Space-Optimal Naming in Population Protocols

Janna Burman[1]
LRI, Université Paris-Sud, CNRS, Université Paris-Saclay, France
janna.burman@lri.fr

Joffroy Beauquier
LRI, Université Paris-Sud, CNRS, Université Paris-Saclay, France
joffroy.beauquier@lri.fr

Devan Sohier
LI-PaRAD, Université de Versailles, Université Paris-Saclay, France
devan.sohier@uvsq.fr

Abstract

The distributed *naming problem*, assigning unique names to the nodes in a distributed system, is a fundamental task. This problem is nontrivial, especially when the amount of memory available for the task is low, and when requirements for fault-tolerance are added.

The considered distributed communication model is *population protocols*. In this model, a priori anonymous and indistinguishable mobile nodes (called agents), communicate in pairs and in an asynchronous manner (according to a fairness condition). Fault-tolerance is addressed through *self-stabilization*, in terms of arbitrary initialization of agents.

This work comprises a comprehensive study of the necessary and sufficient state space conditions for naming. The problem is studied under various combinations of model assumptions: *weak or global fairness*, arbitrary or uniform initialization of agents, existence or absence of a distinguishable agent (arbitrarily initialized or not), possibility of breaking symmetry in pair-wise interactions (*symmetric or asymmetric transitions*). For each possible combination of these assumptions, either an impossibility is proven or the necessary *exact* number of states (per mobile agent) is determined and an appropriate space-optimal naming protocol is presented.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases networks of passively mobile agents, population protocols, deterministic naming, self-stabilization, exact space complexity, tight lower bounds, global and weak fairness

Digital Object Identifier 10.4230/LIPIcs.DISC.2019.9

Related Version A full version of the paper is available at [8], `https://hal.inria.fr/hal-01790517`.

1 Introduction

The population protocol model was introduced as a minimalist model for mobile sensor networks where the computational devices, called agents, communicate by pair-wise interactions without (almost) any control on their communication schedule [1].[2] The model originally assumed that the memory of agents is constant, i.e., independent of the population size n. Due to this, agents are prohibited from storing unique identifiers.

Many limitations have been discovered due to this restriction, and a lot of work has been devoted to study the power added by relaxing it. For example, it was obtained in [10] that a non-semi-linear predicate can be computed starting from $\Theta(\log\log n)$ bits of agent's memory (allowing $\Theta(\log^{O(1)} n)$ identifiers), while the original population protocols compute only exactly the semi-linear predicates [2]. Furthermore, it was shown that using $\Theta(\log n)$

[1] corresponding author
[2] Basically, at each step a pair of agents is scheduled to interact (subject to a fairness condition) and each agent observes the other's state updating its own according to the transition function.

bits per agent, allowing to assign unique identifiers (names) to agents, already permits to compute exactly all the symmetric predicates in the class $NSPACE(n \log n)$ (what is equivalent to the power of $O(n \log n)$ space Turing machine) [15, 10]. Following these results, a comprehensive study in [7] provided a complete hierarchy establishing complexity classes for the cases going from non-identified agents (the original population protocol model) to uniquely identified ones, passing by the case of homonyms.

Another line of works where the possibility of having names plays an important role concerns fault-tolerant population protocols. As a motivating scenario one can think of reliability critical mobile sensor networks (not necessary of a very large scale), which may be hardly accessible and have to recover automatically to the correct behavior after faults. In the framework of *self-stabilization* [13, 3], i.e. when the reliability concerns transient faults (e.g., memory or communication errors), it was shown that a linear (on n) state space is necessary for the realization of many tasks. Interestingly, in these cases, many proposed solutions perform a sort of naming mechanism. This concerns for example the self-stabilizing tasks of leader election [9], counting [5, 16, 4] and deterministic oscillation [11]. In [12], for computing semi-linear predicates while tolerating a known constant number of transient and crash faults, the algorithm approximately divides the population into a predefined number of groups, actually creating named homonyms. Finally, some other fault-tolerant population protocols assume that names are given a priori, e.g., [15, 18].

These observations suggest that the task of naming in population protocols should receive a particular attention. This work presents a comprehensive study of this problem. It focuses on the necessary and sufficient state space conditions for naming under all possible combinations of a set of classical model assumptions, like existence of a leader, *weak* or *global* fairness, uniform or arbitrary initialization and symmetry of transition rules. Note that a choice of a combination affects the type and the level of difficulty of breaking symmetry or of achieving fault-tolerance. These parameters, their interest and effect are all discussed below.

