Towards Efficient Verification of Population Protocols

Michael Blondin
Technische Universität München
blondin@in.tum.de

Javier Esparza
Technische Universität München
esparza@in.tum.de

Stefan Jaax
Technische Universität München
jaax@in.tum.de

Philipp J. Meyer
Technische Universität München
meyerphi@in.tum.de

ABSTRACT

Population protocols are a well established model of computation by anonymous, identical finite state agents. A protocol is well-specified if from every initial configuration, all fair executions of the protocol reach a common consensus. The central verification question for population protocols is the *well-specification problem*: deciding if a given protocol is well-specified. Esparza et al. have recently shown that this problem is decidable, but with very high complexity: it is at least as hard as the Petri net reachability problem, which is EXPSPACE-hard, and for which only algorithms of non-primitive recursive complexity are currently known.

In this paper we introduce the class WS^3 of well-specified strongly-silent protocols and we prove that it is suitable for automatic verification. More precisely, we show that WS^3 has the same computational power as general well-specified protocols, and captures standard protocols from the literature. Moreover, we show that the membership problem for WS^3 reduces to solving boolean combinations of linear constraints over $\mathbb{N}$. This allowed us to develop the first software able to automatically prove well-specification for *all* of the infinitely many possible inputs.

CCS CONCEPTS

• **Networks → Protocol testing and verification**; • **Theory of computation → Logic and verification**;

KEYWORDS

population protocols; automated verification; termination

1 INTRODUCTION

Population protocols [1, 2] are a model of distributed computation by many anonymous finite-state agents. They were initially introduced to model networks of passively mobile sensors [1, 2], but are now also used to describe chemical reaction networks (see e.g. [7, 19]).

PODC '17, July 25-27, 2017, Washington, DC, USA

ACM ISBN 978-1-4503-4992-5/17/07…$15.00
http://dx.doi.org/10.1145/3087801.3087816

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 b for a given initial configuration C_0 if in all fair executions starting at C_0, all agents eventually agree to b — so, intuitively, population protocols compute by reaching consensus under a certain fairness condition. A protocol is *well-specified* if it computes a value for each of its infinitely many initial configurations (also called *inputs*). The predicate computed by a protocol is the function that assigns to each input the corresponding consensus value. In a famous series of papers, Angluin *et al.* [1, 2] have shown that well-specified protocols compute exactly the predicates definable in Presburger arithmetic [1–4].

In this paper we search for efficient algorithms for the *well-specification problem*: Given a population protocol, is it well-specified? This is a question about an infinite family of finite-state systems. Indeed, for every input the semantics of a protocol is a finite graph with the reachable configurations as nodes. Deciding if the protocol reaches consensus for a fixed input only requires to inspect one of these graphs, and can be done automatically using a model checker. This approach has been followed in a number of papers [6, 8, 20, 24], but it only shows well-specification for some inputs. There has also been work in formalizing well-specification proofs in interactive theorem provers [10], but this approach is not automatic: a human prover must first come up with a proof for each particular protocol.

Recently, the second author, together with other co-authors, has shown that the well-specification problem is decidable [13]. That is, there is an algorithm that decides if for all inputs the protocol stabilizes to a boolean value. The proof uses deep results of the theory of Petri nets, a model very close to population protocols. However, the same paper shows that the well-specification problem is at least as hard as the reachability problem for Petri nets, a famously difficult problem. More precisely, the problem is known to be EXPSPACE-hard, and all known algorithms for it have non-primitive recursive complexity [22]. In particular, there are no stable implementations of any of these algorithms, and they are considered impractical for nearly all applications.

For this reason, in this paper we search for a class of well-specified protocols satisfying three properties:

(a) *No loss of expressive power*: the class should compute all Presburger-definable predicates.

(b) *Natural*: the class should contain most protocols discussed in the literature.
(c) *Feasible membership problem*: membership for the class should have reasonable complexity.

The class *WS* of all well-specified protocols obviously satisfies (a) and (b), but not (c). So we introduce a new class WS^3, standing for *Well-Specified Strongly Silent* protocols. We show that WS^3 still satisfies (a) and (b), and then prove that the membership problem for WS^3 is in the complexity class DP; the class of languages L such that $L = L_1 \cap L_2$ for some languages $L_1 \in \mathsf{NP}$ and $L_2 \in \mathsf{coNP}$. This is a dramatic improvement with respect to the EXPSPACE-hardness of the membership problem for *WS*.

Our proof that the problem is in DP reduces membership for WS^3 to checking (un)satisfiability of two systems of boolean combinations of linear constraints over the natural numbers. This allowed us to implement our decision procedure on top of the constraint solver Z3 [9], yielding the first software able to automatically prove well-specification for *all* inputs. We tested our implementation on the families of protocols studied in [6, 8, 20, 24]. These papers prove well-specification for some inputs of protocols with up to 9 states and 28 transitions. Our approach proves well-specification for all inputs of protocols with up to 20 states in less than one second, and protocols with 70 states and 2500 transitions in less than one hour. In particular, we can automatically prove well-specification for *all* inputs in less time than previous tools needed to check *one single large input*.

The verification problem for population protocols naturally divides into two parts: checking that a given protocol is well specified, and checking that a given well-specified protocol computes the desired predicate. While in this paper we are concerned with well-specification, our implementation is already able to solve the second problem for all the families of protocols described above. This is achieved by adding to the second system of constraints used to check well-specification further linear constraints describing the sets of input configurations for which the protocol should return *true* or *false*. An extension of the software that, given a protocol and an arbitrary Presburger predicate, checks whether the protocol computes the predicate, requires to solve implementation problems related to Presburger arithmetic, and is left for future research.

The paper is organized as follows. Section 2 contains basic definitions. Section 3 introduces an intermediate class WS^2 of silent well-specified protocols, and shows that its membership problem is still as hard as for *WS*. In Section 4, we characterize WS^2 in terms of two properties which are then strengthened to define our new class WS^3. We then show that the properties defining WS^3 can be tested in NP and coNP, and so that membership for WS^3 is in DP. Section 5 proves that WS^3-protocols compute all Presburger predicates. Section 6 reports on our experimental results, and Section 7 presents conclusions.

