0 stabilise to b. The *well-specification problem* for PPUD asks, given a PPUD $\mathcal{P}$, whether $\mathcal{P}$ is well-specified. Given a PPUD $\mathcal{P}$ and a function $\Pi\colon \Gamma_{\mathsf{init}} \to \{\top, \bot\}$, $\mathcal{P}$ *computes* Π if, for every $\gamma_0 \in \Gamma_{\mathsf{init}}$, every fair run of $\mathcal{P}$ starting in γ_0 stabilises to $\Pi(\gamma_0)$.

▶ **Example 2.** Consider the following PPUD $\mathcal{P}$. Its set of states is $Q = \{\ell_0, \ell_1, \mathsf{f}_0, \mathsf{f}_1, \mathsf{dead}\}$, with $I = \{\ell_1\}$, $O(\ell_0) = O(\mathsf{f}_0) = \top$, $O(\ell_1) = O(\mathsf{f}_1) = O(\mathsf{dead}) = \bot$ and its transitions are:

$$\forall b, b' \in \{0,1\},\ (\ell_b, \ell_{b'}) \mapsto (\ell_{b \oplus b'}, \mathsf{f}_{b \oplus b'}) \qquad \forall b, b' \in \{0,1\},\ (\ell_b, \mathsf{f}_{b'}) \mapsto (\ell_b, \mathsf{f}_b)$$
$$\forall q, q' \in Q,\ (q, q') \mapsto_= (\mathsf{dead}, \mathsf{dead}) \qquad \forall q \in Q,\ (q, \mathsf{dead}) \mapsto (\mathsf{dead}, \mathsf{dead})$$

where $\mapsto_=$ denotes that the data of the agents must be equal, $\mapsto$ without subscript means no condition on data (or equivalently, the transition exists both for equality and disequality), and $\oplus$ denotes the XOR operator. $\mathcal{P}$ is well-specified and computes the function Π that is equal to $\top$ whenever there is an even number of appearing data and they all have exactly one corresponding agent. To see this, if there are two agents of equal datum, then all fair runs eventually have all agents on dead and stabilise to $\bot$. Otherwise, there will eventually be a single agent in $\{\ell_0, \ell_1\}$, and it will be on ℓ_b if and only if the number of agents has parity b, in which case all other agents will eventually go to f_b and the run stabilises to $\bot$ if $b = 1$ (odd number of agents) and to $\top$ if $b = 0$ (even number of agents).

A more interesting but also more complex example is the majority protocol described in [10, Section 3]; it computes whether a datum has the absolute majority, i. e., strictly more agents than all other data combined.

Well-specification is the fundamental verification problem for population protocols. However, as we will see in Section 3, this problem is undecidable for PPUD.

▶ **Theorem 3.** *The well-specification problem for PPUD is undecidable.*

This motivates the study of the restricted class of *immediate observation* PPUD.

[1] One often considers that a finite run $\gamma_f \xrightarrow{*} \gamma_\ell$ is fair when there is no γ such that $\gamma_\ell \to \gamma$. In the following, we rule out this possibility by implicitly assuming that, for all $q_1, q_2 \in Q$ and $\bowtie\ \in \{=, \neq\}$, it holds that $((q_1, q_2), \bowtie, (q_1, q_2)) \in \Delta$, and ignoring the trivial cases of runs with at most one agent.

2.2 Immediate Observation Protocols

Immediate observation protocols [4] are a restriction of population protocols where, when two agents interact, one of the two agents does not change its state. The restriction of the model with data to immediate observation was first considered in [10].

▶ **Definition 4.** *An* immediate observation population protocol with unordered data *(or* IOPPUD*) is a PPUD $\mathcal{P} = (Q, \Delta, I, O)$ where every transition $\delta \in \Delta$ is of the form $(q_1, q_2, \bowtie, q_1, q_3)$, with $q_1, q_2, q_3 \in Q$ and $\bowtie\ \in \{=, \neq\}$, i. e., the first agent does not change its state.*

For IOPPUD, we denote a transition $(q_1, q_2, \bowtie, q_1, q_3)$ by $q_2 \xrightarrow{\bowtie q_1} q_3$. If we have a step $\gamma \to \gamma'$ with transition $q_2 \xrightarrow{\bowtie q_1} q_3$ that involves agents $a, a_o \in \mathbb{A}$ where a is the agent moving from q_2 to q_3 and a_o is the agent in q_1, we denote it by $\gamma \xrightarrow{\bowtie a_o}_a \gamma'$. We say that agent a *observes* agent a_o, and call a_o the *observed* agent. Intuitively, a “observes” a_o and reacts, whereas a_o may not even know it has been observed.

▶ **Example 5.** Consider the following IOPPUD $\mathcal{P} = (Q, \Delta, I, O)$, with $Q := \{q_0, q_1, q_2, q_3\}$, $I := \{q_0, q_1\}$, $O(q_3) = \top$, $O(q) = \bot$ for all $q \neq q_3$, and transitions in Δ as follows:

$$q_0 \xrightarrow{=q_1} q_2 \quad q_1 \xrightarrow{=q_0} q_2 \quad q_2 \xrightarrow{\neq q_2} q_3 \qquad \forall q \in \{q_1, q_2\}, \forall \bowtie\ \in \{=, \neq\}, q \xrightarrow{\bowtie q_3} q_3$$

This protocol is well-specified: from $\gamma_0 \in \Gamma_{\mathsf{init}}$, all fair runs stabilise to $\top$ if two data have agents on both q_0 and q_1, and all fair runs stabilise to $\bot$ otherwise. Indeed, if there is a datum with agents on both q_0 and q_1, by fairness eventually an agent with this datum is sent to q_2; if there are two such data, then eventually some agent covers q_3, and then all agents are sent to q_3 and the run stabilises to $\top$. Conversely, if it is not the case, then q_3 cannot be covered and all fair runs stabilise to $\bot$.

Let $\rho\colon \gamma_{start} \xrightarrow{*} \gamma_{end}$ be a run. Agent a_o is *internally observed* (resp. *externally observed*) in ρ if ρ contains a step of the form $\gamma_1 \xrightarrow{=a_o}_a \gamma_2$ (resp. $\gamma_1 \xrightarrow{\neq a_o}_a \gamma_2$); it is *observed* if one of the two cases holds. Similarly, a datum d is *observed* in ρ if an agent a with $\mathbf{dat}(a) = d$ is observed in ρ; we define similarly a datum being internally or externally observed.

While the set of functions that can be computed by PPUD remains an open question, it is known that IOPPUD exactly compute interval predicates [10], defined as follows. Let S be a finite set. A *simple interval predicate* over S is a formula ψ of the form $\dot\exists \mathsf{d}_1, \ldots, \mathsf{d}_m, \bigwedge_{q \in S} \bigwedge_{j=1}^{m} \#(q, \mathsf{d}_j) \in [A_{q,j}, B_{q,j}]$ where, for all $q \in S$ and $j \in [1, m]$, we have $A_{q,j} \in \mathbb{N}$ and $B_{q,j} \in \mathbb{N} \cup \{+\infty\}$. The dotted quantifiers quantify over *pairwise distinct* data. Formally, given a protocol $\mathcal{P}$ with set of states Q such that $S \subseteq Q$ and given $\gamma \in \Gamma$, the predicate ψ is *satisfied* by γ if there exist pairwise distinct data $d_1, \ldots, d_m \in \mathbb{D}$ such that for all $q \in S$ and $j \in [1, m]$, it holds that $\gamma^{\#}_{d_j}(q) \in [A_{q,j}, B_{q,j}]$ (resp. $\gamma^{\#}_{d_j}(q) \in [A_{q,j}, B_{q,j})$ in the case that $B_{q,j} = +\infty$). An *interval predicate* over S is a Boolean combination φ of simple interval predicates over S; we define that φ is *satisfied* by a configuration γ if the simple interval predicates satisfied by γ satisfy the Boolean combination.

▶ **Theorem 6** ([10], Theorem 18 and Corollary 29). *Given a finite set I, the functions computed by IOPPUD with set of initial states*[2] *I are exactly the interval predicates over I.*

▶ **Example 7.** The protocol described in Example 2 and the majority protocol of [10, Section 3] cannot be turned into immediate observation protocols, as they compute functions that cannot be expressed as interval predicates. The immediate observation protocol from Example 5 computes the following interval predicate, which is actually a simple interval predicate:

$$\dot\exists \mathsf{d}_1, \mathsf{d}_2,\ (\#(q_0, \mathsf{d}_1) \geq 1) \wedge (\#(q_1, \mathsf{d}_1) \geq 1) \wedge (\#(q_0, \mathsf{d}_2) \geq 1) \wedge (\#(q_1, \mathsf{d}_2) \geq 1).$$

[2] This does not limit the number of states of said protocols, as their set of states Q may be larger than I.

Given a simple interval predicate $\psi = \dot{\exists}\mathsf{d}_1, \ldots, \mathsf{d}_m, \bigwedge_{q \in I} \bigwedge_{j=1}^{m} \#(q, \mathsf{d}_j) \in [A_{q,j}, B_{q,j}]$, we define its *width* as m, its *height* h as the maximum of all finite $A_{q,j}$ and $B_{q,j}$, and its *size* as $|I| \cdot m \cdot \log(h)$. We also define the *width* (resp. *height*) of an interval predicate as the maximum of the widths (resp. heights) of its simple interval predicates, and its *size*, measuring the space taken by its encoding, as the sum of their sizes plus its number of Boolean operators.

▶ Remark 8. In [10], predicates refer to an input alphabet Σ, which is converted into initial states using an input mapping. For convenience, we have not included the input alphabet in our model, which is why we arbitrarily fix a set of initial states I in Theorem 6.

2.3 Generalised Reachability Expressions

We define a general class of specifications, called generalised reachability expressions, which are formulas constructed using interval predicates as atoms and using union, complement, Post^*, and Pre^* as operators. This concept is inspired by [29, Section 2.4], although our choice of atoms is more general and adapted to the data setting.

▶ **Definition 9.** *Let $\mathcal{P} = (Q, \Delta, I, O)$ be a protocol.*
Generalised Reachability Expressions *(GRE) over $\mathcal{P}$ are produced by the grammar*

$$E ::= \varphi \mid E \cup E \mid \overline{E} \mid \mathsf{Post}^*(E) \mid \mathsf{Pre}^*(E),$$

where φ ranges over interval predicates over Q.