The first parameter is the nature of the assumed fairness, weak or global. The formal definitions and an example illustrating the difference between the two appear in the model section. Intuitively, while global fairness ensures that an infinitely often reachable configuration is unavoidable, weak fairness only ensures that every pair of agents interacts infinitely often. Global fairness can be viewed as a way for modeling randomized systems (without introducing randomization explicitly in the model). This explains why it is generally easier to get solutions under this fairness. However, such randomization cannot be always assumed to be available, in particular in reliability critical systems.

A second parameter is the symmetry nature of the (transition) rules of a protocol. With *symmetric rules*, two interacting agents in the same state stay in identical states. With *asymmetric rules*, they can take different states. This latter assumption was the original one proposed for population protocols and motivated by asymmetric wireless communications. The symmetric rules assumption is more general (weaker), and many motivating scenarios can be found for it in nature inspired population protocols, social networks (when equity is an issue) or in networks with symmetric wireless communication.[3]

A third parameter is the presence or absence of a unique leader (a distinguishable agent),

[3] Note that an asymmetric population protocol can be transformed into a symmetric one using the transformer of [6]. However, this transformer requires global fairness and doubles the number of states per agent. This makes it frequently inadequate for obtaining a space efficient symmetric solution from an asymmetric one (in terms of exact space complexity), and certainly inadequate under weak fairness.

which is obviously also useful for the sake of breaking symmetry. In the context of sensor networks, this agent may represent a (possibly mobile) base station, having augmented resources comparing to the tiny mobile agents. From this perspective and similarly to previous studies (e.g., [20, 4, 16]), we are not concerned with the space complexity of this agent.

A fourth parameter is related to the initialization of system agents, i.e., of the leader (if present) and of the other agents (called here *mobile*). In case of mobile agents, if initialization is assumed, it is always *uniform*, i.e., to the same value for every mobile agent. In case of arbitrary initialization, the given solution stabilizes starting from an *arbitrary* configuration (i.e., resulting from any number of transient faults). In this case it is called self-stabilizing [13]. The weaker initialization assumption is (e.g., only the leader is initialized, or nobody), the stronger the system is against transient faults and the more adapted to repetitive execution of a task (requiring less or no re-initialization).

Finally, the focus of this work is on deterministic protocol design, useful whenever random behavior is inappropriate or deterministic guarantees are required. Moreover, as many previous works [20, 11, 4, 16, 5], we perform an *exact* state space analysis. On the negative side, this requires especially careful analysis and design, in contrast to, e.g., the asymptotic analysis case. On the positive side, the exact analysis is extremely relevant in the cases of particularly memory-limited devices, small sized networks and self-stabilizing protocols. The less volatile memory is used by a self-stabilizing protocol, the less (probabilistically less frequently) it is vulnerable to corruptions.

Contribution

Thus, we investigate the naming problem under all the possible combinations of the parameters described above. All the negative results (impossibilities and lower bounds) are presented in Section 3, while all the positive ones (state-optimal solutions) are given in Section 4. Table 1 gives a synthetic view of these results. For each case, it indicates first the statement establishing the feasibility, either by proving impossibility or by construction of a space-optimal protocol. In the latter case, the table also indicates the optimal (used) number of states and the relevant statement of the lower-bound.

The results are expressed in terms of P - a known upper bound on the number of mobile agents n (i.e., the maximum number of agents destined to be named). This technical assumption is done for dealing with bounded protocols (similarly to the related studies, e.g., [16, 5]). P can be seen as a function of the manufactured memory size in each (undistinguishable) mobile agent.[4] Nevertheless, the results (and the lower bounds in particular) can be equivalently expressed in terms of n, when considering a particular population size.

