Acta Informatica (2017) 54:191–215
DOI 10.1007/s00236-016-0272-3

ORIGINAL ARTICLE

Verification of population protocols

Javier Esparza[1] · **Pierre Ganty**[2] · **Jérôme Leroux**[3] · **Rupak Majumdar**[4]

Received: 26 January 2016 / Accepted: 8 June 2016 / Published online: 17 June 2016

Abstract Population protocols (Angluin et al. in PODC, 2004) are a formal model of sensor networks consisting of identical mobile devices. Two devices can interact and thereby change their states. Computations are infinite sequences of interactions satisfying a strong fairness constraint. A population protocol is well specified if for every initial configuration C of devices, and every computation starting at C, all devices eventually agree on a consensus value depending only on C. If a protocol is well specified, then it is said to compute the predicate that assigns to each initial configuration its consensus value. While the computational power of well-specified protocols has been extensively studied, the two basic verification problems remain open: Is a given protocol well specified? Does a given protocol compute a given predicate? We prove that both problems are decidable by reduction to the reachability problem of Petri nets. We also give a new proof of the fact that the predicates computed by well-defined protocols are those definable in Presburger arithmetic (Angluin et al. in *PODC*, 2006).

Mathematics Subject Classification C.2.2 · D.2.4 · F.3.1

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

✉ Pierre Ganty
pierreganty@gmail.com

1 TUM, Munich, Germany

2 IMDEA Software Institute, Madrid, Spain

3 LaBRI, CNRS, Univ. Bordeaux, Bordeaux, France

4 MPI-SWS, Kaiserslautern, Germany

1 Introduction

Population protocols [2,3] are a model of distributed computation by anonymous, identical finite-state agents. While they were initially introduced to model networks of passively mobile sensors [2,3], they capture the essence of distributed computation in diverse areas such as trust propagation [13] and chemical reactions [31].

In each computation step of a population protocol, a fixed number of agents are chosen nondeterministically, and their states are updated according to a joint transition function. Since agents are anonymous and identical, the global state of a protocol is completely determined by the number of agents at each local state, called a configuration. A protocol computes a boolean value for a given initial configuration if in all fair executions starting at it, all agents eventually agree to this value—so, intuitively, population protocols compute by reaching consensus. An execution is fair if it is finite and cannot be extended, or it is infinite and satisfies the following condition: if C appears infinitely often in the execution, then every step enabled at C is taken infinitely often in the execution. Given a set of inputs (typically a set of vectors of natural numbers), and a mapping that assigns to each input an initial configuration, the predicate computed by a protocol is the function that assigns to each input the boolean value computed by the protocol from the corresponding initial configuration. If the protocol does not reach consensus for some input, then we say it is ill specified and does not compute any predicate.

Much of the work on population protocols has concentrated on characterizing the predicates computable by well-specified protocols. In particular, Angluin et al. [2,3] gave explicit well-specified protocols to compute every predicate definable in Presburger arithmetic, and showed in a later paper (with a different set of authors) that they cannot compute anything else, i.e., well-specified population protocols compute exactly the Presburger-definable predicates [5,7].

Since it is easy to erroneously design protocols that are not well specified, one can ask two natural verification questions: Given a population protocol, is it well specified? Given a population protocol and a Presburger predicate (represented by a Presburger formula), does the protocol compute the predicate? We call them the *well-specification* and *fitting* problems.

The semantics of a population protocol is an infinite family of finite-state transition systems, one for each possible input. Deciding if the protocol reaches consensus for a fixed input only requires to inspect one of these finite transition systems, and can be done automatically using a model checker. This approach has been followed in [10,11,32,33], but it only proves the correctness of a protocol for a finite number of (typically small) inputs. Alternatively, one can also formalize a proof of well specification in a theorem prover [12], but this approach is not automatic: a human prover must first come up with a proof for each particular protocol.

Since the well-specification problem asks if consensus is reached for all inputs, and there are infinitely many of them, it is not obviously decidable; in fact, similar questions are undecidable for many parameterized systems [8]. Moreover, techniques based on algorithms for the coverability problem of Petri nets, or on well-quasi-orders—which have been used to prove decidability of many parameterized verification problems [1,17]—cannot be directly applied to the well specification and fitting problems. Loosely speaking, the reason is that the set of initial configurations from which all agents eventually agree on a value is not necessarily upward- nor downward-closed.

Despite these difficulties, in the first part of the paper we show that the well-specification and fitting problems are decidable and recursively equivalent to the reachability problem for Petri nets.

In the second part of the paper we study the *tailor* problem: Given a well-specified protocol, returns a Presburger formula for the predicate computed by it. To solve the problem, we introduce a notion of certificate (of well-specification) of a protocol. We provide algorithms that, given a protocol and an advice string decide if the string is a certificate, and extract from it a Presburger formula of the predicate computed by the protocol. The overall algorithm for the tailor problem just enumerates all advice strings, checks if they are a certificate, and if so computes a formula. However, this algorithm may not terminate if a protocol happens to have no certificates. So we also show that this is not the case: every well-specified protocol has at least one certificate. The proof relies on several recent results from the theory of Petri nets: the existence of Presburger-definable inductive sets that separate unreachable markings [22], the effective Presburger-definability of the mutual reachability relation [23,26], and a result from the theory of accelerations [25]. Finally, along the way we obtain a new proof of the main theorem of [5,7] showing that well-specified protocols can only compute Presburger-definable predicates.

The paper is organized as follows. Section 2 presents some preliminaries. Section 3 introduces population protocols and defines the well-specification, fitting, and tailor problems. Section 4 describes the connection between population protocols and Petri nets. Sections 5 and 6 reduce the well-specification and fitting problems to the reachability problem for Petri nets, and Sect. 7 presents reductions in the other direction. Finally, Sect. 8 presents our certificate-based algorithm for the tailor problem.

2 Preliminaries: Presburger sets, semilinear sets, multisets

Presburger arithmetic and Presburger sets Presburger arithmetic is the first-order theory of addition, i.e., the first-order theory of the natural numbers with addition as only function, and equality as only predicate. A set $\mathbf{S} \subseteq \mathbb{N}^d$ is a *Presburger-definable*, or just a *Presburger set*, if there exists a formula $\mathtt{F}(x_1, \ldots, x_d)$ with free variables $x_1, \ldots, x_d$ such that $\mathtt{F}(n_1, \ldots, n_d)$ is true iff $(n_1, \ldots, n_d) \in \mathbf{S}$.

Semi-linear sets A set $\mathbf{L} \subseteq \mathbb{N}^d$ is *linear* if there is a *base* or *root vector* $\mathbf{b}$ and a finite set $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ of *periods* such that $\mathbf{L} = \{\mathbf{b} + \sum_{i=1}^{n} \lambda_i \mathbf{p}_i \mid (\lambda_1, \ldots, \lambda_n) \in \mathbb{N}^d\}$. We write $\mathbf{L} = (\mathbf{b}; \mathbf{P})$, and say that the pair $(\mathbf{b}; \mathbf{P})$ is a *linear representation* of $\mathbf{L}$. A set $\mathbf{S}$ is *semi-linear set* if it is a finite union of linear sets, and the set of its linear representations is called a *semi-linear representation* of $\mathbf{S}$.

It is well known that the semi-linear sets and the Presburger sets coincide [19]. In particular, semi-linear sets are effectively closed under Boolean operations and emptiness, inclusion, and equivalence of semi-linear sets are all decidable.

Multisets A *multiset* on a finite set E is a mapping $M : E \to \mathbb{N}$. For $e \in E$, $M(e)$ denotes the number of occurrences of element e in M. Operations on $\mathbb{N}$ like addition, subtraction, or comparison, are extended to multisets by defining them component wise. The set of all multisets over E is denoted $\mathbb{N}^E$. Given $e \in E$, we denote $\mathbf{e} \in \mathbb{N}^E$ the multiset consisting of one occurrence of element e, that is, the multiset satisfying $\mathbf{e}(e) = 1$ and $\mathbf{e}(e') = 0$ for every $e' \neq e$. The *support* of a multiset $M \in \mathbb{N}^E$, denoted by $\mathrm{Sup}(M)$, is the set $\{e \in E \mid M(e) > 0\}$.

Given a total order $e_1 \prec e_2 \prec \cdots \prec e_n$ on E, a multiset M can be represented by the vector $(M(e_1), \ldots, M(e_n))$, and a set $\mathcal{M}$ of multisets by a set of vectors. A set of multisets over a finite set E is Presburger (resp. linear, semi-linear) if its corresponding set of vectors is Presburger (resp. linear, semi-linear)

3 Population protocols

A *population* P on a finite set E is a non-empty multiset on E, i.e., $P \in \mathbb{N}^E$ and $P \neq \emptyset$. Thus $P(e) > 0$ for some $e \in E$, which is equivalent to $\Sigma_{e \in E} P(e) > 0$. The set of all populations on E is denoted by $\text{Pop}(E)$.

Example 1 Let $E = \{a, b\}$. The set of populations $\{P \in \text{Pop}(E) \mid P(a) \geq P(b)\}$ is Presburger, since it is denoted by the Presburger formula $\exists Y : X_a = Y + X_b \wedge X_b > 0$. It is easy to see that the set of populations $\{P \in \text{Pop}(E) \mid P(a) = P(b)^2\}$ is not Presburger. □

3.1 Protocol schemes

A *protocol scheme* $\mathcal{A} = (Q, \Delta)$ consists of a finite non-empty set Q of states and a set $\Delta \subseteq Q^4$. If $(q_1, q_2, q_1', q_2') \in \Delta$, we write $(q_1, q_2) \mapsto (q_1', q_2')$ and call it a *transition*. The populations of $\text{Pop}(Q)$ are called *configurations*. Intuitively, a configuration C describes a collection of identical finite-state *agents* with Q as set of states, containing $C(q)$ agents in state q for every $q \in Q$. Pairs of agents interact using transitions from Δ.[1] Formally, given two configurations C and C' and a transition $\delta = (q_1, q_2) \mapsto (q_1', q_2')$, we write $C \xrightarrow{\delta} C'$ if

$$C \geq (\mathbf{q}_1 + \mathbf{q}_2) \text{ holds, } \quad \text{and} \quad C' = C - (\mathbf{q}_1 + \mathbf{q}_2) + (\mathbf{q}_1' + \mathbf{q}_2') \ .$$

From the definition of step, it is easily seen that, inside the tuple $(q_1, q_2, q_1', q_2') \in \Delta$, the ordering between q_1 and q_2 and between q_1' and q_2' is irrelevant. We write $C \xrightarrow{w} C'$ for a sequence $w = \delta_1 \ldots \delta_k$ of transitions if there exists a sequence $C_0, \ldots, C_k$ of configurations satisfying $C = C_0 \xrightarrow{\delta_1} C_1 \cdots \xrightarrow{\delta_k} C_k = C'$. We also write $C \rightarrow C'$ if $C \xrightarrow{\delta} C'$ for some transition $\delta \in \Delta$, and call $C \rightarrow C'$ a *step*. We say that C' is *reachable from* C if $C \xrightarrow{w} C'$ for some (possibly empty) sequence w of transitions. Further, two configurations C, C' are *mutually reachable* if C is reachable from C' and C' is reachable from C. We have:

Lemma 1 *For every configuration C, the set of configurations reachable from C is finite.*

Proof Follows immediately from the fact that an interaction does not create or destroy agents, just changes their current states. Since Q is finite, there are only finitely many configurations C' satisfying $\sum_{q \in Q} C(q) = \sum_{q \in Q} C'(q)$. □

Example 2 (Debating philosophers) We consider a protocol scheme $\mathcal{A} = (XQ, \Delta)$ with agents called “philosophers”. A group of philosophers debate about a thesis, say, “do animals have rights?”. At each point in time a philosopher is tired or rested, denoted by `T` and `R`, respectively, and is for or against the thesis, denoted by `F` and `A`. The set Q contains four states

$$Q = \{\mathtt{T}, \mathtt{R}\} \times \{\mathtt{F}, \mathtt{A}\} \ .$$

The interactions between the philosophers model the following behavior. When two philosophers meet, they compare their positions and update their current state as follows:

(i) Philosophers with the same opinion do not debate and stay in their current state.
(ii) Rested philosophers convince tired opponents of anything.

[1] While protocol schemes model pairwise interactions only, one can model k-way interactions for a fixed $k > 2$ by adding additional states.

(iii) If two philosophers in the same physical condition debate, the one for animal rights convinces the one against and they both are tired after the debate.