2 PRELIMINARIES

Multisets. A *multiset* over a finite set E is a mapping $M : E \to \mathbb{N}$. The set of all multisets over E is denoted $\mathbb{N}^E$. For every $e \in E$, $M(e)$ denotes the number of occurrences of e in M. We sometimes denote multisets using a set-like notation, e.g. $\{\!|f, g, g|\!\}$ is the multiset M such that $M(f) = 1$, $M(g) = 2$ and $M(e) = 0$ for every $e \in E \setminus \{f, g\}$. The *support* of $M \in \mathbb{N}^E$ is $[\![M]\!] \stackrel{\text{def}}{=} \{e \in E : M(e) > 0\}$. The *size* of $M \in \mathbb{N}^E$ is $|M| \stackrel{\text{def}}{=} \sum_{e \in E} M(e)$. Addition and comparison are extended to multisets componentwise, i.e. $(M + M')(e) \stackrel{\text{def}}{=} M(e) + M'(e)$ for every $e \in E$, and $M \leq M' \stackrel{\text{def}}{\iff} M(e) \leq M(e)$ for every $e \in E$. We define multiset difference as $(M \ominus M')(e) \stackrel{\text{def}}{=} \max(M(e) - M'(e), 0)$ for every $e \in E$. The empty multiset is denoted $\mathbf{0}$, and for every $e \in E$ we write $\boldsymbol{e} \stackrel{\text{def}}{=} \{\!|e|\!\}$.

Population protocols. A *population* P over a finite set E is a multiset $P \in \mathbb{N}^E$ such that $|P| \geq 2$. The set of all populations over E is denoted by $\mathrm{Pop}(E)$. A *population protocol* is a tuple $\mathcal{P} = (Q, T, \Sigma, I, O)$ where

- Q is a non-empty finite set of *states*,
- $T \subseteq Q^2 \times Q^2$ is a set of *transitions* such that for every $(p, q) \in Q^2$ there exists at least a pair $(p', q') \in Q^2$ such that $(p, q, p', q') \in T$,
- Σ is a non-empty finite *input alphabet*,
- $I : \Sigma \to Q$ is the *input function* mapping input symbols to states,
- $O : Q \to \{0, 1\}$ is the *output function* mapping states to boolean values.

Following the convention of previous papers, we call the populations of $\mathrm{Pop}(Q)$ *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 $q \in Q$, and at least two agents in total.

Pairs of agents[1] interact using transitions. For every $t = (p, q, p', q') \in T$, we write $(p, q) \mapsto (p', q')$ to denote t, and we define $\mathrm{pre}(t) \stackrel{\text{def}}{=} \{\!|p, q|\!\}$ and $\mathrm{post}(t) \stackrel{\text{def}}{=} \{\!|p', q'|\!\}$. For every configuration C and transition $t \in T$, we say that t is *enabled* at C if $C \geq \mathrm{pre}(t)$. Note that by definition of T, every configuration enables at least one transition. A transition $t \in T$ enabled at C can *occur*, leading to the configuration $C \ominus \mathrm{pre}(t) + \mathrm{post}(t)$. Intuitively, a pair of agents in states $\mathrm{pre}(t)$ move to states $\mathrm{post}(t)$. We write $C \xrightarrow{t} C'$ to denote that t is enabled at C and that its occurrence leads to C'. A transition $t \in T$ is *silent* if $\mathrm{pre}(t) = \mathrm{post}(t)$, i.e., if it cannot change the current configuration.

For every sequence of transitions $w = t_1 t_2 \cdots t_k$, we write $C \xrightarrow{w} C'$ if there exists a sequence of configurations $C_0, C_1, \ldots, C_k$ such that $C = C_0 \xrightarrow{t_1} C_1 \cdots \xrightarrow{t_k} C_k = C'$. We also write $C \to C'$ if $C \xrightarrow{t} C'$ for some transition $t \in T$, and call $C \to C'$ a *step*. We write $C \xrightarrow{*} C'$ if $C \xrightarrow{w} C'$ for some $w \in T^*$. We say that C' is *reachable from* C if $C \xrightarrow{*} C'$. An *execution* is an infinite sequence of configurations $C_0 C_1 \cdots$ such that $C_i \to C_{i+1}$ for every $i \in \mathbb{N}$. An execution $C_0 C_1 \cdots$ is *fair* if for every step $C \to C'$, if $C_i = C$ for infinitely many indices $i \in \mathbb{N}$, then $C_j = C$ and $C_{j+1} = C'$ for infinitely many indices $j \in \mathbb{N}$. We say that a configuration C is

- *terminal* if $C \xrightarrow{*} C'$ implies $C = C'$, i.e., if every transition enabled at C is silent;
- a *consensus configuration* if $O(p) = O(q)$ for every $p, q \in [\![C]\!]$.

[1] While protocols only model interactions between two agents, k-way interactions for a fixed $k > 2$ can be simulated by adding additional states.

For every consensus configuration C, let $O(C)$ denote the unique output of the states in $[\![C]\!]$. An execution $C_0C_1\cdots$ *stabilizes* to $b \in \{0,1\}$ if there exists $n \in \mathbb{N}$ such that C_i is a consensus configuration and $O(C_i) = b$ for every $i \geq n$.

Predicates computable by population protocols. Every *input* $X \in \mathrm{Pop}(\Sigma)$ is mapped to the configuration $I(X) \in \mathrm{Pop}(Q)$ defined by

$$I(X)(q) \stackrel{\text{def}}{=} \sum_{\substack{\sigma\in\Sigma \\ I(\sigma)=q}} X(\sigma) \text{ for every } q \in Q.$$

A configuration C is said to be *initial* if $C = I(X)$ for some input X. A population protocol is *well-specified* if for every input X, there exists $b \in \{0,1\}$ such that every fair execution of $\mathcal{P}$ starting at $I(X)$ stabilizes to b. We say that $\mathcal{P}$ *computes* a predicate φ if for every input X, every fair execution of $\mathcal{P}$ starting at $I(X)$ stabilizes to $\varphi(X)$. It is readily seen that $\mathcal{P}$ computes a predicate if and only if it is well-specified.

Example 2.1. We consider the majority protocol of [3] as a running example. Initially, agents of the protocol can be in either state A or B. The protocol computes whether there are at least as many agents in state B as there are in state A. The states and the input alphabet are $Q = \{A, B, a, b\}$ and $\Sigma = \{A, B\}$ respectively. The input function is the identity function, and the output function is given by $O(B) = O(b) = 1$ and $O(A) = O(a) = 0$. The set of transitions T consists of:

$$t_{AB} = (A,B) \mapsto (a,b)$$
$$t_{Ab} = (A,b) \mapsto (A,a)$$
$$t_{Ba} = (B,a) \mapsto (B,b)$$
$$t_{ba} = (b,a) \mapsto (b,b)$$

and of silent transitions for the remaining pairs of states. Transition t_{AB} ensures that every fair execution eventually reaches a configuration C such that $C(A) = 0$ or $C(B) = 0$. If $C(A) = 0 = C(B)$, then there were initially equally many agents in A and B. Transition t_{ba} then acts as tie breaker, resulting in a terminal configuration populated only by b. If, say, $C(A) > 0$ and $C(B) = 0$, then there were initially more As than Bs, and t_{Ab} ensures that every fair execution eventually reaches a terminal configuration populated only by A and a.