Notice that, first of all, the table concerns the case of *arbitrarily initialized mobile agents*. Together with that, it is also relevant to the case of *uniform initialization of mobile agents*. It is obvious for the positive results (the protocols stay correct). However, somewhat surprisingly, the impossibilities and lower bounds also hold under this stronger assumption, but with only one exception. The exception concerns symmetric rules under weak fairness and with an initialized leader, i.e., when a leader is present and can be initialized together with the other agents (refer to the table). Then, there is a simple naming protocol with only P states per

[4] The parameter P is used in the protocols, but this explicit usage can be frequently replaced by an alert mechanism announcing that the bound will be shortly reached.

mobile agent (Proposition 14), instead of the necessary $P + 1$ states with arbitrary initialized mobile agents. In contrast with this simple protocol for completely initialized case, the analysis of tight negative and positive results assuming no initialization of mobile agents is much more complex (see Section 3.1 and Propositions 16 and 17).

Additional remarks can be made about the table. First, under weak fairness and without leader, no symmetric deterministic protocol is capable of breaking symmetry, and thus naming is impossible. Second, if asymmetric rules are allowed, in all cases, there is a space-optimal solution with P states. On the contrary, with symmetric rules, the norm is $P + 1$ states, excluding two cases: when there is an initialized leader (in both cases) and (1) either global fairness or (2) uniform initialization of mobile agents is assumed. In both cases, the problem can be solved with P states per agent.

Table 1 Synthesis of the relevant statements establishing the feasibility of naming and the necessary (optimal) state space, under different model parameters. If not stated otherwise, the indicated results hold for both arbitrarily an uniformly initialized mobile agents.

	symmetric rules		**asymmetric rules**
	weak fairness	**global fairness**	**weak/global fairness**
no leader	impossible (Prop. 1)	Prop. 13, with $P + 1$ states (Prop. 3)	Prop. 12, with P states
non-initialized leader	Prop. 16, with $P + 1$ states (Prop. 4)	Prop. 13, with $P + 1$ states (Prop. 4)	Prop. 12, with P states
initialized leader	*non-initialized agents:* Prop. 16, with $P + 1$ states (Theorem 11); *initialized agents:* Prop. 14, with P states	Prop. 17, with P states	Prop. 12, with P states

Note. Due to the space constraints, several proofs and the additional related work discussion have been moved to [8]. Though, the most relevant work is mentioned above.

2 Model and Notations

We adopt the model of population protocols [1] as a basic model. A system consists of a collection $\mathcal{A}$ of pairwise interacting agents, also called *population*. Each agent can represent a sensing and communicating mobile device. Among the agents, there can be a unique distinguishable agent called the *leader* which can be as powerful as needed, in contrast with the resource-limited non-leader agents. The non-leader agents are also called *mobile*, interchangeably. The size of the population n is the number of the mobile agents. It is unknown (a priori) to the agents, contrary to the upper bound parameter $P \geq n$. Each agent has a state taken from a set of states Q (depending on P), the same for all mobile agents, but generally different for the leader.

A *(population) protocol* can be modeled as a finite transition system defined by transitions between *configurations*, where a configuration is a vector of states of all the agents. The transitions between configurations are defined by pairwise interactions between agents. If, in a configuration C, two agents x and y interact (meet), s.t. x is in state p and y in state q, they execute a *transition rule* $(p, q) \to (p', q')$. As a result, x changes its state from p to p' and y from q to q'. The new configuration C', resulting from this state changes, is said to be reachable from C (in one step), denoted by $C \to C'$. If $p = p'$ and $q = q'$, the corresponding transition is said *null* (such transition rules are specified by default), and non-null otherwise.[5] If there is a sequence of configurations $C = C_0, C_1, \dots, C_k = C'$, such that $C_i \to C_{i+1}$ for all $i, 0 \leq i < k$, we say that C' *is reachable* from C, denoted $C \xrightarrow{*} C'$.

The transition rules are *deterministic*, if for every pair of states (p, q), there is exactly one (p', q') such that $(p, q) \to (p', q')$. We consider only deterministic transitions and thus, only *deterministic protocols*. Transitions and protocols can be *symmetric* or *asymmetric*. Symmetric means that, if $(p, q) \to (p', q')$ is a transition rule, then $(q, p) \to (q', p')$ is also a transition rule. In particular, if $(p, p) \to (p', q')$ is symmetric, $p' = q'$. A protocol is called symmetric, if all its transition rules are symmetric, and asymmetric otherwise.