Given a GRE E, we define the set of configurations defined by E, denoted $[\![E]\!]_{\mathcal{P}}$, as the set containing all configurations of $\mathcal{P}$ that satisfy the formula, where the predicates are interpreted as above and the other operators are interpreted naturally (the overline denotes set complementation). This set is denoted $[\![E]\!]$ when $\mathcal{P}$ is clear from context.

The *length* $|E|$ of a GRE E is its number of operators. Letting $\varphi_1, \ldots, \varphi_k$ be the interval predicates used as atoms in E, the *norm* $||E||$ of E is the maximum of the heights and widths of the φ_i. Its *size* is the sum of the sizes of the φ_i plus $|E|$. The *emptiness problem for GRE* asks, given as input a protocol $\mathcal{P}$ and a GRE E over $\mathcal{P}$, whether $[\![E]\!]_{\mathcal{P}} = \emptyset$. We will show in Section 6 that, for IOPPUD, this problem is decidable.

▶ **Theorem 10.** *The emptiness problem for GRE over IOPPUD is in* ExpSpace.

We now argue that this decidability result is powerful, as it implies decidability of many classic problems on IOPPUD. We start with well-specification. We use the notation $\dot{\forall}\mathsf{d}, \varphi$ as a short form for $\neg\dot{\exists}\mathsf{d}, \neg\varphi$. Given a PPUD $\mathcal{P}$ and $b \in \{\top, \bot\}$, let $\mathsf{Out}_b :=$ $\dot{\forall}\mathsf{d}, \bigwedge_{q \notin O^{-1}(\{b\})} \#(q, \mathsf{d}) = 0$ be the GRE for b-consensus configurations; moreover, let $\mathsf{Stable}_b := \overline{\mathsf{Pre}^*(\overline{\mathsf{Out}_b})}$ be the GRE for *stable* b-consensus, i. e., configurations from which all runs lead to a b-consensus.

▶ **Proposition 11.** *Let $\mathcal{P}$ be a PPUD, $E_{\mathsf{ws}} := \Gamma_{\mathsf{init}} \cap \mathsf{Pre}^*(\overline{\mathsf{Pre}^*(\mathsf{Stable}_\top)}) \cap \mathsf{Pre}^*(\overline{\mathsf{Pre}^*(\mathsf{Stable}_\bot)})$. $\mathcal{P}$ is well-specified if and only if $[\![E_{\mathsf{ws}}]\!]_{\mathcal{P}} = \emptyset$.*

Proof. First, $\Gamma_{\mathsf{init}} = [\![\dot{\forall}\mathsf{d}, \bigwedge_{q \notin I} \#(q, d) = 0]\!]$ and E_{ws} is indeed a GRE over $\mathcal{P}$. For every $\gamma \in \Gamma$, $\mathsf{Post}^*(\gamma)$ is finite as all configurations reachable from γ have the same number of agents. Therefore, a fair run ρ that visits $\mathsf{Pre}^*(\mathcal{S})$ infinitely often for $\mathcal{S} \subseteq \Gamma$ must visit $\mathcal{S}$ infinitely often. Let $\gamma_0 \in \Gamma_{\mathsf{init}}$ and $b \in \{\top, \bot\}$; it suffices to prove that there is a fair run from γ_0 that does *not* stabilise to b if and only if $\gamma_0 \in [\![\mathsf{Pre}^*(\overline{\mathsf{Pre}^*(\mathsf{Stable}_b)})]\!]$. If, from γ_0, one can reach $\gamma \notin [\![\mathsf{Pre}^*(\mathsf{Stable}_b)]\!]$, then one can build a fair run from γ_0 that first goes to γ, and then forever performs arbitrary steps in a fair way; since $\gamma \notin [\![\mathsf{Pre}^*(\mathsf{Stable}_b)]\!]$, it will stay in $[\![\mathsf{Pre}^*(\overline{\mathsf{Out}_b})]\!]$, so by fairness it visits $[\![\overline{\mathsf{Out}_b}]\!]$ infinitely often and does not stabilise to b. Conversely, if there is a fair run that does not stabilise to b, then it never visits $[\![\mathsf{Stable}_b]\!]$, hence by fairness it eventually stops visiting $[\![\mathsf{Pre}^*(\mathsf{Stable}_b)]\!]$; this proves that it visits a configuration $\gamma \in [\![\overline{\mathsf{Pre}^*(\mathsf{Stable}_b)}]\!]$, and γ is reachable from γ_0 hence $\gamma_0 \in [\![\mathsf{Pre}^*(\overline{\mathsf{Pre}^*(\mathsf{Stable}_b)})]\!]$. ◀

Many other problems can be expressed as emptiness problems for GRE; we list a few.

- The *correctness* problem for IOPPUD asks, given an IOPPUD $\mathcal{P} = (Q, I, O, \Delta)$ and an interval predicate φ over I, whether $\mathcal{P}$ computes φ. This can be equivalently phrased as $[\![\varphi^{-1}(b) \cap \mathsf{Pre}^*(\overline{\mathsf{Pre}^*(\mathsf{Stable}_b)})]\!] = \emptyset$ for all $b \in \{\top, \bot\}$, where $\varphi^{-1}(b)$ is the set of initial configurations that φ maps to b. Note that the previous expression is a GRE because, in Definition 9, we chose as atoms interval predicates such as φ.
- The *set-reachability* problem (called cube-reachability in [5]) asks, given two sets of configurations $\mathcal{S}_1, \mathcal{S}_2$, whether $\mathcal{S}_2$ is reachable from $\mathcal{S}_1$; this typically expresses safety problems where $\mathcal{S}_2$ represents "bad configurations" that must not be reached. If $\mathcal{S}_1 = [\![E_1]\!]$ and $\mathcal{S}_2 = [\![E_2]\!]$, then this amounts to checking whether $[\![E_1 \cap \mathsf{Pre}^*(E_2)]\!]$ is empty.
- The *home-space problem* asks, given a protocol $\mathcal{P}$ and a set of configurations $\mathcal{H}$, whether $\mathcal{H}$ can be reached from every configuration reachable from an initial configuration. If $\mathcal{H}$ can be expressed as a GRE E, then it suffices to check whether $[\![\mathsf{Post}^*(\Gamma_{\mathsf{init}})]\!] \subseteq [\![\mathsf{Pre}^*(E)]\!]$. This problem has been studied in Petri nets [22], but also in probabilistic settings, for example in [11] for asynchronous shared-memory systems; indeed, in systems with uniform probabilistic schedulers where $\mathsf{Post}^*(\gamma_0)$ is finite for every initial configuration γ_0, this problem is equivalent to asking whether the probability of reaching $\mathcal{H}$ is equal to 1.

Theorem 10 entails that, for IOPPUD, all these problems are decidable and in EXPSPACE.

3 Undecidability of Verification of Population Protocols with Unordered Data

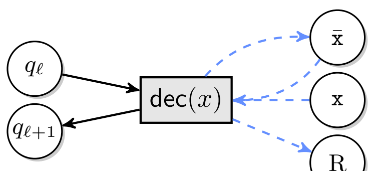
In this section, we establish that the most fundamental verification problem for PPUD, i. e., the well-specification problem, is already undecidable.

▶ **Theorem 3.** *The well-specification problem for PPUD is undecidable.*

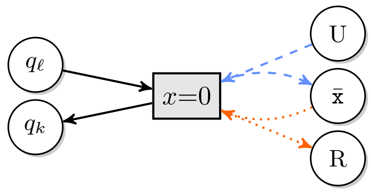
We proceed by reduction from the halting problem for 2-counter machines with zero-tests, a famously undecidable problem [26]. Here, we give a proof sketch. The detailed reduction can be found in [7], which is the full version of this paper.

We fix a 2-counter machine and build a protocol $\mathcal{P}$ which is *not* well-specified if and only if the counter machine halts. A 2-*counter machine* performs increments, decrements and zero-tests on two counters. The main difficulty are the zero-tests. Let us first recall how increments and decrements are simulated in many prior undecidability results for population protocols and Petri nets [21, 27]. The protocol has a control part $Q_{CM} := \{q_i \mid i \in [1, n]\}$ where a single *instruction agent* evolves; this part has one state per instruction of the machine. Increments and decrements are simulated as follows: The instruction agent interacts with states of $\{R\} \cup \{\mathtt{x}, \mathtt{y}\}$, where R is a *reservoir state* and $\mathtt{x}$ and $\mathtt{y}$ are states in which the number of agents represents the value of the counters. For example an increment on counter x moves one agent from the reservoir R to $\mathtt{x}$ and advances the instruction agent to the next instruction. The reservoir is hence implicitly assumed to start with arbitrarily many agents.

The main difficulty is that one does not want to take the $= 0$ branch of a zero-test when the value of the counter is non-zero. Actually, similar to [21, 27], we will not prevent the existence of such runs. Instead, our protocol will have "violating" runs which take the wrong



(a) Gadget for performing $\mathsf{dec}(x)$. Agents of the blue dashed edges must have the same datum.



(b) Gadget for $x{=}0$ branch of "`if` $x{=}0$ `then goto` k". The same agent takes the two blue dashed edges and the same agent takes the orange dotted edges.

Figure 1 For simplicity, we use Petri net notation: circles are states, rectangles are Petri net transitions. To encode this into our protocols, we split each transition into pairwise interactions.

branch of a zero-test, but our well-specification check will consider only *violation-free* runs. The correctness of the reduction is then established in two steps: The CM halts if and only if some *violation-free* run to the halting state exists, and this is true if and only if our protocol is not well-specified. We establish the connection between non-well-specification and the existence of a *violation-free* run in our protocol.

In the first place, we guarantee that every initial configuration of our protocol has a fair run stabilising to $\perp$, so that $\mathcal{P}$ is not well-specified if and only if there exists a fair run which does not stabilise to $\perp$[3]. Second, we introduce *violation detection*, a mechanism which guarantees that *fair* runs which contain a violation stabilise to $\perp$, hence preserving well-specification. To do so, we add a sink state $q_\perp$, which has output $\perp$ and is attracting, i. e., all other states have a transition to $q_\perp$ available when observing that $q_\perp$ is non-empty. Violation detection then entails adding transitions into $q_\perp$ that will be available infinitely often if the run (or its initial configuration) contained a violation. By fairness, any run containing a violation will then eventually put an agent into $q_\perp$, and hence, because $q_\perp$ is attracting, the run ends in a deadlock with all agents in $q_\perp$. In particular, any fair run containing a violation will output $\perp$ as claimed. There are two types of violation detection.