3 WELL-SPECIFIED SILENT PROTOCOLS

Silent protocols[2] were introduced in [12]. Loosely speaking, a protocol is silent if communication between agents eventually ceases, i.e. if every fair execution eventually stays in the same configuration forever. Observe that a well-specified protocol need not be silent: fair executions may keep alternating from a configuration to another as long as they are consensus configurations with the same output.

More formally, we say that an execution $C_0C_1\cdots$ is *silent* if there exists $n \in \mathbb{N}$ and a configuration C such that $C_i = C$ for every $i \geq n$. A population protocol $\mathcal{P}$ is *silent* if every fair execution of $\mathcal{P}$ is silent, regardless of the starting configuration. We call a protocol that is well-specified and silent a *WS^2-protocol*, and denote by WS^2 the set of all WS^2-protocols.

[2]Silent protocols are also referred to as *protocols with stabilizing states* and silent transitions are called *ineffective* in [17, 18].

Example 3.1. As explained in Example 2.1, every fair execution of the majority protocol is silent. This implies that the protocol is silent. If, for example, we add a new state b' where $O(b') = 1$, and transitions $(b,b) \mapsto (b',b')$, $(b',b') \mapsto (b,b)$, then the protocol is no longer silent since the execution where two agents alternate between states b and b' is fair but not silent.

Being silent is a desirable property. While in arbitrary protocols it is difficult to determine if an execution has already stabilized, in silent protocols it is simple: one just checks if the current configuration only enables silent transitions. Even though it is was not observed explicitely, the protocols introduced in [1] to characterize the expressive power of population protocols belong to WS^2. Therefore, WS^2-protocols can compute the same predicates as general ones.

Unfortunately, a slight adaptation of [14, Theorem 10] shows that the complexity of the membership problem for WS^2-protocols is still as high as for the general case:

Proposition 3.2. *The reachability problem for Petri nets is reducible in polynomial time to the membership problem for WS^2. In particular, membership for WS^2 is* EXPSPACE*-complete.*

To circumvent this high complexity, we will show in the next section how WS^2 can be refined into a smaller class of well-specified protocols with the same expressive power, and a membership problem of much lower complexity.

4 A FINER CLASS OF SILENT WELL-SPECIFIED PROTOCOLS: WS^3

It can be shown that WS^2-protocols are exactly the protocols satisfying the two following properties:

- Termination: for every configuration C, there exists a terminal configuration C' such that $C \stackrel{*}{\to} C'$.
- Consensus: for every initial configuration C, there exists $b \in \{0,1\}$ such that every terminal configuration C' reachable from C is a consensus configuration with output b, i.e. $C \stackrel{*}{\to} C'$ implies $O(C') = b$.

We will introduce the new class WS^3 as a refinement of WS^2 obtained by strengthening Termination and Consensus into two new properties called LayeredTermination and StrongConsensus. We introduce these properties in Section 4.1 and Section 4.2, and show that their decision problems belong to NP and coNP respectively.

Before doing so, let us introduce some useful notions. Let $\mathcal{P} = (Q, T, \Sigma, I, O)$ be a population protocol. For every $S \subseteq T$, $\mathcal{P}[S]$ denotes the *protocol induced by* S, i.e. $\mathcal{P}[S] \stackrel{\text{def}}{=} (Q, S \cup T', \Sigma, I, O)$ where $T' \stackrel{\text{def}}{=} \{(p,q,p,q) : p,q \in Q\}$ is added to ensure that any two states can interact. Let $\to_S$ denote the transition relation of $\mathcal{P}[S]$. An *ordered partition* of T is a tuple $(T_1, T_2, \ldots, T_n)$ of nonempty subsets of T such that $T = \bigcup_{i=1}^{n} T_i$ and $T_i \cap T_j = \emptyset$ for every $1 \leq i < j \leq n$.

4.1 Layered termination

We replace Termination by a stronger property called LayeredTermination, and show that deciding LayeredTermination belongs to NP. The definition of LayeredTermination is inspired by the typical structure of protocols found in the literature. Such

protocols are organized in layers such that transitions of higher layers cannot be enabled by executing transitions of lower layers. In particular, if the protocol reaches a configuration of the highest layer that does not enable any transition, then this configuration is terminal. For such protocols, Termination can be proven by showing that every (fair or unfair) execution of a layer is silent.

Definition 4.1. A population protocol $\mathcal{P} = (Q, T, \Sigma, I, O)$ satisfies LayeredTermination if there is an ordered partition

$$(T_1, T_2, \ldots, T_n)$$

of T such that the following properties hold for every $i \in [n]$:

(a) For every configuration C, every (fair or unfair) execution of $\mathcal{P}[T_i]$ starting at C is silent.

(b) For every configurations C and C', if $C \xrightarrow{*}_{T_i} C'$ and C is terminal in $\mathcal{P}[T_1 \cup T_2 \cup \cdots \cup T_{i-1}]$, then C' is also terminal in $\mathcal{P}[T_1 \cup T_2 \cup \cdots \cup T_{i-1}]$.

Example 4.2. The majority protocol satisfies LayeredTermination. Indeed, consider the ordered partition (T_1, T_2), where

$$T_1 = \{(A, B) \mapsto (a, b), (A, b) \mapsto (A, a)\}$$
$$T_2 = \{(B, a) \mapsto (B, b), (b, a) \mapsto (b, b)\}.$$

All executions of $\mathcal{P}[T_1]$ and $\mathcal{P}[T_2]$ are silent. For every terminal configuration C of $\mathcal{P}[T_1]$, we have $[\![C]\!] \subseteq \{A, a\}$ or $[\![C]\!] \subseteq \{B, a, b\}$. In the former case, no transition of T_2 is enabled; in the latter case, taking transitons of T_2 cannot enable T_1.

As briefly sketched above, LayeredTermination implies Termination. In the rest of this section, we prove that checking LayeredTermination is in NP. We do this by showing that conditions (a) and (b) of Definition 4.1 can be tested in polynomial time.

We recall a basic notion of Petri net theory recast in the terminology of population protocols. For every step $C \xrightarrow{t} C'$ and every state q of a population protocol, we have $C'(q) = C(q) + \mathrm{post}(t)(q) - \mathrm{pre}(t)(q)$. This observation can be extended to sequences of transitions. Let $|w|_t$ denote the number of occurrences of transition t in a sequence w. We have $C'(q) = C(q) + \sum_{t \in T} |w|_t \cdot (\mathrm{post}(t)(q) - \mathrm{pre}(t)(q))$. Thus, a necessary condition for $C \xrightarrow{w} C'$ is the existence of some $\boldsymbol{x} : T \to \mathbb{N}$ such that

$$C'(q) = C(q) + \sum_{t \in T} \boldsymbol{x}(t) \cdot (\mathrm{post}(t)(q) - \mathrm{pre}(t)(q)). \qquad (1)$$

We call (1) the *flow equation* for state q.