Let $(p_1, q_1) \to (p_2, q_2)$, $(p_2, q_2) \to (p_3, q_3)$, ..., $(p_{k-2}, q_{k-2}) \to (p_{k-1}, q_{k-1})$, $(p_{k-1}, q_{k-1}) \to (p_k, q_k)$ be a sequence of rules of a protocol. Then, we shortly write $(p_1, q_1) \xrightarrow{*} (p_k, q_k)$ to denote the successive application of the rules of the sequence, to the same pair of agents, initially in states p_1 and q_1, leading them to p_k and q_k, respectively. We sometimes call an agent in state p a p-state agent, or just a p-agent.

An *execution* of a protocol is an infinite sequence of configurations and transitions $C_0, t_1, C_1, t_2, C_2, t_3, \dots$ such that C_0 is the starting configuration and for each $i \geq 0$, $C_i \to C_{i+1}$, t_i being the transition between two particular agents used to reach C_{i+1} from C_i. When the actual transitions are irrelevant, we denote the execution by $C_0, C_1, C_2, \dots$. A *segment* or a *sub-execution* is a sub-sequence of an execution. The *trace of transitions* of a sub-execution $C_0, t_1, C_1, t_2, C_2, \dots t_k, C_k$ is a sequence $t_1, t_2, t_3, \dots t_k$ of transitions, and the corresponding *trace of transition rules* is the sequence $r_1, r_2, r_3, \dots r_k$ such that r_i is the transition rule applied in t_i. In a real distributed execution, interactions of distinct agents are independent and could take place simultaneously (in parallel), but when writing down an execution we order those simultaneous interactions arbitrarily.

An execution is said *weakly fair*, if every pair of agents in $\mathcal{A}$ interacts infinitely often. An execution is said *globally fair*, if for every two configurations C and C' such that $C \to C'$, if C occurs infinitely often in the execution, then C' also occurs infinitely often in the execution. This also implies that, if in an execution there is an infinitely often reachable configuration, then it is infinitely often reached [2]. Note that an execution where pairs of agents interact according to some probabilistic distribution is globally fair with probability 1 [17].

A simple example allows to better understand the difference between weak and global (or probabilistic) fairness. Consider a population of 3 agents. Each agent can be white or black, and initially one agent is black and the two others are white. Consider also the protocol in which, when two white agents interact, they both become black and when two agents of different colors interact, they exchange their colors. It is easy to see that there is an infinite weakly fair execution in which there is always one black and two white agents (the black color "jump" indefinitely from agent to agent). On the contrary, every globally fair execution terminates in a configuration in which the 3 agents are black, because otherwise there would be infinitely many configurations during an execution from which the "all black" configuration could be reached, without ever being reached (contradicting global fairness).

A *(static) problem* is defined by a predicate $\mathcal{D}$ on configurations. A population protocol

[5] For simplicity, in some cases, we do not present protocols under the form of transition rules, but under the equivalent form of a pseudo-code.

$\mathcal{P}$ is said to *solve a problem* $\mathcal{D}$, if and only if every execution of $\mathcal{P}$ reaches a configuration satisfying the conditions defining $\mathcal{D}$ and stays in such configurations forever after. When this happens, we say that the protocol has *stabilized* (or *terminated*), and a *terminal configuration* has been reached. A *self-stabilizing protocol* is a protocol that stabilizes from an arbitrary configuration (i.e., from a configuration where *any* agent, *including the leader*, can be in any possible state).

In the *naming* problem, each mobile agent x has a variable, also called a *name*, that eventually does not change and such that no two agents have the same name. Mobile agents having the same state (thus the same name) are called *homonyms.*

We consider *uniform* or *semi-uniform* protocols (cf. [14, 19]) in the sense that all agents, except the leader (whence semi-), are a priori indistinguishable and interact according to the same transition rules. Moreover, given an upper bound P on n, the protocol functions similarly for any n. Thus, given the bound P, by the definition of naming, an obvious lower bound on the state space of a mobile agent (for solving naming) is P.

3 Negative Results

In this section, we clarify some boundaries on the possibility to obtain symmetric naming. They are summarized in Table 1 and useful for establishing the space-optimality (tightness) of the solutions presented in the next section. We proceed from simple to more intricate results, gradually adding assumptions helping in breaking symmetry (making harder to proof impossibilities). We conclude this section by Theorem 11 - a subtle result stating impossibility to get a P state symmetric protocol even with an initialized leader, but non-initialized mobile agents and under weak fairness.

