

Mode d'emploi pour la vérification des protocoles de population[†]

Julien Clement, Carole Delporte-Gallet, Hugues Fauconnier,
and Mihaela Sighireanu

¹*LIAFA, Université Paris Diderot and CNRS, France*
{jclément, cd, hf, sighirea}@liafa.jussieu.fr

Cet article est consacré à la vérification basée sur le modèle (*model-checking*) du modèle des protocoles de population introduit par Angluin et al. [AAD⁺06]. Ces deux dernières années, la vérification des protocoles de population par *model-checking* a fait l'objet de nombreuses études et de nouveaux outils ont été proposés ([PLD08, BFPvdP08, LPSZ09, SLDP09, CMS10]). Nous montrons dans cet article que, dans une certaine mesure, les outils classiques de *model-checking* tels que Spin et Prism peuvent être utilisés pour effectuer les vérifications. Pour cela, nous appliquons la technique d'abstraction par comptage [GS92] pour obtenir des modèles abstraits de protocoles de population qui peuvent être vérifiés efficacement par les outils de *model-checking* existants.

Le problème essentiel pour la complexité de la vérification des protocoles de population concerne la condition d'équité forte. Cette dernière ne peut être utilisée directement avec Spin même pour des exemples de relativement petite taille. Nous montrons qu'on peut cependant remplacer dans de nombreux cas cette équité par une équité faible efficacement vérifiable en Spin. Plus notable encore, nous montrons que la vérification avec la condition d'équité forte est équivalente à un problème de vérification d'un modèle probabiliste. Ainsi, le *model-checker* probabiliste Prism s'avère être un outil de vérification adapté aux protocoles de population.

Keywords: protocole de population, abstraction par comptage, model-checking, Spin, Prism

1 Introduction

Population protocols [AAD⁺06] are a formal model for sensor networks. Since such networks should become a part of the every day life, it is important to ensure their correctness. This task is now possible in a fully automatic way due to the important progress of verification techniques. Proposed in the early eighties, the model-checking technique [EC80, QS81] allows to check complex correctness properties for models with finite number of states. The system to be verified is formalized in some high level formal language, e.g., CSP or Petri Nets, to obtain a model. The requirements of the system are specified using some logical language, e.g. LTL [Pnu77], to obtain a property. Model-checking techniques explore exhaustively the model in order to check that each state of this model satisfies the given property (a reference for model-checking techniques is [CGP99]). High performance model-checkers, e.g., Spin [Hol03], are now available. They are able to deal with systems of more than 10^{20} states. Moreover, the model-checking techniques have been extended to deal with more complex finite models like the probabilistic ones, e.g., in the Prism [KNP04] model-checker.

Unfortunately, some models of realistic systems have a number of states which exceeds the capacity of finite model-checkers. Thus, the research have focus on designing abstraction techniques to reduce the model-checking problem of huge systems to the model-checking of smaller systems. One of the first abstraction techniques proposed is the counting abstraction [GS92]. It reduces systems with many identical processes running in parallel to a system which keeps track only of the number of processes which are in some particular state. This abstraction has shown its efficiency for the model-checking of safety properties in cache coherence protocols.

[†]This work has been accepted for publication at ICDCS 2011 [CDGFS11].

In this work, we apply the counting abstraction technique and finite model-checking tools to verify population protocols. Due to lack of space, we refer the reader to the long version [CDGFS11] for formal definitions, proofs and experimental results.

The model of population protocols (PP) [AAD⁺06] involves individual agents with a very simple behavior, which can be seen as a finite state machine. An important property is the uniformity of the protocol, i.e., the fact that the protocol description is independent of the number of agents (called also the population size) or their identity. When two agents come into range of each other (“meet”), they can exchange information. The agents are anonymous and move in an asynchronous way. At the beginning of the protocol, each agent receives a piece of input. The goal of the protocol is to stabilize each agent into a state in which it outputs the value to be computed by the protocol. Thus, the specification of a population protocol includes the “correct stabilization” (CS) property which requires that each agent stabilizes its computation to an output value which corresponds to a function on the input assigned to agents.