Proposition 4.3. *Let $\mathcal{P} = (Q, T, \Sigma, I, O)$ be a population protocol. Deciding whether an ordered partition $(T_1, T_2, \ldots, T_n)$ of T satisfies condition (a) of Definition 4.1 can be done in polynomial time.*

Proof. Let $i \in [n]$ and let U_i be the set of non silent transitions of T_i. It can be shown that $\mathcal{P}[T_i]$ is non silent if and only if there exists $\boldsymbol{x} : U_i \to \mathbb{Q}$ such that $\sum_{t \in U_i} \boldsymbol{x}(t) \cdot (\mathrm{post}(t)(q) - \mathrm{pre}(t)(q)) = 0$ and $\boldsymbol{x}(q) \geq 0$ for every $q \in Q$, and $\boldsymbol{x}(q) > 0$ for some $q \in Q$. Therefore, since linear programming is in P, we can check for the (non) existence of an appropriate rational solution $\boldsymbol{x}_i$ for every $i \in [n]$. □

We show how to check condition (b) of Definition 4.1 in polynomial time. Let $U \subseteq T$ be a set of transitions. A configuration $C \in \mathrm{Pop}(Q)$ is *U-dead* if for every $t \in U$, $C \xrightarrow{t} C'$ implies $C' = C$. We say that $\mathcal{P}$ is *U-dead from* $C_0 \in \mathrm{Pop}(Q)$ if every configuration reachable from C_0 is U-dead, i.e. $C_0 \xrightarrow{*} C$ implies that C is U-dead. Finally, we say that $\mathcal{P}$ is *U-dead* if it is U-dead from every U-dead configuration $C_0 \in \mathrm{Pop}(Q)$.

Proposition 4.4. *Let $\mathcal{P} = (Q, T, \Sigma, I, O)$ be a population protocol. Deciding whether an ordered partition $(T_1, \ldots, T_n)$ of T satisfies condition (b) of Definition 4.1 can be done in polynomial time.*

Proof. Let $i \in [n]$ and let $U \stackrel{\text{def}}{=} T_1 \cup T_2 \cup \cdots \cup T_{i-1}$. $\mathcal{P}[T_i]$ satisfies condition (b) if and only if $\mathcal{P}[T_i]$ is U-dead. The latter can be tested in polynomial time through the following characterization: $\mathcal{P}[T_i]$ is *not* U-dead if and only if there exist $t \in T_i$ and non silent $u \in U$ such that for every non silent $u' \in U$:

$$\mathrm{pre}(u') \not\leq \mathrm{pre}(t) + (\mathrm{pre}(u) \ominus \mathrm{post}(t)). \qquad \square$$

Propositions 4.3 and 4.4 yield an NP procedure to decide LayeredTermination. Indeed, it suffices to guess an ordered partition and to check whether it satisfies conditions (a) and (b) of Definition 4.1 in polynomial time.

Corollary 4.5. *Deciding if a protocol satisfies LayeredTermination is in* NP.

4.2 Strong consensus

To overcome the high complexity of reachability in population protocols, we strengthen Consensus by replacing the reachability relation in its definition by an *over-approximation*, i.e., a relation $\dashrightarrow$ over configurations such that $C \xrightarrow{*} C'$ implies $C \dashrightarrow C'$. Observe that the flow equations provide an over-approximation of the reachability relation. Indeed, as mentioned earlier, if $C \xrightarrow{*} C'$, then there exists $\boldsymbol{x} : T \to \mathbb{N}$ such that $(C, C', \boldsymbol{x})$ satisfies all of the flow equations. However, this over-approximation alone is too crude for the verification of protocols.

Example 4.6. For example, let us consider the configurations $C = \{\!|A, B|\!\}$ and $C' = \{\!|a, a|\!\}$ of the majority protocol. The flow equations are satisfied by the mapping $\boldsymbol{x}$ such that $\boldsymbol{x}(t_{AB}) = \boldsymbol{x}(t_{Ab}) = 1$ and $\boldsymbol{x}(t_{Ba}) = \boldsymbol{x}(t_{ba}) = 0$. Yet, $C \xrightarrow{*} C'$ does not hold.

To obtain a finer reachability over-approximation, we introduce so-called traps and siphons constraints borrowed from the theory of Petri nets [11, 15, 16] and successfully applied to a number of analysis problems (see e.g. [5, 15, 16]). Intuitively, for some subset of transitions $U \subseteq T$, a U-trap is a set of states $P \subseteq Q$ such that every transition of U that removes an agent from P also moves an agent into P. Conversely, a U-siphon is a set $P \subseteq Q$ such that every transition of U that moves an agent into P also removes an agent from P. More formally, let ${}^\bullet R \stackrel{\text{def}}{=} \{t \in T : [\![\mathrm{post}(t)]\!] \cap R \neq \emptyset\}$ and $R^\bullet \stackrel{\text{def}}{=} \{t \in T : [\![\mathrm{pre}(t)]\!] \cap R \neq \emptyset\}$. U-siphons and U-traps are defined as follows:

Definition 4.7. A subset of states $P \subseteq Q$ is a *U-trap* if $P^\bullet \cap U \subseteq {}^\bullet P$, and a *$U$-siphon* if ${}^\bullet P \cap U \subseteq P^\bullet$.

For every configuration $C \in \mathrm{Pop}(Q)$ and $P \subseteq Q$, let $C(P) \stackrel{\text{def}}{=} \sum_{q \in P} C(q)$. Consider a sequence of steps $C_0 \xrightarrow{t_1} C_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} C_n$

where $t_1, \ldots, t_n \in U$. It follows from Definition 4.7 that if some transition t_i moves an agent to a U-trap P, then $C_j(P) > 0$ for every $j \geq i$. Similarly, if some transition t_i removes an agent from a U-siphon, then $C_j(P) > 0$ for every $j < i$. In particular:

Observation 4.8. *Let $U \subseteq T$, and let C and C' be configurations such that $C \xrightarrow{*}_U C'$. For every U-trap P, if $C'(P) = 0$, then ${}^\bullet P \cap U = \emptyset$. For every U-siphon P, if $C(S) = 0$, then $P^\bullet \cap U = \emptyset$.*

We obtain a necessary condition for $C \xrightarrow{*}_U C'$ to hold, which we call potential reachability:

Definition 4.9. Let C, C' be two configurations, let $\boldsymbol{x} : T \to \mathbb{N}$, and let $U = [\![\boldsymbol{x}]\!]$. We say that C' is *potentially reachable from C through $\boldsymbol{x}$*, denoted $C \overset{\boldsymbol{x}}{\dashrightarrow} C'$, if

(a) the flow equation (1) holds for every $q \in Q$,
(b) $C'(P) = 0$ implies ${}^\bullet P \cap U = \emptyset$ for every U-trap P, and
(c) $C(P) = 0$ implies $P^\bullet \cap U = \emptyset$ for every U-siphon P.

