Non-deterministic Population Protocols (Extended Version)

Joffroy Beauquier, Janna Burman, Laurent Rosaz, Brigitte Rozoy

Non-deterministic Population Protocols (Extended Version)

Joffroy Beauquier[1,2], Janna Burman[1,2], Laurent Rosaz[1], and Brigitte Rozoy[1,2]

[1] LRI - CNRS UMR 8623, Université Paris Sud XI, Bât 650, 91405 Orsay cedex, France,
{jb, burman, rozoy, rosaz}@lri.fr, fax: +33 (0)1 69 15 65 79, tel: +33 (0)1 69 15 68 60
[2] INRIA Saclay - Ile de France, Grand Large project

Abstract. In this paper we show that, in terms of generated output languages, non-deterministic *population protocols* are strictly more powerful than deterministic ones. Analyzing the reason for this negative result, we propose two slightly enhanced models, in which non-deterministic population protocols can be *exactly* simulated by deterministic ones. First, we consider a model in which interactions are not only between couples of agents, but also between triples and in which non-uniform initial states are allowed. We generalize this transformation and we prove a general property for a model with interactions between any number of agents. Second, we simulate any non-deterministic population protocol by a deterministic one in a model where a *configuration* can have an *empty output*.

Non-deterministic and deterministic population protocols are then compared in terms of inclusion of their output languages, that is, in terms of solvability of problems. Two transformations, which realize this inclusion, are presented. The first one uses (again) the natural model with interactions of triples, but does not need non-uniform initial states. As before, this result is generalized for the natural model with interactions between any number of agents. The second transformation is a parameterized one with parameters depending on the transition graph of the considered non-deterministic protocol and on the population.

Note that the transformations in the paper apply to a whole class of non-deterministic population protocols (for a proposed model), in contrast with the transformations proposed in previous works, which apply only to a specific sub-class of protocols (satisfying a so called "elasticity" condition).

Keywords: networks of mobile agents, population protocols, non-determinism vs. determinism

1 Introduction

Population protocols have been introduced [2] as a computation model (of functions or predicates) for asynchronous networks of simple (anonymous, resource limited) mobile agents, interacting pairwise. A characterization of what can be computed in this model is given in [4], namely the first order predicates in Presburger arithmetic. There, the protocols are assumed to be deterministic, meaning that, when two agents interact, there is a unique executable transition. The computational power of non-deterministic population protocols has been only partially studied in [1, 5].

The question concerning the comparison, in terms of computability and expressiveness, of deterministic and non-deterministic machines is a natural question in all computation models. Concerning population protocols, this question appears at different levels.

As population protocols were originally introduced in the context of function and predicate computation [2], at the first level, the question is whether or not deterministic and non-deterministic population protocols compute the same functions and the same predicates (in the sense of [2, 4]).

The second level concerns expressiveness in general. Some common method to define *any problem* (and not only a problem of function or predicate computation) is to define a set of (correct) *execution sequences* (see, e.g., [9]). An execution of a population protocol generates an *output sequence*, each configuration being associated to an output value. Thus, a problem can be also defined by the set of (correct) output sequences. Then, a population protocol can be defined *to solve a problem*, if its (non-empty) set of output sequences is

included in the set of output sequences characterizing the problem.[3] At this level, the question is whether or not non-deterministic population protocols solve the same family of problems as the deterministic ones. In other words, are they equivalent in terms of the problems they can solve? This issue is not only theoretical. Indeed, implementing non-determinism is usually made by randomization. However, non-determinism is not randomization. Why using randomization, if a deterministic solution for the same problem is available? At the same time, designing a non-deterministic solution is sometimes easier and more elegant than the equivalent deterministic one. Thus, the availability of automatic transformers of non-deterministic protocols into deterministic ones, could be of some help for a developer. Note that such transformers are generally a by-product of the study on expressiveness.

The third level concerns the generating power of population protocols. In finite automata and language theory, a label is associated to each transition, so that an execution generates a word, and an automaton produces a language. The Rabin and Scott construction [8] shows that, in terms of generated languages, non-deterministic and deterministic finite automata are equivalent, both generating the family of regular languages. However, language theory is related to programming language analysis and compilation, so its tools and outcomes hardly apply in a model of mobile agents. For instance, the definitions of non/determinism in population protocols are very different. Even if the rules defining a population protocol are deterministic, the resulting global transition system is not, because of the unpredictable interactions assumed between the agents. What is more relevant for the study of the generating power of population protocols is to consider and compare (for *equivalence* or *inclusion*) the generated languages of output sequences.

To motivate the study on this third level, note that proving *inclusion* of the output language of a protocol in the output language of another one implies that the former protocol solves the same problem as the latter. This may appear useful in practice, as already explained before in context of solvability of problems. At the same time, having *equivalence* of generated output languages can be also of practical help. For instance, if an implementation of a deterministic version of a protocol is preferable to that of a non-deterministic one (e.g., due to some development cost reasons), it can be useful and even necessary to have the same set of output sequences generated by the corresponding deterministic protocol. For instance, one reason may be efficiency in terms of time complexity. That is, e.g., the average complexity or the complexity of prevalent execution scenarios could be much better when concerning the larger set of executions/output sequences. Another reason may be the necessity to perform statistics over the whole set of execution/output sequences that can be generated by the non-deterministic protocol. Thus, it will be helpful to study whether the deterministic version of the protocol generates the same language.

Now, we summarize what is already known and what are the new results in this paper about population protocols in terms of the three types of questions explained above. First, the question about the computational power (in terms of predicate or function computability) of non-deterministic population protocols has been already raised in [3]. One can consider it received an answer in [1], where it is actually only stated that the non-deterministic population protocols compute exactly the Presburger predicates, exactly like the deterministic ones.[4]

In the context of problem solvability in general, (rather than in the context of computability of predicates or functions), [5] proves that the protocols solving the, so called, *elastic problems* (*elastic behaviors*, in terms of [5]), have a deterministic counterpart solving the same problem. To define elastic problem, first, define the repetition closure of a sequence t as the set of sequences obtainable from t by repeating each element of t one or more times. Extend this definition to a set of sequences O by taking the union of repetition closures of every sequence $t \in O$. Then, O is said *elastic* if it is closed by repetition closure. An elastic problem is a problem characterized by an elastic set of output sequences. Note however that this result of [5] does not imply that the output language of the deterministic counterpart is included in the output language of the

[3] One can see the terms "problem", "output sequence" and "solving a problem" as equivalent to the terms "behavior", "output trace" and "implementing a behavior", respectively. These terms are used in the literature about population protocols as well (see, e.g., [5, 6]).

[4] In some unpublished submitted version, one can find only the sketch of proof of the statement. There, the proof uses a transformation technique also used in [5] and one can understand how a complete equivalence proof would use this transformation.

given non-deterministic protocol. Still, one can deduce the following different result for some smaller class of protocols that we call *strongly elastic*.

A population protocol is strongly elastic, if for every rule $(p, q) \to (p', q')$ of the protocol, there is an idempotent rule $(p, q) \to (p, q)$. Then, it can be easily deduced from [5] that, if a problem is elastic and if there exists a strongly elastic non-deterministic population protocol solving this problem, then there exists a transformation giving a deterministic population protocol solving the same problem and moreover, with an output language included in the output language of the non-deterministic protocol.

