

Advances in Parameterized Verification of Population Protocols

Javier Esparza^(✉)

Technische Universität München, Munich, Germany
esparza@in.tum.de

Abstract. Population protocols (Angluin et al. *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, much less is known about how to verify their correctness: Given a population protocol, is it well specified? Given a population protocol and a predicate, does the protocol compute the predicate? Given a well-specified protocol, can we automatically obtain a symbolic representation of the predicate it computes? We survey our recent work on this problem.

Population protocols [3,4] are a model of distributed computation by anonymous, identical finite-state agents. While they were initially introduced to model networks of passively mobile sensors [3,4], they capture the essence of distributed computation in diverse areas such as trust propagation [16] and chemical reaction networks [27], a popular model for theoretical chemistry.

The Model. A protocol has a finite set of states Q , and a set of transitions of the form $(q, q') \mapsto (r, r')$, where $q, q', r, r' \in Q$. At each computation step of a population protocol a scheduler selects two agents, observes their current states, say q_1 and q_2 , selects a rule with (q_1, q_2) on the left-hand side, say $(q_1, q_2) \mapsto (q_3, q_4)$, and updates the states of the agents to q_3 and q_4 ¹. Since agents are anonymous and identical, it is irrelevant which agent moves to which state. Further, the global state of a protocol is completely determined by the number of agents at each state, i.e., by a mapping $Q \rightarrow \mathbb{N}$. Such a mapping is called a *configuration*.

Without loss of generality, one can assume that for every two states q_1, q_2 the protocol has at least one transition of the form $(q_1, q_2) \mapsto (q_3, q_4)$ for some

¹ Since I am often asked this question, let me mention that extensions to k agents for some $k > 2$ can be easily simulated by protocols with $k = 2$.

states q_3, q_4 (perhaps the transition $(q_1, q_2) \mapsto (q_1, q_2)$ that does not change the current configuration). Then the protocol cannot block, and all computations have an infinite number of steps. Schedulers are requested to satisfy a fairness condition: if a configuration C appears infinitely often in the execution, then every transition enabled at C is taken infinitely often in the execution. This condition approximates the behavior of a probabilistic scheduler that selects agents at random, either uniformly or according to some fixed probability distribution.

Computing with Population Protocols. Population protocols are machines for the distributed computation of predicates $\mathbb{N}^n \rightarrow \{0, 1\}$. In the simplest version, a predicate $P(x_1, \dots, x_n)$ of arity n is computed by a protocol with *initial states* q_1, \dots, q_n . Given $k_1, \dots, k_n \in \mathbb{N}^n$, the value $P(k_1, \dots, k_n)$ is computed by preparing the initial configuration C_0 given by $C_0(q_i) = k_i$ for every $1 \leq i \leq n$ and $C_0(q) = 0$ for every non-initial state q , and letting the protocol run. Intuitively, $P(k_1, \dots, k_n) = b$ holds if in *every* fair execution starting at C_0 , all agents eventually agree to b and do not ever change their mind—so, loosely speaking, population protocols compute by reaching consensus. But what does it mean that the agents agree to b ? For this, we define a mapping that assigns to every state $q \in Q$ an *opinion* $O(q) \in \{0, 1\}$, with the intended meaning that agents at state q have opinion $O(q)$. Therefore, we have $P(k_1, \dots, k_n) = b$ if every fair computation $C_0 C_1 C_2 \dots$ reaches a configuration C_i such that for every $j \geq i$ and for every state q satisfying $C_j(q) > 0$, the state q satisfies $O(q) = b$.

Observe that not every protocol computes a predicate. It is easy to construct protocols such that for some initial configuration C_0 some fair computation never agrees to a value, or for which two fair computations agree to two different values. We call such protocols *ill specified*, and say that a protocol is *well specified* if it is not ill specified.

Computing Power. Most of the work on population protocols carried out by the distributed computing community has concentrated on characterizing the predicates computable by well-specified protocols. In particular, Angluin et al. [3, 4] gave explicit well-specified protocols to compute every predicate definable in Presburger arithmetic, the first-order theory of addition, 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]. There is also extensive work on the computational power and runtime of protocols. For example, protocols for any Presburger-definable predicate satisfying a certain runtime guarantee are shown in [6], while other papers have designed particularly efficient protocols for specific predicates (see e.g. [2]).