Example 4.10. Let us reconsider Example 4.6. Let $U = [\![\boldsymbol{x}]\!] = \{t_{AB}, t_{Ab}\}$ and $P = \{A, b\}$. Recall that $t_{AB} = (A, B) \mapsto (a, b)$ and $t_{Ab} = (A, b) \mapsto (A, a)$. We have $P^\bullet \cap U = U$ which implies that P is a U-trap. This means that Definition 4.9(b) is violated as $C'(P) = 0$ and ${}^\bullet P \cap U = U \neq \emptyset$. Therefore, $\wr A, B \wr \overset{\boldsymbol{x}}{\dashrightarrow} \wr a, a \wr$ does not hold.

We write $C \dashrightarrow C'$ if $C \overset{\boldsymbol{x}}{\dashrightarrow} C'$ for some $\boldsymbol{x} : T \to \mathbb{N}$. As an immediate consequence of Observation 4.8, for every configurations C and C', if $C \xrightarrow{*} C'$, then $C \dashrightarrow C'$. This allows us to strengthen Consensus by redefining it in terms of potential reachability instead of reachability:

Definition 4.11. A protocol satisfies StrongConsensus if for every initial configuration C, there exists $b \in \{0, 1\}$ such that every terminal configuration C' potentially reachable from C is a consensus configuration with output b, i.e. $C \dashrightarrow C'$ implies $O(C') = b$.

Since the number of U-traps and U-siphons of a protocol can be exponential in the number of states, checking trap and siphon constraints by enumerating them may take exponential time. Fortunately, this can be avoided. By definition, it follows that the union of two U-traps is again a U-trap, and similarly for siphons. Therefore, given a configuration C, there exists a unique maximal U-siphon $P_{\max}$ such that $C(P_{\max}) = 0$, and a unique maximal U-trap $P'_{\max}$ such that $C(P'_{\max}) = 0$. Moreover, $P_{\max}$ and $P'_{\max}$ can be computed in linear time by means of a simple greedy algorithm (see e.g. [11, Ex. 4.5]). This simplifies the task of checking traps and siphons constraints, and yields a coNP procedure for testing StrongConsensus:

Proposition 4.12. *Deciding if a protocol satisfies StrongConsensus is in* coNP.

Proof. Testing whether a protocol *does not* satisfy StrongConsensus can be done by guessing $C_0, C, C' \in \mathrm{Pop}(Q)$, $b \in \{0, 1\}$, $q, q' \in Q$ and $\boldsymbol{x}, \boldsymbol{x}' : T \to \mathbb{N}$, and testing whether

(a) C_0 is initial, C is terminal, C' is terminal, $q \in [\![C]\!]$, $q' \in [\![C']\!]$, $O(q) \neq O(q')$, and
(b) $C_0 \overset{\boldsymbol{x}}{\dashrightarrow} C$ and $C_0 \overset{\boldsymbol{x}'}{\dashrightarrow} C'$.

Since there is no *a priori* bound on the size of C_0, C, C' and $\boldsymbol{x}, \boldsymbol{x}'$, we guess them carefully. First, we guess whether $D(p) = 0, D(p) = 1$ or $D(p) \geq 2$ for every $D \in \{C_0, C, C'\}$ and $p \in Q$. This gives enough information to test (a). Then, we guess $[\![\boldsymbol{x}]\!]$ and $[\![\boldsymbol{x}']\!]$. This allows to test traps/siphons constraints as follows. Let $U \overset{\text{def}}{=} [\![\boldsymbol{x}]\!]$, let $P_{\max}$ be the maximal U-trap such that $C(P_{\max}) = 0$, and let $P'_{\max}$ be the maximal U-siphon such that $C_0(P'_{\max}) = 0$. Conditions (b) and (c) of Definition 4.9 hold if and only if ${}^\bullet(P_{\max}) \cap U = \emptyset$ and $(P'_{\max})^\bullet \cap U = \emptyset$, which can be tested in polynomial time. The same is done for $\boldsymbol{x}'$. If (a) and siphons/traps constraints hold, we build the system $\mathcal{S}$ of linear equations/inequalities obtained from the conjunction of the flow equations together with the constraints already guessed. By standard results on integer linear programming (see e.g. [23, Sect. 17]), if $\mathcal{S}$ has a solution, then it has one of polynomial size, and hence we may guess it. □

4.3 WS^3-protocols

We say that a protocol belongs to WS^3 if it satisfies LayeredTermination and StrongConsensus. Since $WS^3 \subseteq WS^2 \subseteq WS$ holds, every WS^3-protocol is well-specified. Recall that a language L belongs to the class DP [21] if there exist languages $L_1 \in$ NP and $L_2 \in$ coNP such that $L = L_1 \cap L_2$. By taking L_1 and L_2 respectively as the languages of population protocols satisfying LayeredTermination and StrongConsensus, Corollary 4.5 and Proposition 4.12 yield:

Theorem 4.13. *The membership problem for WS^3-protocols is in* DP.

5 WS^3 IS AS EXPRESSIVE AS WS

In a famous result, Angluin *et al.* [3] have shown that a predicate is computable by a population protocol if and only if it is definable in Presburger arithmetic, the first-order theory of addition [1, 3]. In particular, [1] constructs protocols for Presburger-definable predicates by means of a well-known result: *Presburger-definable* predicates are the smallest set of predicates containing all threshold and remainder predicates, and closed under boolean operations. A *threshold predicate* is a predicate of the form

$$P(x_1, \ldots, x_k) = \left(\sum_{i=1}^{k} a_i x_i < c \right),$$

where $k \geq 1$ and $a_1, \ldots, a_k, c \in \mathbb{Z}$. A *remainder predicate* is a predicate of the form

$$P(x_1, \ldots, x_k) = \left(\sum_{i=1}^{k} a_i x_i \equiv c \pmod{m} \right),$$

where $k \geq 1$, $m \geq 2$ and $a_1, \ldots, a_k, c, m \in \mathbb{Z}$. Here, we show that these predicates can be computed by WS^3-protocols, and that WS^3 is closed under negation and conjunction. As a consequence, we obtain that WS^3-protocols are as expressive as WS, the class of all well-specified protocols.