However, the transformation in [5] does not provide the equality between the output languages of the strongly elastic non-deterministic population protocol and of its deterministic transformed version. In this paper, we study a way to obtain such an equality for population protocols in general (Sec. 3). Unfortunately, we come with a counter example (Sec. 3.1). When studying carefully the reason for this negative result, it appears that a natural way for simulating the non-determinism in the transitions of a non-deterministic population protocol is to use the non-determinism in the interactions between the agents. The negative result comes from the fact that, when there are not enough possible interactions between agents, a high degree of non-determinism in the transitions cannot be simulated.

In order to circumvent this negative result, we propose (in Sec. 3.2) to increase the number of possible interactions by allowing interactions between more than two agents. Without changing the total number of agents, this allows more non-determinism. As a matter of fact, we prove that a non-deterministic population protocol with pairwise interactions can be exactly simulated by a deterministic population protocol with three agent interactions, under the assumption that the initial states of the agents may be different. We show how this result can be generalized to k agent interactions, for any integer $k > 1$.

A second attempt to obtain equality of output languages consists in modifying slightly the definition of what can be an output value of a configuration (Sec. 3.3). Thus, an *empty output* value for a configuration is introduced such that, when it appears in the output sequence, it is taken as an identity element. We show that, in this extended model, the equality of output languages of non-deterministic and deterministic population protocols is obtained.

The results about equality of output languages yield also results about inclusion. However, we try to weaken the assumptions (that are made to obtain equality) in order to obtain stronger results about inclusion. It happens that, when considering interactions with more than two agents, we do not need non-uniform initial agent states, as we require for obtaining equality (see Sec. 4.1). This involves that, if the model does not restrict the number of agents in the interactions, non-deterministic and deterministic population protocols are equivalent, in terms of solvability of problems.

Finally, in the original model of pairwise interactions, we extend the result of inclusion for strongly elastic population protocols to the whole class. Contrary to [5], we cannot provide a transformation that is independent from the non-deterministic population protocol it transforms. We believe that such a general transformation does not exist and we explain the reason why (see appendix). However, for getting the inclusion result, we do not need this independence. Indeed, in Sec. 4.2, we propose a parameterized transformation. The parameters are the considered non-deterministic protocol and the population (the number of agents and their interaction graph). Given these parameters, the transformation constructs a deterministic population protocol solving the same problem as the given non-deterministic one.

2 Basic Model and Notations

As a basic model, we use the model of population protocols, as defined in [5, 6]. A *population* $\hat{\mathcal{A}}$ consists of a set $\mathcal{A}$ of $\mathbf{n}$ agents together with a weakly connected directed graph $G(\mathcal{A}, E)$. An agent represents a finite state sensing device and $\mathbf{n}$ is unknown to the agents. $G(\mathcal{A}, E)$ is called the *interaction (or communication) graph*, where $E \subseteq \mathcal{A} \times \mathcal{A}$. An edge $(u, v) \in E$ represents the possibility of a communication (an interaction) between u and v in which u is the initiator and v is the responder.

Population protocols can be modeled as *transition systems*. Thus, each agent is represented by the same finite transition system. The states of agents are from a finite set Q. Each agent has a constant *input value* and different agents may have different input values. For simplicity and as we assume constant input values,

we consider the inputs as a part of the state of an agent. There is an *output value* associated to each state of an agent. A transition function δ maps each element of $Q \times Q$ to a subset of $Q \times Q$. Let $(p, q) \in Q \times Q$. If $(p', q') \in \delta(p, q)$, then $(p, q) \rightarrow (p', q')$ is called a *transition*, and $(p, q) \rightarrow \delta(p, q)$ is called a *rule*. When, two agents u, in state p, and v, in state q, interact (meet), respectively playing the roles of initiator and responder, they execute a transition $(p, q) \rightarrow (p', q')$ such that $(p', q') \in \delta(p, q)$. As a result, u changes its state from p to p' and v from q to q'. It is possible that $p = p'$ and/or $q = q'$. The transition function and the protocol are *deterministic*, if $\delta(p, q)$ always contains just one pair of states (in other words, if each rule provides just one transition). Otherwise, if $|\delta(p, q)| = k > 1$, δ and the protocol are said *non-deterministic* (then u, v execute one of the k transitions in $\delta(p, q)$ chosen non-deterministically). Let us call k the *degree of non-determinism* of the rule $(p, q) \rightarrow \delta(p, q)$. Let $d = \max_{(p,q)\in Q\times Q}\{|\delta(p, q)|\}$ be the degree of non-determinism of δ and of the protocol. For simplicity, for any non-deterministic protocol, if for some $(p, q) \in Q \times Q, |\delta(p, q)| < d$, we duplicate some pairs of states in $\delta(p, q)$ in order to obtain $|\delta(p, q)| = d$. Thus, w.l.o.g., we assume that $\forall (p, q) \in Q \times Q, |\delta(p, q)| = d$.

A population protocol is also a finite transition system whose states are called *configurations*. A *configuration* is a mapping $C : \mathcal{A} \rightarrow Q$. A subset of configurations $\mathcal{C}_0$ defines the *initial configurations*. We say that C *goes to* C' via pair (interaction) $\pi = (u, v)$, denoted $C \xrightarrow{\pi} C'$, if the pair $(C'(u), C'(v))$ is in $\delta(C(u), C(v))$ and for all $w \in \mathcal{A} \setminus \{u, v\}$, $C'(w) = C(w)$. We say that C can go to C' in one step (or C' is *reachable in one step* from C), denoted $C \rightarrow C'$, if $C \xrightarrow{\pi} C'$ for some edge $\pi \in E$. If there is a sequence of configurations $C = C_0, C_1, \ldots, C_k = C'$, such that $C_i \rightarrow C_{i+1}$ for all $i, 0 \leq i < k$, we say that C' *is reachable* from C, denoted $C \xrightarrow{*} C'$.

An *execution* is an infinite sequence of configurations $C_0, C_1, C_2, \ldots$ such that $C_0 \in \mathcal{C}_0$ and for each i, $C_i \rightarrow C_{i+1}$. The *output of a configuration* C is the multi-set of the output values of agents in C. The *output word (or the output trace)* of an execution $e = C_0, C_1, C_2 \ldots$ is a sequence $O_0, O_1, O_2, \ldots$ resulting from the concatenation of the successive outputs of the configurations of e. That is, for all $i \geq 0$, O_i is the output of the configuration C_i. The set of output words of a protocol P is called the *(generated) output language* of the protocol and denoted by $L(P)$.

Let P_1 and P_2 be two protocols with sets of states Q_1 and $Q_1 \times Q'$ respectively, for some set Q'. For a state $s_2 = [s_1\ s'] \in Q_1 \times Q'$ of P_2, where $s_1 \in Q_1$ and $s' \in Q'$, $\Pi_{P_1}(s_2) = s_1$. That is, $\Pi_{P_1}(s_2)$ denotes the state of P_1 which is the projection of s_2 on P_1 (in other words, which is the mapping of s_2 to the state component of P_1). We extend the notation of Π in the natural way to configurations, sets of states or configurations, rules, transitions and executions.

A *problem* is defined by some conditions on executions, or equivalently by the sub-set of the executions that satisfy the conditions. As an output word associated to an execution can be defined to be the execution sequence itself (by defining the output of each agent as being the whole state), a problem can be well defined by giving conditions only on output words. Thus, w.l.o.g. and for the sake simplicity, *we assume that a problem is defined by conditions on output words, i.e., by a sub-set $\mathcal{B}$ of output words*. A population protocol is said to *solve a problem*, if and only if the set of its output words $\mathcal{O}$ is non-empty and each output word $o \in \mathcal{O}$ satisfies the conditions defining the problem, i.e., $o \in \mathcal{B}$ or equivalently, $\mathcal{O} \subseteq \mathcal{B}$ (see, e.g., [9]).