The first result is obtained by observing a completely symmetric weakly fair execution where at each step the population transits from one uniform configuration (all agents are in the same state) to the other, by applying each time a symmetric rule between two homonyms.

▶ **Proposition 1.** *Under weak fairness and without leader (even with a uniform initialization of mobile agents), symmetric naming is impossible in the population protocol model.*

Proof. By contradiction, assume that such a symmetric protocol $\mathcal{PP}$ exists. With or without an initialization, consider a possible starting configuration where each agent is in state s_1 and the population size is even (this also corresponds to a uniform initialization). We build a weakly fair infinite execution of $\mathcal{PP}$ during which no configuration with agents in distinct states is reached. This execution can be described by phases. In the first phase, the agents are matched in pairs and interact accordingly. As the protocol is symmetric, after this first phase, each agent is in some state s_2, the same for all agents. In the next phase, the agents are matched again in pairs, but differently from the previous phase, and interact accordingly. After this phase, each agent is in some state s_3, the same for all agents. The execution continues in such phases such that eventually every agent has interacted with every other. From now on, such interaction sequence is repeated infinitely often, satisfying weak fairness. However, this execution never assigns distinct names to agents. ◀

The following lemma states properties of the transition rules of any P state symmetric naming protocol. Its short proof is given below. The lemma is used in the proofs of the next two propositions: the first one assumes no existing leader, while the second one proves impossibility of a self-stabilizing symmetric naming even with a leader.

▶ **Lemma 2.** *In any symmetric naming protocol using only P states per agent, the only possible non-null transition rules between two non-leader agents are between two homonyms.*

Proof. Otherwise, in a population of P agents, after stabilization (every state in $\{1, \ldots, P\}$ is assigned to some agent), mobile agents would be able to change their states and hence their names. This is a contradiction to the assumed stabilization. ◀

Under the conditions of the next proposition and by the lemma above, the only non-null transitions are between two homonyms (resulting in another two homonyms). Such transitions are useless for eliminating repeated names, implying the result.

▶ **Proposition 3.** *Even with a uniform initialization of agents, but without a leader, and using only P states per agent, it is impossible to obtain symmetric naming in the population protocol model, under weak or global fairness.*

▶ **Proposition 4.** *There is no symmetric naming protocol with P states per agent with an arbitrarily initialized leader (or without it), under weak or global fairness.*

Proof. Assume by contradiction that such a protocol exists. By Proposition 3, a leader is necessary. Consider a population of P mobile agents and a starting configuration C_0 (with possibly uniformly initialized mobile agents) that has homonyms (with states in $\{1, \ldots, P\}$). By Lemma 2, only transitions with a leader can eliminate homonyms (the only possible non-null symmetric transition rules between homonyms create necessarily two other homonyms). Thus, there is a finite execution sequence e, starting from C_0, during which the leader renames interacting mobile agents, and finally stabilizes to a correct naming (where every name from $\{1, \ldots, P\}$ is assigned to some mobile agent). Denote by C_e and by s_e the configuration and the state of the leader, respectively, at the end of e.

Then, assume a starting configuration C'_0 (with possibly uniformly initialized mobile agents, as before) containing only homonyms in state s (in $\{1, \ldots, P\}$) and with the leader in state s_e. There must be a sequence of interactions between the leader and mobile agents starting from C'_0 which ends up with a transition during which an interacting agent changes its name. Such a sequence of interactions exists also starting from C_e (with either weak or global fairness), because in C_e any possible state exists in the population. This contradicts the assumption that the protocol has stabilized starting from C_e in e. Hence, the proposition follows. ◀

3.1 Impossibility of P state symmetric naming of arbitrarily initialized mobile agents, under weak fairness, even with an initialized leader

For proving this stronger impossibility result, let us assume, for the sake of contradiction, that such a solution exists. Thus, denote by $Name$ any symmetric protocol solving the naming problem under weak fairness (for any $n \leq P$), with arbitrarily initialized P-state mobile agents. By Proposition 3, under such conditions, a leader is necessary. Moreover, by Proposition 4, such an agent has to be initialized. So, in this sub-section, we assume such initialized unique leader. In the following, some additional necessary properties of protocol $Name$ are proven and finally imply the impossibility of its existence (see Theorem 11).

