# An Experimental Scenario on Algorithmic Verification of Population Protocols

August 25, 2009

## 1 Introduction

*Population Protocols* (PP) [1, 2] are primitive-form distributed algorithms appropriate for sensor systems whose nodes are equivalent to communicating finite-state machines. The PP model was extended in [6] to form the *Mediated Population Protocol* (MPP) model, which is strictly stronger in terms of computational power. MPP's ability to decide graph languages was partially studied in [4]. Finally, [9] proposed the *Community Protocol* model, in which agents are equipped with unique IDs. For the interested reader, [3, 8] constitute nice introductions to the area.

We believe that for applying our protocols to real-critical application scenarios some form of computer-aided *verification* is necessary. Even if a protocol is followed by a formal proof of correctness it would be safer to verify its code before loading it to the real sensor nodes.

## 2 Research Targets

It seems that the easiest (but not easy) place to start the investigation of verification is the *basic* population protocol model [1], in which the *communication graph* is always directed and complete (that is, all ordered pairs of nodes are allowed communication). In this model symmetry allows a configuration to be safely represented as a $|Q|$-vector, where $Q$ is the set of protocol states, whose components sum up to $n$ (the size of the population). That is, each component of the vector $C$ corresponds to a state $q$ from $Q$ and expresses the number of agents that are in state $q$ under configuration $C$.

But talking about verification of a protocol $\mathcal{A}$ is pointless in lack of the specifications of $\mathcal{A}$. Imagine for example the OR protocol (see e.g. [5, 7]) in which the transition function is defined as follows: "If at least one of the interacting agents is in state 1 then both go to 1, otherwise they both remain to 0". The specifications of the OR protocol can in fact be *efficiently decided*: "If at least one agent in the population gets input 1 then all agents output the value 1, otherwise all agents output 0". Clearly, there exists a polynomial-time computable function $S$ that given an input $(c, \{0, 1\}, n)$ can tell whether $c$ is a legal input assignment for the OR protocol (by checking whether it is a 2-vector whose components sum up to $n$), and if it is return 1 or 0 by counting the number of 1s, otherwise answer "no". If the specifications of a protocol cannot be efficiently decided, then a little thought reveals that the resulting input is in general huge rendering the verification process meaningless.

We can now define the computational problem B-VERIFICATION ('B' is from "basic") as follows: "Given a population protocol $\mathcal{A}$ for the basic model, a polynomial-time computable function $S$ representing the specifications of $\mathcal{A}$, and an integer $n \geq 2$ determine whether $\mathcal{A}$ conforms to its specifications for the complete digraph on $n$ nodes (agents)".

Our reasearch targets can be summerized as follows.

- Algorithmic design of a small number of deterministic verifiers (one or two) for the B-VERIFICATION problem or its special cases (see e.g. in the next section the *binary* special case).
- Implement the designed verifiers.
- Experiment with the verifiers: Evaluate experimentally their performance/correctness.
- Exploit the experimental results as feedback for further algorithmic design.

## 3 An Algorithmic Starting Point

As a starting point we propose two verifiers, one for the B-VERIFICATION problem restricted to predicates and one for its binary special case.

The *transition graph* $D(\mathcal{A}, \mathcal{P})$ (see [2, 7]) of a population protocol $\mathcal{A}$ running on population $\mathcal{P}$ is a digraph with possible self-loops whose nodes are all possible population configurations and $(C, C') \in E(D)$ (we use $D$ instead of $D(\mathcal{A}, \mathcal{P})$ for simplicity) iff $C$ *can go in one step to* $C'$ (see for example [1, 6] for the definition of the "can go in one step to" relation).

If $\mathcal{A}$ is designated to compute a predicate (that is, the output alphabet is $\{0, 1\}$) then there exists an obvious algorithm based on exhaustive search of the transition graph: "For every legal input $x$ compute the initial configuration corresponding to it (denoted $I(x)$) and check whether all directed paths initiating from $I(x)$ lead to a *final strongly connected component* (see [2]) of $D$ in which the output of all configuration-nodes is $S(x)$".