The *transition graph* $G(P, \hat{\mathcal{A}})$ of a protocol P running in population $\hat{\mathcal{A}}$ is a directed graph whose nodes are all possible population configurations and whose edges are all possible transitions on those nodes. A strongly connected component of a directed graph is *final* iff no edge leads from a node in the component to a node outside.

As originally for population protocols, we assume a strong fairness condition on the executions that is called *global fairness*. An execution is said globally fair, if for every two configurations C and C' such that $C \rightarrow C'$, if C occurs infinitely often in the execution, then C' also occurs infinitely often in the execution.

3 Results about Equality of Output Languages

In this section, we study the strong relation of equality between the sets of output languages of deterministic and non-deterministic population protocols. First, we give a negative result (Theorem 1, Sec. 3.1) showing

that in the basic model of Sec. 2, in terms of the equality between the sets of generated output languages, non-deterministic protocols are more powerful. Then, in sections 3.2 and 3.3, we propose two model extensions that allow to circumvent this negative result.

3.1 A Negative Result

The following example provides some simple preliminary intuition for the result stated in Theorem 1 below. Consider a population of two agents in the initial configuration (q_0, q_0) and the non-deterministic protocol P with two rules $(q_0, q_0) \to \{(q_0, q_0), (q_1, q_1)\}$ and $(q_1, q_1) \to (q_0, q_0)$. Assume that the output of an agent in P is its state. Then, each output word is an infinite concatenation of the output sequences of the form $(q_0, q_0)^k, (q_1, q_1)$, for any positive integer k. Thus, P has an output language of infinite size. However, the output language of any deterministic protocol executing on two agents (with finite states) has a finite size. This example proves the theorem below for two agent populations. Note that the same argument is wrong in larger populations, since then, intuitively, there exists a non-determinism in the choice of interactions that can lead to an infinite output language size. In the proof of the theorem, we give an example that works for a population of any size $\mathbf{n}$.

Theorem 1. *Given a population of size* $\mathbf{n}$*, the set of output languages of non-deterministic population protocols strictly contains the corresponding set of deterministic population protocols.*

Proof. It consists in exhibiting an example of a non-deterministic protocol whose output language is not equal to the output language of any deterministic protocol.

Consider a population of $\mathbf{n}$ agents and a non-deterministic population protocol P. Let $t = \mathbf{n}\cdot(\mathbf{n}-1)$. Let P to have a single non-deterministic rule $(p_0, p_0) \to \{(p_1, p_1), (p_2, p_2), \ldots, (p_{t+1}, p_{t+1})\}$, and $t+1$ deterministic rules $(p_1, p_1) \to (p_0, p_0)$, $(p_2, p_2) \to (p_0, p_0)$, $\ldots$, $(p_{t+1}, p_{t+1}) \to (p_0, p_0)$. Let o_i be the output value associated to a state p_i. We choose $o_i = p_i$. The initial configuration of P is $C_0 = (p_0, p_0, p_0, p_0)$. Note that the output of C_0 is the multi-set $M_0 = \{p_0, p_0, p_0, p_0\}$. Assume, for the sake of contradiction, that there is a deterministic population protocol P' such that $L(P') = L(P)$.

In P, consider all the prefixes of execution of a type $e\_ = (C_0, C)$. There are exactly $t + 1$ different configuration output prefixes corresponding to these execution prefixes. Now, consider the concatenation of two such prefixes $e\_e\_$, which is also a prefix of a possible execution in P. The number of different configuration output prefixes for $e\_e\_$ is $(t+1)^2$, and more generally, for the concatenation of k prefixes of $e\_$, the number is $(t+1)^k$. Denote by H_k the set of these output prefixes ($|H_k| = (t+1)^k$). Since $L(P') = L(P)$, all the prefixes in H_k are also output prefixes of P'. However, if P' has a single configuration with *output* $\{p_0, p_0, p_0, p_0\} = M_0$, P' can "generate" only t output prefixes of length 2, starting from C_0. More generally, P' can "generate" only t^k output prefixes composed by concatenation of k output prefixes of length 2, starting from C_0. The number $t = \mathbf{n} \cdot (\mathbf{n} - 1)$ is the maximum number of different pairs of states that $\mathbf{n}$ agents can have (with the distinction between initiator and responder).

Thus, since $|H_k| = (t+1)^k > t^k$, but $L(P') = L(P)$, P' has necessarily more than one configuration with output $\{p_0, p_0, p_0, p_0\} = M_0$. Assume then that P' has r different configurations with output M_0. Each of them "generates" at most t^k different concatenated (k times) output prefixes of length 2 starting with C_0. That is, P' can "generate" at most $r \cdot t^k$ different such output prefixes. However, since $L(P') = L(P)$, $r \cdot t^k$ must be at least as large as $(t + 1)^k$. This involves that r is at least as large as $\frac{(t+1)^k}{t^k}$ and that, for *every* integer $k \geq 1$. A contradiction arises from the fact that r is bounded by the number of the configurations of P' which is finite. □

An immediate corollary from the proof of Theorem 1, is that one of the reasons for the theorem correctness is the assumption that each agent in population protocols has only a finite size state. One can also notice that the negative result comes from the fact that, when there are not enough possible interactions between agents, a high degree of non-determinism in the transitions cannot be simulated by any deterministic protocol. That is why, increasing the number of agents in an interaction, as in Sec. 3.2, allows to overcome the negative result.

3.2 Equality with Interactions of More than Two Agents

One way to increase the degree of non-determinism through the interactions of agents is to consider a more general population protocol model, where the interactions concern more than only two agents. The issue of considering such a generalization was raised already in [2], but to our knowledge, it was not dealt in the literature in the context of non-deterministic protocols as in this work.Thus, to obtain the desired equality of output languages, we consider interactions involving more than two agents. Roughly, the idea is to assign a different integer from $[1, d]$ to each agent, where d is the degree of non-determinism of the given non-deterministic protocol. Then, we simulate deterministically the k^{th} ($k \in [1, d]$) choice of a non-deterministic transition between two agents u and v, by an interaction between three agents u, v and an agent holding the integer k.

Let us denote by PP_k the model of population protocols in which possible interactions are between k agents or less. The definition of PP_k follows the definition of the basic model of population protocols in Sec. 2. However, for PP_k, we should generalize the definition of the transition function δ and the notion of initiator and responder. During an interaction of k' agents, $u_1, u_2, \ldots, u_{k'}$, $2 \leq k' \leq k$, we say that u_1 is the initiator, u_2 is the primary responder, u_3 is the secondary responder, and so on. Now, δ maps each element in $Q^{k'}$, for each $2 \leq k' \leq k$, to a subset of $Q^{k'}$. We first prove the following result.