Threshold. We describe the protocol given in [1] to compute the threshold predicate $\sum_{i=1}^{k} a_i x_i < c$. Let

$$v_{\max} \overset{\text{def}}{=} \max(|a_1|, |a_2|, \ldots, |a_k|, |c| + 1)$$

and define

$$f(m,n) \stackrel{\text{def}}{=} \max(-v_{\max}, \min(v_{\max}, m+n))$$
$$g(m,n) \stackrel{\text{def}}{=} (m+n) - f(m,n)$$
$$b(m,n) \stackrel{\text{def}}{=} (f(m,n) < c)$$

The protocol is $\mathcal{P}_{\text{thr}} \stackrel{\text{def}}{=} (Q, T, \Sigma, I, O)$, where

$$Q \stackrel{\text{def}}{=} \{0,1\} \times [-v_{\max}, v_{\max}] \times \{0,1\}$$
$$\Sigma \stackrel{\text{def}}{=} \{x_1, x_2, \ldots, x_k\}$$
$$I(x_i) \stackrel{\text{def}}{=} (1, a_i, a_i < c) \text{ for every } i \in [k]$$
$$O(\ell, n, o) \stackrel{\text{def}}{=} o \text{ for every state } (\ell, n, o),$$

and T contains

$$(1,n,o), (l,n',o') \mapsto (1, f(n,n'), b(n,n')), (0, g(n,n'), b(n,n'))$$

for every $n, n' \in [-v_{\max}, v_{\max}]$, $\ell, o, o' \in \{0,1\}$. Intuitively, a state (ℓ, n, o) indicates that the agent has value n, opinion o, and that it is a leader if and only if $\ell = 1$. When a leader q and a state r interact, r becomes a non leader, and q increases its value as much as possible by substracting from the value of r. Moreover, a leader can change the opinion of any non leader.

Proposition 5.1. *$\mathcal{P}_{thr}$ satisfies StrongConsensus.*

Proof. Let $\text{val}(q) \stackrel{\text{def}}{=} n$ for every state $q = (\ell, n, o) \in Q$, and let $\text{val}(C) \stackrel{\text{def}}{=} \sum_{q \in Q} C(q) \cdot \text{val}(q)$ for every configuration $C \in \text{Pop}(Q)$. The following holds for every $C, C' \in \text{Pop}(Q)$:

(a) If $(C, C', \boldsymbol{x})$ is a solution to the flow equations for some $\boldsymbol{x} : T \to \mathbb{N}$, then $\text{val}(C) = \text{val}(C')$.
(b) If C, C' are terminal, C and C' contain a leader, and $\text{val}(C) = \text{val}(C')$, then $O(C) = O(C')$.

Suppose for the sake of contradiction that $\mathcal{P}$ does not satisfy StrongConsensus. There exist $C_0, C, C' \in \text{Pop}(Q)$, $q, q' \in Q$ and $\boldsymbol{x}, \boldsymbol{x}' : T \to \mathbb{N}$ such that $C_0 \xrightarrow{\boldsymbol{x}} C$, $C_0 \xrightarrow{\boldsymbol{x}'} C'$, C_0 is initial, C and C' are terminal consensus configurations, $q \in [\![C]\!]$, $q' \in [\![C']\!]$ and $O(q) \neq O(q')$. Note that $(C_0, C, \boldsymbol{x})$ and $(C_0, C', \boldsymbol{x}')$ both satisfy the flow equations. Thus, by (a), $\text{val}(C) = \text{val}(C_0) = \text{val}(C')$. Since C_0 is initial, it contains a leader. Since the set of leaders forms a U-trap for every $U \subseteq T$, and $(C_0, C, \boldsymbol{x})$ and $(C_0, C', \boldsymbol{x})$ satisfy trap constraints, C and C' contain a leader. By (b), C and C' are consensus configurations with $O(C) = O(C')$, which is a contradiction. □

Proposition 5.2. *$\mathcal{P}_{thr}$ satisfies LayeredTermination.*

Proof. Let $L_0 \stackrel{\text{def}}{=} \{(1,x,0) : c \leq x \leq v_{\max}\}$, $L_1 \stackrel{\text{def}}{=} \{(1,x,1) : -v_{\max} \leq x < c\}$, $N_0 \stackrel{\text{def}}{=} \{(0,0,0)\}$ and $N_1 \stackrel{\text{def}}{=} \{(0,0,1)\}$. It can be shown that the following ordered partitions satisfy layered termination for $c > 0$ and $c \leq 0$ respectively:

$$T_1 \stackrel{\text{def}}{=} \{t \in T : \text{pre}(t) \neq \multiset{q, r} \text{ for all } q \in L_0, r \in N_1\},$$
$$T_2 \stackrel{\text{def}}{=} T \setminus T_1, \text{ and}$$
$$S_1 \stackrel{\text{def}}{=} \{t \in T : \text{pre}(t) \neq \multiset{q, r} \text{ for all } q \in L_1, r \in N_0\},$$
$$S_2 \stackrel{\text{def}}{=} T \setminus S_1.$$ □

Remainder. We give a protocol for the remainder predicate

$$\sum_{i=1}^{k} a_i x_i \equiv c \pmod{m}.$$

The protocol is $\mathcal{P}_{\text{rmd}} = (Q, T, \Sigma, I, O)$, where

$$Q \stackrel{\text{def}}{=} [0, m) \cup \{\text{true}, \text{false}\}$$
$$\Sigma \stackrel{\text{def}}{=} \{x_1, x_2, \ldots, x_k\}$$
$$I(x_i) \stackrel{\text{def}}{=} a_i \bmod m \text{ for every } i \in [k]$$
$$O(q) \stackrel{\text{def}}{=} \begin{cases} 1 \text{ if } q \in \{c, \text{true}\} \\ 0 \text{ otherwise} \end{cases}$$

and where T contains the following transitions for every $n, n' \in [0, m)$ and $b \in \{\text{false}, \text{true}\}$:

$$(n, n') \mapsto (n + n' \bmod m, n + n' \bmod m = c) \quad \text{and}$$
$$(n, b) \mapsto (n, n = c).$$

In the the full version of this paper[3] we show that $\mathcal{P}_{\text{rmd}}$ belongs to WS^3 by adapting the proof for $\mathcal{P}_{\text{thr}}$.