First, we want to only mark those runs as violation-free that start in initial configurations where $U \in Q$ has at most one agent of each datum. To do so, we make agents remember whether their initial state was U or not (by encoding it into the state space), and, from every state, we add a transition to $q_\perp$ such that this transition is enabled when an agent who started in U observes another agent of same datum that also started in U.

Second, we want to detect violations which consist in falsely simulating a zero-test, as discussed above. Here our technique shares some similarities with [27]. Let $c \in \{x, y\}$ be a counter; for every zero-test of the counter machine, we add two types of transitions to the protocol. The first type simulates the $c \neq 0$ branch and can be taken by the instruction agent upon interacting with some agent on state $\mathtt{c}$; by contrast, the $c = 0$ branch can always be taken. However, if it was taken with $c \neq 0$, then violation detection will eventually detect this. For this mechanism, we introduce a counter control state $\bar{\mathtt{c}} \in Q$. At any point in time, $\bar{\mathtt{c}}$ contains one agent, similar to the instruction agent. The crux of our violation detection is that only agents which share the datum with the agent in $\bar{\mathtt{c}}$ will be allowed to move in and out of state $\mathtt{c}$, as illustrated in Figure 1a.

The $= 0$ branch of a zero-test is depicted in Figure 1b. It replaces the agent on $\bar{\mathtt{c}}$ with an agent with fresh datum from state U. Thus, when the $c = 0$ branch is taken, any remaining agent in $\mathtt{c}$ is stuck in $\mathtt{c}$ as it will never again share datum with the agent in $\bar{\mathtt{c}}$. Violation detection then sends an agent in $\mathtt{c}$ to $q_\perp$ upon observing an agent in $\bar{\mathtt{c}}$ with different datum.

[3] This can be done with the addition of a fresh state that is the only initial state and that has internal transitions to all former initial states and an internal transition to a sink state that has output $\perp$.

Now that we have violation detection in place, it only remains to explain the connection to halting. The halting instruction q_n in Q_{CM} is the only state with output $\top$. Hence, any run not outputting $\bot$ must contain an agent in the halting instruction at some point, and be violation-free by the above. That is, the counter machine reached the halting state without violations. Conversely, if the machine halts, one can build a finite run that puts an agent into the halting state without any violation occurring. The corresponding configuration is then a deadlock, and hence the extension to an infinite run (by staying there forever) is a fair run not outputting $\bot$. This proves that well-specification is undecidable for PPUD, which motivates restricting ourselves to immediate observation PPUD.

4 An Analysis of Immediate Observation Protocols with Data

To obtain our complexity bounds on the emptiness problem for GRE, we first show some transformations on runs that allow us to bound the number of observed agents. All runs that we consider in this section are finite, and we therefore write them as $\gamma_1 \to \cdots \to \gamma_m$ or $\gamma_{start} \xrightarrow{*} \gamma_{end}$. In the rest of this section, we fix an IOPPUD $\mathcal{P} = (Q, \Delta, I, O)$.

We introduce some notation for agents in runs. Let $\rho\colon \gamma_1 \xrightarrow{*} \gamma_m$ and $d \in \mathbb{D}$. We let $\mathbb{A}_\rho$ be the set of agents appearing in ρ, and set $\mathbb{A}^d_\rho := \{a \in \mathbb{A}_\rho \mid \mathbf{dat}(a) = d\}$. We let $\mathbb{A}^d_{\rho,o}$ be the set of agents with datum d that are observed in ρ, i. e., the $a_o \in \mathbb{A}^d_\rho$ such that there exists a step $\gamma \xrightarrow{\bowtie a_o}_a \gamma'$ in ρ. For all $q_1, q_m \in Q$, we let $\mathbb{A}^d_{\rho,q_1,q_m}$ be the set of agents with datum d that start in q_1 and end in q_m, i. e., the $a \in \mathbb{A}^d_\rho$ such that $\gamma_1(a) = q_1$ and $\gamma_m(a) = q_m$. Moreover, we let $\mathbb{D}_\rho := \{d \in \mathbb{D} \mid \mathbb{A}^d_\rho \neq \emptyset\}$ be the set of data appearing in ρ. We may omit ρ in the subscript if the run is clear from the context.

4.1 Bounds on the Number of Observed Agents per Datum

Let $\rho\colon \gamma_1 \to \cdots \to \gamma_m$ be a run. For $i \in [1, m]$, we call $\gamma_i \to \gamma_{i+1}$ the *i-th step* in ρ. Let $\rho[\to i]$ (resp. $\rho[i \to]$) denote the prefix of ρ ending on its i-th configuration (resp. the suffix of ρ starting on its i-th configuration). Let $a, b \in \mathbb{A}_\rho$. Agent a is *active* in the i-th step if $\gamma_i \xrightarrow{\bowtie a_o}_a \gamma_{i+1}$ for some agent a_o. Otherwise, a is *idle* in that step. We say b *copies* a in ρ if after every step $\gamma_i \xrightarrow{\bowtie a_o}_a \gamma_{i+1}$ in ρ via some transition t, there is a step $\gamma_{i+1} \xrightarrow{\bowtie a_o}_b \gamma_{i+2}$ via t and, additionally, b is idle in every step not immediately following an active step of a.

The following lemma allows us to add agents to a run that copy an agent of the same datum.

▶ **Lemma 12** (Agents copycat). *Let $\rho\colon \gamma_{start} \xrightarrow{*} \gamma_{end}$ be a run. Let $a \in \mathbb{A}_\rho$ and $\tilde{a} \in \mathbb{A} \setminus \mathbb{A}_\rho$ with $\mathbf{dat}(a) = \mathbf{dat}(\tilde{a})$. Then there exist configurations $\tilde{\gamma}_{start}, \tilde{\gamma}_{end}$ and a run $\tilde{\rho}\colon \tilde{\gamma}_{start} \xrightarrow{*} \tilde{\gamma}_{end}$ such that:*

(i) *$\mathbb{A}_{\tilde{\rho}} = \mathbb{A}_\rho \uplus \{\tilde{a}\}$, and for all $a' \in \mathbb{A}_\rho$, $\tilde{\gamma}_{start}(a') = \gamma_{start}(a')$ and $\tilde{\gamma}_{end}(a') = \gamma_{end}(a')$;*
(ii) *$\tilde{\gamma}_{start}(\tilde{a}) = \gamma_{start}(a)$ and $\tilde{\gamma}_{end}(\tilde{a}) = \gamma_{end}(a)$;*
(iii) *$\tilde{a}$ is not observed in $\tilde{\rho}$.*

Proof. We let $\tilde{\gamma}_{start}$ be such that $\tilde{\gamma}_{start}(\tilde{a}) = \gamma_{start}(a)$ and $\tilde{\gamma}_{start}(a') = \gamma_{start}(a')$ for all $a' \neq \tilde{a}$. We construct $\tilde{\rho}$ by going through ρ step by step, making $\tilde{a}$ copy a: whenever ρ takes a step $\xrightarrow{\bowtie a_o}_a$, then we take this step followed by step $\xrightarrow{\bowtie a_o}_{\tilde{a}}$ to $\tilde{\rho}$. We can do so because $\mathbf{dat}(\tilde{a}) = \mathbf{dat}(a)$ and because agent $a_o \neq a$ has not moved and thus can be observed again. These are the only steps where $\tilde{a}$ is involved, hence it is never observed. ◀

The following result shows that, given a run ρ, we can construct a new run with a small subset of the agents of $\mathbb{A}_\rho$ such that, for all $d \in \mathbb{D}$ and all states q_1 and q_2, if there is a d-agent starting in q_1 and ending in q_2 in ρ, then this is also true in the new run. We refer to [7] for proof details.

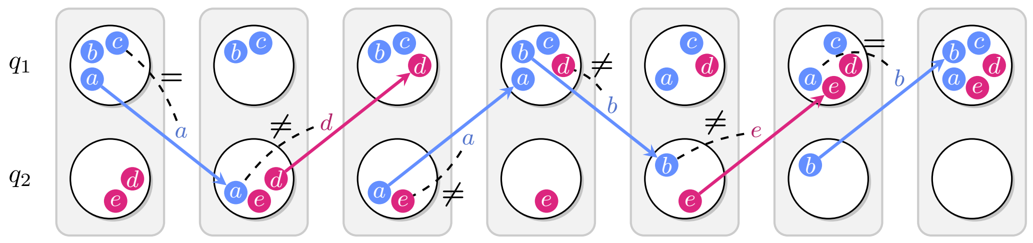


Figure 2 An example of a run with six steps on a protocol with two states q_1, q_2. $\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}$ denote agents; $\mathsf{a}, \mathsf{b}, \mathsf{c}$ have the same datum and d, e have the same datum. Dashed lines are observations.

▶ **Lemma 13** (Agents core). *Let $\rho\colon \gamma_{start} \xrightarrow{*} \gamma_{end}$ be a run. Then there exist configurations γ'_{start}, γ'_{end} and a run $\rho'\colon \gamma'_{start} \xrightarrow{*} \gamma'_{end}$ with $\mathbb{A}_{\rho'} \subseteq \mathbb{A}_\rho$ such that:*

(i) *for all $a \in \mathbb{A}_{\rho'}$, $\gamma'_{start}(a) = \gamma_{start}(a)$ and $\gamma'_{end}(a) = \gamma_{end}(a)$;*
(ii) *for all $d \in \mathbb{D}$ and $q_s, q_e \in Q$, if $\mathbb{A}^d_{\rho,q_s,q_e} \neq \emptyset$, then $\mathbb{A}^d_{\rho',q_s,q_e} \neq \emptyset$;*
(iii) *for all $d \in \mathbb{D}$, we have $|\mathbb{A}^d_{\rho'}| \leq |Q|^3$.*

Proof sketch. We adapt the bunch argument from the case of IO protocols without data [19]. Suppose there is $d \in \mathbb{D}$ and $q_s, q_e \in Q$ such that $|\mathbb{A}^d_{\rho,q_s,q_e}| > |Q|$. Let $\mathcal{R}$ be the set of states visited by agents of $\mathbb{A}^d_{\rho,q_s,q_e}$ in ρ. Notice that $|\mathcal{R}| \leq |Q|$. We define a family $(a_q)_{q\in\mathcal{R}}$ of pairwise distinct agents such that reducing $\mathbb{A}^d_{\rho,q_s,q_e}$ in ρ to $(a_q)_{q\in\mathcal{R}}$ still yields a valid run.