Accordingly, the set Δ is defined as follows where $\alpha, \beta \in \{\texttt{F}, \texttt{A}\}, \alpha \neq \beta$ and $\texttt{X}, \texttt{Y} \in \{\texttt{R}, \texttt{T}\}$:

$$(\texttt{X}\alpha, \texttt{Y}\alpha) \mapsto (\texttt{X}\alpha, \texttt{Y}\alpha) \qquad (\texttt{R}\alpha, \texttt{T}\beta) \mapsto (\texttt{R}\alpha, \texttt{T}\alpha) \qquad (\texttt{X}\alpha, \texttt{X}\beta) \mapsto (\texttt{TF}, \texttt{TF}) \ .$$

We represent configurations as tuples indicating the number of philosophers in each state. Here is a possible infinite step sequence of the protocol scheme.

$$\begin{array}{c}\texttt{RF RA TF TA}\\ (\;3\;\;3\;\;0\;\;0\;)\end{array} \rightarrow \begin{array}{c}\texttt{RF RA TF TA}\\ (\;2\;\;2\;\;2\;\;0\;)\end{array} \rightarrow \begin{array}{c}\texttt{RF RA TF TA}\\ (\;2\;\;2\;\;1\;\;1\;)\end{array} \rightarrow \begin{array}{c}\texttt{RF RA TF TA}\\ (\;1\;\;1\;\;3\;\;1\;)\end{array} \rightarrow$$
$$\begin{array}{c}\texttt{RF RA TF TA}\\ (\;0\;\;0\;\;5\;\;1\;)\end{array} \rightarrow \begin{array}{c}\texttt{RF RA TF TA}\\ (\;0\;\;0\;\;6\;\;0\;)\end{array} \rightarrow \begin{array}{c}\texttt{RF RA TF TA}\\ (\;0\;\;0\;\;6\;\;0\;)\end{array} \rightarrow \begin{array}{c}\texttt{RF RA TF TA}\\ (\;0\;\;0\;\;6\;\;0\;)\end{array} \cdots \quad \square$$

3.2 Configuration graphs

The *configuration graph* of a protocol scheme $\mathcal{A}$ is the infinite directed graph $(\text{Pop}(Q), \rightarrow$ having the populations over Q as nodes and the pairs (C, C') such that $C \rightarrow C'$ as edges. Consider the partition $\{\text{Pop}(Q)_i\}_{i\geq 1}$ of $\text{Pop}(Q)$, where $\text{Pop}(Q)_i = \{C \in \text{Pop}(Q) \mid \sum_{q\in Q} C(q) = i\}$. (Note that i starts at 1 because every population contains at least one agent. Populations with ony one agent are not interesting, but they make the definition of the predicate computed by a population protocol more natural, see page 9.) Since interactions do not create or destroy agents, the set $\{\rightarrow_i\}_{i\geq 1}$, where $\rightarrow_i = \rightarrow \cap \text{Pop}(Q)_i^2$, is also a partition of $\rightarrow$. Therefore $(\text{Pop}(Q), \rightarrow)$ consists of the infinitely many disjoint and finite subgraphs $\{(\text{Pop}(Q)_i, \rightarrow_i)\}_{i\geq 1}$.

A *strongly connected component* (SCC) of the configuration graph is a maximal set of mutually reachable configurations. An SCC is a *bottom SCC* if it is closed under reachability, i.e., if C belongs to the SCC and C' is reachable from C, then C' also belongs to the SCC. A configuration is a *bottom configuration* if it belongs to a bottom SCC of the configuration graph.

Example 3 (Debating philosophers) In the step sequence of Example 2 the number of philosophers remains constant at 6, and so all its configurations belong to $\text{Pop}(Q)_6$.

We prove that the set $\mathcal{B}$ of bottom configurations of the debating philosophers is $\mathcal{B} = \mathcal{B}_{\texttt{F}} \cup \mathcal{B}_{\texttt{A}}$, where

$$\mathcal{B}_{\texttt{F}} = \{\, (rf, 0, tf, 0) \ \mid rf + tf > 0 \,\} \quad \text{and} \quad \mathcal{B}_{\texttt{A}} = \{\, (0, ra, 0, ta) \ \mid ra + ta > 0 \,\} \ .$$

(Observe that in the configurations of $\mathcal{B}$ all philosophers have the same opinion: in $\mathcal{B}_{\texttt{F}}$ they are all for animal rights, and in $\mathcal{B}_{\texttt{A}}$ against them.) If $C \in \mathcal{B}$, then the only possible step is $C \rightarrow C$. So $\{C\}$ is a bottom SCC, and C is a bottom configuration. It remains to prove that if $C \notin \mathcal{B}$ then C is not a bottom configuration. For this it suffices to exhibit a configuration C' reachable from C such that C is not reachable from C' (i.e., C' is reachable from C, but C and C' are not mutually reachable). Let $C = (rf, ra, tf, ta) \notin \mathcal{B}$. We consider several cases:

– $rf > 0$ and $ra > 0$. Then $C \rightarrow C'$ for $C' = (rf - 1, ra - 1, tf + 2, ta)$ (two rested philosophers debate and get tired) but, since no rules turn a tired philosopher into a rested one, C is not reachable from C'.
– $rf > 0$ and $ra = 0$. Since $C \notin \mathcal{B}$ we have $ta > 0$, and so $C' = (rf, 0, tf + ta, 0)$ is reachable from C (a rested philosopher for animal rights convinces all tired philosophers against them), but not vice versa.

- $rf = 0$ and $ra > 0$. Since $C \notin \mathcal{B}$ we have $tf > 0$, and so $C' = (0, ra, 0, tf + ta)$ is reachable from C (a rested philosopher against animal rights convinces all tired philosophers in favor of them), but not vice versa.
- $rf = 0$ and $ra = 0$. Since $C \notin \mathcal{B}$ we have $tf, ta > 0$, and so $C' = (0, 0, tf + ta, 0)$ is reachable from C (a tired philosopher for animal rights convinces all tired philosophers against them), but not vice versa. □

3.3 Executions and fair executions

An *execution* of $\mathcal{A}$ is a finite or infinite sequence of configurations $C_0, C_1, \ldots$ such that $C_i \to C_{i+1}$ for each $i \geq 0$. Following Angluin et al. [2,3], we introduce a notion of fair execution. Loosely speaking, if C appears infinitely often in a fair execution, then every step enabled at C is taken infinitely often in the execution. Formally, an execution $C_0, C_1, \ldots$ is *fair* if it is finite and cannot be extended, or it is infinite and for every step $C \to C'$, if $C_i = C$ for infinitely many indices $i \geq 0$, then $C_j = C$ and $C_{j+1} = C'$ for infinitely many indices $j \geq 0$. Thanks to Lemma 1 we show in Lemma 3 that every fair execution reaches a strongly connected component (SCC) of $(\mathrm{Pop}(Q), \to)$ and never leaves it. Observe that the fairness condition subsumes transition-based weak fairness.

Remark 2 Our notion of fairness is based on configurations, and does not coincide with transition-based weak fairness (if a transition is enabled at infinitely many configurations, then it is also taken infinitely often). To illustrate the difference, consider the protocol scheme with states $\{q_1, q_2, r_1, r_2, s\}$ and transitions

$$\delta_{qr} = (q_1, r_1) \mapsto (q_1, r_1) \quad \begin{array}{ll} \delta_{q_1} = (q_1, s) \mapsto (q_2, s) & \delta_{r_1} = (r_1, s) \mapsto (r_2, s) \\ \delta_{q_2} = (q_2, s) \mapsto (q_1, s) & \delta_{r_2} = (r_2, s) \mapsto (r_1, s) \ . \end{array}$$

Consider the configuration that puts one agent in each of q_1, r_2, and s, that is $\mathbf{q}_1 + \mathbf{r}_2 + \mathbf{s}$. From this configuration we can execute the infinite sequence of transitions $w = (\delta_{q_1}\delta_{r_2}\delta_{r_1}\delta_{q_2})^{\omega}$. It is easy to see that during the execution of w the transition δ_{qr} is never enabled, and so the execution is weakly fair in the transition-based sense. However, it is not fair in the configuration-based sense. Indeed, while $\mathbf{q}_1 + \mathbf{r}_2 + \mathbf{s}$ is visited infinitely often, only δ_{q_1} is executed from it, even though it also enables δ_{r_2}. □

The following Lemma 3 formalizes a fundamental property of fair executions: they eventually reach a bottom SCC of the configuration graph, and then visit each of its states infinitely often (actually, if the execution is finite, then the bottom SCC consists of just one state without successors; intuitively, the execution reaches this state, and stays there "forever").

Lemma 3 *For every fair execution $C_0, C_1, \ldots$ there is an index $i \geq 0$ such that C_i is a bottom configuration, and the set $\{C_j \mid j \geq i\}$ is a bottom SCC of the configuration graph.*

Proof If the execution is finite, then, since it cannot be extended, its last configuration is a bottom SCC with one single node and no outgoing transitions. If the execution is infinite, then the fairness condition forces it to eventually leave every non-bottom SCC it enters. So there is an index $i \geq 0$ such that C_j is a bottom configuration for every $j \geq i$, and so $\{C_j \mid j \geq i\}$ is included in a bottom SCC $\mathcal{S}$. Now let C be an arbitrary configuration of $\mathcal{S}$. By Lemma 1 the set $\mathcal{S}$ is finite, and so there is a number k such that for every $j \geq i$, the configuration C

is reachable from C_j in at most k steps. A simple induction on k shows that, by fairness, C is contained in the execution. So $\mathcal{S} = \{C_j \mid j \geq i\}$. □

Example 4 (Debating philosophers) It is easy to see that the infinite sequence of steps shown in Example 2 is a fair execution. Many other executions are not fair (for example, the infinite execution $(2, 1, 0, 0)^\omega$ where no two philosophers with diverging opinions get to debate). A less trivial example is $(3, 3, 0, 0) \left((2, 2, 2, 0)\ (2, 2, 1, 1) \right)^\omega$. □

3.4 Population protocols

We define what it means for a protocol scheme to compute a predicate $\Pi\colon \mathrm{Pop}(\Sigma) \to \{0, 1\}$, where Σ is a non-empty, finite set of *input variables*. Before presenting formal definitions, we give some intuition.

The first step is to add to a protocol scheme an *input mapping* $I\colon \mathrm{Pop}(\Sigma) \to \mathrm{Pop}(Q)$ and an *output mapping* $O\colon \mathrm{Pop}(Q) \to \{0, \bot, 1\}$. The input mapping assigns to an input $X \in \mathrm{Pop}(\Sigma)$ an initial configuration $I(X)$ of the protocol scheme, and the output mapping assigns to a configuration C an output, which can be either 0, 1, or $\bot$. Here $\bot$ stands for "undefined" or "no output".

Intuitively, imagine that an operator is in charge of computing a boolean for each input $X \in \mathrm{Pop}(\Sigma)$ with the help of a machine implementing the protocol scheme. Upon receiving X, the operator first applies the input mapping to it, obtains the configuration $C = I(X)$, allocates $C(q)$ agents to each state q of the scheme/machine, and runs it from this initial configuration, letting it produce a fair execution. The machine has two lamps for the outputs 1 and 0. The b-lamp is switched on whenever the current configuration C satisfies $O(C) = b$, and switched off otherwise. By definition, the execution of the machine outputs $b \in \{0, 1\}$ if it eventually stabilizes to b, meaning that from some moment on the b-lamp stays on forever (that is, from some moment on the execution only visits configurations C such that $O(C) = b$).

For a given input X some fair execution starting at $C(X)$ may not stabilize to 0 or 1. Or two different fair executions starting at $C(X)$ may stabilize to 0 and 1, respectively. Then we say that the scheme is *ill specified*. More precisely: If there is at least one input for which at least one fair execution does not stabilize to 0 or 1, or for which two fair executions stabilize to 0 and to 1, respectively, then the scheme is ill specified, and "does not compute any predicate".

If a scheme is well specified, then for every input X all fair computations from $I(X)$ stabilize to the same boolean b_X, and we define the predicate computed by the protocol as the mapping Π given by $\Pi(X) = b_X$.

Example 5 (Debating philosophers) We define input and output mappings for the debating philosophers. For the set of inputs we choose $\Sigma = \{\mathtt{F}, \mathtt{A}\}$ (For and Against). So a population over Σ models a population of philosophers, specifying how many are for and against animal rights. We represent a population with f philosophers for and a philosophers against animal rights by the pair (f, a).

As input mapping we choose the function $I\colon \mathrm{Pop}(\Sigma) \to \mathrm{Pop}(Q)$ given by

$$\begin{array}{cccccc} & \mathtt{RF} & \mathtt{RA} & \mathtt{TF} & \mathtt{TA} & \\ I(f, a) = (& f & a & 0 & 0 &) \end{array}$$

In other words, the mapping assigns to (f, a) a population with f rested philosophers supporting animal rights, a rested philosophers against animal rights, and no tired philosophers.