Negation and conjunction. Let $\mathcal{P}_1 = (Q_1, T_1, \Sigma, I_1, O_1)$ and $\mathcal{P}_2 = (Q_2, T_2, \Sigma, I_2, O_2)$ be WS^3-protocols computing predicates φ_1 and φ_2 respectively. We may assume that $\mathcal{P}_1$ and $\mathcal{P}_2$ are defined over identical Σ, for we can always extend the input domain of threshold/remainder predicates by variables with coefficients of value zero. The predicate $\neg\varphi_i$ can be computed by replacing O_i by the new output function O_i' such that $O_i'(q) \stackrel{\text{def}}{=} \neg O_i(q)$ for every $q \in Q_i$. To compute $\varphi_1 \wedge \varphi_2$, we build an asynchronous product where steps of $\mathcal{P}_1$ and $\mathcal{P}_2$ can be executed independently.

More formally, the *conjunction* of $\mathcal{P}_1$ and $\mathcal{P}_2$ is defined as the population protocol $\mathcal{P} \stackrel{\text{def}}{=} (Q, S, I, \Sigma, O)$ such that $Q \stackrel{\text{def}}{=} Q_1 \times Q_2$, $S \stackrel{\text{def}}{=} S_1 \cup S_2$, $I(\sigma) \stackrel{\text{def}}{=} (I_1(\sigma), I_2(\sigma))$ and $O(p,q) \stackrel{\text{def}}{=} O_1(p) \wedge O_2(q)$ where

$$S_1 \stackrel{\text{def}}{=} \{(p,r), (p',r') \mapsto (q,r), (q',r') : (p,p',q,q') \in T_1, r, r' \in Q_2\},$$
$$S_2 \stackrel{\text{def}}{=} \{(r,p), (r',p') \mapsto (r,q), (r',q') : (p,p',q,q') \in T_2, r, r' \in Q_1\}.$$

In the full version of this paper. we show that $\mathcal{P}$ is in WS^3 since terminal/consensus configurations, flow equations, and traps and siphons constraints are preserved by projections from $\mathcal{P}$ onto $\mathcal{P}_1$ and $\mathcal{P}_2$.

[3] https://arxiv.org/abs/1703.04367

Threshold			
$v_{\max}$	$\lvert Q\rvert$	$\lvert T\rvert$	Time
3	28	288	8.0
4	36	478	26.5
5	44	716	97.6
6	52	1002	243.4
7	60	1336	565.0
8	68	1718	1019.7
9	76	2148	2375.9
10	84	2626	timeout

Remainder			
m	$\lvert Q\rvert$	$\lvert T\rvert$	Time
10	12	65	0.4
20	22	230	2.8
30	32	495	15.9
40	42	860	79.3
50	52	1325	440.3
60	62	1890	3055.4
70	72	2555	3176.5
80	82	3320	timeout

Majority			
	$\lvert Q\rvert$	$\lvert T\rvert$	Time
	4	4	0.1

Flock of birds [6]			
c	$\lvert Q\rvert$	$\lvert T\rvert$	Time
20	21	210	1.5
25	26	325	3.3
30	31	465	7.7
35	36	630	20.8
40	41	820	106.9
45	46	1035	295.6
50	51	1275	181.6
55	56	1540	timeout

Broadcast			
	$\lvert Q\rvert$	$\lvert T\rvert$	Time
	2	1	0.1

Flock of birds [8]			
c	$\lvert Q\rvert$	$\lvert T\rvert$	Time
50	51	99	11.8
100	101	199	44.8
150	151	299	369.1
200	201	399	778.8
250	251	499	1554.2
300	301	599	2782.5
325	326	649	3470.8
350	351	699	timeout

Table 1: Results of the experimental evaluation where $|Q|$ denotes the number of states, $|T|$ denotes the number of non silent transitions, and the time to prove membership for WS^3 is given in seconds.

6 EXPERIMENTAL RESULTS

We have developed a tool called `Peregrine`[4] to check membership in WS^3. Peregrine is implemented on top of the SMT solver Z3 [9].

`Peregrine` reads in a population protocol $\mathcal{P} = (Q, T, \Sigma, I, O)$ and constructs two sets of constraints. The first set is satisfiable if and only if LayeredTermination holds, and the second is unsatisfiable if and only if StrongConsensus holds.

For LayeredTermination, our tool `Peregrine` iteratively constructs constraints checking the existence of an ordered partition of size 1,2, ..., $|T|$ and decides if they are satisfiable. To check that the execution of a layer is silent, the constraints mentioned in the proof of Proposition 4.3 are transformed using Farkas' lemma (see e.g. [23]) into a version that is satisfiable if and only if all the executions of the layer are silent. Also, the constraints for condition (b) of Definition 4.1 are added.

For StrongConsensus, `Peregrine` initially constructs the constraints for the flow equation for three configurations C_0, C_1, C_2 and vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, with additional constraints to guarantee that C_0 is initial, C_1 and C_2 are terminal, and C_1 and C_2 are consensus of different values. If these constraints are unsatisfiable, the protocol satisfies StrongConsensus. Otherwise, `Peregrine` searches for a U-trap or U-siphon to show that either $C_0 \xrightarrow{\boldsymbol{x}_1} C_1$ or $C_0 \xrightarrow{\boldsymbol{x}_2} C_2$ does not hold. If, say, a U-siphon S is found, then `Peregrine` adds the constraint $C_0(S) > 0$ to the set of initial constraints. This process is repeated until either the constraints are unsatisfiable and StrongConsensus is shown, or all possible U-traps and U-siphons are added, in which case StrongConsensus does not hold. We use this refinement-based approach instead of the coNP approach described in Proposition 4.12, as that could require a quadratic number of variables and constraints, and we generally expect to need a small number of refinement steps.

We evaluated `Peregrine` on a set of benchmarks: the threshold and remainder protocols of [2], the majority protocol of [3], the broadcast protocol of [8] and two versions of the flock of birds[5] protocol from [6, 8]. We checked the parametrized protocols for increasing values of their primary parameter until we reached a timeout. For the threshold and remainder protocols, we set the secondary parameter c to 1 since it has no incidence on the size of the protocol, and since the variation in execution time for different values of c was negligible. Moreover, we assumed that all possible values for a_i were present in the inputs, which represents the worst case.

All experiments were performed on the same machine equipped with an Intel Core i7-4810MQ CPU and 16 GB of RAM. The time limit was set to 1 hour. The results are shown in Table 1. In all cases where we terminated within the time limit, we were able to show membership for WS^3. Generally, showing StrongConsensus took much less time than showing LayeredTermination, except for the flock of birds protocols, where we needed linearly many U-traps.