We iterate through $\mathcal{R}$ as follows. Let $q \in \mathcal{R}$ and let f be the first moment q is reached in ρ, i. e., the minimal index such that there exists an $a \in \mathbb{A}^d_{\rho,q_s,q_e}$ with $\gamma_f(a) = q$. Let ℓ be the last moment q is occupied in ρ, i. e., the maximal index such that there exists an $a \in \mathbb{A}^d_{\rho,q_s,q_e}$ with $\gamma_\ell(a) = q$. Let α_q be the agent in $\mathbb{A}^d_{\rho,q_s,q_e}$ that reaches q first, i. e., $\gamma_f(\alpha_q) = q$, and let β_q be the agent in $\mathbb{A}^d_{\rho,q_s,q_e}$ that leaves q last, i. e., $\gamma_l(\beta_q) = q$. Note that these agents do not have to be distinct. We pick a fresh agent $a_q \notin \mathbb{A}_\rho$ with $\mathbf{dat}(a_q) = d$ and modify ρ as follows. We let a_q copy α_q in $\rho[\to f]$, then a_q stays idle until β_q leaves q (for the last time) and then a_q copies β_q in $\rho[\ell \to]$. We do this for every $q \in \mathcal{R}$.

Then, for every step in which an a_o in $\mathbb{A}^d_{\rho,q_s,q_e}$ is observed in state q, let a_q be observed instead, i. e., replace steps $\xrightarrow{\bowtie a_o}_a$ with $\xrightarrow{\bowtie a_q}_a$. Finally, remove all the agents of $\mathbb{A}^d_{\rho,q_s,q_e}$ from the run, and identify (or substitute) each a_q with a distinct agent in $\mathbb{A}^d_{\rho,q_s,q_e}$, so that $(a_q)_{q\in\mathcal{R}} \subseteq \mathbb{A}^d_{\rho,q_s,q_e}$. We do this for every $d \in \mathbb{D}$ and $q_s, q_e \in Q$ such that $|\mathbb{A}^d_{\rho,q_s,q_e}| > |Q|$. ◀

▶ **Example 14.** Consider the run ρ depicted in Figure 2. Applying Lemma 13 on ρ yields a new run ρ' with 4 agents instead of 5. Indeed, let d denote the datum of a, b and c; we have $|\mathbb{A}^d_{\rho,q_1,q_1}| = |\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}| = 3$ whereas $|Q| = 2$. In ρ, agents a and b successively go from q_1 to q_2 and back to q_1. In ρ', these two agents are replaced by a single agent (named b again) who goes to q_2 on the first step and only leaves q_2 on the last step. In ρ', the new agent b is observed by d in the second step, and by e in the penultimate step.

4.2 Bounds on the Number of Observed Data

Given a run and a datum d appearing in it, we define the *trace* of d in ρ as the function $\mathtt{tr}^d_\rho\colon Q^2 \to \mathbb{N}$ such that for all $q_1, q_2 \in Q$, it holds that $\mathtt{tr}^d_\rho(q_1, q_2) = |\mathbb{A}^d_{\rho,q_1,q_2}|$. For each pair of states q_1, q_2, the trace counts the number of d-agents starting in q_1 and ending in q_2. For example, the trace of the run ρ of Example 14, with d the datum of agents a, b and c, is such that $\mathtt{tr}^d_\rho(q_1, q_1) = 3$ and $\mathtt{tr}^d_\rho(q, q') = 0$ for all $(q, q') \neq (q_1, q_1)$. The trace is the information we need to copy data: if there is a datum d with trace *tr* in a run, then we can add data to the run that mimic d and have the same trace. The following lemma echoes Lemma 12.

▶ **Lemma 15** (Data copycat). *Let $\rho\colon \gamma_{start} \xrightarrow{*} \gamma_{end}$ be a run. Let $d \in \mathbb{D}_\rho$ and $\tilde{d} \in \mathbb{D} \setminus \mathbb{D}_\rho$. Then there exist configurations $\tilde{\gamma}_{start}$, $\tilde{\gamma}_{end}$ and a run $\tilde{\rho}\colon \tilde{\gamma}_{start} \xrightarrow{*} \tilde{\gamma}_{end}$ such that:*

(i) *$\mathbb{A}_{\tilde{\rho}} = \mathbb{A}_\rho \uplus \mathbb{A}^{\tilde{d}}_{\tilde{\rho}}$, and for all $a \in \mathbb{A}_\rho$, $\tilde{\gamma}_{init}(a) = \gamma_{init}(a)$ and $\tilde{\gamma}_{end}(a) = \gamma_{end}(a)$,*

(ii) *$\mathtt{tr}^{\tilde{d}}_{\tilde{\rho}} = \mathtt{tr}^{d}_{\rho}$ and $\mathtt{tr}^{d'}_{\tilde{\rho}} = \mathtt{tr}^{d'}_{\rho}$ for all $d' \neq \tilde{d}$,*

(iii) *$\tilde{d}$ is not externally observed in $\tilde{\rho}$.*

Proof. For all $q_s, q_e \in Q$ and all $a \in \mathbb{A}^d_{q_s,q_e}$, we add an agent $\tilde{a}$ with datum $\tilde{d}$ in q_s at the start. We do this in a way similar to Lemma 12: after every step $\xrightarrow{\neq a_o}_a$ in ρ, we insert a step $\xrightarrow{\neq a_o}_{\tilde{a}}$, and after every step $\xrightarrow{= a_o}_a$ in ρ, we insert a step $\xrightarrow{= \tilde{a}_o}_{\tilde{a}}$. We thus maintain the fact that each added agent $\tilde{a}$ is in the same state as its counterpart a. In particular, they are in the same state at the end of the run. This yields a run $\tilde{\rho}$ with $\mathtt{tr}^{\tilde{d}}_{\tilde{\rho}} = \mathtt{tr}^{d}_{\rho}$, and such that for all $d' \neq \tilde{d}$, $\mathtt{tr}^{d'}_{\tilde{\rho}} = \mathtt{tr}^{d'}_{\rho}$. Since $\tilde{d} \notin \mathbb{D}_\rho$, it is not externally observed in $\tilde{\rho}$. ◀

Like we showed for the agents, we show that we can reduce the number of data in a run. We lift the proof strategy of Lemma 13 from agents to data, exploiting the sets of data with equal traces. We refer to [7] for proof details.

▶ **Lemma 16** (Data core). *Let $\rho\colon \gamma_{start} \xrightarrow{*} \gamma_{end}$ be a run and let K be a number such that there are at most K agents of each datum in ρ. Then there exist configurations γ'_{start}, γ'_{end}, a run $\rho'\colon \gamma'_{start} \xrightarrow{*} \gamma'_{end}$, and a subset of data $\mathbb{D}_{\rho'} \subseteq \mathbb{D}_\rho$ such that:*

(i) *for all $d \in \mathbb{D}_{\rho'}$ and all agents a of datum d, $\gamma_{start}(a) = \gamma'_{start}(a)$ and $\gamma_{end}(a) = \gamma'_{end}(a)$,*

(ii) *for all $d \in \mathbb{D}_\rho$, there exists $d' \in \mathbb{D}_{\rho'}$ such that $\mathtt{tr}^{d'}_{\rho'} = \mathtt{tr}^{d}_{\rho}$,*

(iii) *$|\mathbb{D}_{\rho'}| \leq (K+1)^{|Q|^3+|Q|^2}$.*

Proof sketch. We define the notion of split trace. The split trace of a datum d at the i-th configuration of a run ρ maps every triple of states (q_1, q_2, q_3) to the number of d-agents that are in q_1 at the start of ρ, then in q_2 in the i-th configuration, and finally in q_3 at the end. Since there are at most K agents per datum, there are at most $(K+1)^{|Q^2|}$ possible traces and $M = (K+1)^{|Q^3|}$ possible split traces.

For every trace tr, if there are more than M data that have trace tr in ρ, we apply a similar argument to Lemma 13: we select one datum for each possible split trace, and use it to cover all external observations of agents whose datum matches that split trace. We remove the other data, and show that this is still a valid run. The bound on the total number of data comes from the number of traces and split traces. ◀

▶ **Corollary 17.** *For every run $\rho\colon \gamma_{start} \xrightarrow{*} \gamma_{end}$, there exists a run $\tilde{\rho}\colon \gamma_{start} \xrightarrow{*} \gamma_{end}$ such that for all $d \in \mathbb{D}$, it holds that $|\mathbb{A}^d_{\tilde{\rho},o}| \leq |Q|^3$ and that agents of at most $(|Q|^3+1)^{|Q|^3+|Q|^2}$ different data are externally observed.*

Proof. We first apply Lemma 13 to ρ to obtain $\rho^{(1)}\colon \gamma^{(1)}_{start} \xrightarrow{*} \gamma^{(1)}_{end}$ over the same data such that for all $d \in \mathbb{D}$, it holds that $|\mathbb{A}^d_{\rho^{(1)}}| \leq |Q|^3$. Then we apply Lemma 16 to obtain $\rho^{(2)}\colon \gamma^{(2)}_{start} \xrightarrow{*} \gamma^{(2)}_{end}$ with at most $(|Q|^3+1)^{|Q|^3+|Q|^2}$ data. By Lemma 13-(i) and Lemma 16-(i), the remaining agents have the same initial and final states in ρ and $\rho^{(2)}$. It remains to put back the agents and data we removed, without increasing the number of externally observed data or observed agents per datum.

By Lemma 16-(ii), every trace of a datum in $\rho^{(1)}$ appears as the trace of a datum in $\rho^{(2)}$. Thus, it is possible to re-add data of $\mathbb{D}_{\rho^{(1)}}$ to $\mathbb{D}_{\rho^{(2)}}$ using repeated applications of Lemma 15. By Lemma 15-(iii), this does not add any external observation. So we obtain a run $\tilde{\rho}^{(1)}$ from

$\gamma^{(1)}_{start}$ to $\gamma^{(1)}_{end}$ such that at most $(|Q|^3+1)^{|Q|^3+|Q|^2}$ data are externally observed by Lemma 15-(iii). Recall that there are at most $|Q|^3$ agents per datum in $\gamma^{(1)}_{start}$ by Lemma 13-(iii); in particular there are at most $|Q|^3$ observed agents per datum in $\tilde{\rho}^{(1)}$.