Our contribution is to highlight the use of the existing model-checking techniques to check the CS property for protocols with finite fixed sizes. For this, we use the counting abstraction to reduce the PP model to a vector addition system model, which is a model computationally equivalent to classical place/transition Petri Nets. We show that this abstraction allows the verification of the CS property in a more efficient manner. We also give sufficient conditions to check the CS property under a fairness constraint weaker than the one required in [AAD⁺06]. These theoretical results allow us to verify, using the Spin tool, a benchmark of PP models for population size greater than 10^3 . In a second approach, we highlight that the fairness of the PP model can be exactly captured for the CS property in a probabilistic model where probabilities strictly greater than some $\varepsilon > 0$ are assigned to transitions. Then, we show that Prism can deal very efficiently with the model-checking of this property on the abstract model of PP.

2 Population Protocol Model and its Abstraction

In this section, we briefly introduce the population protocol model. More details are available in [AAD⁺06]. For a population of k agents, $\Pi = \{\pi_1, \dots, \pi_k\}$ denotes the set of agents.

Definition 1 A *population protocol* (PP) is specified as a six-tuple $\mathcal{P} = (Q, X, Y, \iota, \omega, \delta)$ which contains a finite set Q of possible agent states, an input assignment $\iota : X \rightarrow Q$ mapping the finite set X of inputs to the agents’ state, an output assignment $\omega : Q \rightarrow Y$ mapping the agents’ states to a finite set of outputs Y , and a transition relation $\delta \subseteq Q^4$.

In this definition, the initial state of an agent is fixed (using ι) by an input value in X received by the agent. In each state q , an agent outputs a value in Y given by the mapping ω . The interaction between agents follows the rules described by the relation δ : if two agents in states q_1 and q_2 meet each other, they change into states q_3 resp. q_4 , where $(q_1, q_2, q_3, q_4) \in \delta$. We also use notation $q_1 || q_2 \rightarrow q_3 || q_4$ for the elements of δ .

The semantics of a PP model is given by a labelled transition system as follows. A configuration s of the protocol is a mapping $\Pi \rightarrow Q$ specifying the state of each agent. A transition between configurations $s \xrightarrow{\ell} s'$ takes place if $\exists \pi_i, \pi_j \in \Pi$ s.t. $t = s(\pi_i) || s(\pi_j) \rightarrow s'(\pi_i) || s'(\pi_j) \in \delta$, $\ell = (\{i, j\}, t)$ and $\forall \pi \in \Pi \setminus \{\pi_i, \pi_j\}. s(\pi) = s'(\pi)$. From an initial configuration s_0 , a computation C from s_0 is $s_0, \ell_0, s_1, \ell_1, \dots$ with $s_i \xrightarrow{\ell_i} s_{i+1}$.

One of the characteristics of the PP model is that the order in which pairs of agents interact is unpredictable. In order to model this aspect, one may imagine the presence of a scheduler which is scheduling the interactions. This scheduler may force two agents to never interact. In the presence of such scheduler, the PP has no chance to compute its goal. Thus, Angluin et al. [AAD⁺06] require that the scheduler allows only strong globally fair computations.

Definition 2 A computation C is *strong globally fair* (GF) iff for every s and s' such that $s \rightarrow s'$, if $s = s_i$ for infinitely many i in C , then $s_j = s'$ for infinitely many j in C i.e., $(s \rightarrow s' \wedge \square \diamond s) \Rightarrow \square \diamond s'$.

Under the GF requirement for the scheduler, a protocol of size k stably computes a function $f : X^k \rightarrow Y$ if, for every input $\vec{x} \in X^k$, every GF computation starting in $\iota(\vec{x})$ stably outputs $f(\vec{x})$, i.e., $\iota(\vec{x}) \wedge \text{GF} \Rightarrow \square \diamond (\omega(s) = f(\vec{x}))$. The important result of Angluin et al. [AAER07] is that a predicate (i.e., function with

co-domain $Y = \{0, 1\}$) is stably computable by the PP model iff it can be defined as a first-order formula in Presburger logic.