Theorem 2. *Consider a non-deterministic population protocol P_1 with the degree of non-determinism d. Let $\hat{\mathcal{A}}$, be any population with $\mathbf{n} \geq d+2$ and a complete interaction graph. Given a protocol P_1 executing on $\hat{\mathcal{A}}$ in PP_2 (the basic model of Sec. 2), there exists a deterministic population protocol P_2 executing on $\hat{\mathcal{A}}$ in PP_3 (with $d+2$ non-uniform initial states)[5] and generating the same output language as P_1.*

The proof consists in, first, constructing for any population protocol P_1 a deterministic population protocol P_2 and then, in proving that $L(P_1) = L(P_2)$. Thus, we first construct P_2. As explained in Sec. 2, we assume, w.l.o.g., that all the rules of P_1 have the same degree d of non-determinism. The state of an agent in P_2 is a couple $[p\ c]$, where p is a state of P_1 (p is the projection of $[p\ c]$ on P_1, denoted $\Pi_{P_1}([p\ c]) = p$), and c is an input value of P_2 which is an integer in $[1, m], m = d+2$. The purpose of c is to serve as a *switch* value to decide deterministically, in P_2, on a transition of a non-deterministic rule of P_1. For every initial configuration C_0 of P_1, there is one initial configuration C_0' of P_2 such that $\Pi_{P_1}(C_0') = C_0$. We make an important assumption about the input value c. During an execution, each value in $[1, m]$ is the input value c of at least one agent (in all this section, we assume $\mathbf{n} \geq m$). The output of the state $[p\ c]$ in P_2 is the output of p in P_1. To each rule $(p, p') \rightarrow \{(p_1, p_1'), (p_2, p_2'), ..., (p_d, p_d')\}$ of P_1, the construction associates three types of deterministic rules of P_2:

i. For c in $[1, d]$, $([p\ c_1], [p'\ c_2], [q\ c]) \rightarrow ([p_c\ c_1], [p'_c\ c_2], [q\ c])$
ii. For c_1 in $[1, d]$, $([p\ c_1], [p'\ c_2]) \rightarrow ([p_{c_1}\ c_1], [p'_{c_1}\ c_2])$
iii. For c in $\{d+1, d+2\}$, $([p\ c_1], [p'\ c_2], [q\ c]) \rightarrow ([p_{c_2}\ c_1], [p'_{c_2}\ c_2], [q\ c])$

The intuition behind the rules of P_2 is, for any pair of states (p_2, p_2') with $\Pi_{P_1}(p_2, p_2') = (p, p')$, to be able to simulate any possible transition in the set $\delta(p, p')$ of P_1, and this by executing only *one* transition of P_2. For obtaining the equality of output languages, it is important to be able to execute exactly one transition for this purpose.[6] Thus, the rule of type i. serves to execute a projected transition $(p, p') \overset{(u,v)}{\rightarrow} (p_c, p'_c)$, for two agents u, v, in the case where there exists another agent holding the switch input value c. Otherwise, the rules of type ii. and iii. are provided for the case where the same "needed" switch value is unique and held either by the initiator or by the primary responder (respectively). Below, we prove the equality of the output languages of P_1 and P_2. For that, we first prove the following basic lemma that actually validates the intuitive ideas explained above.

[5] Note that this assumption cannot be used to assign identifiers to agents, if $\mathbf{n} \gg d+2$. As for population protocols, it is generally assumed that $\mathbf{n} \gg |Q|$, that implies that $\mathbf{n} \gg d+2$. In any case, we show in the sequel (Sec. 4.1) that this assumption can be dropped to obtain a weaker property of inclusion for the output languages.

[6] Note, however, that by changing the model definitions, e.g., for the output words of a protocol, as in Sec. 3.3, it is possible to drop this requirement when still having the equality of output languages for non/deterministic protocols.

Lemma 1. *Let C_2 be a configuration of the deterministic protocol P_2 given by the construction above. Let $C_1 = \Pi_{P_1}(C_2)$ and let C_1' be any configuration of P_1 such that $C_1 \to C_1'$. Then, there exist a configuration C_2' of P_2 and $C_2 \to C_2'$ such that $C_1' = \Pi_{P_1}(C_2')$.*

Proof. Let C_1' be reachable from C_1 by executing a transition $(p, p') \to (p_i, p_i')$, corresponding to the i^{th} choice in $\delta(p, p')$. As C_1 is a projection of C_2, in C_2, there are two agents, one in a state $[p\ c_p]$, and another one in a state $[p'\ c_{p'}]$. If $c_p = i$, then applying rule ii. of P_2, in configuration C_2, gives a configuration C_2' whose projection is C_1'. Otherwise, if $c_{p'} = i$, then, by the construction of P_2, there are at least two agents with switch value equal to either $d+1$ or $d+2$. Then, rule iii. can be applied in C_2, which results in configuration C_2' whose projection is C_1'. In case neither c_p, nor $c_{p'}$ is equal to i, there is at least one additional agent with a switch value equal to i. Then, rule i. can be applied in C_2, which results in configuration C_2' whose projection is C_1'. Thus, in all cases, C_2' is reachable from C_2. □

Proof (of Theorem 2). To prove the theorem, we first show that given any globally fair execution e_2 of P_2, its projection $e_1 = \Pi_{P_1}(e_2)$ is a globally fair execution of P_1 and thus the output word of e_2 is in the output language of P_1. Then, we show that for every globally fair execution e_1 of P_1, there is a globally fair execution of P_2 whose projection on P_1 is e_1 and thus, the output word of e_1 is in the output language of P_2.

Thus, let e_2 be a globally fair execution of P_2. The projection $e_1 = \Pi_{P_1}(e_2)$ is an execution of P_1, since, by construction, the projection of each transition of P_2 is a transition of P_1. In the following, we show that e_1 is globally fair. Let C_1 be a configuration of P_1 appearing infinitely often in e_1, and let C_1' be a configuration reachable in one step from C_1, $C_1 \to C_1'$. Then, since e_1 is the projection of e_2, there are infinitely many configurations appearing in e_2, whose projection is C_1. Thus and by the finiteness of the states of the agents, there is such a configuration C_2, appearing infinitely often in e_2. By Lemma 1, a configuration C_2' of P_2 whose projection is C_1' is reachable from C_2 in one step. As e_2 is globally fair, C_2' appears infinitely often in e_2. Thus C_1' appears infinitely often in e_1. That proves that e_1 is globally fair.

Now consider a globally fair execution e_1 of P_1. Consider the prefix of e_1 of length r, e_1^r, for some integer $r \geq 1$ and assume (by induction on r) that there exists a segment e_2^r , prefix of an execution of P_2, with projection e_1^r on P_1 (the basis of the induction, for $r = 1$, holds by construction). Assume that $e_1^{r+1} = (e_1^{r-1}, C_1, C_1')$ and $e_2^r = (e_2^{r-1}, C_2)$. By Lemma 1, a configuration C_2' of P_2 whose projection is C_1' is reachable from C_2 in one step. Thus, there is a prefix of an execution of P_2, e_2^{r+1}, whose projection on P_1 is e_1^{r+1}. Thus, by induction, an execution e_2 whose projection is e_1 can be built. As e_1 is globally fair and as the switch values are constant, e_2 is also globally fair. □

The result of Theorem 2 can be generalized for any k.