By Lemma 13-(ii), for each datum d and states q_s, q_e, if there is a d-agent a such that $\gamma_{start}(a) = q_s$ and $\gamma_{end}(a) = q_e$ then there is an agent a' such that $\gamma^{(1)}_{start}(a') = q_s$ and $\gamma^{(1)}_{end}(a') = q_e$ in ρ. Therefore, due to Lemma 13-(ii), we can apply Lemma 12 repeatedly to add back the missing agents in $\tilde{\rho}^{(1)}$ and obtain a run $\tilde{\rho}$ from γ_{start} to γ_{end}. By Lemma 12-(iii), this does not add any observation. As a result, we obtain a run from γ_{start} to γ_{end} in which at most $(|Q|^3+1)^{|Q|^3+|Q|^2}$ data are externally observed and for all datum d, at most $|Q|^3$ d-agents are observed. ◀

5 From Expressions to Containers

In this section, we define the technical notions of boxes and containers, which are meant to represent sets of configurations defined by counting agents and data up to some thresholds. In Proposition 21, we will prove that the set of configurations defined by a generalised reachability expression E can be described as a union of containers whose thresholds are exponential in the length of E and polynomial in its norm. To do so, we will leverage the bounds on the number of observed agents from Section 4 to bound the description of the GRE $\mathsf{Post}^*(F)$ with respect to the one of GRE F. The key result of Proposition 21 will be used in Section 6 to obtain the decidability of the emptiness problem for GRE.

5.1 Equivalence of Predicates and Containers

In this subsection, we fix an IOPPUD $\mathcal{P} = (Q, \Delta, I, O)$.

Let $n, M \in \mathbb{N}$. An *n-box* is a vector $\mathsf{b}\colon Q \to [0, n]$. Given a configuration γ and a datum $d \in \mathbb{D}$, we define *the n-box of d in γ* as $\lceil \gamma, d \rceil^n\colon Q \to [0, n]$ such that for all $q \in Q$, $\lceil \gamma, d \rceil^n(q) = \min\{n, \gamma^{\#}_d(q)\}$; in words, the n-box of d truncates the number of agents of d if it exceeds n. We write $\mathbf{Boxes}_n$ for the set of all n-boxes. We define the equivalence relation $\equiv_n$ over $\Gamma \times \mathbb{D}$ by $(\gamma_1, d_1) \equiv_n (\gamma_2, d_2)$ whenever $\lceil \gamma_1, d_1 \rceil^n = \lceil \gamma_2, d_2 \rceil^n$. An equivalence class of $\equiv_n$ is a set of the form $\{(\gamma, d) \in \Gamma \times \mathbb{D} \mid \lceil \gamma, d \rceil^n = \mathsf{b}\}$ for $\mathsf{b} \in \mathbf{Boxes}_n$; we represent such an equivalence class for $\equiv_n$ by the associated n-box b.

To lift this concept to data, we count the number of data with the same n-box up to bound M. The (n, M)*-container* of a configuration γ is the function $\lceil \gamma \rceil^{n,M}\colon \mathbf{Boxes}_n \to [0, M]$ such that $\lceil \gamma \rceil^{n,M}(\mathsf{b}) = \min\big\{M, \big|\{d \in \mathbb{D} \mid \lceil \gamma, d \rceil^n = \mathsf{b}\}\big|\big\}$ for all $\mathsf{b} \in \mathbf{Boxes}_n$. We define the equivalence relation $\equiv_{n,M}$ over Γ by $\gamma_1 \equiv_{n,M} \gamma_2$ whenever $\lceil \gamma_1 \rceil^{n,M} = \lceil \gamma_2 \rceil^{n,M}$. An equivalence relation for $\equiv_{n,M}$ is the preimage of some (n, M)-container by the previously described function; we represent such an equivalence class by the associated (n, M)-container. Figure 3 illustrates the function mapping a given configuration to its container.

In all the following, we use the terms n-boxes and (n, M)-containers to designate both the vectors and the equivalence classes of $\equiv_n$ and $\equiv_{n,M}$ that they represent. For instance, we write *union of n-boxes* for the union of the corresponding equivalence classes of $\equiv_n$.

The partition of Γ into (n, M)-containers becomes finer as n and M grow.

▶ **Lemma 18.** *Let $n_1, n_2, M_1, M_2 \in \mathbb{N}$. If $n_1 \le n_2$ and $M_1 \le M_2$, then every (n_1, M_1)-container is a union of (n_2, M_2)-containers.*

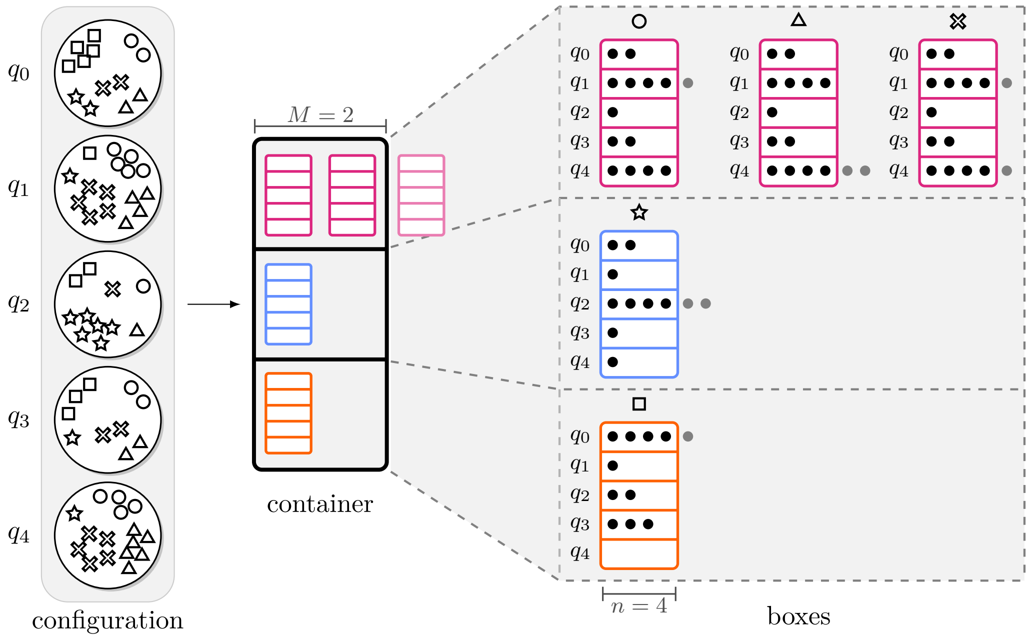


Figure 3 How a configuration is mapped to a (4, 2)-container. Here, the protocol has five states $q_0, \dots, q_4$. Five distinct data appear in the configuration, and they are represented using symbols.

Algorithmically, we represent an n-box as a list of appearing states with associated numbers from $[1, n]$ encoded in binary. Similarly, we represent an (n, M)-container as a list of appearing n-boxes with associated numbers from $[1, M]$ encoded in binary.

In fact, interval predicates exactly describe finite unions of containers.

▶ **Proposition 19.** *The sets of configurations defined by interval predicates of height at most n and width at most M are exactly the sets formed by unions of (n, M)-containers.*

Proof sketch. For the translation from predicates to containers, consider a simple interval predicate $\dot{\exists}\mathsf{d}_1, \dots, \mathsf{d}_M,\ \bigwedge_{q \in Q} \bigwedge_{j=1}^{M} \#(q, \mathsf{d}_j) \in [A_{q,j}, B_{q,j}]$ of height n. This predicate cannot distinguish data mapped to the same n-box, hence cannot distinguish configurations in the same equivalence class for $\equiv_{n,M}$, i. e., (n, M)-containers. The same directly extends to interval predicates.

For the other direction, we prove that a given (n, M)-container can be expressed as an interval predicate of height at most n and width at most M. To do so, given a box $\mathsf{b} \in \mathbf{Boxes}_n$ and $m \le M$, we define the simple interval predicate $\varphi_{\mathsf{b},\ge m}$ expressing that at least m data are mapped to box b. Formally, $\varphi_{\mathsf{b},\ge m} := \dot{\exists}\mathsf{d}_1, \dots, \mathsf{d}_m,\ \bigwedge_{q \in Q} \bigwedge_{j=1}^{m} \#(q, \mathsf{d}_j) \in [A_q, B_q]$, where, for all $q \in Q$, $A_q := \mathsf{b}(q)$, $B_q := \mathsf{b}(q)$ if $\mathsf{b}(q) < n$ and $B_q := +\infty$ if $\mathsf{b}(q) = n$. This predicate has height at most n and width at most M. A Boolean combination of such predicates allows us to express an (n, M)-container. We refer to [7] for a detailed proof. ◀

We therefore have two equivalent representations. Both are useful: interval predicates allow us to express properties more naturally, but containers are more convenient for the proofs in the remainder of this section. While they are equally expressive, each can be much more succinct than the other, as stated below. We refer to [7] for details.

▶ Remark 20. Containers can be exponentially more succinct than interval predicates, while interval predicates can be doubly exponentially more succinct than unions of containers.

5.2 A Translation from Expressions to Containers

Based on the translation from interval predicates to containers from Proposition 19, we can now show that for all generalised reachability expressions E over an IOPPUD $\mathcal{P}$, the set $[\![E]\!]_{\mathcal{P}}$ is a union of (n, M)-containers with n and M bounded in terms of E and $\mathcal{P}$.

▶ **Proposition 21.** *There is a polynomial function* $\mathtt{poly}\colon \mathbb{N} \to \mathbb{N}$ *such that for all IOPPUD* $\mathcal{P}$ *and GRE* E*, the set* $[\![E]\!]_{\mathcal{P}}$ *is a union of* $\Big(||E|| \cdot \big(\mathtt{poly}(|\mathcal{P}|)\big)^{|E|}, ||E||^{\mathtt{poly}(|\mathcal{P}|)\cdot|E|^2}\Big)$*-containers.*

The detailed proof of Proposition 21 can be found in [7]. We show the result by structural induction on E. The base case, when E is an interval predicate, is provided by Proposition 19. For the induction step, handling Boolean operators is straightforward; the difficulty lies in operators Pre^* and Post^*. This is handled by the following lemma, which relies on the bounds from Section 4.