The above algorithm can be made easily pseudo-polynomial in the *binary* special case where the input alphabet and the set of states of $\mathcal{A}$ are binary (and so is the output alphabet when focusing on predicates) and we do not know yet if there is a polynomial-time algorithm even for this simple case. Moreover, what happens in the general case is an even bigger mystery. It could very well be that B-VERIFICATION is coNP-complete (at first sight, it seems easier to provide a succinct "no" witness, that is, a failing computation counterexample, than a witness showing that all computational paths are correct; however, this remains a conjecture and as happens with every conjecture "we do not know" [Jack Edmonds, 1966]).

Our distant target is to solve the B-VERIFICATION problem. We believe that this will give a first insight into the complexity and algorithmic techniques of solving the verification problem on the many now existing extensions of the basic population protocol model.

## 4 Methodology

The experimental approach comes immediately into play. Implementing and experimentally evaluating the first obvious algorithms that we have for some special cases of B-VERIFICATION will provide valuable feedback to the theoretical work to be done.

Specifically, we will begin by formally designing the verifiers alluded to above and possibly others that we have not yet come up with. Then we will implement those verifiers possibly in some procedural language like C (or C++). There are many implementation questions here to be answered. For example, we have to agree in what form/respresentation the specification algorithm $S$ will be provided as part of the input and what constitutes an easily manageable protocol representation. Should it be a *transition matrix* (e.g. some bidimensional array)? Moreover, we have to devise and implement population protocols that will be provided as input to our verifiers. In [5] the simulation tool *PPSIM* was developed for experimental verification of protocols. Some protocols have already been implemented there. But possibly more will need to implemented, since some of them where based on probabilistic arguments that do not seem to fit well in the B-VERIFICATION problem (stability is not required there to be reached always, but instead with high probability). Our verifiers will be tested on a variety of different inputs and their performance will be recorded and plotted in order to obtain useful information about their functionality. For example, we would be very happy if we could design an exponential in the worst-case algorithm for the general B-VERIFICATION problem that would be shown by extensive simulations to perform well in practice. Finally, we want to investigate how we could possibly exploit PPSIM to evaluate the applicability of protocols. This means that protocols proven incorrect by the verifiers could be supplied as input to PPSIM, which in turn would provide us with statistical results about several executions of the protocols. In this way we may salvage many erroneous protocols yet satisfactory for "not so critical" applications.

## References

[1] D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In *23rd Annual ACM Sympsium on Principles of Distributed Computing* (*PODC*), pages 290-299, New York, NY, USA, 2004. ACM.

[2] D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4): 235-253, 2006.

[3] J. Aspnes and E. Ruppert. An introduction to population protocols. *Bulletin of the European Association for Theoretical Computer Science*, 93:98-117, October 2007. Columns: *Distributed Computing*, Editor: M. Mavronicolas.

[4] I. Chatzigiannakis, O. Michail, and P. G. Spirakis. Decidable Graph Languages by Mediated Population Protocols. To appear in *Distributed Computing*, *23nd International Symposium*, *DISC*, Elche, Spain, Sept. 2009. (Also FRONTS Technical Report FRONTS-TR-2009-16, `http://fronts.cti.gr/aigaion/?TR=80`)

[5] I. Chatzigiannakis, O. Michail, and P. G. Spirakis. Experimental verification and performance study of extremely large sized population protocols. FRONTS Technical Report FRONTS-TR-2009-3, `http://fronts.cti.gr/aigaion/?TR=61`, Jan. 2009.

[6] I. Chatzigiannakis, O. Michail, and P. G. Spirakis. Mediated Population Protocols. In *36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 363-374, Rhodes, Greece, 2009.

[7] I. Chatzigiannakis, S. Dolev, S. Fekete, O. Michail, and P. G. Spirakis. Not All Fair Probabilistic Schedulers are Equivalent. FRONTS Technical Report FRONTS-TR-2009-29, `http://fronts.cti.gr/aigaion/?TR=93`, Jun. 2009.

[8] I. Chatzigiannakis, O. Michail, and P. G. Spirakis. Recent Advances in Population Protocols. FRONTS Technical Report FRONTS-TR-2009-21, `http://fronts.cti.gr/aigaion/?TR=85`, June 2009. To appear in *34th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, August 24-28, 2009, Novy Smokovec, High Tatras, Slovakia.

[9] R. Guerraoui and E. Ruppert. Names Trump Malice: Tiny Mobile Agents Can Tolerate Byzantine Failures. In *36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 484-495, Rhodes, Greece, 2009.