Theorem 3. *Consider any population $\hat{\mathcal{A}}$ with $\mathbf{n} \geq d + k^2$ and complete interaction graph. For any non-deterministic population protocol on $\hat{\mathcal{A}}$ in PP_k, there exists a deterministic population protocol on $\hat{\mathcal{A}}$ in PP_{k+1} (with $d + k^2$ non-uniform initial states) with the same output language.*

Proof Sketch. For the general case, we propose two kinds of transformation protocols in PP_{k+1}. One, denoted P_3, is a generalization of the protocol P_2 (given above for $k = 2$). Thus, in P_3, for any $k > 1$, $m = d + k^2$. During any execution, each value in $[1, m]$ is the input value c of at least one agent. To each rule of P_1, $(p_1, p_2, \ldots, p_{k'}) \to \{(p_{11}, p_{12}, \ldots, p_{1k'}), (p_{21}, \ldots, p_{2k'}), \ldots, (p_{d1}, \ldots, p_{dk'})\}$, for $2 \leq k' \leq k$, the construction associates two types of deterministic rules of P_3:

i. For c in $[1, d]$,
$([p_1\ c^1], [p_2\ c^2], \ldots, [p_{k'}\ c^{k'}], [q\ c]) \to ([p_{c1}\ c^1]), [p_{c2}\ c^2], \ldots, [p_{ck'}\ c^{k'}], [q, c])$
ii. For c in $[d + x \cdot k + 1, d + x \cdot k + k]$ and for any integer x, $0 \leq x < k'$,
$([p_1\ c^1], [p_2\ c^2], \ldots, [p_k'\ c^{k'}], [q\ c]) \to ([p_{c^x 1}\ c^1], [p_{c^x 2}\ c^2], \ldots, [p_{c^x k'}\ c^{k'}], [q\ c])$

Another transformation protocol to simulate the non-deterministic protocol P_1, denoted P_3', differs from P_3 by the value of m, the conditions on the inputs and by the transition function δ. Thus, for P_3', $m = d$. During any execution, each value in $[1, m]$ is the input value c of at least $k + 1$ agents. To each rule of P_1, $(p_1, p_2, \ldots, p_{k'}) \to \{(p_{11}, p_{12}, \ldots, p_{1k'}), (p_{21}, \ldots, p_{2k'}), \ldots, (p_{d1}, \ldots, p_{dk'})\}$, for $2 \leq k' \leq k$, the construction

associates the following deterministic rule of P'_3:
$([p_1\ c^1], [p_2\ c^2], \ldots, [p_{k'}\ c^{k'}], [q\ c]) \rightarrow ([p_{c1}\ c^1], [p_{c2}\ c^2], \ldots, [p_{ck'}\ c^{k'}], [q\ c])$. To see the correctness of the transformations, notice that Lem. 1 holds also for P_3 and P'_3. That is, given any configuration C_3 of the transformed protocol (P_3 or P'_3) and its projection C_1 on P_1, for any C'_1 such that $C_1 \rightarrow C'_1$, there exists a configuration C'_3 such that $C_3 \rightarrow C'_3$, and $C'_1 = \Pi_{P_1}(C'_3)$. The rest of the correctness proof follows the proof of Theorem 2.[7] □

3.3 Equality by Simulation with Empty Outputs

Theorem 1 states that there is no *Rabin and Scott*-like construction for population protocols, at least with the original definitions of [5, 6]. We note that the negative property strongly depends on the definition of what an output value can be. We think this definition can be changed, without reappraisal of the basic model of population protocols. In the sequel, we investigate the way of modifying the definition of the output of a configuration, in order to get an equivalence result for the output languages. The idea we develop is to consider an *empty output* ϵ for a configuration, serving as an identity element in the monoid generated by output values of configurations. That is, we allow the empty output ϵ to be a possible output value for a configuration such that for any segment of an output word o, $(o, \epsilon) = (\epsilon, o) = o$.

Intuitively, this idea of introducing empty outputs in the model can be helpful in the following way. For instance, assume that agents, in the deterministic protocol (simulating the non-deterministic one), hold different integers used as a switch to indicate one of the possible non-deterministic choices. These switch values can be changed by the protocol. A problem arises when an agent u in a state p, holding the switch value c, interacts with an agent v in a state q, but the non-deterministic choice c' has to be simulated to obtain a specific output word (to obtain equality of output languages with the non-deterministic protocol). In this situation, one would like to perform some transitions (called *null-transitions*, in the sequel) to obtain a configuration where the switch value c' is in u and the rest of the states of u and v stays unchanged. However, the outputs of the intermediary configurations reached by these null-transitions are repetitions of the same value. This may result in an output word that is not in the output language of the corresponding non-deterministic protocol. With empty outputs, it is possible to remove such repetitions of the same output and obtain the equality of the output languages. Notice that a difficulty comes from the fact that the same configuration can be reached either by a null or a non-null-transition. In the first case, it is required to output the empty output, but not in the second.

Theorem 4. *In terms of generated output languages, the non-deterministic and the deterministic population protocols are equivalent in the model allowing empty outputs for configurations.*

To prove the theorem, we present a general technique to transform the rules of *any* non-deterministic population protocol P_1 into the deterministic rules of a population protocol P_3, in the model with empty outputs. Next, we prove that $L(P_1) = L(P_3)$ (see Theorem 5). The transformation we propose, denoted D, takes as an input a protocol P_1 and another deterministic transformation D'. It is required that D' applied to P_1, denoted $D'(P_1)$, results in a deterministic protocol P_2 satisfying conditions defined in Property 1 below. We write $P_2 = D'(P_1)$ and $P_3 = D(D'(P_1))$. In the sequel, we show that there exists such a transformation D', e.g., the transformation presented in [5] (see Lem. 4). Recall that this transformation (in [5]) only applies to some sub-class of protocols and does not provide the equality of languages for non/deterministic protocols even for this sub-class.

We use the following definitions to state Property 1 and to define D. Let P and P' be two protocols with sets of states Q and $Q \times Q''$ respectively, for some set Q''. A transition t of P', $(p, q) \rightarrow (p', q')$, is called a *(P-)null-transition*, if $(p, q) \neq (p', q')$, but $\Pi_P(p, q) = \Pi_P(p', q')$. Two consecutive and different configurations C_1, C_2 in an execution of P' are called *(P-)similar*, if C_2 is obtained from C_1 by a P-null-transition (that is, $\Pi_P(C_1) = \Pi_P(C_2)$).

[7] The required memory for an agent in P_3 is larger than the one in P'_3. However, when $k \ll d$, P_3 may be more advantageous. In this case, the state space requirements for the two transformations differ only slightly, though the minimum number of agents required by P'_3 may be much larger than the one of P_3.

Property 1. Let P_1 be any non-deterministic population protocol with a set of states Q_1. Protocol P_2 is said to satisfy Property 1, if it satisfies the following conditions:

1. The protocol P_2 is a deterministic protocol with a set of states $Q_1 \times Q'$, for some set Q'. The projection of the rules of P_2 on P_1, is the set of rules of P_1.
2. The output of a configuration C in P_2 is the output of the configuration $\Pi_{P_1}(C)$ of P_1.
3. For every initial configuration C_0 of P_1, there is one initial configuration C'_0 of P_2 such that $\Pi_{P_1}(C'_0) = C_0$.
4. For every two configurations C, C' of P_2, if $C \rightarrow C'$, then $C \neq C'$.
5. Let C_2 be a configuration of P_2 such that $C_1 = \Pi_{P_1}(C_2)$. Let C'_1 be a configuration of P_1 such that $C_1 \rightarrow C'_1$. Then, there exists a configuration C'_2 such that $C_2 \xrightarrow{*} C'_2$ and $\Pi_{P_1}(C'_2) = C'_1$. In addition, C'_2 is reachable from C_2 using a finite number of null transitions of P_2, except for the last transition that results in C'_2.