Equivalence classes for fixed values of n and M do not behave well with respect to the reachability relation, in the sense that it can happen that $\gamma_{start} \xrightarrow{*} \gamma_{end}$ and $\gamma_{start} \equiv_{n,M} \chi_{start}$, but there is no $\chi_{end} \equiv_{n,M} \gamma_{end}$ such that $\chi_{start} \xrightarrow{*} \chi_{end}$. However, this will hold if we take some margin on the equivalence relation of configurations at the start; the following two functions express this margin. For all $n, M \in \mathbb{N}$, let $f(n) := (n + |\mathcal{P}|^3) \cdot |\mathcal{P}|$ and $g(n, M) := \big(M + (|\mathcal{P}|^3 + 1)^{|\mathcal{P}|^3+|\mathcal{P}|^2}\big)(n + 1)^{|\mathcal{P}|}$.

The following lemma states that, if a set of configurations C cannot distinguish $\equiv_{n,M}$-equivalent configurations, then $\mathsf{Pre}^*(C)$ cannot distinguish $\equiv_{f(n),g(n,M)}$-equivalent configurations. In other words, if C is a union of $\equiv_{n,M}$-equivalence classes, (i.e., of (n, M)-containers), then $\mathsf{Pre}^*(C)$ is a union of $\equiv_{f(n),g(n,M)}$-equivalence classes.

▶ **Lemma 22.** *For all* $n, M \in \mathbb{N}$ *and all configurations* $\gamma_{start}, \gamma_{end}, \chi_{start} \in \Gamma$*, if there is a run* $\rho\colon \gamma_{start} \xrightarrow{*} \gamma_{end}$ *and* $\gamma_{start} \equiv_{f(n),g(n,M)} \chi_{start}$*, then there is a configuration* $\chi_{end} \in \Gamma$ *with* $\gamma_{end} \equiv_{n,M} \chi_{end}$ *and a run* $\pi\colon \chi_{start} \xrightarrow{*} \chi_{end}$*.*

Proof sketch. We first apply Corollary 17 to ρ, so that we can assume that ρ has a limited number of externally observed data and of observed agents per datum.

In this proof sketch, we first handle the case with only one datum. Then, we explain how to generalise this. We refer to [7] for proof details.

Suppose that all agents in γ_{start} and χ_{start} share a single datum d, and suppose $(\gamma_{start}, d) \equiv_{f(n)} (\chi_{start}, d)$. Let $\mathbb{A}_\gamma$ and $\mathbb{A}_\chi$ be the agents in γ_{start} and χ_{start}, respectively. For all $q, q' \in Q$, we set $\mathbb{A}_\gamma^{q\to} := \{a \in \mathbb{A}_\gamma \mid \gamma_{start}(a) = q\}$, $\mathbb{A}_\chi^{q\to} := \{a \in \mathbb{A}_\chi \mid \chi_{start}(a) = q\}$, $\mathbb{A}_\gamma^{\to q'} := \{a \in \mathbb{A}_\gamma \mid \gamma_{end}(a) = q'\}$, and $\mathbb{A}_\gamma^{q\to q'} := \mathbb{A}_\gamma^{q\to} \cap \mathbb{A}_\gamma^{\to q'}$.

Our aim is to assign to each agent in χ_{start} an agent in γ_{start} to mimic. To do so, we construct a mapping $\nu\colon \mathbb{A}_\chi \to \mathbb{A}_\gamma$ such that

(A) for all $a \in \mathbb{A}_\chi$, we have $\chi_{start}(a) = \gamma_{start}\big(\nu(a)\big)$,
(B) for all $a' \in \mathbb{A}_\gamma$ observed in ρ, we have $\nu^{-1}(a') \neq \emptyset$, and
(C) for all $q' \in Q$, we have $|\nu^{-1}(\mathbb{A}_{\gamma,d}^{\to q'})| = |\mathbb{A}_{\gamma,d}^{\to q'}|$, or $|\nu^{-1}(\mathbb{A}_{\gamma,d}^{\to q'})| \geq n$ and $|\mathbb{A}_{\gamma,d}^{\to q'}| \geq n$.

We build ν separately on each set $\mathbb{A}_\chi^{q\to}$ by defining, for each $q \in Q$, a mapping $\nu_q\colon \mathbb{A}_\chi^{q\to} \to \mathbb{A}_\gamma^{q\to}$. Let $q \in Q$. As $(\chi_{start}, d) \equiv_{f(n)} (\gamma_{start}, d)$, either $|\mathbb{A}_\gamma^{q\to}| = |\mathbb{A}_\chi^{q\to}|$, or both $|\mathbb{A}_\gamma^{q\to}|$ and $|\mathbb{A}_\chi^{q\to}|$ are at least $f(n)$. If $|\mathbb{A}_\gamma^{q\to}| = |\mathbb{A}_\chi^{q\to}|$, we let ν_q form a bijection between $\mathbb{A}_\chi^{q\to}$ and $\mathbb{A}_\gamma^{q\to}$. Consider now the second case, where $|\mathbb{A}_\gamma^{q\to}|$ and $|\mathbb{A}_\chi^{q\to}|$ are at least $f(n)$. We aim at selecting, for every $q' \in Q$, a set $A_{q\to q'} \subseteq \mathbb{A}_\gamma^{q\to q'}$ of agents that must be copied in π. If $|\mathbb{A}_\gamma^{q\to q'}| \leq n$, then we let $A_{q\to q'} := \mathbb{A}_\gamma^{q\to q'}$. Otherwise, we first put in $A_{q\to q'}$ all agents in $\mathbb{A}_\gamma^{q\to q'}$ that are observed in ρ, at most $|\mathcal{P}|^3$ in total by Corollary 17. If $|A_{q\to q'}| < n$, we add arbitrary agents from $\mathbb{A}_\gamma^{q\to q'}$ to $A_{q\to q'}$ until $|A_{q\to q'}| \geq n$. Either way, we have selected $A_{q\to q'}$ of size at most $|\mathcal{P}|^3 + n$ for each q', hence at most $f(n)$ agents in total. For every q', we have $|A_{q\to q'}| \leq |\mathbb{A}_\gamma^{q\to q'}|$, and either $|A_{q\to q'}| = |\mathbb{A}_\gamma^{q\to q'}|$ or the two sets have size more than n.

We now build ν_q such that its image over $\mathbb{A}_\chi^{q\to}$ is $\bigcup_{q'\in Q} A_{q\to q'}$. We build this in two steps. First, we assign to each $\bigcup_{q'\in Q} A_{q\to q'}$ one antecedent by ν_q in $\mathbb{A}_\chi^{q\to}$. This is possible because $|\bigcup_{q'\in Q} A_{q\to q'}| \leq f(n) \leq |\mathbb{A}_\chi^{q\to}|$. We then identify some q'' such that $|A_{q\to q''}| > n$ and map all remaining agents of $\mathbb{A}_\chi^{q\to}$ to an arbitrary agent in $A_{q\to q''}$. Such a q'' exists because $|\mathbb{A}_\gamma^{q\to}| \geq f(n) \geq n \cdot |\mathcal{P}|$, so there is a q' such that $|\mathbb{A}_\gamma^{q\to q'}| \geq n$, and hence $|A_{q\to q'}| \geq n$ by construction.

This concludes the construction of ν. It remains to prove that ν fulfils Items A–C. Items A and B are immediate from the definition. We prove Item C. Let $q' \in Q$. We distinguish two cases:

- if $|\mathbb{A}_\gamma^{q\to q'}| < n$ for all $q \in Q$, then for all q, we have $|\nu^{-1}(\mathbb{A}_\gamma^{q\to q'})| = |\nu^{-1}(A_{q\to q'})| = |A_{q\to q'}| = |\mathbb{A}_\gamma^{q\to q'}|$, so $|\nu^{-1}(\mathbb{A}_\gamma^{\to q'})| = |\mathbb{A}_\gamma^{\to q'}|$;
- if $|\mathbb{A}_\gamma^{q\to q'}| \geq n$ for some $q \in Q$, then $|A_{q\to q'}| \geq n$, so $|\nu^{-1}(\mathbb{A}_\gamma^{q\to q'})| \geq n$, and thus both $|\nu^{-1}(\mathbb{A}_\gamma^{\to q'})|$ and $|\mathbb{A}_\gamma^{\to q'}|$ are at least n.

We construct a run π from χ_{start} by copying ρ as follows. For each step of ρ where an agent a performs some transition t, we make $|\nu^{-1}(a)|$ steps in π so that all agents in $\nu^{-1}(a)$ perform transition t one by one. If a observed some agent a', there is a'' in π that can be observed because $\nu^{-1}(a') \neq \emptyset$: we made sure to map an agent to each observed agent in ρ.

For the general case with more data, we similarly construct two mappings μ and ν. First we define μ, which maps each datum d of χ_{start} to one of γ_{start} such that $(\gamma_{start}, \mu(d)) \equiv_{f(n)} (\chi_{start}, d)$. Then, for each datum d, ν maps each agent a with datum d of χ_{start} to one with datum $\mu(d)$ of γ_{start}.

Once μ and ν are defined, we build a run from χ_{start} to a configuration χ_{end} in which each agent a mimics the behaviour of $\nu(a)$ in ρ. We make sure that agents (resp. data) observed in ρ have agents (resp. data) mapped to them, so that we can take the same transitions in ρ and π. The construction of ν ensures that, for all data d, we have $(\gamma_{end}, \mu(d)) \equiv_{f(n)} (\chi_{end}, d)$. The construction of μ ensures that $\gamma_{end} \equiv_{f(n),g(n,M)} \chi_{end}$. ◀

6 Decidability and Complexity Bounds

6.1 Decidability in Exponential Space

In this section, we use the results on GRE from Section 5 to provide an ExpSpace upper bound for the emptiness problem for GRE. In the following, we assume that the representation of a GRE E takes $|E| + \log(||E||)$ space.

We first prove that we can decide membership of a configuration (encoded in a naive way) in a GRE in PSpace. A configuration is represented *data-explicitly* if it is represented as a list of vectors of $\mathbb{N}^Q$, one vector for each datum. The *size* of this representation is $k \cdot |\mathcal{P}| \cdot log(m)$ where k is the number of data and m is the number of agents appearing in γ.