To scale up the verification task, we use as abstract model for the PP model the *vector addition system* model [KM69].

Definition 3 A *vector addition system* (VAS) is a pair $\mathcal{A} = (\mathbf{c}, D)$ where $\mathbf{c} = \langle c_1, \dots, c_n \rangle$ is a finite vector of integer variables called counters, and $D \subseteq \mathbb{N}^n \times \mathbb{Z}^n$ is a finite set of guarded translations (μ, τ) s.t. $\mu + \tau \geq \mathbf{0}$.

To obtain a VAS model from a PP model, we apply the *counting abstraction* [GS92]. Intuitively, a configuration s is abstracted by a vector \mathbf{c} indexed by Q such that $\mathbf{c}[q]$ is the number of agents in state q in s . This abstraction is possible due to the uniformity of the protocol : the behavior of an agent depends only on its state and not on its index. We prove (see [CDGFS11]) that the counting abstraction preserves the CS property and thus the verification of the CS property on the PP model can be transferred into a verification of the same property in the abstract PP model.

3 Verification under Weak Fairness

The verification of the CS property on the abstract model of PP for finite population sizes can be done using existing finite state model-checkers. We consider the Spin tool [Hol03] which is one of the most efficient model-checkers for finite-state systems. To check a property on a model, Spin translates the property into a Büchi automaton which size is exponential in the size of the formula. This fact prevents us to encode the GF constraint in the formula in order to select only GF computations. Fortunately, Spin provides an algorithm which verifies a property on the set of “weakly fair” computations of a model. The selection of these computations is done on-the-fly and very efficiently.

Definition 4 A computation C is *weakly fair* (WF) iff for every ℓ , if it exists j and s'_i with $s_i \xrightarrow{\ell} s'_i \in \delta$ for any $i \geq j$, then $s_k \xrightarrow{\ell} s_{k+1}$ for infinitely many $k > j$ in C , i.e., $\diamond \square \ell$ is enabled $\Rightarrow \square \diamond \ell$ is fired.

Thus, we study the conditions on which the WF constraint can be used for the verification of the abstract PP models. The problem with the use of the WF constrain is that we can obtain false negatives since the set of WF computations includes the set of GF computations. We provide two sufficient conditions on the transition system of the abstract PP model to eliminate the false negatives. These conditions constrain the strongly connected components (SCC) of the induced transition system.

Theorem 1 (Correction of verification under WF) Let \mathcal{P} be a PP model whose abstract model has an induced transition system \mathcal{A} which satisfies one of the following constraints :

1. \mathcal{A} has only sink strongly connected components.
2. All the SCC of \mathcal{A} have the same output value.

Then \mathcal{P} stably computes f under the GF constraint if \mathcal{A} stably computes f under the WF constraint

4 Probabilistic Verification

An alternative way to introduce fairness constraints in the PP model is to consider probabilistic schedulers. This alternative is considered by Angluin et al. [AAE08] which define the notion of probabilistic computation for a PP model. The simplest and more natural probabilistic scheduler proposed in [AAD⁺06] schedules ordered pairs of agents.

Definition 5 A *random pairing* (RP) scheduler chooses, in each state of a computation, the ordered pair of agents which interact in a random independent and uniform manner from all ordered pairs of agents.

The RP scheduler does not combine well with the counter abstraction because this abstraction collapses states whose sets of agents is the same. Therefore, we have to consider a scheduler whose decision is based on the transitions of the PP which are also transitions of the abstract VAS model.

Definition 6 The *random ruling* (RR) scheduler chooses, in each state of the computation, one enabled transition $t \in \delta$ of the PP in a random uniform and independent manner.