Definition of the transformation D**.** The main idea of the transformation D is to simulate $P_2 = D'(P_1)$ (satisfying Property 1) while eliminating the effect of the P_1-null-transitions of P_2 in the output words. This is done by introducing empty outputs, for obtaining the equality of output languages. Now, we define the protocol $P_3 = D(P_2)$. Let Q_1 and $Q_2 = Q_1 \times Q'$ be the sets of states of P_1 and P_2, respectively. Starting from P_2, we build a deterministic population protocol P_3, which has a lot of similarities with P_2, but differs mainly in the definition of states (configurations) and configuration outputs. The set of states of P_3 is $Q_3 = Q_2 \times Q_2$. Then, a configuration of P_3 can be viewed as a pair (C^*, C) of configurations of P_2. For every transition (rule) of P_2, $(p, q) \rightarrow (p', q')$, D associates a transition $([p^*\ p], [q^*\ q]) \rightarrow ([p\ p'], [q\ q'])$ of P_3. Thus, iff $C \rightarrow C'$ in P_2, then $(C^*, C) \rightarrow (C, C')$ in P_3. In an execution of P_3, a component C^* of a configuration (C^*, C) can be viewed as the previous configuration in the corresponding execution of P_2, and C can be viewed as the actual configuration. The reason of doing that is to be able to locate P_1-similar configurations in an execution (resulting from the null-transitions in P_2) and "eliminate" their output from the output word. Thus, the output of a configuration (C^*, C) is defined to be the empty output ϵ, if C^* and C are P_1-similar. Otherwise, the output of (C^*, C) is the output of C in P_2 (which is the output of $\Pi_{P_1}(C)$ in P_1, by Property 1). For every initial configuration C_0 of P_2, (C_0, C_0) is the initial configuration of P_3.

Theorem 5. *Consider a population protocol model allowing empty outputs for configurations. Let P_1 be any non-deterministic population protocol and a protocol $P_2 = D'(P_1)$ (satisfying Property 1). Let $P_3 = D(P_2)$. Then, $L(P_1) = L(P_3)$.*

We prove the theorem by proving the following two lemmas.

Lemma 2. *Consider a population protocol model allowing empty outputs for configurations. Let P_1 be any non-deterministic population protocol. Let $P_2 = D'(P_1)$ and $P_3 = D(P_2)$. Then, the output word of any globally fair execution of P_3 is the output word of a globally fair execution of P_1 (that is, $L(P_3) \subseteq L(P_1)$).*

Proof. Let E be an (infinite) globally fair execution of P_3 and w its output word. In E, we replace any two consecutive configurations $(C^*, C), (C, C')$, by C, C' and a starting configuration (C_0, C_0) by C_0. Then, in the resulting sequence E_2, we replace any sequence of consecutive similar configurations by the first configuration of the sequence. Let E' be the resulting sequence. By the construction of P_3 and the definition of the empty output, the output word of E' is w. The projection of E' on P_1 is an execution e of P_1. To prove the lemma we show that e is globally fair for P_1 and thus, its output, which is also w, is an output word of P_1. Note that by construction of D, E_2 is a globally fair execution of P_2. Now, consider a configuration c of P_1 appearing infinitely often in e, and assume that there is a configuration c', reachable from c in one step. Since e is the projection of E', there exist in E', and then in E_2, infinitely many configurations whose projection is c. Since the set of configurations of P_2 is finite, at least one of these configurations, C, appears infinitely often in E_2. By Property 1, condition 5, there is a configuration C' such that $C \xrightarrow{*} C'$ using only P_1-null-transitions but the last, and such that $c' = \Pi_{P_1}(C')$. As E_2 is globally fair, the sequence of configurations from C to C', using only null-transitions but the last, appears infinitely often in E_2. Thus, by construction of e, c' appears infinitely often in e. That involves that e is globally fair. □

Lemma 2 is sufficient for proving the inclusion of the output languages of P_3 and P_1, involving that the class of problems solved by non-deterministic and deterministic population protocols is the same (when empty outputs are allowed). For proving the equality of the output languages, we need to prove the converse inclusion.

Lemma 3. *Consider a population protocol model allowing empty outputs for configurations. Let P_1 be any non-deterministic population protocol. Let $P_2 = D'(P_1)$ and $P_3 = D(P_2)$. Then, the output word of any globally fair execution of P_1 is the output word of a globally fair execution of P_3 (that is, $L(P_1) \subseteq L(P_3)$).*

Proof. Consider a globally fair execution $e_1 = C_0, C_1, C_2, C_3, \ldots$ of P_1. First, we build a corresponding globally fair execution e_2 of P_2. By a simple induction, and using conditions 5 and 3 of Property 1, there is $e_2 = C'_0, e'_0, C'_1, e'_1, C'_2, \ldots$ such that $\forall i \geq 0$, e'_i is a finite sequence of P_1-similar configurations and $C'_i = \Pi_{P_1}(C_i)$. As e_1 is globally fair and the states of P_2 are finite, each e'_i can be chosen such that e_2 is globally fair. Now, we build an execution $e_3 = (C'_0, C'_0)(C'_0, C''_1)(C''_1, C''_2)(C''_2, C''_3)\ldots$ of P_3 such that, $\forall i \geq 2$, $(C''_{i-1}, C''_i) \rightarrow (C''_i, C''_{i+1})$ in e_3, if $C''_i \rightarrow C''_{i+1}$ in e_2. It is easy to see that if e_2 is globally fair, then e_3 is globally fair. By construction of D, the output word of e_3 is the output word of e_1. Thus, the lemma holds. □

Lemma 4. *Given any non-deterministic population protocol P_1 and the transformation D' presented in [5], $D'(P_1) = P_2$ is a deterministic protocol satisfying Property 1.*

Proof. All conditions of Property 1, except the last, trivially hold for P_2, by the construction of D' in [5]. Condition 5 holds by lemmas 3.1 and 3.2 in [5] and the fact that the statements of these lemmas are achieved by executing P_1-null-transitions only, as it is shown in their proofs. □

4 Results about Inclusion of Output Languages

In this section, we consider the weaker requirement of inclusion for output languages of non/deterministic protocols. As noted in the introduction, [5] gives a transformation of any strongly elastic non-deterministic population protocol into a deterministic one, whose output language is included in the output language of the former. Dropping the "elasticity" condition, without introducing empty output values, is difficult (we explain the reason for that in the appendix).

One way to solve this difficulty is to use a parameterized transformation, presented in Sec. 4.2 below. In contrast with the transformation of [5], it uses non-uniform initial states for agents and depends on the transition graph of a given non-deterministic protocol and on the population.

Another, more natural (and coherent with population protocols) way to obtain inclusion of output languages is, like in Sec. 3.2, to allow interactions with three (or more) agents. The idea is to use the secondary responder with a required switch value to be always able to execute deterministically any possible transition of the non-deterministic transition function. It appears that, when only inclusion is required (in contrast with Sec. 3.2), it is not necessary for the switch values to be initially distinct. Indeed, we provide a protocol (Protocol 1 below) that, starting from a symmetrical initial configuration, distributes the different switch values between the agents. The idea of Prot. 1 is to generalize a (circulating) leader election population protocol proposed in [1] to manage several (instead of one) leader marks (which we call here tokens). Note that Prot. 1 cannot be used to obtain equality.