▶ **Proposition 23.** *The following problem is decidable in* PSpace*: given a PPUD $\mathcal{P}$, a GRE E, and a configuration γ described data-explicitly, decide if $\gamma \in [\![E]\!]_{\mathcal{P}}$.*

We refer to the full version [7] for the proof. It uses a relatively straightforward induction on E to show that this problem can be decided in polynomial space using a recursive algorithm (with a polynomial whose degree does not depend on E). For the case where $E = \mathsf{Post}^*(F)$, we rely on the fact that the numbers of agents and data remain the same throughout a run;

we therefore can guess the configuration γ' such that $\gamma' \in [\![F]\!]_{\mathcal{P}}$ (which can be checked with a recursive call) and $\gamma \xrightarrow{*} \gamma'$ (which can be checked by exploration of the graph containing configurations with as many agents and data as γ). The case $E = \mathsf{Pre}^*(F)$ is similar.

Proposition 23 allows us to check if a given configuration of a PPUD is in the set described by a GRE[4]. In the case of IOPPUD, Proposition 21 allows us to search for a witness configuration within some bounded set, yielding decidability.

▶ **Theorem 10.** *The emptiness problem for GRE over IOPPUD is in* EXPSPACE.

Proof. Suppose $[\![E]\!]_{\mathcal{P}}$ is not empty. By Proposition 21, it contains an (n, M)-container cont with $n := ||E|| \cdot \mathtt{poly}(|\mathcal{P}|)^{|E|}$ and $M := ||E||^{\mathtt{poly}(|\mathcal{P}|)\cdot|E|^2}$. We construct a configuration $\gamma \in \mathsf{cont}$ as follows. For each n-box b, we select $\mathsf{cont}(\mathsf{b})$ many data such that over all n-boxes, the selected data are pairwise distinct. Then, for each n-box b, each state $q \in Q$ of $\mathcal{P}$, and each datum d_{b} selected for b, we put $\mathsf{b}(q)$ many agents with datum d_{b} in q. Note that the configuration γ is in cont, and the number of agents it contains it at most $n \cdot |\mathcal{P}| \cdot |\mathbf{Boxes}_n| \cdot M$. We have $|\mathbf{Boxes}_n| = (n+1)^{|\mathcal{P}|} = ||E|| \cdot \mathtt{poly}(|\mathcal{P}|)^{|E||\mathcal{P}|}$. We assumed at the beginning of Section 6 that the encoding of E uses memory $|E| + \log(||E||)$. As a result, n, M, $|\mathcal{P}|$ and $|\mathbf{Boxes}_n|$ are all at most exponential in the size of the input. Therefore, if $[\![E]\!]_{\mathcal{P}}$ is not empty, then it contains a configuration with at most exponentially many agents. We can guess the data-explicit description of such a configuration in non-deterministic exponential space, and then check that the guessed configuration is in $[\![E]\!]_{\mathcal{P}}$ in exponential space by Proposition 23 (we apply the PSPACE algorithm on an exponential input). As a result, deciding emptiness of $[\![E]\!]_{\mathcal{P}}$ is in NEXPSPACE, which is identical with EXPSPACE. ◀

6.2 A Lower Complexity Bound

We now provide the following lower complexity bound.

▶ **Theorem 24.** *The emptiness problem for GRE over IOPPUD is* coNEXPTIME*-hard.*

Proof sketch. We proceed by reduction from the problem of tiling an exponentially large grid, a NEXPTIME-complete problem [28], to the complement of the emptiness problem for GRE. We refer to [7] for proof details.

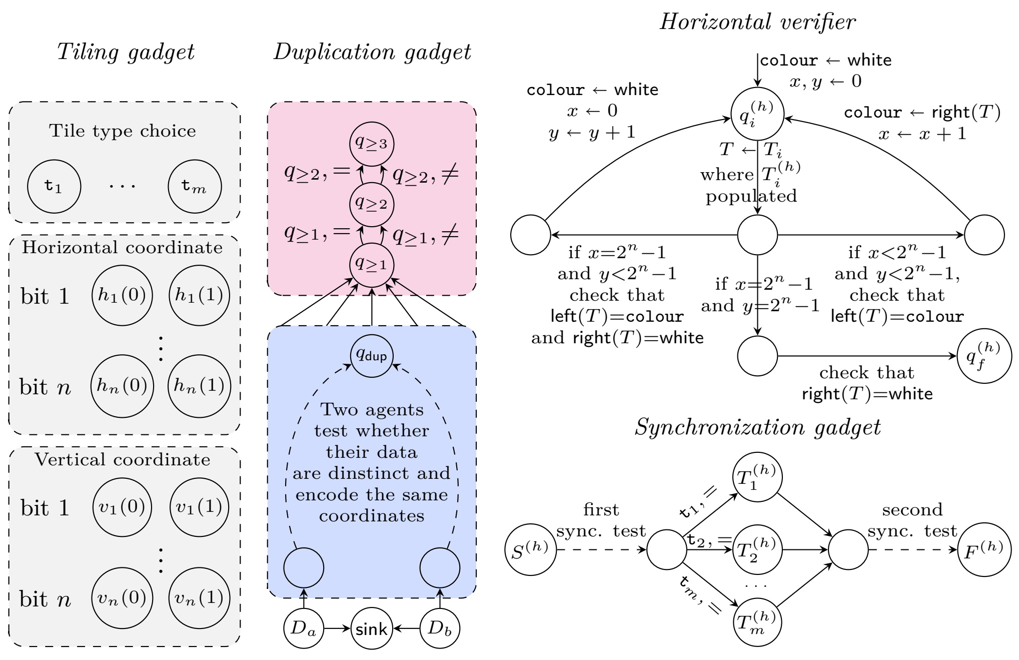
A *tiling instance* is a tuple $(2^n, \mathcal{C}, \mathcal{T})$, with $n \geq 1$, $\mathcal{C}$ a finite set of *colours* with special colour white, and $\mathcal{T} = \{t_1, \dots, t_m\} \subseteq \mathcal{C}^4$ a finite set of *tiles*. We can view a tile as a square whose four edges are coloured. The *tiling problem* asks whether there is a *tiling*, that is, a mapping $\tau \colon [0, 2^n{-}1] \times [0, 2^n{-}1] \to \mathcal{T}$ such that the colours of neighbouring tiles match and the borders of the grid are white.

Given a tiling instance $(2^n, \mathcal{C}, \mathcal{T})$, we build an instance $(\mathcal{P}, E)$ of the emptiness problem for GRE. In $\mathcal{P}$, witness tilings can be encoded in the configurations, and we construct $(\mathcal{P}, E)$ such that $[\![E]\!]$ contains exactly the configurations that correspond to a correctly encoded witness tiling. More precisely, $\gamma \in [\![E]\!]$ when:

(Cond1) for all $(i, j) \in [0, 2^n - 1]^2$, some datum encodes coordinates (i, j) and a tile type;
(Cond2) for all $(i, j) \in [0, 2^n - 1]^2$, there is at most one datum encoding (i, j);
(Cond3) the mapping $[0, 2^n - 1]^2 \to \mathcal{C}$ defined by the data is a tiling.

The GRE E will be of the form of a conjunction, i. e., a list of *constraints* that the configuration must satisfy. Our first constraint is $\overline{\mathsf{Pre}^*(\mathsf{Pres}(q_\perp))}$ where $q_\perp$ is a special error state and $\mathsf{Pres}(q)$ is the GRE expressing that some agent is in q. This forbids, in $[\![E]\!]$, configurations from which $q_\perp$ can be covered.

[4] This implies that the emptiness problem for GRE over PPUD, while undecidable due to Theorem 3, is semi-decidable: one can simply enumerate all configurations and test membership for each of them.



Tiling gadget
Tile type choice
t_1 … t_m
Horizontal coordinate
bit 1 $h_1(0)$ $h_1(1)$
bit n $h_n(0)$ $h_n(1)$
Vertical coordinate
bit 1 $v_1(0)$ $v_1(1)$
bit n $v_n(0)$ $v_n(1)$
Duplication gadget
$q_{\geq 3}$
$q_{\geq 2}, =$
$q_{\geq 2}, \neq$
$q_{\geq 2}$
$q_{\geq 1}, =$
$q_{\geq 1}, \neq$
$q_{\geq 1}$
q_{dup}
Two agents test whether their data are dinstinct and encode the same coordinates
D_a
sink
D_b
Horizontal verifier
colour ← white $x, y \leftarrow 0$
$q_i^{(h)}$
colour ← white $x \leftarrow 0$ $y \leftarrow y+1$
colour ← right(T) $x \leftarrow x+1$
$T \leftarrow T_i$ where $T_i^{(h)}$ populated
if $x{=}2^n{-}1$ and $y{<}2^n{-}1$ check that left(T)=colour and right(T)=white
if $x{=}2^n{-}1$ and $y{=}2^n{-}1$
if $x{<}2^n{-}1$ and $y{<}2^n{-}1$, check that left(T)=colour
check that right(T)=white
$q_f^{(h)}$
Synchronization gadget
$S^{(h)}$
first sync. test
$t_1, =$
$t_2, =$
$t_m, =$
$T_1^{(h)}$
$T_2^{(h)}$
$T_m^{(h)}$
second sync. test
$F^{(h)}$

Figure 4 Partial depiction of the protocol constructed in Theorem 24.

(Cond1) is obtained using the *tiling gadget* in Figure 4. States $\mathtt{t}_1, \dots, \mathtt{t}_m$ represent the available tiles of $\mathcal{T}$, and coordinate states allow for a binary representation of the horizontal and vertical coordinates of a square in the grid. For a datum d, the agents of datum d in the coordinate states encode the position of the square corresponding to d, and an agent of datum d in state $\mathtt{t}_i$ indicates that the square in the grid corresponding to d should be coloured according to tile t_i. Configurations in $[\![E]\!]$ are not allowed to have two agents of same datum playing the same role; otherwise, one of them may observe the other and go to $q_\perp$. In particular, each datum has at most $2n+1$ agents in the tiling gadget.

To obtain (Cond2), we use a *duplication gadget*, partially represented in Figure 4. We enforce that any configuration in $[\![E]\!]$ has one agent of each datum in D_a, one in D_b and none in the rest of the duplication gadget. The blue part implements a test (depicted in [7, Figure 5b]) where two agents of distinct data, one from D_a and one from D_b, may test that their data encode the same coordinates; if this is the case, they may go to q_{dup}. If there are more than two agents in the blue part, this test is not reliable but $q_{\geq 3}$ can be covered. (Cond2) can therefore be achieved by enforcing that configurations in $[\![E]\!]$ are *not* in $\mathsf{Pre}^*(\mathsf{Pres}(q_{\mathsf{dup}}) \cap \overline{\mathsf{Pre}^*(\mathsf{Pres}(q_{\geq 3}))})$.