As output mapping we choose the function $O\colon \text{Pop}(Q) \to \{0, \bot, 1\}$ given by

$$O(rf, ra, tf, ta) = \begin{cases} 1 & \text{if } ra + ta = 0 \text{ (all philosophers are for animal rights)} \\ 0 & \text{if } rf + tf = 0 \text{ (all philosophers are against animal rights)} \\ \bot & \text{otherwise} \end{cases}$$

□

After this informal introduction, we now present some formal definitions.

Input and output mappings Formally, an *input mapping* of a protocol scheme $\mathcal{A} = (Q, \Delta)$ is a function $I\colon \text{Pop}(\Sigma) \to \text{Pop}(Q)$ that maps each input population X to a configuration of $\mathcal{A}$. The set of *initial configurations* is $\mathcal{I} = \{I(X) \mid X \in \text{Pop}(\Sigma)\}$. An *output mapping* of O is a function $O\colon \text{Pop}(Q) \to \{0, \bot, 1\}$ that associates to each configuration C of $\mathcal{A}$ an output value in $\{0, \bot, 1\}$. A configuration C on Q such that $O(C) = b$ for some $b \in \{0, \bot, 1\}$ is called a *b-configuration*.

If input and output mappings can be arbitrary functions, even non computable ones, then any problem involving them is bound to be undecidable. For this reason we introduce "reasonable" classes of input and output mappings.

An input mapping I is *Presburger* if the set of pairs $(X, C) \in \text{Pop}(\Sigma) \times \text{Pop}(Q)$ such that $C = I(X)$ is definable in Presburger arithmetic. An output mapping O is *Presburger* if the same holds for the set of pairs $(C, b) \in \text{Pop}(Q) \times \{0, \bot, 1\}$ such that $O(C) = b$.

A *population protocol* is a triple $(\mathcal{A}, \mathtt{I}, \mathtt{O})$, where $\mathcal{A}$ is a protocol scheme, and $\mathtt{I}(X, C)$ and $\mathtt{O}(C, b)$ are formulas of Presburger arithmetic denoting a Presburger input mapping I and a Presburger output mapping O, respectively.

Presburger input and output mappings are still quite general. Many papers only consider a more restricted class. An input mapping I is *simple* if for every input variable $\sigma \in \Sigma$ there exists a state $q_\sigma \in Q$ such that

$$I(X) = \sum_{\sigma \in \Sigma} X(\sigma)\, \mathbf{q}_\sigma$$

for every input population X on Σ. Intuitively, if I is simple then each input variable is assigned a state, and the operator can prepare the initial configuration for the input X by going through all input variables σ, and putting $X(\sigma)$ agents in the state corresponding to σ.

Similarly, an output mapping O is *simple* if there exists a partition (Q_0, Q_1) of Q such that

$$O(C) = \begin{cases} 0 & \text{if } \text{Sup}(C) \subseteq Q_0 \\ 1 & \text{if } \text{Sup}(C) \subseteq Q_1 \\ \bot & \text{otherwise} \end{cases}$$

for every configuration C. Notice that O is well defined because $\text{Sup}(C) \neq \emptyset$.

Example 6 (Debating philosophers) The input mapping given in Example 5 is Presburger. Indeed, if we represent the input X satisfying $X(\mathtt{F}) = f$ and $X(\mathtt{A}) = a$ by the pair (f, a), and the configuration C satisfying $C(\mathtt{RF}) = rf$, $C(\mathtt{RA}) = ra$, $C(\mathtt{TF}) = tf$, and $C(\mathtt{TA}) = ta$ by the vector (rf, ra, tf, ta), then the set of pairs (X, C) such that $I(X) = C$ is defined by the Presburger formula

$$\mathtt{I}(f, a, rf, ra, tf, ta) := (rf = f \wedge ra = a \wedge tf = 0 \wedge ta = 0)$$

The output mapping is also Presburger. Moreover, both mappings are simple. Indeed, for I we have

$$I(f, a) = f\,\mathtt{RF} + a\,\mathtt{RA}$$

For the output mapping simply consider $Q_0 = \{\mathtt{RA}, \mathtt{TA}\}$ and $Q_0 = \{\mathtt{RF}, \mathtt{TF}\}$. $\square$

Remark 4 A particular case of protocols with Presburger input and output mappings are population protocols with leader [4,6]. In these protocols the initial configuration contains one agent, called the *leader*, occupying a distinguished initial state q_l not initially occupied by any other agent. This corresponds to the input mapping $I(X) = \mathbf{q}_l + \sum_{\sigma \in \Sigma} X(\sigma)\,\mathbf{q}_\sigma$ which is obviously Presburger. $\square$

Stabilization and well-specified protocols An execution $C_0, C_1, \ldots$ *stabilizes to* b for a given $b \in \{0, \perp, 1\}$ if there exists $n \in \mathbb{N}$ such that $O(C_m) = b$ for every $m \geq n$ (if the execution is finite, then this means for every m between n and the length of the execution). Notice that there may be many different executions from a given configuration C_0, each of which may stabilize to 0, 1, or $\perp$, or not stabilize at all.

A population protocol $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ is *well specified* if for every input population $X \in \mathrm{Pop}(\Sigma)$, every fair execution of $\mathcal{A}$ starting at $I(X)$ stabilizes to the same value $b \in \{0, 1\}$. Otherwise, the protocol is *ill specified*. Finally, population protocol *computes* a predicate Π if for every $X \in \mathrm{Pop}(\Sigma)$, every fair execution of $\mathcal{A}$ starting at $I(X)$ stabilizes to $\Pi(X)$. It follows easily from the definitions that a protocol computes a predicate iff it is well specified.

Example 7 (Debating philosophers) Let us show that the population protocol $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ of the debating philosophers is well specified. By Lemma 3, every fair execution eventually gets trapped in bottom configurations, that is, in configurations of $\mathcal{B} = \mathcal{B}_{\mathtt{F}} \cup \mathcal{B}_{\mathtt{A}}$, where $\mathcal{B}_{\mathtt{F}}$ and $\mathcal{B}_{\mathtt{A}}$ were computed in Example 3:

$$\mathcal{B}_{\mathtt{F}} = \{\, (rf, 0, tf, 0) \mid rf + tf > 0 \,\} \quad \text{and} \quad \mathcal{B}_{\mathtt{A}} = \{\, (0, ra, 0, ta) \mid ra + ta > 0 \,\}$$

Since in the configurations of $\mathcal{B}$ all philosophers have the same opinion, it is not possible to move from $\mathcal{B}_{\mathtt{F}}$ to $\mathcal{B}_{\mathtt{A}}$, or vice versa. So a fair execution gets trapped either in $\mathcal{B}_{\mathtt{F}}$ or in $\mathcal{B}_{\mathtt{A}}$, and therefore every fair execution stabilizes to 0 or 1.

It remains to show that for every fixed initial configuration $C_0 = (rf_0, ra_0, 0, 0)$, either all fair executions starting at C_0 get trapped in $\mathcal{B}_{\mathtt{F}}$, or they all get trapped in $\mathcal{B}_{\mathtt{A}}$. We prove that they get trapped in $\mathcal{B}_{\mathtt{A}}$ if $rf_0 \geq ra_0$, and in $\mathcal{B}_{\mathtt{F}}$ otherwise.

Let $\mathcal{C}_1 = \{(tf, ra, rf, ta) \mid rf < ra\}$. By direct inspection of the transitions, if $C \in \mathcal{C}_1$ and $C \to C'$, then $C' \in \mathcal{C}_1$. Therefore, if $rf_0 \geq ra_0$ then a fair execution starting at C_0 gets trapped in configurations of $\mathcal{B} \cap \mathcal{C}_1$, and so only in configurations of $\mathcal{B}_{\mathtt{A}}$.

Let $\mathcal{C}_2 = \{(tf, ra, rf, ta) \mid tf \geq ra \wedge tf + rf > 0\}$ By direct inspection of the transitions, if $C \in \mathcal{C}_2$ and $C \to C'$, then $C' \in \mathcal{C}_2$. (For the transition $(\mathtt{R}\alpha, \mathtt{T}\beta) \mapsto (\mathtt{R}\alpha, \mathtt{T}\alpha)$, observe that if the transition is enabled then $rf > 0$.) Assume $C_0 = (tf_0, ra_0, 0, 0)$ satisfies $tf_0 \geq ra_0$. Since configurations contain at least one agent, we have $tf_0 > 0$ and so $C_0 \in \mathcal{C}_2$. Therefore, a fair execution starting at C_0 gets trapped in configurations of $\mathcal{B} \cap \mathcal{C}_2$, and so only in configurations of $\mathcal{B}_{\mathtt{F}}$.

So the protocol of the debating philosophers is well specified, hence it computes a predicate $\Pi\colon \mathrm{Pop}(\{\mathtt{F}, \mathtt{A}\}) \to \{0, 1\}$. This predicate is just the *majority* predicate: $\Pi(f, a) = 1$ iff $f \geq a$. $\square$

3.5 Verification problems

Angluin et al. [2,3] showed that well-specified population protocols can compute all Presburger predicates. Later, Angluin, Aspnes and Eisenstat [5,7] proved by means of an involved argument that they can only compute Presburger predicates. However, a protocol can be ill specified, or be well specified but compute a predicate different from the one intended. Finally, given a protocol we would like to obtain a Presburger formula for the predicate it computes. So we study the following three problems.

- The *well-specification problem*: given a population protocol $(\mathcal{A}, \mathtt{I}, \mathtt{O})$, is it well specified?
- The *fitting problem*: given a population protocol $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ and a Presburger predicate Π, does $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ compute Π?
- The *tailor problem*: given a well-specified population protocol $(\mathcal{A}, \mathtt{I}, \mathtt{O})$, compute (in the standard sense) a Presburger formula for the predicate computed (in the population protocol sense) by $(\mathcal{A}, \mathtt{I}, \mathtt{O})$.

Note that the fitting problem does not assume $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ to be well specified. Consequently, if $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ does not compute Π then either the population protocol is ill specified, or it stabilizes to $b \in \{0, 1\}$ for some input $X \in \text{Pop}(\Sigma)$ such that $\Pi(X) = 1 - b$.

In the rest of the paper we obtain the following results:

- The well-specification and fitting problems are Turing-reducible in to the reachability problem for Petri nets.
  In other words, we show that both problems can be solved with the help of an oracle for the reachability problem for Petri nets. In particular, this proves that both problems are decidable.
- The reachability problem for Petri nets can be reduced in polynomial time to the (complements of the) well-specification or the fitting problems.
- There is an algorithm for the tailor problem.
  This algorithm can also be used to solve the well-specification and fitting problems. However, it consists of two semi-decision algorithms, and currently we do not know of any reduction to the reachability problem. As a corollary of this algorithm we obtain an alternative proof to the result of Angluin et al. [5,7].

4 Population protocols as Petri nets

The computation of a population protocol can be simulated by an associated Petri net. This allows us to apply results on Petri nets to population protocols.

A Petri net $N = (P, T, F)$ consists of a finite set P of *places*, a finite set T of *transitions*, and a *flow function* $F\colon (P \times T) \cup (T \times P) \to \mathbb{N}$. The *preset* of a transition t is the multiset ${}^\bullet t$ of places given by ${}^\bullet t(p) = F(p, t)$ and its *postset* the multiset ${}^\bullet t$ given by $t^\bullet(p) = F(t, p)$. A *marking* $M \in \mathbb{N}^P$ is a multiset on the set P of places and we say that M puts $M(p)$ *tokens* in place p. A transition $t \in T$ is *enabled at* marking M, written $M\,[t\rangle$, if ${}^\bullet t \leq M$. A transition t that is enabled at M can *fire*, yielding the marking $M' = M - {}^\bullet t + t^\bullet$. We denote this fact as $M\,[t\rangle\,M'$. We extend enabledness and firing inductively to words of transitions as follows. Let $w = t_1 \ldots t_k$ be a finite word of transitions $t_j \in T$. We write $M\,[w\rangle\,M'$ if there exists a sequence $M_0, \ldots, M_k$ of markings such that $M = M_0\,[t_1\rangle\,M_1 \cdots [t_k\rangle\,M_k = M'$, and say that M' is *reachable from* M.

Given a Petri net $N = (P, T, F)$, a set $\mathcal{M}$ of markings, and a language $W \subseteq T^*$, we introduce the sets:

$$post_N(\mathcal{M}, W) = \{M' \in \mathbb{N}^P \mid \exists M \in \mathcal{M}\, \exists w \in W\colon M\,[w\rangle\, M'\}$$
$$pre_N(\mathcal{M}, W) = \{M \in \mathbb{N}^P \mid \exists M' \in \mathcal{M}\, \exists w \in W\colon M\,[w\rangle\, M'\}\ .$$

When $W = T^*$ these sets are denoted by $post^*_N(\mathcal{M})$ and $pre^*_N(\mathcal{M})$, respectively.