4.1 Inclusion with Interactions of More than Two Agents

Thus, we propose a deterministic protocol Prot. 1 that distributes tokens of m ($\mathbf{n} \geq m \geq 1$) different types (represented by integers in $[1, m]$) between $\mathbf{n}$ agents. By Lem. 5 proven below, eventually, there is exactly one token of each type and every agent holds at most one token.[8]

It is assumed that there are at least m agents and that initially, each agent holds one token of each type. Note that these initial states are uniform and the protocol works in any PP_k model, for any integer $k > 1$.

[8] Note that the property given by the lemma does not state that eventually the same token stays with the same agent. On the contrary, the protocol ensures that the tokens are always exchanged between the agents (lines 2-3). This makes the protocol work for populations with interaction graph of *any* topology.

Protocol 1 - to distribute m tokens of different types between n agents

Initialization:

Every agent x has a set $T_x = \{t_1, t_2, \ldots, t_m\}$ of $m \geq 1$ different tokens. For every pair of agents x, y, $T_x = T_y$.

1: **when** an initiator x interacts with a primary responder y **do**
2: **if** $(T_x \cap T_y) = \emptyset \bigwedge |T_x| = |T_y|$ **then**
3: $T' \leftarrow T_x, T_x \leftarrow T_y, T_y \leftarrow T'$ // *exchange the tokens*
4: // *distribute the tokens*
5: **if** $(T_x \cap T_y) \neq \emptyset \bigwedge (T' \leftarrow (T_x \cap T_y) = \{t'_1, t'_2, \ldots, t'_{|T'|}\})$ **then**
6: $T_y \leftarrow T_y \setminus \{t'_1, t'_2, \ldots, t'_{\lceil \frac{|T'|}{2} \rceil}\}$
7: **if** $|T' > 1|$ **then**
8: $T_x \leftarrow T_x \setminus \{t'_{\lceil \frac{|T'|}{2} \rceil + 1}, \ldots, t'_{|T'|}\}$
9: **if** $(T_x \cap T_y) = \emptyset \bigwedge (dif \leftarrow |T_x| - |T_y|) > 1 \wedge (T_x = \{t^x_1, t^x_2, \ldots, t^x_{|T_x|}\})$ **then**
10: $size_x \leftarrow |T_x|, size_y \leftarrow |T_y|$
11: $T_x \leftarrow \{t^x_1, \ldots, t^x_{\lceil \frac{dif}{2} \rceil + size_y}\}$
12: $T_y \leftarrow T_y \cup \{t^x_{\lceil \frac{dif}{2} \rceil + size_y + 1}, \ldots, t^x_{size_x}\}$

Lemma 5. *Eventually, in each configuration reached by an execution of Prot. 1, there is exactly one token of each type and every agent holds at most one token.*

Proof. Let us denote lines 2-3 of the protocol, by action 1; lines 5-8, by action 2; and lines 9-12, by action 3. Note that, according to all the actions, the number of tokens held by an agent is non-increasing. Thus and because of the initial configuration, two interacting agents can have at most two tokens of a similar type (each agent, at most one such token). When two agents, holding tokens of a similar type, interact as initiator and primary responder, action 2 is applied. Then, exactly one of the two tokens of the similar type remains in one of the agents. No action removes a token appearing only in one of the interacting agents. Thus, there is always at least one token of type r, for every type $1 \leq r \leq m$.

Now, assume for the sake of contradiction that there are two tokens of the same type existing during an infinite suffix of some execution. However, by action 1 and the global fairness assumption, eventually, two agents, each having one of these two tokens, interact (as initiator and primary responder), and action 2 is applied. Thus, only one of these tokens remains. Then, if still at least two tokens of the same considered type exist, a similar execution repeats, until only one token of the type remains - a contradiction.

Finally, assume for the sake of contradiction that there is an agent u with at least two tokens in infinitely many configurations. By the paragraph above, eventually, for each type, there is at most one token of the type. Then, first, there are exactly $m \leq \mathbf{n}$ tokens (u holding at least two of them) and thus, there are some agents with no tokens. Second, only action 3 or 1 can be applied. Action 1 can only move these two tokens together from agent to agent. Then, if the tokens are still at one agent, an agent holding them, eventually interacts (by the global fairness) with an agent with no token (they interact as initiator and primary responder) and action 3 is applied. If the two tokens are still at one agent after this interaction (it can happen in case where there are more than two tokens in u), a similar execution repeats until the two are finally "separated" by action 3 - a contradiction. □

Now, given a non-deterministic population protocol P_1 in PP_2 (or in PP_k, for any integer $k > 1$), we build a deterministic version PDI of P_1 (given by Protocol 2 below) such that $L(PDI) \subseteq L(P_1)$ (see Theorem 6). That is, the deterministic protocol PDI solves the same problems as P_1. To build PDI, we combine Prot. 1 with protocol P_2 (for PP_3), or with its generalization P_3 (for PP_{k+1}), constructed in Sec. 3.2. In the following, either of these protocols is denoted by P^*. In Sec. 3.2, P^* is constructed in such a way that $L(P_1) = L(P^*)$. For that, in particular, non-uniform initial states are assumed by the transformations in Sec. 3.2. Here, to achieve only inclusion, we can drop this assumption with the help of Prot. 1.

Thus, the composition PDI is obtained by taking the Cartesian product of the state sets of Prot. 1 and P^*, and by updating the states for each protocol independently. The output of a configuration C in PDI is the output of the configuration $\Pi_{P^*}(C)$ in P^*.

Protocol 2 PDI - deterministic transformation in PP_{k+1} with uniform initial states

Initialization:

Initialize the variables of Prot. 1 and the switch variable c_x of P^* to 1 ($c_x \leftarrow 1$, for every agent $x \in \mathcal{A}$). Initialize the projection of states of PDI on P^* as in P^*.

1: **when** interaction occurs **do**
2: ⟨execute transition of Prot. 1⟩
3: ⟨execute transition of P^*⟩
4: **for all** agent x in the interaction **do**
5: **if** $|T_x| > 1$ **then**
6: $c_x \leftarrow 1$
7: **else if** $T_x = \{t_i\}$ **then**
8: $c_x \leftarrow i$

Theorem 6. *Consider any population $\hat{\mathcal{A}}$ with $\mathbf{n} \geq d + k^2$ and with complete interaction graph. Let P_1 be a non-deterministic population protocol in PP_k (for any integer $k > 1$) on population $\hat{\mathcal{A}}$. Then, the protocol PDI, given by Protocol 2, is the deterministic version of P_1 on $\hat{\mathcal{A}}$ in the model of PP_{k+1} such that $L(PDI) \subseteq L(P_1)$. That is, the deterministic protocol PDI solves the same problems for $\hat{\mathcal{A}}$, in PP_{k+1}, as the non-deterministic protocol P_1, in PP_k.*

Proof. In the composition PDI, P^* reads the variables of Prot. 1 (the content of the set of tokens T), on each interaction (lines 5,7, Prot. 2). However, Prot. 1 neither reads, nor writes in the variables of P^*. Thus, for PDI, the conditions of a *fair composition* [7, 9] holds, as well as Lem. 5. Thus and by line 8, eventually, the requirement of P^* on the switch values c is satisfied. That is, eventually, each value in $[1, m]$ is the value c of at least one agent. Thus, *eventually*, Lem. 1 holds for the projection of PDI on P^*. Then, by Lem. 1, exactly as in the proof of Theorem 2, one proves that given any globally fair execution e_4 of PDI, $\Pi_{P_1}(e_4)$ is a globally fair execution of P_1. Then, the output word of e_4 is in the output language of P_1. □

4.2 Inclusion with Parameterized Transformation

The phenomenon described in the appendix makes difficult to prove the inclusion result by exhibiting a general transformation, in the original model of population protocols. We even conjecture that such a transformation does not exist. Thus, in this section, we present a parameterized transformation, the parameter depending on the transition graph of the non-deterministic protocol that we want to transform for a given population.