As an extension, we also tried proving correctness after proving membership in WS^3. For this, we constructed constraints for the existence of an input X and configuration C with $I(X) \xrightarrow{\boldsymbol{x}} C$ and $\varphi(X) \neq O(C)$. We were able to prove correctness for all the protocols in our set of benchmarks. The correctness check was faster than the well-specification check for broadcast, majority, threshold and both flock of birds protocols, and slower for the remainder protocol, where we reached a timeout for $m = 70$.

7 CONCLUSION AND FURTHER WORK

We have presented WS^3, the first class of well-specified population protocols with a membership problem of reasonable complexity (i.e. in DP) and with the full expressiveness of well-specified protocols. Previous work had shown that the membership problem for the general class of well-specified protocols is decidable, but at least EXPSPACE-hard with algorithms of non primitive recursive complexity.

[4] `Peregrine` and benchmarks are available from https://gitlab.lrz.de/i7/peregrine/.

[5] The variant from [8] is referred to as *threshold-n* by its authors.

We have shown that WS^3 is a natural class that contains many standard protocols from the literature, like flock-of-birds, majority, threshold and remainder protocols. We implemented the membership procedure for WS^3 on top of the SMT solver Z3, yielding the first software able to automatically prove well-specification of population protocols for *all* (of the infinitely many) inputs. Previous work could only prove partial correctness of protocols with at most 9 states and 28 transitions, by trying exhaustively a *finite* number of inputs [6, 8, 20, 24]. Our algorithm deals with all inputs and can handle larger protocols with up to 70 states and over 2500 transitions.

Future work will concentrate on three problems: improving the performance of our tool; automatically deciding if a WS^3-protocol computes the predicate described by a given Presburger formula; and the diagnosis problem: when a protocol does not belong to WS^3, delivering an explanation, e.g. a non-terminating fair execution. We think that our constraint-based approach provides an excellent basis for attacking these questions.

ACKNOWLEDGMENTS

We wish to thank David de Frutos Escrig and Pierre Ganty for pointing out some mistakes in an earlier version of this paper.

REFERENCES

[1] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. 2004. Computation in networks of passively mobile finite-state sensors. In *Proc. 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*. 290–299. https://doi.org/10.1145/1011767.1011810

[2] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. 2006. Computation in networks of passively mobile finite-state sensors. *Distributed Computing* 18, 4 (2006), 235–253. https://doi.org/10.1007/s00446-005-0138-3

[3] Dana Angluin, James Aspnes, and David Eisenstat. 2006. Stably computable predicates are semilinear. In *Proc. 25th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. 292–299. https://doi.org/10.1145/1146381.1146425

[4] Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. 2007. The computational power of population protocols. *Distributed Computing* 20, 4 (2007), 279–304. https://doi.org/10.1007/s00446-007-0040-2

[5] Konstantinos Athanasiou, Peizun Liu, and Thomas Wahl. 2016. Unbounded-Thread Program Verification using Thread-State Equations. In *IJCAR (Lecture Notes in Computer Science)*, Vol. 9706. Springer, 516–531.

[6] Ioannis Chatzigiannakis, Othon Michail, and Paul G. Spirakis. 2010. Algorithmic Verification of Population Protocols. In *Proc. 12th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*. 221–235.

[7] Ho-Lin Chen, Rachel Cummings, David Doty, and David Soloveichik. 2014. Speed Faults in Computation by Chemical Reaction Networks. In *DISC (Lecture Notes in Computer Science)*, Vol. 8784. Springer, 16–30.

[8] Julien Clément, Carole Delporte-Gallet, Hugues Fauconnier, and Mihaela Sighireanu. 2011. Guidelines for the Verification of Population Protocols. In *ICDCS*. IEEE Computer Society, 215–224.

[9] Leonardo Mendonça de Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In *TACAS (Lecture Notes in Computer Science)*, Vol. 4963. Springer, 337–340. Z3 is available at https://github.com/Z3Prover/z3.

[10] Yuxin Deng and Jean-François Monin. 2009. Verifying Self-stabilizing Population Protocols with Coq. In *Proc. 3rd IEEE International Symposium on Theoretical Aspects of Software Engineering (TASE)*. 201–208. https://doi.org/10.1109/TASE.2009.9

[11] Jörg Desel and Javier Esparza. 1995. *Free choice Petri nets*. Cambridge University Press.

[12] Shlomi Dolev, Mohamed G. Gouda, and Marco Schneider. 1999. Memory requirements for silent stabilization. *Acta Informatica* 36, 6 (1999), 447–462.

[13] Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar. 2015. Verification of Population Protocols. In *Proc. 26th International Conference on Concurrency Theory (CONCUR)*. 470–482.

[14] Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar. 2016. Model Checking Population Protocols. In *FSTTCS (LIPIcs)*, Vol. 65. 27:1–27:14.

[15] Javier Esparza, Ruslán Ledesma-Garza, Rupak Majumdar, Philipp J. Meyer, and Filip Niksic. 2014. An SMT-Based Approach to Coverability Analysis. In *CAV (Lecture Notes in Computer Science)*, Vol. 8559. Springer, 603–619.

[16] Javier Esparza and Stephan Melzer. 2000. Verification of Safety Properties Using Integer Programming: Beyond the State Equation. *Formal Methods in System Design* 16, 2 (2000), 159–189.

[17] Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. 2011. Mediated population protocols. *Theor. Comput. Sci.* 412, 22 (2011), 2434–2450.

[18] Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. 2012. Terminating Population Protocols via Some Minimal Global Knowledge Assumptions. In *Stabilization, Safety, and Security of Distributed Systems - 14th International Symposium, SSS 2012, Toronto, Canada, October 1-4, 2012. Proceedings*. 77–89.

[19] Saket Navlakha and Ziv Bar-Joseph. 2015. Distributed information processing in biological and computational systems. *Commun. ACM* 58, 1 (2015), 94–102. https://doi.org/10.1145/2678280

[20] Jun Pang, Zhengqin Luo, and Yuxin Deng. 2008. On Automatic Verification of Self-Stabilizing Population Protocols. In *Proc. 2nd IEEE/IFIP International Symposium on Theoretical Aspects of Software Engineering (TASE)*. 185–192. https://doi.org/10.1109/TASE.2008.8

[21] Christos H. Papadimitriou. 2007. *Computational complexity*. Academic Internet Publ.

[22] Sylvain Schmitz. 2016. The complexity of reachability in vector addition systems. *SIGLOG News* 3, 1 (2016), 4–21.

[23] Alexander Schrijver. 1986. *Theory of Linear and Integer Programming*. John Wiley & Sons.

[24] Jun Sun, Yang Liu, Jin Song Dong, and Jun Pang. 2009. PAT: Towards Flexible Verification under Fairness. In *Proc. 21st International Conference on Computer Aided Verification (CAV)*. 709–714. https://doi.org/10.1007/978-3-642-02658-4_59