Using the lemma below, the next proposition establishes an important property of any protocol $Name$ - the existence of a unique state m, called *sink*. This particular state satisfies the following three conditions: (1) $(m, m) \rightarrow (m, m)$; (2) for every state $s \in Q$, there is a transition rule sequence $(s, s) \xrightarrow{*} (m, m)$; (3) for any $n < P$, no mobile agent is assigned a name m at stabilization (i.e., m does not appear infinitely often in executions with $n < P$).

▶ **Lemma 5.** *Consider any weakly fair execution $e = C_1, C_2, C_3, \ldots, C_j, \ldots$ of $Name$ on a population $\mathcal{A}$ of size $n < P$. There is an integer k such that, for any $j \geq k$, no mobile agent is in a state $m \in Q$ such that there is a sequence of transitions of $Name$ $(m, m) \xrightarrow{*} (m, m)$.*

Proof. Let us assume, by contradiction, that there are infinitely many configurations in e with a mobile agent in state m. Since there is a finite number of agents, there is a particular mobile agent x in $\mathcal{A}$ which is in state m in infinitely many configurations. Let $C_{j_1}, C_{j_2}, C_{j_3}, \ldots$ be these configurations such that $e = e_1, C_{j_1}, e_2, C_{j_2}, e_3, C_{j_3}, \ldots$ W.l.o.g., we choose these configurations such that, in every execution segment e_i, every agent in $\mathcal{A}$ interacts with every other (this is possible with weak fairness).

Now consider a population $\mathcal{A}' = \mathcal{A} \cup \{x'\}$ of size $n+1$. To prove the lemma, we will construct a weakly fair execution e' of $Name$ in population $\mathcal{A}'$ where no agent can distinguish e' from e, and where consequently $Name$ wrongly names the agents. Precisely, in e', x and x' will be simultaneously in state m in infinitely many configurations.

We construct e' based on e. First, we assume that in e', x' is in state m in the starting configuration, and $e' = e'_1, C'_{j_1}, e^m, e'_2, C'_{j_2}, e^m, e'_3, C'_{j_3}, e^m, \ldots$ Every segment e'_i follows exactly the same transition sequence as in e_i. In every segment $e'_{2r+1}, C'_{j_{2r+1}}$ (for $r \geq 0$) the interactions are exactly the same as in $e_{2r+1}, C_{j_{2r+1}}$, and x' does not interact. However, in $e'_{2r}, C'_{j_{2r}}$, all the interactions are as in $e_{2r+2}, C_{j_{2r}}$, but the interactions with x. In this case, x is replaced by x' in the appropriate state, and x does not interact. Finally, e^m is an execution segment where only x and x' interact. They both start in state m, performing the sequence $(m,m) \overset{*}{\rightarrow} (m,m)$. The configurations at the beginning and at the end of e^m are identical. The construction of e' ensures that in every C'_{j_i}, both x and x' are in the state m.

It is easy to verify that e' is possible. In particular, this is because, at the end of every segment e'_i, C'_{j_i}, e^m, both x and x' are in the state m, so they can be exchanged in the following transitions of e'_{i+1}. Moreover, e' is weakly fair, because x' interacts with x in every e^m; in every e'_{2r+1} and e'_{2r+2}, x and x', respectively, interact with every other agent (by the assumption on e_i); and all the other agents interact with all the others infinitely often, by the later arguments and by weak fairness of e.

Finally, in e', $Name$ does not name x and x' differently. This is a contradiction to the assumption that $Name$ is a correct naming protocol. ◀

▶ **Proposition 6.** *In any protocol $Name$, there is a sink state.*

Proof. As $Name$ is symmetric, two interacting agents, both in some state $s \in Q$, execute some symmetric transition of the form $(s,s) \rightarrow (s_1,s_1)$. If they meet several times successively, there is a possible sequence of transitions $(s,s) \rightarrow (s_1,s_1) \rightarrow (s_2,s_2) \rightarrow (s_3,s_3) \ldots$ As mobile agents are finite state, for some $j > i \geq 1$, $s_i = s_j$, i.e. $(s_i,s_i) \overset{*}{\rightarrow} (s_i,s_i)$. By Lem. 5, $s_i = m$ s.t. m does not appear infinitely often in executions with $n < P$. As there are at least $P-1$ states appearing infinitely often in an execution with $n = P-1$, there is at most one such possible state m in a P state protocol. This implies correctness of conditions (2) and (3) of the sink definition.