Finally, we explain how (Cond3) is achieved; we describe only how the horizontal (left-right) borders are verified. We use a gadget, named *horizontal verifier* in Figure 4. In this gadget, a single agent, called *verifier*, is in charge of verifying that colours of left-right borders match. The verifier uses $2n$ auxiliary agents to encode two variables $x, y \in [0, 2^n - 1]$ in binary. Again, transitions to $q_\perp$ detect when two agents play the same role, so that there is only one verifier and so that variables x and y can be implemented faithfully. The initialisation $x = y = 0$ is enforced as a constraint in E. We now sketch how the verifier reads the encoded tiling; to do that, it must synchronise with the datum encoding (x, y).

This is done using the synchronisation gadget of Figure 4. In $[\![E]\!]$, all agents in the synchronisation gadget are in $S^{(h)}$. Moreover, we add a constraint in E so that $\gamma \in [\![E]\!]$ requires that there is a run from γ where all agents in the synchronisation gadget end in

$F^{(h)}$ and where the verifier ends in $q_f^{(h)}$. The synchronisation tests guarantee that, whenever there is an agent in $T_i^{(h)}$, this agent's datum encodes square (i, j) where i is equal to the current value of x and j is equal to the current value of y. The synchronisation is challenging to design because the values of x and y may change throughout a run and only one bit can be tested at a time. However, as proved in [7, Lemma 37], this can be achieved by having a first synchronisation test that checks equality of bits from most to least significant, and a second test that checks equality from least to most significant. ◀

6.3 Discussion on Complexity Gaps

We now discuss some complexity gaps left open by this paper. First, there remains a complexity gap for the emptiness problem for GRE, which is known to be between coNExpTime (Theorem 24) and ExpSpace (Theorem 10). Closing the gap appears challenging. On one hand, if the problem is below ExpSpace, then this probably requires developing new techniques. On the other hand, proving ExpSpace-hardness does not seem easy. In particular, the synchronisation techniques from Theorem 24 assumes that each datum synchronises only once with the verifier. This synchronisation technique would not suitable for, e. g., multiple interactions between the head and the cells of a Turing machine.

Another, arguably more important open question is the exact complexity of well-specification, which is only known to be between PSpace (model without data, [19]) and ExpSpace (Theorem 10). On the one hand, it is unclear whether relevant configurations can be stored in polynomial space.

▷ Claim 25. The number of data that need to be considered for well-specification may be exponential.

The claim is formalised and proven in [7]. As a consequence, proving that the problem is in PSpace cannot be achieved with a procedure that explicitly stores configurations. On the other hand, in order to build a reduction from the tiling problem as in Theorem 24, we need a new idea to enforce that *at most* one datum encodes each tile. In Theorem 24, we had states q_{dup} and $q_{\geq 3}$ and duplication meant being able to cover q_{dup} and, at the same, forbid that $q_{\geq 3}$ can ever be covered in the future. We do not know how to encode this constraint when working with an instance of well-specification.

7 Conclusion

We have studied the verification of population protocols with unordered data [10], an extension of population protocols where agents carry data from an infinite unordered set. We first proved that the well-specification problem is undecidable (Theorem 3), which then led us to consider the restriction to protocols with immediate observation. This subclass was defined in [10], where the authors proved that these protocols compute exactly the interval predicates. We defined a general class of problems on this model, which consists in deciding the existence of a configuration satisfying a so-called generalised reachability expression; this class of problems subsumes many classic problems, one of which is well-specification. Despite its generality, we showed the problem to be decidable in exponential space (Theorem 10); we also provided a coNExpTime lower bound. A remaining open question is the exact complexity of well-specification for immediate observation population protocols with unordered data, which is located between PSpace (model without data, [19]) and ExpSpace (Theorem 10).

References

1 Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L. Rivest. Time-space trade-offs in population protocols. In *28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 2560–2579, 2017. `doi:10.1137/1.9781611974782.169`.

2 Dan Alistarh and Rati Gelashvili. Recent algorithmic advances in population protocols. *SIGACT News*, 49(3):63–73, 2018. `doi:10.1145/3289137.3289150`.

3 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. In *33rd Annual ACM Symposium on Principles of Distributed Computing, PODC 2004*, pages 290–299. ACM, 2004. `doi:10.1145/1011767.1011810`.

4 Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Comput.*, 20(4):279–304, 2007. `doi:10.1007/S00446-007-0040-2`.

5 A. R. Balasubramanian, Lucie Guillou, and Chana Weil-Kennedy. Parameterized analysis of reconfigurable broadcast networks. In *Foundations of Software Science and Computation Structures - 25th International Conference, FoSSaCS 2022*, pages 61–80, 2022. `doi:10.1007/978-3-030-99253-8_4`.

6 A. R. Balasubramanian and Chana Weil-Kennedy. Reconfigurable broadcast networks and asynchronous shared-memory systems are equivalent. In *12th International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2021*, pages 18–34, 2021. `doi:10.4204/EPTCS.346.2`.

7 Steffen van Bergerem, Roland Guttenberg, Sandra Kiefer, Corto Mascle, Nicolas Waldburger, and Chana Weil-Kennedy. Verification of population protocols with unordered data. *CoRR*, abs/2405.00921, 2024. `arXiv:2405.00921`.

8 Roderick Bloem, Swen Jacobs, Ayrat Khalimov, Igor Konnov, Sasha Rubin, Helmut Veith, and Josef Widder. *Decidability of Parameterized Verification.* Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2015. `doi:10.2200/S00658ED1V01Y201508DCT013`.

9 Michael Blondin, Javier Esparza, Stefan Jaax, and Philipp J. Meyer. Towards efficient verification of population protocols. *Formal Methods Syst. Des.*, 57(3):305–342, 2021. `doi:10.1007/S10703-021-00367-3`.

10 Michael Blondin and François Ladouceur. Population protocols with unordered data. In *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023*, pages 115:1–115:20, 2023. `doi:10.4230/LIPICS.ICALP.2023.115`.

11 Patricia Bouyer, Nicolas Markey, Mickael Randour, Arnaud Sangnier, and Daniel Stan. Reachability in networks of register protocols under stochastic schedulers. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016*, pages 106:1–106:14, 2016. `doi:10.4230/LIPICS.ICALP.2016.106`.

12 Philipp Czerner, Roland Guttenberg, Martin Helfrich, and Javier Esparza. Fast and succinct population protocols for Presburger arithmetic. *J. Comput. Syst. Sci.*, 140:103481, 2024. `doi:10.1016/J.JCSS.2023.103481`.

13 Wojciech Czerwinski and Lukasz Orlikowski. Reachability in vector addition systems is Ackermann-complete. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 1229–1240, 2021. `doi:10.1109/FOCS52979.2021.00120`.

14 Robert Elsässer and Tomasz Radzik. Recent results in population protocols for exact majority and leader election. *Bull. EATCS*, 126, 2018. URL: `http://bulletin.eatcs.org/index.php/beatcs/article/view/549/546`.

15 Javier Esparza. Keeping a crowd safe: On the complexity of parameterized verification (invited talk). In *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France*, pages 1–10, 2014. `doi:10.4230/LIPICS.STACS.2014.1`.

16 Javier Esparza. Population protocols: Beyond runtime analysis. In *Reachability Problems - 15th International Conference, RP 2021*, pages 28–51, 2021. doi:10.1007/978-3-030-89716-1_3.

17 Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar. Verification of population protocols. *Acta Informatica*, 54(2):191–215, 2017. doi:10.1007/S00236-016-0272-3.

18 Javier Esparza, Stefan Jaax, Mikhail A. Raskin, and Chana Weil-Kennedy. The complexity of verifying population protocols. *Distributed Comput.*, 34(2):133–177, 2021. doi:10.1007/S00446-021-00390-X.

19 Javier Esparza, Mikhail A. Raskin, and Chana Weil-Kennedy. Parameterized analysis of immediate observation Petri nets. In *Application and Theory of Petri Nets and Concurrency - 40th International Conference, PETRI NETS 2019*, pages 365–385, 2019. doi:10.1007/978-3-030-21571-2_20.

20 Lucie Guillou, Corto Mascle, and Nicolas Waldburger. Parameterized broadcast networks with registers: from NP to the frontiers of decidability. In *Foundations of Software Science and Computation Structures - 27th International Conference, FoSSaCS 2024*, pages 250–270, 2024. doi:10.1007/978-3-031-57231-9_12.

21 Petr Jancar. Undecidability of bisimilarity for Petri nets and some related problems. *Theor. Comput. Sci.*, 148(2):281–301, 1995. doi:10.1016/0304-3975(95)00037-W.

22 Petr Jancar and Jérôme Leroux. The semilinear home-space problem is Ackermann-complete for Petri nets. In *34th International Conference on Concurrency Theory, CONCUR 2023*, pages 36:1–36:17, 2023. doi:10.4230/LIPICS.CONCUR.2023.36.

23 Ranko Lazic, Thomas Christopher Newcomb, Joël Ouaknine, A. W. Roscoe, and James Worrell. Nets with tokens which carry data. In *28th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, ICATPN 2007*, pages 301–320, 2007. doi:10.1007/978-3-540-73094-1_19.

24 Jérôme Leroux. The reachability problem for Petri nets is not primitive recursive. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 1241–1252, 2021. doi:10.1109/FOCS52979.2021.00121.

25 Jérôme Leroux and Sylvain Schmitz. Reachability in vector addition systems is primitive-recursive in fixed dimension. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019*, pages 1–13, 2019. doi:10.1109/LICS.2019.8785796.

26 Marvin L. Minsky. *Computation: Finite and Infinite Machines.* Prentice-Hall, Inc., 1967.

27 Fernando Rosa-Velardo and David de Frutos-Escrig. Decidability and complexity of Petri nets with unordered data. *Theor. Comput. Sci.*, 412(34):4439–4451, 2011. doi:10.1016/J.TCS.2011.05.007.

28 François Schwarzentruber. The complexity of tiling problems. *CoRR*, abs/1907.00102, 2019. arXiv:1907.00102.

29 Chana Weil-Kennedy. *Observation Petri Nets.* PhD thesis, Technical University of Munich, Germany, 2023. URL: https://nbn-resolving.org/urn:nbn:de:bvb:91-diss-20230320-1691161-1-3.