The study of the complexity of problems on Petri nets requires we define the size of the input. It is not necessary to define these sizes in details since they are quite standard. It suffices to know that numbers are encoded in binary.

The *reachability problem* for Petri nets asks, given a Petri net N and two markings M, M' of N, whether M' is reachable from M, or equivalenty whether $M' \in post_N(\{M\})$. The problem is known to be decidable [30], with a cubic Ackermanian complexity upper bound [27], and EXPSPACE-hard [29]. It is open whether the problem has an algorithm that runs in elementary time, i.e., in k-EXPTIME for some number k independent of the input.

Given two sets $\mathcal{M}, \mathcal{M}'$ of markings, we say that $\mathcal{M}'$ is reachable from $\mathcal{M}$, $\mathcal{M}'$ if there are $M \in \mathcal{M}$ and $M' \in \mathcal{M}'$ such that M' is reachable from M. The reachability problem for Presburger-definable sets of markings is also decidable:

Theorem 5 *Let N be a Petri net, and let ϕ, ϕ' be two Presburger formulas denoting sets $\mathcal{M}, \mathcal{M}'$ of markings of N. The problem whether $\mathcal{M}'$ is reachable from $\mathcal{M}$ can be reduced to the reachability problem for Petri nets, and is thus decidable.*

Proof Since similar reductions are well known (see e.g. [20]), we only sketch the argument. Let d be the number of places of N. Markings of N can then be represented as vectors of $\mathbb{N}^d$. Since $\mathcal{M}$ and $\mathcal{M}'$ are Presburger definable, they are semi-linear [19], and we can compute in triple exponential time in N semi-linear representations for $\mathcal{M}$ and $\mathcal{M}'$.

Let $\{(r_1; P_1), \ldots, (r_n; P_n)\}$ and $\{(r'_1, P'_1), \ldots, (r'_m, P'_m)\}$ be semi-linear representations of $\mathcal{M}$ and $\mathcal{M}'$. We sketch the behavior of a Petri net $\widehat{N}$ with an initial marking $\widehat{M}$ that, loosely speaking, nondeterministically generates an initial marking M_0 of N, simulates N on this marking, nondeterministically stops the simulation at some point in time, and nondeterministically checks if the marking M reached by N when the simulation is stopped belongs to $\mathcal{M}'$.

The marking M_0 is generated as follows. Initially $\widehat{N}$ nondeterministically fires a transition from a set $\{t_1, \ldots, t_n\}$, containing a transition for each linear set in the representation of $\mathcal{M}$. After firing, say, transition t_i, the net proceeds to nondeterministically generate a marking of (r_i, P_i) where, say, $P_i = \{p_{i1}, \ldots, p_{ik}\}$. For this it first fires a transition that puts r_i tokens in the places of N, and then it proceeds to repeatedly fire transitions $t_{i1}, \ldots, t_{ik}$ such that the firing of t_{ij} adds p_{ij} tokens to the places of N. The net can stop these firings at any time by nondeterministically choosing to fire a transition *start*, after which it starts simulating N.

The simulation is stopped nondeterministically by firing a transition *stop*. Let M be the marking of N after the simulation stops. The net nondeterministically guesses that M belongs to the linear set (r'_j, P'_j) of the representation of $\mathcal{M}'$ by firing a transition t'_i for some $1 \leq i \leq m$. Assume $P'_i = \{p'_{i1}, \ldots, p'_{ik'}\}$. The net proceeds to nondeterministically check the guess by first firing a transition that removes r'_i tokens from the places of N, and then repeatedly firing transitions $t'_{i1}, \ldots, t'_{ik'}$, where the firing of t'_{il} removes p'_{ij} tokens from the places of N. If the guess is correct, i.e., if M belongs to the linear set (r'_j, P'_j), then the net can reach the empty marking; otherwise, the nondeterministic check gets stuck at some marking different from the empty marking. Therefore, the empty marking can be reached from $\widehat{M}$ iff some marking of $\mathcal{M}'$ is reachable from some marking of $\mathcal{M}$. □

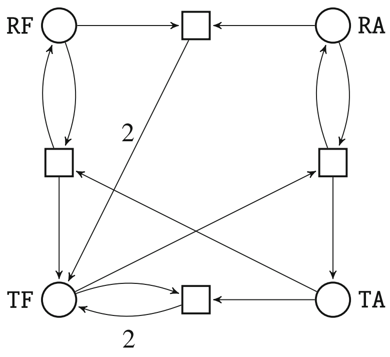

Fig. 1 Petri net for the protocol scheme of the debating philosophers. Transitions t such that ${}^\bullet t = t^\bullet$ are not shown

Given a protocol scheme $\mathcal{A} = (Q, \Delta)$, we define the Petri net $N(\mathcal{A}) = (Q, \Delta, F)$, whose places and transitions are the states and transitions of the protocol, respectively, and ${}^\bullet\delta = \{q_1, q_2\}$, $\delta^\bullet = \{q_1', q_2'\}$ for every $\delta = (q_1, q_2) \mapsto (q_1', q_2')$ in Δ. Note that a configuration of the protocol scheme $\mathcal{A}$ is a marking of the Petri net $N(\mathcal{A})$. Further, whenever $C \xrightarrow{\delta} C'$ for configurations C and C', we have $C\,[\delta\rangle\, C'$ in the Petri net, and vice versa. Figure 1 shows the Petri net for the protocol scheme of the debating philosophers. Transitions t such that ${}^\bullet t = t^\bullet$ (whose firing does not change the current marking) have been omitted.

5 The well-specification problem is decidable

We first characterize the ill specified population protocols in terms of the bottom configurations of their configuration graphs.

Definition 6 Let $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ be a population protocol, and let $\mathcal{B}$ be the set of bottom configurations of its configuration graph. We define $\mathcal{B}_0$ as the set of configurations $C \in \mathcal{B}$ such that for every configuration C' in the same SCC as C the equality $O(C) = 0$ holds[2]. The set $\mathcal{B}_1$ is defined analogously.

Lemma 7 *A population protocol* $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ *is ill specified iff*

(1) $\mathcal{B} \setminus (\mathcal{B}_0 \cup \mathcal{B}_1)$ *is reachable from* $\mathcal{I}$*, or*
(2) $\mathcal{I}$ *contains a configuration* $C \in \mathcal{I}$ *such that both* $\mathcal{B}_0$ *and* $\mathcal{B}_1$ *are reachable from* C.

Proof By definition, a protocol is ill specified iff

(a) some fair execution starting at a configuration of $\mathcal{I}$ does not stabilize to either 0 or 1; or
(b) two fair executions starting at the same configuration of $\mathcal{I}$ stabilize to 0 and 1, respectively.

We prove that (a) holds iff (1) holds, and (b) holds iff (2) holds.

(a) $\Leftrightarrow$ (1). By Lemma 3, a fair execution eventually gets trapped in a bottom SCC $\mathcal{S}$ of the configuration graph, and visits infinitely often every configuration of $\mathcal{S}$. Therefore, the execution does not stabilize to either 0 or 1 iff either $O(C) = \bot$ for some $C \in \mathcal{S}$, or $\mathcal{S}$ contains two configurations C_1, C_2 such that $O(C_1) \neq O(C_2)$. In both cases we have $\mathcal{S} \cap (\mathcal{B}_0 \cup \mathcal{B}_1) = \emptyset$, and so $\mathcal{S} \subseteq \mathcal{B} \setminus (\mathcal{B}_0 \cup \mathcal{B}_1)$.

[2] Observe that we do *not* define $\mathcal{B}_0$ as the set of configurations $C \in \mathcal{B}$ such that $O(C) = 0$.

(b) $\Leftrightarrow$ (2). By Lemma 3, two executions that stabilize to 0 and 1 get trapped in two bottom SCCs $\mathcal{S}_0$ and $\mathcal{S}_1$, and visit all configurations of these SCCs infinitely often. So, we have $O(C_0) = 0$ for every $C_0 \in \mathcal{S}_0$, and $O(C_1) = 1$ for every $C_1 \in \mathcal{S}_1$. It follows $\mathcal{S}_0 \subseteq \mathcal{B}_0$ and $\mathcal{S}_1 \subseteq \mathcal{B}_1$, and so both $\mathcal{B}_0$ and $\mathcal{B}_1$ are reachable from some $C \in \mathcal{I}$. □

This lemma reduces the ill specification problem to reachability questions for the sets $\mathcal{I}, \mathcal{B}, \mathcal{B}_0$, and $\mathcal{B}_1$. We use some results of Petri net theory to prove that all these sets are effectively Presburger, which allows us to apply Theorem 5.

The *reachability relation* of a Petri net N is the binary relation over the markings of N containing the pairs (M, M') such that M' is reachable from M. Similarly, the *mutual reachability relation* of N is the binary relation containing the pairs (M, M') such that M' is reachable from M and M is reachable from M' (equivalently, the pairs (M, M') such that M and M' belong to the same SCC of the reachability graph). It is easy to see that the reachability and mutual reachability relations are closed under addition: if (M_1, M_1') and (M_2, M_2') belong to the relation, then so does $(M_1 + M_2, M_1' + M_2')$. Further, and contrary to the reachability relation, the mutual reachability relation is an equivalence relation. A result of Eilenberg and Schützenberger about rational sets in commutative monoids [14] shows that every equivalence relation closed under sum is Presburger-definable, and so the mutual reachability relation of any Petri net (but not the reachability relation!) is Presburger-definable (Hirshfeld [21] gave a short proof). However, the proofs of this result are non-constructive. We show how to overcome this problem using results of Leroux [23,26].

Definition 8 ([23,26]) A Petri net N is *globally cyclic* if for every two markings M, M', the marking M' is reachable from M iff M and M' are mutually reachable. (In other words: N is globally cyclic if its reachability and mutual reachability relations coincide.)

Leroux and Sutre [28] studied globally flat counter machines, and showed that their reachability relation is effectively semilinear (Theorem 4.3 and Corollary 4.4). Further, they show that globally cyclic Petri nets (seen as a class of counter machines with one single state and one counter for each place) are globally flat (Proposition 5.4). Since a relation is semilinear iff it is Presburger-definable [19], we obtain:

Theorem 9 ([28]) *The mutual reachability relation of a globally cyclic Petri net is effectively Presburger-definable.*

It remains to extend this result to arbitrary Petri nets. For this we show that every Petri net can be effectively transformed into a globally cyclic Petri net with the same mutual reachability relation. The proof is based on the following notion:

Definition 10 ([23,26]) Let t be a transition of a Petri net N. The *domain of reversibility* of t, denoted D_t, is the set of markings M such that there exists a firing sequence σ satisfying $M\,[t\rangle\, M'\,[\sigma\rangle\, M$.

Since $M\,[t\rangle\, M'\,[\sigma\rangle\, M$ implies $M + L\,[t\rangle\, M' + L\,[\sigma\rangle\, M + L$ for every marking L, the domain of reversibility of a transition is an upward-closed set of markings. By Dickson's lemma, a domain of reversibility D_t has a finite set of minimal elements $\min(D_t)$. We now resort to the following result of Leroux [26]:

Theorem 11 ([26], Theorem 10.1) *Let N be a Petri net of size n, and let t be a transition of N. Every marking of* $\min(D_t)$ *has size at most* $2^{2^{O(n)}}$.

Closely following an idea of Bouziane and Finkel [9], we now can prove:

Theorem 12 *There is an algorithm that, given a Petri net N, constructs a globally cyclic Petri net N' having the same mutual reachability relation as N.*

Proof We can easily extract from Theorem 11 an algorithm that constructs the set $\min(D_t)$ for every transition t: Enumerate all markings M of size $2^{2^{O(n)}}$ that enable t, compute for each of them the marking M' such that $M\,[t\rangle\,M'$, and check that M and M' are mutually reachable using the algorithm introduced in [23]. We now use the sets $\min(D_t)$ to construct the globally cyclic net N'.

The sets of places of N' and N coincide. For every transition t of N and for every marking $L \in \min(D_t)$, we add to N' a transition t_L satisfying ${}^\bullet(t_L) = L$ and $(t_L)^\bullet = L'$, where L' is the marking such that $L\,[t\rangle\,L'$.

We show that N and N' have the same mutual reachability relation. Assume that M and M' are mutually reachable in N'. Then there is a firing sequence $M_1 \left[t^{(1)}_{L_1}\right\rangle M_2 \cdots M_n \left[t^{(n)}_{L_n}\right\rangle M_{n+1}$ in N' such that $M_1 = M_{n+1} = M$ and $M_i = M'$ for some $1 \le i \le n$. By the observation above, we have $M_1 \left[t^{(1)}\right\rangle M_2 \cdots M_n \left[t^{(n)}\right\rangle M_{n+1}$ in N, and so M and M' are mutually reachable.