Verification Problems for Population Protocols. The verification community is interested in other 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? Given a well-specified protocol, can we automatically obtain a symbolic representation of the predicate it computes? It is also interested in properties concerning the executions of a protocol, seen not as a device for the distributed computation of a predicate, but as a concurrent system. In particular, we are interested in checking if a protocol satisfies a given temporal logic specification expressed in, for example, Linear Temporal Logic, one of the standard specification languages. All these problems are challenging because of their *parameterized* nature. The semantics of a population protocol is an *infinite* family of finite-state transition systems, one for each possible input. Therefore, any of the problems above is actually a question of the form: Do all members of the family satisfy a certain property?

Deciding the property for one single member requires to inspect only one finite transition system, and can be done automatically using a model checker. This is the approach that verification researchers investigated first [11, 12, 28, 31], but it only proves the correctness of a protocol for a finite number of inputs. Alternatively, one can also formalize a proof of well specification in a theorem prover [15], but this approach is not automatic: a human prover must first come up with a proof for each particular protocol. Can we go beyond this? The verification community has studied parameterized verification problems for a long time (see [9, 17]), and so at first sight one could feel optimistic. Unfortunately, the techniques developed in this area cannot be applied to the analysis of population protocols. It is worth to spend some lines explaining why.

Classical Parameterized Verification Techniques Do Not Work. Many simple parameterized models of computation have Turing power, and so all interesting analysis problems are undecidable for them [8]. For example, given a Turing machine M and an input x , we can easily construct a little finite-state program that simulates a tape cell. The program has a boolean variable indicating whether the head is on the cell or not, a variable storing the current tape symbol, and a third variable storing the current control state when the head is on the cell. A agent running the program communicates with its left and right neighbors by message passing. If M accepts x , then it does so using a finite number N of tape cells. Therefore, the instance of the system containing N agents eventually reaches a configuration in which the value of the control-state variable of a agent is a final state of M . On the contrary, if M does not accept x , then no instance, however large, ever reaches such a configuration. So the reachability problem for parameterized programs is undecidable. However, this proof sketch contains the sentence “the program communicates with its left and right neighbors”. This is achieved by giving agents an *identity*, typically a number in the range $[1..N]$. This number appears as a parameter i in the code, and so it is not the case that all agents execute *exactly* the same code, but the code where the parameter is instantiated with the agent identity. Since in population protocols agents have no identity, the argument above does not apply. So, are parameterized verification questions for systems consisting of completely

identical agents decidable, and, if this is the case, can these results be applied to population protocols?

The answer to the first question is positive but, unfortunately, the second is not. Many questions about systems of identical agents can be proved decidable using the theory of well-quasi orders [1, 21], which have become the standard mathematical tool for parameterized verification. Loosely speaking, the theory can be applied when the specification to be checked satisfies a monotonicity property: if the instance consisting of N agents satisfies the property, then for every K the instance with $N + K$ agents will also satisfy it. However, the central verification questions for population protocols, like well-specification, are not monotonic. This is due to the notion of fairness inherently present in population protocols. While an execution of the instance of the protocol running with N agents is also an execution of the instance with $N + K$ agents (the K additional agents just do not participate in the execution), the same is not true for *fair* executions, because the new K processes cannot be indefinitely left out. This problem can be overcome for some simple notions of fairness, but not for the particularly demanding one required by population protocols.

Parameterized Verification of Population Protocols. In 2014 I started an initiative to attack the parameterized verification of population protocols, which has been (and continues to be) quite successful. Together with my colleagues and students Michael Blondin, Pierre Ganty, Stefan Jaax, Jérôme Leroux, Rupak Majumdar, and Philipp J. Meyer, we have

- obtained decidability and undecidability results that establish the limits of what is algorithmically achievable, and several complexity bounds showing that many questions have very high complexity; and
- initiated the study of population protocols that are *good for verification*: classes of protocols with the same expressive power as the general class, but with more tractable verification problems.

In the rest of this note I summarize our results.

Decidability of Central Verification Questions. In [18], a paper published at CONCUR 2015, we showed that the three central verification problems for population protocols are all decidable:

- (a) Given a population protocol, is it well specified?
- (b) Given a population protocol and a Presburger predicate (represented by a Presburger formula), does the protocol compute the predicate?
- (c) Given a well-specified protocol, can we automatically obtain a symbolic representation of the predicate it computes?

We call (b) the *fitting problem* (whether a given protocol *fits* the specification), and (c) the *tailor problem* (*tailoring* a protocol that fits the specification). The proofs relied on breakthrough results by Jérôme Leroux on the theory of Petri nets, a model very close to population protocols [22–25]. Shortly after submitting

this paper we were able to substantially improve our results: We showed that questions (a) and (b) are recursively equivalent to the reachability problem for Petri nets, eliminating the need for the most advanced and complicated of Leroux’s results (the ones of [22]). These improvements are contained in the journal version of [18], recently published in *Acta Informatica* [20] (I do not recommend to read [18] any more!).