We show that, as in the non-probabilistic case, to verify that a predicate f is computed almost surely when probabilities are strictly positive consists in searching the closed strongly connected components in the transition system. Thus, we obtain the equivalence of the RR and RP schedulers w.r.t. the stably computation property as a consequence of the ergodic theorem on finite discrete Markov chains.

Proposition 1 *A PP \mathcal{P} stably computes a predicate $f \Leftrightarrow \mathcal{P}$ computes f almost surely with an RP scheduler.*

This result is the theoretical basis for the use of probabilistic model-checkers to verify that \mathcal{P} stably computes a predicate f even for a non randomized model of population protocols. Indeed, we have to verify that the abstract model of \mathcal{P} computes f almost surely with some scheduler assigning strictly positive probabilities to transitions.

We use the Prism tool [KNP04] to perform probabilistic model-checking. For each protocol, we encode its abstract VAS model in a discrete Markov chain model of Prism. The probabilities on transitions are not fixed, thus Prism associates to each transition enabled at some state a uniform distribution of probabilities. This model is checked against the property $\diamond \square f$ with probability at least one, i.e., almost surely.

5 Conclusion

We demonstrate that standard, finite state model-checking tools may help to verify that PP stably computes some predicate. In addition, these tools may be used to verify other properties of population protocols, i.e., properties on computations that can be expressed by LTL formulas. Such properties may be (1) all agents are infinitely often in some state, or (2) all computations are private in some PP with privacy [DGFG07].

Références

- [AAD⁺06] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, pages 235–253, 2006.
- [AAE08] D. Angluin, J. Aspnes, and D. Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21(2) :87–102, 2008.
- [AAER07] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4) :279–304, 2007.
- [BFPvdP08] R. Bakhshi, W. Fokkink, J. Pang, and J. van de Pol. Leader election in anonymous rings : Franklin goes probabilistic. In *IFIP TCS*, volume 273 of *IFIP*, pages 57–72. Springer-Verlag, 2008.
- [CDGFS11] J. Clement, C. Delporte-Gallet, H. Fauconnier, and M. Sighireanu. Guidelines for verification of population protocols. In *HAL-00565090, to appear in ICDCS*, 2011.
- [CGP99] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. MIT Press, 1999.
- [CMS10] I. Chatzigiannakis, O. Michail, and P.G. Spirakis. Algorithmic verification of population protocols. In *Proceedings of SSS*, volume 6366 of *LNCS*, pages 221–235. Springer, 2010.
- [DGFG07] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and E. Ruppert. Secretive birds : Privacy in population protocols. In *OPODIS*, LNCS, pages 329–342. Springer, 2007.
- [EC80] E.A. Emerson and E.M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *ICALP*, volume 85 of *LNCS*, pages 169–181. Springer, 1980.
- [GS92] S.M. German and A.P. Sistla. Reasoning about systems with many processes. *J. ACM*, 39(3) :675–735, 1992.
- [Hol03] G. Holzmann. *Spin model checker : the primer and reference manual*. Addison-Wesley Professional, 2003.
- [KM69] R.M. Karp and R.E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2) :147–195, 1969.
- [KNP04] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM : A hybrid approach. *Journal STTT*, 6(2) :128–142, 2004.
- [LPSZ09] Y. Liu, J. Pang, J. Sun, and J. Zhao. Verification of population ring protocols in PAT. In *TASE*, pages 81–89. IEEE Computer Society, 2009.
- [PLD08] J. Pang, Z. Luo, and Y. Deng. On automatic verification of self-stabilizing population protocols. In *TASE*, pages 185–192. IEEE Computer Society, 2008.
- [Pnu77] A. Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE, 1977.
- [QS81] J.-P. Queille and J. Sifakis. Iterative methods for the analysis of petri nets. In *Proceedings of APN*, volume 52 of *Informatik-Fachberichte*, pages 161–167. Springer, 1981.
- [SLDP09] J. Sun, Y. Liu, J.S. Dong, and J. Pang. Pat : Towards flexible verification under fairness. In *CAV*, volume 5643 of *LNCS*, pages 709–714. Springer, 2009.