Now, let M, M' be two mutually reachable markings of N. Then there is a firing sequence $M_1 \left[t^{(1)}\right\rangle M_2 \cdots M_n \left[t^{(n)}\right\rangle M_{n+1}$ such that $M_1 = M_{n+1} = M$ and $M_i = M'$ for some $1 \le i \le n$. Then M_i and M_{i+1} are mutually reachable for every $1 \le i \le n$, and so $M_i \in D_{t^{(i)}}$ for every $1 \le i \le n$. It follows that $M_1 \left[t^{(1)}_{L_1}\right\rangle M_2 \cdots M_n \left[t^{(n)}_{L_n}\right\rangle M_{n+1}$ is a firing sequence of N' for some markings $L_1, \ldots, L_n$ such that $L_i \le M_i$ and $L_i \in \min(D_{t_i})$ for every $1 \le i \le n$. So M and M' are also mutually reachable in N'.

Finally, we show that N' is globally cyclic by considering two markings M and M' such that $M\,[t_L\rangle\,M'$ for some transition t of N and some marking $L \in \min(D_t)$. From $M\,[t_L\rangle\,M'$ we derive $M \ge L$. Hence $M \in D_t$. It follows that M is in the domain of reversibility of t. Thus M and M' are mutually reachable in N. Since N and N' have the same mutal reachability relation, we derive that M and M' are mutually reachable in N'. Thus N' is globally cyclic. □

Finally, combining Theorems 9 and 12 we obtain:

Theorem 13 *The mutual reachability relation of a Petri net N is effectively Presburger-definable.*

Using this theorem, we can easily derive an algorithm to construct Presburger formulas for $\mathcal{B}$, $\mathcal{B}_0$, and $\mathcal{B}_1$.

Proposition 14 *There is an algorithm that takes as input a protocol scheme and returns Presburger formulas denoting the sets $\mathcal{B}$, $\mathcal{B}_0$, and $\mathcal{B}_1$.*

Proof We show that the predicate $\mathtt{B}(C)$ associated to the set of bottom configurations is definable in Presburger arithmetic. Let us introduce the predicate $\mathtt{MR}(C, C')$ associated to the mutual reachability relation. Theorem 13 shows that $\mathtt{MR}(C, C')$ is effectively Presburger. Now, we just observe that C is a bottom configuration iff for every configuration C' such that C and C' are mutually reachable and for every C'' such that $C' \to C''$, we have C and C'' are also mutually reachable:

$$\mathtt{B}(C) = \forall C'\, \forall C'' \colon (\mathtt{MR}(C, C') \wedge C' \to C'') \Rightarrow \mathtt{MR}(C, C'') \ .$$

We claim that $\mathcal{B}_b$ is a Presburger set of configurations. To prove this, we just notice that $\mathcal{B}_b$ is denoted by the following formula:

$$\mathtt{B}_b(C) = \mathtt{B}(C) \wedge \forall C' \colon \mathtt{MR}(C, C') \Rightarrow \mathtt{O}(C', b) \ .$$

□

Together with Theorem 5, Proposition 14 shows that we decide reachability questions between $\mathcal{I}$ (which is a Presburger set by definition), and the sets of bottom configurations. The next theorem reduces conditions (1) and (2) of Lemma 7 to such questions.

Theorem 15 *The ill specification problem is Turing-reducible to the reachability problem for Petri nets, and thus decidable.*

Proof Given a population protocol $(\mathcal{A}, \mathtt{I}, \mathtt{O})$, we show that conditions (1) and (2) of Lemma 7 are reducible to the reachability problem for Petri nets. To check condition (1) we proceed as follows:

– Using Proposition 14, compute a Presburger formula $\phi_\perp$ denoting the set $\mathcal{B} \setminus (\mathcal{B}_0 \cup \mathcal{B}_1)$.
– Apply Theorem 5 to the net $N(\mathcal{A})$ and the formulas $\mathtt{I}$ and $\phi_\perp$.

Checking condition (2) requires some more work. Consider the net $(N(\mathcal{A}) \parallel N(\mathcal{A}))$ obtained by putting two disjoint copies of $N(\mathcal{A})$ side by side. (Formally, if $N(\mathcal{A}) = (P, T, F)$, then we take a net (P', T', F') isomorphic to (P, T, F) and satisfying $(P' \cup T') \cap (P \cup T) = \emptyset$, and let $(N(\mathcal{A}) \parallel N(\mathcal{A})) = (P \cup P', T \cup T', F \cup F')$.) We denote a marking of $(N(\mathcal{A}) \parallel N(\mathcal{A}))$ as (M, M'), meaning that its projections onto P and P' are M and M', respectively. It follows easily from this definition that (M, M') is reachable from (M_0, M_0') in $(N(\mathcal{A}) \parallel N(\mathcal{A}))$ iff M is reachable from M_0 and M' is reachable from M_0' in $N(\mathcal{A})$. In particular, since the non-empty markings of $N(\mathcal{A})$ are the configurations of $\mathcal{A}$, condition (2) of Lemma 7 holds iff there are non-empty markings M_I, M_0, M_1 of $N(\mathcal{A})$ such that

– $M_I \in \mathcal{I}$, $M_0 \in \mathcal{B}_0$, $M_1 \in \mathcal{B}_1$, and
– (M_0, M_1) is reachable from (M_I, M_I) in $(N(\mathcal{A}) \parallel N(\mathcal{A}))$.

So to check condition (2) we proceed as follows:

– Construct a Presburger formula ϕ_{II} denoting the set of markings of $(N(\mathcal{A}) \parallel N(\mathcal{A}))$ of the form $\{(M, M) \mid M \in \mathcal{I}\}$.
  This is possible because the set $\mathcal{I}$ is Presburger.
– Construct a Presburger formula ϕ_{01} denoting the set of markings of $(N(\mathcal{A}) \parallel N(\mathcal{A}))$ of the form $\{(M_0, M_1) \mid M_0 \in \mathcal{B}_0, M_1 \in \mathcal{B}_1\}$.
– Apply Theorem 5 to the net $(N(\mathcal{A}) \parallel N(\mathcal{A}))$ and the formulas ϕ_{II} and ϕ_{01}. □

6 The fitting problem is decidable

We show that the fitting problem is Turing-reducible to the reachability problem for Petri nets.

Theorem 16 *The fitting problem is Turing-reducible to the reachability problem for Petri nets, and thus decidable.*

Proof Let $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ be a population protocol and let $\Pi \colon \mathrm{Pop}(\Sigma) \to \{0, 1\}$ be a Presburger predicate. We reduce the fitting problem to the (complement of the) reachability problem for Presburger sets of markings, and apply Theorem 5.

Recall that $\mathtt{I}(X, C)$ is a formula of Presburger arithmetic that holds iff $I(X) = C$, that is, if $C \in \mathrm{Pop}(Q)$ is the initial configuration for the input $X \in \mathrm{Pop}(\Sigma)$. We define the formulas

$$\mathtt{I}_1(C) = \exists X \colon \mathtt{I}(X, C) \wedge \Pi(X) \qquad \mathtt{I}_0(C) = \exists X \colon \mathtt{I}(X, C) \wedge \neg\Pi(X)$$

and the sets $\mathcal{I}_1, \mathcal{I}_0$ of configurations satisfying $\mathtt{I}_1, \mathtt{I}_0$. So $\mathcal{I}_1$ (resp. $\mathcal{I}_0$) is the set of initial configurations of $\mathcal{A}$ corresponding to the inputs that satisfy Π (resp. do not satisfy Π). Clearly, both sets are Presburger-definable.

Let $\mathcal{B}, \mathcal{B}_0$, and $\mathcal{B}_1$ as in Definition 6. We claim that $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ computes Π iff $\mathcal{B} \setminus \mathcal{B}_0$ is not reachable from $\mathcal{I}_0$, and $\mathcal{B} \setminus \mathcal{B}_1$ is not reachable from $\mathcal{I}_1$. By Lemma 3 and the definition of $\mathcal{B}_0$ and $\mathcal{B}_1$, a fair computation stabilizes to $b \in \{0, 1\}$ iff it gets trapped in a bottom SCC contained in $\mathcal{B}_b$. Therefore, $\mathcal{B} \setminus \mathcal{B}_b$ is not reachable from $\mathcal{I}_b$ iff every fair computation from $\mathcal{I}_b$ stabilizes to b. This proves the claim.

Let $N(\mathcal{A})$ be the Petri net associated to $\mathcal{A}$. By the claim, $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ does not compute Π iff some marking of $\mathcal{B} \setminus \mathcal{B}_0$ is reachable in $N(\mathcal{A})$ from some marking of $\mathcal{I}_0$, or some marking of $\mathcal{B} \setminus \mathcal{B}_1$ is reachable in $N(\mathcal{A})$ from some marking of $\mathcal{I}_1$. Since, by Proposition 14, $\mathcal{B}, \mathcal{B}_0$ and $\mathcal{B}_1$ are effectively computable Presburger sets, so are $\mathcal{B} \setminus \mathcal{B}_0$ and $\mathcal{B} \setminus \mathcal{B}_1$. So the fitting problem reduces to (the complements of) two instances of the reachability problem for Presburger sets. □

7 Lower bounds for the well-specification and fitting problems

We show that the reachability problem for Petri nets can be reduced to the complements of the well-specification and fitting problems.

Theorem 17 *The reachability problem for Petri nets is polynomially reducible to ill-specification problem and to the complement of the fitting problem for population protocols (in both cases even with simple output mappings).*

Proof We proceed by means of a sequence of reductions. First, the reachability problem for Petri nets can be reduced in polynomial time to the *single-place zero-reachability problem* (or SPZRP) [20]:

Given: a Petri net $N = (P, T, F)$, a marking $M_0 \in \mathbb{N}^P$, and a place $z \in P$.
Decide: Is there a marking M reachable from M_0 such that $M(z) = 0$?

Next, we show that SPZRP can be reduced to SPRZP in which N and M_0 satisfy some additional conditions:

(a) $M_0(z) > 0$,
(b) no two transitions of N have the same input and output places (i.e., if ${}^\bullet t_1 = {}^\bullet t_2$ and $t_1{}^\bullet = t_2{}^\bullet$ then $t_1 = t_2$),
(c) the range of the flow function F is $\{0, 1\}$, and
(d) every transition t of N satisfies $1 \leq |{}^\bullet t| \leq 2$ and $1 \leq |t^\bullet| \leq 2$,

Description of the reduction For each of the conditions (a)–(d) we show how to transform an instance N, M_0, and z of SPZRP that does not satisfy the condition into an equivalent instance N', M_0', z' that does. For (a): if $M_0(z) = 0$ then the answer to SPRZP is trivially "yes" (take $M = M_0$), and we can take any positive instance N', M_0', z' satisfying (a). For (b), if N has several transitions with the same input and output places, we can safely remove all but one, without changing the set of reachable markings, and let N' be the result. For (c)

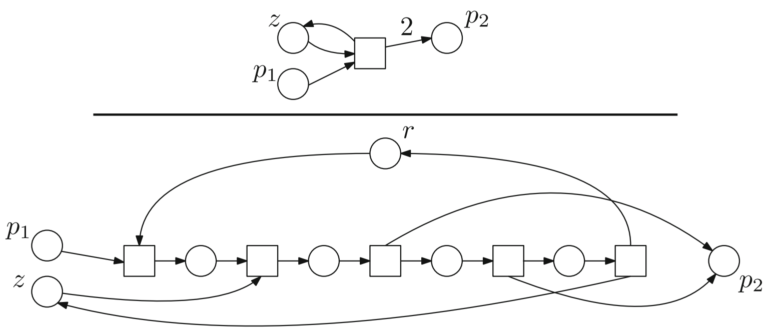

Fig. 2 *Above* the original transition, *below* its gadget

and (d), we choose N' as the result of replacing every transition of N by an adequate "gadget" that simulates the firing of a transition t one place at a time: first the places of ${}^\bullet t$ then those of $t^\bullet$. Figure 2 illustrate the gadget construction (bottom) given the original transition (top); the reader can easily guess the general definition. The special place r is shared by all the gadgets of all transitions. Loosely speaking, r guarantees that at most one gadget is active at a time.

Let $N' = (P', T', F')$ and z' be the result of performing all these transformations. We have $P' = P \cup P_{aux}$, where P_{aux} are the auxiliary places used in the gadgets, hence P_{aux} includes r. Let $M'_0 \in \mathbb{N}^{P'}$ be such that $M'_0 = M_0 + \mathbf{r}$.

The following property is easy to prove. The reachable markings of N and the projections onto P of the reachable markings of N' that put one token in the place r coincide. Loosely speaking, the net N' simulates the firings of transitions of N by executing the corresponding gadget. If N' tries to simulate the firing of a transition of N that is not currently enabled, then the gadget cannot execute and N' reaches a deadlock. The markings of N' with one token in r are those in which every execution of a gadget could be successfully completed.