The reductions from the reachability problem show that there is little hope of finding reasonably efficient algorithms for arbitrary protocols, even small ones: The reachability problem is known to be **EXSPACE**-hard, and all known algorithms for it have non-primitive recursive complexity (actually, the first explicit upper bound for the problem was only obtained very recently, see [29] for a recent survey). In particular, there are no stable implementations of any of these algorithms, and they are considered impractical for nearly all applications.

To solve problem (c), we introduce a notion of certificate of well-specification for 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 problem (c) 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. Since this certificate approach only provides two semi-decision algorithms for problem (c), we know even less concerning its complexity, no upper bound has been given yet.

Decidability of Model Checking Problems. Population protocols have been used to model systems beyond their initial motivation in distributed computing. In particular, they can also model trust propagation [16], evolutionary dynamics [26], or chemical reaction systems [27,30]. These systems do not always aim at the computation of predicates, or, if they do, they do not compute them in the way defined by Angluin et al. [3]. With more diverse applications of population protocols comes also new properties one would like to reason about. For instance, Delporte-Gallet et al. [14] studied privacy in population protocols. They proved (by hand) different properties of specific protocols, like “the system can reach a good configuration without any interaction involving a distinguished agent p_0 ”. For these reasons, in [19], published at FSTTCS 2016, we studied the general model checking problem for population protocols against probabilistic linear-time (LTL) specifications. We use the probabilistic semantics in which the scheduler selecting the agents at each step proceeds stochastically. We assume that each transition carries a label that can be observed whenever it occurs. This assigns to each computation an infinite word over the set of labels. We can then speak of the probability of the set of computations satisfying a given formula ϕ of LTL. We show that the qualitative model checking problem (i.e., deciding if ϕ holds with probability 1) is decidable, while the quantitative problem (deciding if the probability that ϕ holds exceeds a given probability) is undecidable.

Efficiently Verifiable Protocols. The theoretical results of [18,20] on the well-specification problem can be reformulated as: the membership problem for the class WS of well-specified protocols is decidable but at least as hard as the reachability problem for Petri nets. This motivates the search for a class of protocols properly contained in WS and satisfying the following three properties:

- (a) *No loss of expressive power:* the class should compute all Presburger-definable predicates.
- (b) *Naturality:* 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 obviously satisfies (a) and (b), but not (c). In our most recent work, currently under submission and available online in [10], 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 $L_1 \in NP$ and $L_2 \in coNP$. This is a dramatic improvement with respect to the $EXSPACE$ -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 allows us to implement our decision procedure on top of the constraint solver Z3 [13], yielding the first software able to automatically prove well-specification for *all* initial configurations. We have tested our implementation on the families of protocols studied in [11,12,28,31]. These papers prove correctness for some inputs of protocols with up to 9 states and 28 transitions. Our approach proves correctness 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.

Acknowledgments. I want to thank my colleagues Pierre Ganty, Jérôme Leroux, and Rupak Majumdar, my postdoc Michael Blondin, and my PhD students Stefan Jaax and Philipp J. Meyer for agreeing to put their brains to work on these questions. Thank you also to Dejavuth Suwimonterabuth for implementing the little simulator I use in my talks on population protocols.

References

1. Abdulla, P.A., Cerans, K., Jonsson, B., Tsay, Y.-K.: General decidability theorems for infinite-state systems. In: Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science, LICS 1996, pp. 313–321. IEEE Computer Society (1996)
2. Alistarh, D., Gelashvili, R., Vojnovic, M.: Fast and exact majority in population protocols. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2015, pp. 47–56. ACM (2015)

3. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. In: PODC 2004, pp. 290–299. ACM (2004)
4. 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)
5. Angluin, D., Aspnes, J., Eisenstat, D.: Stably computable predicates are semilinear. In: PODC 2006, 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. Bloem, R., Jacobs, S., Khalimov, A., Konnov, I., Rubin, S., Veith, H., Widder, J.: Decidability of parameterized verification. In: *Synthesis Lectures on Distributed Computing Theory*. Morgan & Claypool Publishers (2015)
10. Blondin, M., Esparza, J., Jaax, S., Meyer, P.: Towards efficient verification of population protocols. CoRR, abs/1703.04367 (2017)
11. Chatzigiannakis, I., Michail, O., Spirakis, P.G.: Algorithmic Verification of Population Protocols. In: Dolev, S., Cobb, J., Fischer, M., Yung, M. (eds.) SSS 2010. LNCS, vol. 6366, pp. 221–235. Springer, Heidelberg (2010). doi:[10.1007/978-3-642-16023-3_19](https://doi.org/10.1007/978-3-642-16023-3_19)
12. Clement, J., Delporte-Gallet, C., Fauconnier, H., Sighireanu, M.: Guidelines for the verification of population protocols. In: ICDCS 2011, pp. 215–224 (2011)
13. Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008). doi:[10.1007/978-3-540-78800-3_24](https://doi.org/10.1007/978-3-540-78800-3_24)
14. Delporte-Gallet, C., Fauconnier, H., Guerraoui, R., Ruppert, E.: Secretive birds: privacy in population protocols. In: Tovar, E., Tsigas, P., Fouchal, H. (eds.) OPODIS 2007. LNCS, vol. 4878, pp. 329–342. Springer, Heidelberg (2007). doi:[10.1007/978-3-540-77096-1_24](https://doi.org/10.1007/978-3-540-77096-1_24)
15. Deng, Y., Monin, J.-F.: Verifying self-stabilizing population protocols with Coq. In: TASE 2009, pp. 201–208. IEEE Computer Society (2009)
16. 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)
17. Esparza, J.: Keeping a crowd safe: on the complexity of parameterized verification (invited talk). In: STACS. LIPIcs, vol. 25, pp. 1–10. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik (2014). Corrected version available in CoRR, abs/1405.1841
18. Esparza, J., Ganty, P., Leroux, J., Majumdar, R.: Verification of population protocols. In: *Proceedings of the 26th International Conference on Concurrency Theory (CONCUR)*, pp. 470–482 (2015)
19. Esparza, J., Ganty, P., Leroux, J., Majumdar, R.: Model checking population protocols. In: FSTTCS. LIPIcs, vol. 65, pp. 27:1–27:14 (2016)
20. Esparza, J., Ganty, P., Leroux, J., Majumdar, R.: Verification of population protocols. *Acta Inf.* **54**(2), 191–215 (2017)
21. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! *Theor. Comput. Sci.* **256**(1–2), 63–92 (2001)
22. Leroux, J.: The general vector addition system reachability problem by Presburger inductive invariants. In: LICS 2009, pp. 4–13. IEEE Computer Society (2009)

23. Leroux, J.: Vector addition system reversible reachability problem. In: Katoen, J.-P., König, B. (eds.) CONCUR 2011. LNCS, vol. 6901, pp. 327–341. Springer, Heidelberg (2011). doi:[10.1007/978-3-642-23217-6_22](https://doi.org/10.1007/978-3-642-23217-6_22)
24. Leroux, J.: Presburger vector addition systems. In: LICS 2013, pp. 23–32. IEEE Computer Society (2013)
25. Leroux, J.: Vector addition system reversible reachability problem. Log. Methods Comput. Sci. **9**(1) (2013)
26. Moran, P.A.P.: Random processes in genetics. Math. Proc. Camb. Philos. Soc. **54**(1), 60–71 (1958)
27. Navlakha, S., Bar-Joseph, Z.: Distributed information processing in biological and computational systems. Commun. ACM **58**(1), 94–102 (2014)
28. Pang, J., Luo, Z., Deng, Y.: On automatic verification of self-stabilizing population protocols. In: Proceedings of the 2nd IEEE/IFIP International Symposium on Theoretical Aspects of Software Engineering (TASE), pp. 185–192 (2008)
29. Schmitz, S.: The complexity of reachability in vector addition systems. SIGLOG News **3**(1), 4–21 (2016)
30. Soloveichik, D., Cook, M., Winfree, E., Bruck, J.: Computation with finite stochastic chemical reaction networks. Nat. Comput. **7**(4), 615–633 (2008)
31. Sun, J., Liu, Y., Dong, J.S., Pang, J.: PAT: towards flexible verification under fairness. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 709–714. Springer, Heidelberg (2009). doi:[10.1007/978-3-642-02658-4_59](https://doi.org/10.1007/978-3-642-02658-4_59)

Computer Science – Theory and Applications
12th International Computer Science Symposium in
Russia, CSR 2017, Kazan, Russia, June 8-12, 2017,
Proceedings
Weil, P. (Ed.)
2017, X, 337 p. 30 illus., Softcover
ISBN: 978-3-319-58746-2