Theorem 7. *Given a non-deterministic population protocol P_1 (with a set Q of states) and population $\hat{\mathcal{A}}$ of $\mathbf{n}$ agents, there exists a deterministic population protocol P_2 for population $\hat{\mathcal{A}}$ (with a state space of $\Omega(|Q|^{2\mathbf{n}})$ and using non-uniform initial states) such that $L(P_2) \subseteq L(P_1)$.*

Proof Sketch. Consider the transition graph $G(P_1, \hat{\mathcal{A}})$. Let a *traveling salesmen walk*, TSW, be a closed path in $G(P_1, \hat{\mathcal{A}})$ that starts from the initial configuration, goes through any possible configuration at least once, and gets back to the same initial configuration. As a first step, we assume that $G(P_1, \hat{\mathcal{A}})$ contains TSW. We consider the other case later. TSW can be viewed as a sequence of transitions denoted $S = \sigma_1, \sigma_2, \ldots, \sigma_m$. Note that, for each transition in TSW, the pair of interacting agents is defined. The idea is to construct a population protocol P_2, which is forced to execute a very similar (if, in G, there exist several pathes to reach the same configuration). sequence of transitions S infinitely often.

The state of an agent in P_2 is composed of: a state of P_1; the sequence S; a pointer toward one of the transitions in S; and for each transition in S, an order number and an indicator of a role (initiator/responder). An order number can be 0, which means that the agent has not to execute the corresponding transition. If the order number of the pointed transition is $i \neq 0$, then this transition corresponds to the i^{th} transition, $\sigma_i = (p, p') \to (q, q')$, in S. Moreover, the next transition of the agent (in P_2) is of the type $([p\ i \ldots], [p'\ i \ldots]) \to ([q\ j \ldots], [q'\ k \ldots])$. This transition can be executed only between two agents in states $[p\ i \ldots], [p'\ i \ldots]$ and with appropriate role indicators. After an execution of the transition, the pointer of an agent (participated in the transition) points to the next transition to be executed. This is the next transition in S with a non-zero order number (transition j for initiator, and k for responder, in the example above). The pointer "moves" cyclically in S, ensuring an infinite execution of all the transitions in S.

In the initial configuration of P_2, all agents are in the initial state p_0 of P_1, have the same sequence of transitions S, but have different pointers, different order numbers and roles. This non-uniform information can be provided to agents by, e.g., inputs. Note that in every configuration reached by an execution of P_2, every agent has a unique state and thus, P_2 executes a very similar sequence of transitions in TSW.

It comes directly from the construction that the output word of any globally fair execution of P_2 is the output word of an execution of P_1. So, it remains to prove that the projection on P_1 of any execution of P_2 is globally fair. That comes from the fact that the projection on P_1 of execution of P_2 reach infinitely often all the configurations in TSW, which are by definition all the infinitely often reachable configurations of P_1.

Now, consider the case where there is no TSW in $G(P_1, \hat{\mathcal{A}})$. As a matter of fact, this case is not very different of the previous one. If there are infinite executions, there necessarily exist a final connected component F (see definition in Sec. 2) which admits a TSW for F. Then, P_2 is built by choosing any path f toward F, and by applying the protocol above for $S = fF$, when the pointers in agents move on the transitions of f and then cyclically on F. □

Acknowledgments. The authors would like to thank the reviewers for their thoughtful comments and suggestions.

References

1. D. Angluin, J. Aspnes, M. Chan, H. Jiang, M.J. Fischer, and R. Peralta. Stably computable properties of network graphs. In *DCOSS*, pages 63–74. LNCS 3560, 2005.
2. D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In *PODC*, pages 290–299, 2004.
3. D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *DC*, 18(4):235–253, 2006.
4. D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *DC*, 20 (4):279–304, 2007.
5. D. Angluin, J. Aspnes, M. J. Fischer, and H. Jiang. Self-stabilizing population protocols. *TAAS*, 3(4), 2008.
6. M. Fischer and H. Jiang. Self-stabilizing leader election in networks of finite-state anonymous agents. In *OPODIS*, pages 395–409, 2006.
7. T. Herman. *Adaptivity through Distributed Convergence. (Ph.D. Thesis).* University of Texas at Austin, 1991.
8. M. O. Rabin and D. Scott. Finite automata and their decision problems. 3(2):114, 1959.
9. G. Tel. *Introduction to Distributed Algorithms (2nd ed.).* Cambridge University Press, 2000.

A Appendix

Dropping the "Elasticity" Condition is Difficult

We conjecture that a generic transformation giving the output languages inclusion property, applicable to all non-deterministic population protocols, in the original population protocol model, does not exist. We explain the intuition behind this conjecture.

As noted in the introduction, [5] gives a transformation of any strongly elastic (defined in the introduction) non-deterministic population protocol into a deterministic one, whose output language is included in the output language of the former. Dropping the "elasticity" condition, without introducing empty output values, is difficult.

Consider a non-elastic non-deterministic population protocol P_1 and assume it is a solution to the same problems as a deterministic population protocol P_2. That means that the set of output words of P_2 is included in the set of output words of P_1. As the output values depend on the states of the agents, the states of P_2 are necessarily composite states, with one part in connection with the states of P_1 (ensuring that the output words of P_2 are controlled by P_1), and a second part with supplementary information (counter values, work variables, etc.), whose role is to eliminate the non-determinism. However, the difficulty is not in constructing P_2 in such a way that any execution of P_2 has a strong correspondence with an execution of P_1 (for instance, it can be obtained by ensuring the property that the projection of an execution of P_2 is an execution of P_1). The problem is that, although the execution of P_2 is globally fair for P_2, for the correspondence to be meaningful, the corresponding execution of P_1 has to be globally fair for P_1. Indeed, saying that P_1 solves some problem means that the globally fair executions of P_1 satisfy the specification of the problem, which says nothing about the non-globally fair executions.

To understand why it is not straightforward to overcome the latter problem, consider a configuration C of P_2, which appears infinitely often in an execution E of P_2. In the execution e of P_1 corresponding to E, there is a configuration c appearing infinitely often. From c (in P_1), all the possibilities given by the non-determinism of P_1 can be used (yielding configurations $c_1, c_2, \ldots, c_k$), while from C, the supplementary information (in the state of P_2) forces a particular choice, for being deterministic. With null-transitions (yielding "elasticity"), using all the chose possibilities is easy, since the values for the chose counters and other work variables can be modified by a sequence of transitions, which do not change the projection of the configuration on P_1. However, without null-transitions, the risk is that getting some values for the counters, changes also the projection on P_1. That is, one could have either the right counter values (corresponding to some choice of a non-deterministic rule), but not c as projection; or c as projection, but not with the right counter values. In this case, a possibility of transition given in P_1 would never be given in P_2, involving that the projection of a globally fair execution of P_2 on P_1 would not be globally fair for P_1. Thus, P_2 could not generate the same output words as the globally fair executions of P_1.