It follows easily from the previous that N has a reachable marking M such that $M(z) = 0$ iff N' has a reachable marking M' such that $M'(z') = 0$ and $M'(r) = 1$.

Next, we define a population protocol that is ill-specified exactly when N' has a reachable marking with no token in z' and one token in r.

Description of the population protocol Given $N' = (P', T', F')$, M'_0 and z', we construct in polynomial time a population protocol $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ with Presburger input mapping. We first describe the protocol scheme $\mathcal{A} = (Q, \Delta)$. The set Q of states of the protocol contains

- a state q_p for every place $p \in P'$;
- a state q_t for every transition $t \in T'$; and
- two states $Source$ and $Sink$.

The output mapping O, which is simple, is given by the partition Q_0, Q_1 with $Q_1 = \{q_r\}$ and, thus, $Q_0 = Q \setminus \{q_r\}$. The input mapping $I\colon \text{Pop}(\Sigma) \to \text{Pop}(Q)$ is defined as follows. The set Σ is a singleton $\{\sigma\}$, and I assigns to the number n—a population of $\text{Pop}(\{\sigma\})$—the configuration that puts

- n agents in *Source*;
- $M'_0(p)$ agents in q_p for every place p; and
- 0 agents elsewhere.

Observe that I is a Presburger mapping.

We now describe the transitions of the protocol. By condition (b), we can identify a Petri net transition t and the pair $({}^\bullet t, t^\bullet)$, and so we use the notation $t = ({}^\bullet t, t^\bullet)$. Following (e), we define the set Δ of protocol transitions as follows:

(1) for every Petri net transition $t = (\{p_1, p_2\}, \{p_3, p_4\})$, two protocol transitions

$$(q_{p_1}, q_{p_2}) \mapsto (q_t, \mathit{Sink}) \quad \text{and} \quad \delta_t := (q_t, \mathit{Source}) \mapsto (q_{p_3}, q_{p_4})$$

(2) for every Petri net transition $t = (\{p_1, p_2\}, \{p_3\})$, two protocol transitions

$$(q_{p_1}, q_{p_2}) \mapsto (q_t, \mathit{Sink}) \quad \text{and} \quad \delta_t := (q_t, \mathit{Source}) \mapsto (q_{p_3}, \mathit{Sink})$$

(3) for every Petri net transition $t = (\{p_1\}, \{p_2, p_3\})$, one protocol transition

$$\delta_t := (q_{p_1}, \mathit{Source}) \mapsto (q_{p_2}, q_{p_3})$$

(4) for every Petri net transition $t = (\{p_1\}, \{p_2\})$, one protocol transition

$$\delta_t := (q_{p_1}, \mathit{Source}) \mapsto (q_{p_2}, \mathit{Sink})$$

(5) a transition

$$(q_r, q_z) \mapsto (\mathit{Sink}, q_z) \ .$$

Observe that, for future reference, each Petri net transition t has a corresponding transition in the protocol identified by the label δ_t .

The transitions of (1)–(4) simulate the firing of the Petri net transition t (in the case of transitions in (1)–(2), firing t is simulated by the occurrence, one after the other, of two protocol transitions). Observe that the simulation of a transition t of type (1) can "get stuck": after the occurrence of $(q_{p_1}, q_{p_2}) \mapsto (q_t, \mathit{Sink})$ there may be no agent in *Source*, and then $(q_t, \mathit{Source}) \mapsto (q_{p_3}, q_{p_4})$ cannot occur. This is also true for the transitions of type (2).

Intuitively, the transition $(q_r, q_z) \mapsto (\mathit{Sink}, q_z)$ of (5) turns a configuration with undefined output into one with output 0 "as long as there is at least one token in z and no gadget is executing".

In all cases, simulating the firing of t requires one agent to leave the *Source* state. Since, moreover, no agents ever enter *Source*, each execution of $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ contains only finitely many occurrences of transitions of (1)–(4). Further, since the transition of (5) moves an agent to *Sink*, and no agents ever leave *Sink*, the transitions of (5) also occur only finitely often, actually at most once since r never contains two or more tokens. Therefore all fair executions of $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ are finite.

Assume that some reachable marking M of N' satisfies $M(z') = 0$ and $M(r) = 1$. Let $\tau \in T^*$ be such that $M'_0 [\tau\rangle M$, and let k be the length of τ. Since $M'_0(z) > 0$, we have $k > 0$. We claim that $\mathcal{A}$ has a fair (finite) execution from $I(k\,\boldsymbol{\sigma})$ that does not stabilize. Consider the execution that starts by simulating τ through transitions (1)–(4). At the end of this simulation the protocol reaches a configuration C such that $C(\mathit{Source}){=}0{=}C(q_{z'})$, $C(q_r) = 1$ and $C(\mathit{Sink}) > 0$ (this follows from $k > 0$). Since $C(\mathit{Source}) = 0$, none of the $\{\delta_t\}_{t \in T'}$ transitions can occur from C, and so, by exhaustively executing transitions from (1)–(2), we reach a configuration C' that that does not enable any transition of (1)–(4), still satisfies $C'(\mathit{Source}){=}0{=}C'(q_{z'})$, $C'(q_r) = 1$ and $C'(\mathit{Sink}) > 0$. Since $C'(q_{z'}) = 0$, the configuration C' does not enable the transition of (5) either. So the execution cannot be extended, hence it is fair. Because in C' one agent is in state q_r and some other in *Sink* we have $O(C') = \bot$. So $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ is ill specified.

Assume now that every reachable marking M of N' satisfies $M(z') > 0$ or $M(r) = 0$. We prove that every fair execution stabilizes to 0. Since all fair executions of $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ are finite, given a fair execution $C_0 C_1 \dots C_n$ we have to prove $O(C_n) = 0$ or, equivalently, $C_n(q_r) = 0$. A fair execution can either reach a deadlock with no agent in q_r. In this case, the output of the resulting configuration C is defined to be 0 since $C(q_r) = 0$ and $Q_1 = \{q_r\}$, hence $O(C) = 0$. Or, a fair execution reaches a deadlock with some agent in q_r. Note that, by construction, it must be the case that some agent is in $q_{z'}$ following our assumption on M. But then transition (5) is enabled and thus the configuration with some agent in q_r is not a deadlock. Note that firing (5) necessarily results in a configuration whose output is defined to be 0.

The same reduction shows hardness for the complement of the fitting problem for the predicate *false*. □

8 An algorithm for the Tailor problem

We present an algorithm for the tailor problem (given a well-specified population protocol obtain a Presburger formula for the predicate it computes), based on the notion of *certificates*. A certificate of a protocol $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ is a string x satisfying certain conditions, specified in Sect. 8.1. After defining certificates, we prove the following properties:

(1) If a protocol has a certificate, then it is well specified. Moreover, there is an algorithm that, given a protocol and a certificate, returns a Presburger formula for the predicate computed by the protocol.
(2) There is an algorithm that, given a protocol and a string x, decides if x is a certificate of the protocol.
(3) If a protocol is well specified, then it has a certificate.

These properties immediately lead to an algorithm for the tailor problem: enumerate all strings x; check if x is a certificate using property (2); if x is a certificate, compute a formula for the predicate computed by the protocol using property (1). The termination of the algorithm is ensured by property (3).

After defining certificates in Sect. 8.1, properties (1)–(3) are proved in in three different sections. Property (3) has the most involved proof, and requires to introduce some further results from the theory of Petri nets.

8.1 Certificates

We need some definitions and notations. Let $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ be a population protocol.

– A configuration C of $\mathcal{A}$ is a 0-*configuration* (resp. 1-*configuration*) if $O(C) = 0$ (resp. $O(C) = 1$).
– A set $\mathcal{C}$ of configurations of $\mathcal{A}$ is *inductive* if $C \in \mathcal{C}$ and $C \to C'$ implies $C' \in \mathcal{C}$.
– Given a language $W \subseteq \Delta^*$ and a set $\mathcal{C}$ of configurations, $pre_{\mathcal{A}}(\mathcal{C}, W)$ denotes the set of configurations C such that $C \xrightarrow{w} C'$ for some word $w \in W$ and some $C' \in \mathcal{C}$. We write $pre^*_{\mathcal{A}}(\mathcal{C})$ to denote $pre_{\mathcal{A}}(\mathcal{C}, \Delta^*)$. The definitions for $post_{\mathcal{A}}(\mathcal{C}, W)$ and $post^*_{\mathcal{A}}(\mathcal{C})$ are as expected.

Definition 18 Let $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ be a population protocol such that $\mathcal{A} = (Q, \Delta)$. A *certificate* for the population protocol $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ is a tuple $(\mathtt{S}_0, \mathtt{S}_1, \mathtt{D}_0, \mathtt{D}_1, w_1, \dots, w_k)$, where $\mathtt{S}_0, \mathtt{S}_1, \mathtt{D}_0, \mathtt{D}_1$ are predicates in Presburger arithmetic denoting Presburger sets of configurations $\mathcal{S}_0, \mathcal{S}_1, \mathcal{D}_0, \mathcal{D}_1$, and $w_1, \dots, w_k$ are words in Δ^* denoting the language $W = w_1^* \dots w_k^*$, such that:

(1) $\mathcal{S}_0, \mathcal{S}_1, \mathcal{D}_0, \mathcal{D}_1$ are inductive.
(2) The pair $(\mathcal{I}_0, \mathcal{I}_1)$, where $\mathcal{I}_0 = \mathcal{S}_0 \cap \mathcal{I}$ and $\mathcal{I}_1 = \mathcal{S}_1 \cap \mathcal{I}$, is a partition of $\mathcal{I}$.
(3) $\mathcal{D}_0$ is a set of 0-configurations such that $\mathcal{S}_0 \subseteq pre_{\mathcal{A}}(\mathcal{D}_0, W)$.
(4) $\mathcal{D}_1$ is a set of 1-configurations such that $\mathcal{S}_1 \subseteq pre_{\mathcal{A}}(\mathcal{D}_1, W)$.

Observe that, by condition (2), all initial configurations belong to $\mathcal{S}_0 \cup \mathcal{S}_1$. So, by condition (1), $\mathcal{S}_0 \cup \mathcal{S}_1$ contains all configurations reachable from initial configurations. Condition (3) ensures that from every configuration of $\mathcal{S}_0$ one can reach and get trapped in a set of 0-configurations (because $\mathcal{D}_0$ is inductive); condition (4) is a similar property for 1-configurations.

8.2 Certificates ensure well-specification

We show that if a protocol has a certificate, then it is well specified. Moreover, a Presburger formula for the predicate computed by the protocol can be easily extracted from the certificate.

Lemma 19 *If a population protocol* $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ *has a certificate* $(\mathtt{S}_0, \mathtt{S}_1, \mathtt{D}_0, \mathtt{D}_1, w_1, \ldots, w_k)$*, then the protocol is well specified and computes the predicate* $\Pi\colon \mathrm{Pop}(\Sigma) \to \{0, 1\}$ *defined as follows:*

$$\Pi(X) = \begin{cases} 0 & \text{if } \exists C\colon \mathtt{I}(X, C) \wedge \mathtt{S}_0(C) \\ 1 & \text{if } \exists C\colon \mathtt{I}(X, C) \wedge \mathtt{S}_1(C) \end{cases} .$$

In particular, the algorithm that given a protocol and a certificate outputs the formula $\exists C\colon \mathtt{I}(X, C) \wedge \mathtt{S}_0(C)$ *yields a correct solution to the tailor problem.*

Proof Let $\mathcal{S}_0, \mathcal{S}_1, \mathcal{D}_0, \mathcal{D}_1$ be the Presburger sets of configurations denoted by $\mathtt{S}_0, \mathtt{S}_1, \mathtt{D}_0, \mathtt{D}_1$, respectively. Let $W = w_1^* \ldots w_k^*$. Since $\mathcal{I}_0$ and $\mathcal{I}_1$ form a partition of $\mathcal{I}$, it suffices to prove that every fair execution starting at $\mathcal{I}_b$ stabilizes to b. Let $C \in \mathcal{I}_b$ and let $C_0, C_1, \ldots$ be a fair execution starting at C. By Lemma 3 the execution gets trapped in a bottom SCC. Hence, there exists $n \in \mathbb{N}$ such that C_n is a bottom configuration. As $\mathcal{S}_b$ is inductive, it follows that $C_n \in \mathcal{S}_b$. Moreover, as $\mathcal{S}_b \subseteq pre_{\mathcal{A}}(\mathcal{D}_b, W)$, there exists a word $w \in W$ and a configuration $C' \in \mathcal{D}_b$ such that $C_n \xrightarrow{w} C'$. Since C_n is a bottom configuration, there exists a word $w' \in \Delta^*$ such that $C' \xrightarrow{w'} C_n$. Now, let $m \geq n$. Since C_m is reachable from C_n, it follows that C_m is reachable from C'. As $C' \in \mathcal{D}_b$ and $\mathcal{D}_b$ is inductive, it follows that $C_m \in \mathcal{D}_b$. As $\mathcal{D}_b$ is a set of b-configurations, it follows that $O(C_m) = b$; thus, the execution stabilizes to b. □

Example 8 (Certificate for the parity predicate) We describe a population protocol and show with the help of a certificate that it computes a given predicate. In the following $b \in \{0, 1\}$.