Finally, by contradiction, if $(m,m) \rightarrow (s,s)$ s.t. $s \neq m$, then the previous part of the proof implies $(s,s) \overset{*}{\rightarrow} (s,s)$. As m is proved (above) to be unique, this is a contradiction, implying the correctness of condition (1) of the sink definition. ◀

From now on, the proof assumes the sink state denoted by m, and shows the impossibility for a particular kind of executions, called here *reduced.* In any segment of a reduced execution, each time a pair of $s \neq m$ homonyms appears, it is immediately *reduced* to m, i.e., by applying the sequence of transition rules $(s,s) \overset{*}{\rightarrow} (m,m)$, whose transitions and by extension the sequence itself are called *(homonym) reducing.* Thus, other transitions can take place only when there are no more homonyms. Naturally, any configuration without any homonyms except those in state m is called *reduced.* Notice that, in a reduced execution, there are non-reduced configurations, but only during the reducing transition sequences. For

example, consider the following reduced sub-execution (where l_i represents a leader state): $[1,2,3,4,m,l_1],(l_1,1) \rightarrow (l_2,2),[2,2,3,4,m,l_2],(2,2) \rightarrow (m,m),[m,m,3,4,m,l_2],(l_2,m) \rightarrow (l_3,3),[m,3,3,4,m,l_3],(3,3) \rightarrow (m,m),[m,m,m,4,m,l_3]$. In this example, the reduced configurations are $[1,2,3,4,m,l_1]$,
$[m,m,3,4,m,l_2]$ and $[m,m,m,4,m,l_3]$.

Forcing a reducing sequence of transitions whenever possible, as in a reduced (sub-) execution, does not prevent an execution from being weakly fair. Thus, we can prove the following corollary.

▶ **Corollary 7.** *Given protocol $Name$, any reduced sub-execution of $Name$ can be prolonged to a reduced weakly fair execution of $Name$, i.e., in which $Name$ stabilizes.*

Proof. First, recall that, by Prop. 6, and Lemma 2, the only non-null-transitions of $Name$ are the homonym reducing transitions between non-m-mobile agents and the transitions with the leader. Given a reduced segment, we prolong the execution by forcing mobile agents to interact with every other agent (including the leader) in a "round-robin fashion" (not necessarily in consecutive interactions). Whenever homonyms are created, this interaction pattern is interrupted by the homonym reducing sequence of transitions, and then resumed after the reduction. The execution constructed that way is weakly fair and uses only transitions of $Name$, hence it stabilizes towards a naming. ◀

The following lemma states a basic property of $Name$, in a population of P agents. It uses the notion of equivalent configurations. Two configurations are *equivalent* if both correspond to the same multi-set of states[6] (e.g., $C_1 = [2,3,2,m,l]$ is equivalent to $C_2 = [2,2,3,m,l]$, where l stands for the leader state). This notion is naturally extended to equivalent executions. The lemma shows that, in a population of P agents, if there is a reduced sub-execution in which some particular state $s \neq m$ never appears (in its reduced configurations), than there is also an *equivalent* sub-execution where a particular m-agent never interacts.

▶ **Lemma 8.** *In a population of P agents, consider a reduced sub-execution $e = CC_1C_2\ldots C_k$ of $Name$, starting from a reduced configuration C and such that no agent in state $s \neq m$ (for some s) exists in any reduced configuration of e. Then, there exists a reduced sub-execution $e' = CC'_1C'_2\ldots C'_k$ of $Name$ in which a particular m-agent x never interacts, and, as in e, no agent in state $s \neq m$ exists in any reduced configuration of e'. Moreover, every configuration C'_i of e' is equivalent to the configuration C_i in e.*

Proof. In a population of P agents, in any reduced configuration (of a reduced execution of $Name$), there is at least one m-state agent. Thus in any reduced configuration of e there are at least two m-state agents. Consider C and call one of these two agents agent x.

Let us construct now e', starting in C, with exactly the same trace of transition rules of e. By Prop. 6, and Lemma 2, the only non-null-transitions are homonym reducing transitions between non-m-mobile agents and the transitions with the leader. By a simple induction below, one can see that every transition rule in a trace of e, step by step, can be executed without participation of agent x, and thus added to e'. Since no transition of e creates an s-agent, no such added transition can create an s-agent.