Let $\Sigma = \{\sigma\}$. Abusing language, we identify the mapping $X\colon \mathrm{Pop}(\Sigma) \to \mathbb{N}$ given by $X(\sigma) = n$ with the number n. The *parity predicate* $\Pi : \mathrm{Pop}(\Sigma) \to \{0, 1\}$ is given by $\Pi(n) = 0$ if n is even, and $\Pi(n) = 1$ otherwise.

The protocol $(\mathcal{A}, \mathtt{I}, \mathtt{O})$, where $\mathcal{A} = (Q, \Delta)$, is defined as follows:

- $Q = \{A_0, A_1, P_0, P_1\}$. Agents in $\{A_0, A_1\}$ are *active*, and those in $\{P_0, P_1\}$ are *passive*. Further, agents in $\{A_b, P_b\}$ *carry (the value) b*.
- $\Delta = \{\delta_{x,y}, \delta_x \mid x, y \in \{0, 1\}\}$, where

$$\delta_{x,y} = (A_x, A_y) \mapsto (A_{x+y}, P_{x+y}) \quad \text{and} \quad \delta_x = (A_x, P_{1-x}) \mapsto (A_x, P_x) .$$

  Intuitively, in $\delta_{x,y}$ two active agents add their values modulo 2, and one of them becomes passive; in δ_x an active agent changes the value of a passive agent.

- $I(n) = n\mathbf{A}_1$ for every $n \in \mathbb{N}$. That is, to compute the parity of n the protocol starts with n active agents carrying 1 (n agents in state A_1, and no agents elsewhere).
- $O(C) = b$ if $Sup(C) \subseteq \{A_b, P_b\}$, and $O(C) = \perp$ otherwise. That is, a configuration has output $b \in \{0, 1\}$ if currently all agents carry b, otherwise it has output $\perp$.

We provide a certificate of the fact that the protocol computes Π.

- $\mathtt{D}_b(C) := \big(C(A_b) = 1 \wedge C(A_{1-b}) = 0 \wedge C(P_{1-b}) = 0\big)$.
  Notice that the set of configurations $\mathcal{D}_b$ denoted by $\mathtt{D}_b$ is inductive. In fact, since configurations of $\mathcal{D}_b$ only have one active agent, and all their agents carry the same value b, they enable no transitions.
- $\mathtt{S}_0(C)$ and $\mathtt{S}_1(C)$ are Presburger formulas for "$C(A_1)$ is even" and "$C(A_1)$ is odd". Inspection of Δ immediately shows that the sets $\mathcal{S}_0$ and $\mathcal{S}_1$ denoted by $\mathtt{S}_0(C)$ and $\mathtt{S}_1(C)$ are inductive. Notice that $\mathcal{I} \cap \mathcal{S}_0$ and $\mathcal{I} \cap \mathcal{S}_1$ is a partition of $\mathcal{I}$.
- $W = \delta_{1,1}^* \; \delta_{0,0}^* \; \delta_{1,0}^* \; \delta_0^* \; \delta_1^*$.
  W models a strategy to reach $\mathcal{D}_0 \cup \mathcal{D}_1$ from any configuration. First execute the transition $\delta_{1,1}$ as long as possible, until there is at most one active agent carrying a 1. Then execute $\delta_{0,0}$ as long as possible, until there is at most one active agent carrying a 0. Then execute $\delta_{1,0}$ if possible, reaching a configuration with exactly one active agent carrying a value b. Finally, execute δ_0 as long as possible, followed by δ_1 as long as possible, leading to a configuration in which every passive agent also carries the value b. □

8.3 Checking certificates

Using acceleration technics [16,18,25], we show that the problem of checking if a given tuple is a certificate reduces to the problem of checking if a closed formula of Presburger arithmetic is true, and so decidable.

Lemma 20 *Given a protocol* $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ *and a tuple* $(\mathtt{S}_0, \mathtt{S}_1, \mathtt{D}_0, \mathtt{D}_1, w_1, \ldots, w_k)$*, it is decidable whether the tuple is a certificate of the protocol.*

Proof We show that conditions (1)–(4) of Definition 18 can be effectively expressed in Presburger arithmetic. For (1), a set $\mathcal{M}$ of configurations denoted by a predicate $\mathtt{M}(C)$ in Presburger arithmetic is inductive iff the following Presburger formula is valid:

$$\forall C, C' \colon (\mathtt{M}(C) \wedge C \rightarrow C') \Rightarrow \mathtt{M}(C') \; .$$

So the inductiveness of $\mathcal{S}_0, \mathcal{S}_1, \mathcal{D}_0, \mathcal{D}_1$ is expressible. For (2), $(\mathcal{I}_0, \mathcal{I}_1)$ is a partition of $\mathcal{I}$ iff

$$\forall C \colon (\exists X \colon \mathtt{I}(X, C)) \Leftrightarrow \big((\mathtt{I}_0(C) \wedge \neg \mathtt{I}_1(C)) \vee (\neg \mathtt{I}_0(C) \wedge \mathtt{I}_1(C))\big)$$

is valid, where $\mathtt{I}_b(C) = (\exists X \colon \mathtt{I}(X, C)) \wedge \mathtt{S}_b(C)$. For (3-4), $\mathcal{D}_b$ is a set of b-configurations iff

$$\forall C \colon \mathtt{D}_b(C) \Rightarrow \mathtt{O}(C, b)$$

is valid. It remains to express $\mathcal{S}_b \subseteq pre_{\mathcal{A}}(\mathcal{D}_b, W)$. Observe that for every word $w \in \Delta^*$, the relation $\xrightarrow{w^*}$ defined by $C \xrightarrow{w^*} C'$ if $C \xrightarrow{w^n} C'$ for some $n \in \mathbb{N}$ is effectively definable in Presburger arithmetic. A formal proof is given in [25, LemmaIII.3]. So the inclusion holds iff

$$\forall C_0 \colon \big(\mathtt{S}_b(C_0) \Rightarrow \exists C_1, \ldots, C_k \colon C_0 \xrightarrow{w_1^*} C_1 \cdots \xrightarrow{w_k^*} C_k \; \wedge \mathtt{D}_b(C_k)\big)$$

is valid. □

8.4 Every well-specified protocol has a certificate

We prove that every well-specified protocol has a certificate.

Let $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ be a well-specified protocol. Let $\mathcal{I}_0$ and $\mathcal{I}_1$ be the subsets of initial configurations for which the protocol computes 0 and 1, respectively. Since the protocol is well specified, the pair $(\mathcal{I}_0, \mathcal{I}_1)$ is a partition of $\mathcal{I}$.

We choose $\mathtt{D}_0$ and $\mathtt{D}_1$ as Presburger formulas denoting the sets $\mathcal{B}_0$ and $\mathcal{B}_1$ of $\mathcal{A}$, as defined in Definition 6. These formulas exist and can be computed by Proposition 14, which shows that $\mathcal{B}_0$ and $\mathcal{B}_1$ are effectively Presburger. Observe that, with this choice, $\mathcal{D}_b$ is a set of b-configurations. Moreover, since any configuration reachable from a bottom configuration is also a bottom configuration, $\mathcal{D}_b$ is inductive.

Before choosing the sets $\mathcal{S}_0$ and $\mathcal{S}_1$, let us consider the tentative choice $\mathcal{S}'_0 = post^*_{\mathcal{A}}(\mathcal{I}_0)$, and $\mathcal{S}'_1 = post^*_{\mathcal{A}}(\mathcal{I}_1)$. The sets $\mathcal{S}'_0$ and $\mathcal{S}'_1$ are clearly inductive. Moreover, since the protocol is well specified, we have $\mathcal{S}'_0 \cap \mathcal{I} = \mathcal{I}_0$ and $\mathcal{S}'_1 \cap \mathcal{I} = \mathcal{I}_1$. Indeed, since $(\mathcal{I}_0, \mathcal{I}_1)$ is a partition of $\mathcal{I}$, if $\mathcal{S}'_0 \cap \mathcal{I} \supsetneq \mathcal{I}_0$ then $\mathcal{S}'_0 \cap \mathcal{I}_1 \neq \emptyset$, and so there is a configuration with two fair computations stabilizing to 0 and to 1, contradicting the assumption that the protocol is well specified.

However, we still miss two important properties: $\mathcal{S}'_0$ and $\mathcal{S}'_1$ may not be Presburger sets, and there may be no language W satisfying conditions (3) and (4). At this point we get help from the following two results:

Theorem 21 ([5,7]) *If* $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ *is well specified, then* $\mathcal{I}_0$ *and* $\mathcal{I}_1$ *are Presburger sets.*

Theorem 22 ([24, Lemma 9.1]) *Let N be a Petri net, and let $\mathcal{M}$ and $\mathcal{M}'$ be Presburger sets of markings of N such that $post^*_N(\mathcal{M}) \cap \mathcal{M}' = \emptyset$. There exists a Presburger inductive set of markings $\mathcal{S}$ such that $\mathcal{M} \subseteq \mathcal{S}$ and $\mathcal{S} \cap \mathcal{M}' = \emptyset$.*

Applying Theorem 22 to $\mathcal{M} = \mathcal{I}_0$ and $\mathcal{M}' = \mathcal{I}_1$ (which are Presburger by Theorem 21) yields an inductive *and* Presburger set $\mathcal{S}_0 \supseteq \mathcal{S}'_0$ such that $\mathcal{S}_0 \cap \mathcal{I}_1 = \emptyset$, and therefore $\mathcal{S}_0 \cap \mathcal{I}_1 = \mathcal{I}_0$. Similarly, applying the theorem to $\mathcal{M} = \mathcal{I}_1$ and $\mathcal{M}' = \mathcal{I}_0$, we obtain a corresponding set $\mathcal{S}_1$.

The existence of the bounded language W follows directly from another result of net theory:

Theorem 23 ([25, Corollary XI.3]) *For every Petri net $N = (P, T, F)$ and for every Presburger sets of markings $\mathcal{S}$ and $\mathcal{D}$ such that $\mathcal{S} \subseteq pre^*_N(\mathcal{D})$, there exists a sequence $w_1, \ldots, w_k$ of words in T^* such that the bounded language $W \subseteq w_1^* \ldots w_k^*$ satisfies $\mathcal{S} \subseteq pre_N(\mathcal{D}, W)$.*

Applying the theorem to $\mathcal{S}_0$ and $\mathcal{D}_0$ and to $\mathcal{S}_1$ and $\mathcal{D}_1$, we obtain two languages W_0, W_1. It then suffices to take $W = W_0 W_1$ since $W \supseteq W_0 \cup W_1$.

8.5 Well-specified protocols compute Presburger predicates: a new proof

Angluin et al. have shown—a celebrated result—that well-specified population protocols compute exactly the Presburger-definable predicates [5,7]. The proof that every Presburger definable predicate is computed by some protocol profits from the fact that every formula of Presburger arithmetic is equivalent to a quantifier-free formula with divisibility predicates [15]. Using this result, it suffices to exhibit protocols computing some simple predicates, and prove that predicates computed by population protocols are closed under conjunction and disjunction, which is achieved by a rather straightforward product construction. The other direction, showing that population protocols can only compute Presburger predicates, is far more involved. We show that this direction follows from recent results of Petri net theory obtained by one of the authors. In fact, we slightly generalize the results of Angluin et al. [5,7], which hold for simple input and output mappings, to the more general Presburger mappings.

The proof is based on the notion of *almost semi-linear sets* introduced in [24] that extends the class of semi-linear sets. We do not recall the formal definition of almost semi-linear sets but just results and intuitions about those sets. Formal definitions can be found in [24]. Intuitively, almost semi-linear sets are subsets of $\mathbb{N}^d$ that can be precisely over-approximated by semi-linear sets. Formally, the class of almost-semi-linear sets contains all the semi-linear sets and it is equipped with a function that maps any almost semi-linear set $\mathbf{X}$ to a semi-linear set $\text{lin}(\mathbf{X})$ that contains $\mathbf{X}$, and a function dim that maps any almost semi-linear sets $\mathbf{X}$ to a number in $\{-1, \ldots, d\}$ in such a way $\dim(\mathbf{X}) \leq \dim(\mathbf{Y})$ for every $\mathbf{X} \subseteq \mathbf{Y}$ and $\dim(\mathbf{X}) = -1$ implies $\mathbf{X} = \emptyset$ for every $\mathbf{X}$. Moreover, the class of almost semi-linear sets satisfies the two following results:

Lemma 24 ([24, Corollary 8.4]) *For every non-empty almost semi-linear sets* $\mathbf{X}$ *and* $\mathbf{Y}$ *with an empty intersection, we have:*

$$\dim(\text{lin}(\mathbf{X}) \cap \text{lin}(\mathbf{Y})) < \max\{\dim(\mathbf{X}), \dim(\mathbf{Y})\} .$$

Theorem 25 ([24, Corollary 6.3]) *The sets* $post_N^*(\mathcal{X}) \cap \mathcal{Y}$ *and* $pre_N^*(\mathcal{Y}) \cap \mathcal{X}$ *are almost semi-linear for every Petri net* N *and for every semi-linear sets of markings* $\mathcal{X}, \mathcal{Y}$.

Now, let us introduce the notion of *decomposable* sets defined as a subclass of the almost semi-linear sets. A subset $\mathbf{X}$ of $\mathbb{N}^d$ is said to be *decomposable* if $\mathbf{X} \cap \mathbf{S}$ is almost semi-linear for every semi-linear set $\mathbf{S}$. It follows from Theorem 25 that reachability sets of Petri nets are decomposable.

Lemma 26 *Disjoint decomposable sets* $\mathbf{X}, \mathbf{Y}$ *such that* $\mathbf{X} \cup \mathbf{Y}$ *is semi-linear are semi-linear.*

Proof Let us prove by induction on $r \in \mathbb{N}$ that for every semi-linear set $\mathbf{A}$ such that $\dim(\mathbf{A}) < r$ and for every partition $\mathbf{X}, \mathbf{Y}$ of $\mathbf{A}$ into decomposable sets, the sets $\mathbf{X}$ and $\mathbf{Y}$ are semi-linear. The case $r = 0$ is immediate since in this case $\mathbf{A}$ is empty. Assuming that the statement is true for r, let us prove it for $r + 1$. Consider a semi-linear set $\mathbf{A}$ such that $\dim(\mathbf{A}) = r$, and a partition $\mathbf{X}, \mathbf{Y}$ of $\mathbf{A}$ into decomposable sets. In particular $\mathbf{X}$ and $\mathbf{Y}$ are almost semi-linear. If $\mathbf{X}$ or $\mathbf{Y}$ is empty, then $\mathbf{X}$ and $\mathbf{Y}$ are semi-linear since those sets are either $\emptyset$ or $\mathbf{A}$. So, we can assume that $\mathbf{X}$ and $\mathbf{Y}$ are non-empty. We introduce $\mathbf{S} = \text{lin}(\mathbf{X})$ and $\mathbf{T} = \text{lin}(\mathbf{Y})$ and $\mathbf{A}' = \mathbf{S} \cap \mathbf{T}$. Lemma 24 shows that $\dim(\mathbf{A}') < r$. We introduce the decomposable sets $\mathbf{X}'$ and $\mathbf{Y}'$ defined as $\mathbf{X} \cap \mathbf{A}'$ and $\mathbf{Y} \cap \mathbf{A}'$. Notice that $\mathbf{X}', \mathbf{Y}'$ is a partition of $\mathbf{A}'$. By induction, it follows that $\mathbf{X}'$ and $\mathbf{Y}'$ are semi-linear. Now, just notice that $\mathbf{X} = (\mathbf{S} \backslash \mathbf{A}') \cup \mathbf{X}'$ and $\mathbf{Y} = (\mathbf{T} \backslash \mathbf{A}') \cup \mathbf{Y}'$. We derive that $\mathbf{X}$ and $\mathbf{Y}$ are semi-linear, and the induction is proved. □

We are now ready to prove our main result.

Theorem 27 *For every Petri net* N*, and for every semi-linear sets of markings* $\mathcal{M}, \mathcal{F}_0, \mathcal{F}_1$*: if* $\mathcal{M}_0 = \mathcal{M} \cap pre_N^*(\mathcal{F}_0)$ *and* $\mathcal{M}_1 = \mathcal{M} \cap pre_N^*(\mathcal{F}_1)$ *is a partition of* $\mathcal{M}$*, then* $\mathcal{M}_0$ *and* $\mathcal{M}_1$ *are semi-linear.*

Proof From Theorem 25, it follows that $\mathcal{M}_0$ and $\mathcal{M}_1$ are decomposable. Since those two sets are disjoint and the union is equal to the semi-linear set $\mathcal{M}$, it follows from Lemma 26 that $\mathcal{M}_0$ and $\mathcal{M}_1$ are semi-linear. □

Corollary 28 *Well-specified population protocols only compute Presburger predicates.*

Proof Let $(\mathcal{A}, \mathtt{I}, \mathtt{O})$ be a well-specified protocol. Then $\mathcal{I}$, $\mathcal{B}_0$, and $\mathcal{B}_1$ are semi-linear sets. Let $\mathcal{I}_0 = \mathcal{I} \cap pre^*_N(\mathcal{B}_0)$ and $\mathcal{I}_1 = \mathcal{I} \cap pre^*_N(\mathcal{B}_1)$. Since the protocol is well specified, each configuration of $\mathcal{I}$ can reach exactly one of $\mathcal{B}_0$ and $\mathcal{B}_1$, it follows that $\mathcal{I}_0$ and $\mathcal{I}_1$ is a partition of the semi-linear set $\mathcal{I}$. From Theorem 27, we derive that $\mathcal{I}_0$ and $\mathcal{I}_1$ are semi-linear. Thus the computed predicates are Presburger. □

9 Certificate-based algorithms for well-specification and correctness

Certificates provide an alternative algorithm to decide the well-specification and fitting problems. If we apply our algorithm for the tailor problem to an arbitrary protocol, then two cases are possible: if the protocol is well specified, then the algorithm terminates and returns a Presburger formula for the computed predicate. If the protocol is ill specified, then it has no certificate, and the algorithm does not terminate. In other words, our algorithm for the tailor problem is at the same time a semi-decision procedure for the well-specification problem.

In order to obtain a decision procedure, it suffices to run this semi-decision procedure for well-specification in parallel with a semi-decision procedure for ill-specification. But this second semi-decision procedure is easy to find. Recall that a protocol is ill specified if there is an input X and either

(1) a fair computation starting at the configuration $I(X)$ that does not stabilize, or
(2) two fair computations starting at$I(X)$, and stabilizing to opposite values.

The semi-decision procedure for ill-specification enumerates all inputs X, constructs for each of them the fragment of the reachability graph with root $I(X)$, which is finite by Lemma 1, and examines the bottom SCCs of this graph to decide if conditions (1) or (2) hold.

Since the semi-decision procedure for the well-specification problem returns a Presburger formula for the computed predicate, we can also use this combination of semi-decision procedures to solve the fitting problem: if the protocol is ill specified, then it does not fit any predicate; if the protocol is well specified, then we check whether the Presburger formulas for the intended predicate and the computed predicate are equivalent, which is a decidable problem.

Acknowledgements We thank the CONCUR reviewers for their insightful feedback.

References

1. Abdulla, P.A., Cerans, K., Jonsson, B., Tsay, Y.-K.: General decidability theorems for infinite-state systems. In: LICS'96: Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science, pp. 313–321. IEEE Computer Society (1996)
2. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. In: PODC'04, pp. 290–299. ACM (2004)
3. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. Distrib. Comput. **18**(4), 235–253 (2006)
4. Angluin, D., Aspnes, J., Eisenstat, D.: Fast computation by population protocols with a leader. In: DISC'06, Volume 4167 of LNCS, pp. 61–75. Springer, Berlin (2006)
5. Angluin, D., Aspnes, J., Eisenstat, D.: Stably computable predicates are semilinear. In: PODC'06, pp. 292–299. ACM (2006)
6. Angluin, D., Aspnes, J., Eisenstat, D.: Fast computation by population protocols with a leader. Distrib. Comput. **21**(3), 183–199 (2008)
7. Angluin, D., Aspnes, J., Eisenstat, D., Ruppert, E.: The computational power of population protocols. Distrib. Comput. **20**(4), 279–304 (2007)

8. Apt, K.R., Kozen, D.C.: Limits for automatic verification of finite-state concurrent systems. Inf. Process. Lett. **22**(6), 307–309 (1986)
9. Bouziane, Z., Finkel, A.: Cyclic petri net reachability sets are semi-linear effectively constructible. Electron. Notes Theor. Comput. Sci. **9**, 15–24 (1997)
10. Chatzigiannakis, I., Michail, O., Spirakis, P.G.: Algorithmic verification of population protocols. In: SSS'10, Volume 6366 of LNCS, pp. 221–235. Springer, Berlin (2010)
11. Clement, J., Delporte-Gallet, C., Fauconnier, H., Sighireanu, M.: Guidelines for the verification of population protocols. In: ICDCS'11, pp. 215–224 (2011)
12. Deng, Y., Monin, J.: Verifying self-stabilizing population protocols with coq. In: TASE'09, pp. 201–208. IEEE Computer Society (2009)
13. Diamadi, Z., Fischer, M.J.: A simple game for the study of trust in distributed systems. Wuhan Univ. J. Nat. Sci. **6**(1–2), 72–82 (2001)
14. Eilenberg, S., Schützenberger, M.P.: Rational sets in commutative monoids. J. Algebra **13**(2), 173–191 (1969)
15. Enderton, H.B.: A Mathematical Introduction to Logic. Academic Press, San Diego (2001)
16. Finkel, A., Leroux, J.: How to compose presburger-accelerations: Applications to broadcast protocols. In: FST TCS'02, Volume 2556 of Lecture Notes in Computer Science, pp. 145–156. Springer, Berlin (2002)
17. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere!. Theor. Comput. Sci. **256**(1–2), 63–92 (2001)
18. Fribourg, L., Olsén, H.: Reductions of petri nets and unfolding of propositional logic programs. In: LOPSTR'96, Volume 1207 of Lecture Notes in Computer Science, pp. 187–203. Springer, Berlin (1996)
19. Ginsburg, S., Spanier, E.H.: Semigroups, Presburger formulas, and languages. Pac. J. Math. **16**(2), 285–296 (1966)
20. Hack, M.H.T.: Decidability questions for Petri nets. Technical Report 161, MIT (1976)
21. Hirshfeld, Y.: Congruences in Commutative Semigroups. LFCS, Department of Computer Science, University of Edinburgh, Edinburgh (1994)
22. Leroux, J.: The general vector addition system reachability problem by Presburger inductive invariants. In: LICS'09, pp. 4–13. IEEE Computer Society (2009)
23. Leroux, J.: Vector addition system reversible reachability problem. In: CONCUR'11, Volume 6901 of LNCS, pp. 327–341. Springer, Berlin (2011)
24. Leroux, J.: Vector addition systems reachability problem (a simpler solution). In: Turing-100: The Alan Turing Centenary Conference, Volume 10 of EPiC Series, pp. 214–228. EasyChair (2012)
25. Leroux, J.: Presburger vector addition systems. In: LICS'13, pp. 23–32. IEEE Computer Society (2013)
26. Leroux, J.: Vector addition system reversible reachability problem. Log. Methods Comput. Sci. **9**(1) (2013)
27. Leroux, J., Schmitz, S.: Demystifying reachability in vector addition systems. In: LICS'15, pp. 56–67. IEEE Computer Society (2015)
28. Leroux, J., Sutre, G.: Flat counter automata almost everywhere!. In: Peled, D.A., Tsay, Y. (eds.), Automated Technology for Verification and Analysis, Third International Symposium, ATVA 2005, Taipei, Taiwan, October 4–7, 2005, Proceedings, Volume 3707 of Lecture Notes in Computer Science, pp. 489–503. Springer, Berlin (2005)
29. Lipton, R.: The Reachability Problem is Exponential-Space Hard. Technical Report 62, Department of Computer Science, Yale University (1976)
30. Mayr, E.W.: An algorithm for the general petri net reachability problem. In: STOC'81, pp. 238–246. ACM (1981)
31. Navlakha, S., Bar-Joseph, Z.: Distributed information processing in biological and computational systems. Commun. ACM **58**(1), 94–102 (2014)
32. Pang, J., Luo, Z., Deng, Y.: On automatic verification of self-stabilizing population protocols. In: TASE'08, pp. 185–192. IEEE Computer Society (2008)
33. Sun, J., Liu, Y., Dong, J.S., Pang, J.: PAT: towards flexible verification under fairness. In: CAV'09, Volume 5643 of LNCS, pp. 709–714. Springer, Berlin (2009)