Thus, starting in C, the first transition of e can be executed with any agent, except x, and thus can be added to e (the base of induction). The resulting configuration C'_1 is

[6] or if one is a permutation of the vector components of the other (recall that a configuration is a vector of states of the agents)

equivalent to C_1. Then, by induction, assume that after k transitions added to e', the reached configuration is equivalent to the one reached after transition k in e (without any s-state agent in a reduced configuration). For $k+1$, if a homonym reducing transition takes place in e, x does not participate (it is already reduced) and thus the same transition can be added to e', and an equivalent configuration is reached. Otherwise, if this is a reduced configuration, there is an additional m-state agent, different from x, so any transition with the leader can be executed excluding x, and an equivalent configuration is reached. By such construction of e', every configuration C'_i is equivalent to C_i in e. ◀

Now, we want to prove that $Name$, in a population of size P, can reach a terminal configuration C (with uniquely named agents and in particular with an agent x in some state $s \neq m$), but such that the leader may be unaware of that. Then, the leader should manage to create the possibly missing s-agent for stabilizing towards a configuration where naming is realized. But, once created, this s-agent can disappear by a reduction with the "hidden" s-agent x. This contradicts the fact that the reached configuration C is terminal (see the proof of Theorem 11). This proof uses the following two technical lemmas.

The first lemma (Lem. 9) states that, in some conditions, from a reduced configuration with an s-agent, a reduced configuration without such an agent is reachable. The second symmetric lemma (Lem. 10) states that, if there is some constrained sub-execution to a latter configuration without an s-agent, then there also exists a particular sub-execution from the configuration without an s-agent to the one with it. We use the following definitions to describe these aspects formally.

A configuration C_1 is said to be *far away* from C_2 *by one state* $s \neq m$ (in agent x), if there is an agent x such that $C_1[x] = m$, $C_2[x] = s \neq m$ and $\forall y \in \mathcal{A} \setminus \{x\}, C_1[y] = C_2[y] \neq s$. Then, C_1 is denoted by C_2^{-s} and C_2 by C_1^{+s}. Agent x is called the *pivot*. For example, $C_1 = [1, m, 3, 4, m, l_1]$ is far away by one state 2 from $C_2 = [1, 2, 3, 4, m, l_1]$, and $C_1 = C_2^{-2}, C_2 = C_1^{+2}$.

▶ **Lemma 9.** *Consider a population of size P and two reduced configurations C_1 and C_1^{-s}, far away by state s, in agent x. Consider a given reduced sub-execution $C_1^{-s}e_1^{-s}C_2$ of $Name$ where: (i) there is no s-agent in every reduced configuration in the segment $C_1^{-s}e_1^{-s}$, (ii) agent x does not interact in $C_1^{-s}e_1^{-s}C_2$, (iii) C_2 is reduced and has exactly one s-agent. Then, there exists a reduced sub-execution $C_1e_1C_2^{-s}$ of $Name$ such that exactly one agent in state s exists in every reduced configuration in C_1e_1, and agent x does not interact in C_1 e_1 C_2^{-s}, except in the very last (s-homonym) reducing sequence.*

Proof. Given a sub-execution $C_1^{-s}e_1^{-s}C_2$, the sub-execution C_1e_1 is constructed, starting from a configuration C_1, by applying exactly the trace of transition rules of $C_1^{-s}e_1^{-s}C_2$, on the population excluding the s-state agent x (recall that agent x does not interact in $C_1^{-s}e_1^{-s}C_2$). At the end of the execution constructed till now, there are exactly two s-state homonyms (x and another s-agent). Now, the transitions reducing these homonyms to m are added to reach the desired configuration C_2^{-s}. In this way, the sub-execution $C_1e_1C_2^{-s}$ is obtained. Notice that agent x interacts only in the very last (s-homonym) reducing sequence. ◀

▶ **Lemma 10.** *Consider a population of size P and two reduced configurations C_1 and C_1^{-s}, far away by state s, in agent x. Consider a given reduced sub-execution $C_1e_1C_2^{-s}$ of $Name$ where exactly one agent in state s exists in every reduced configuration in the segment C_1e_1, and agent x does not interact in C_1 e_1 C_2^{-s}, except in the very last (s-homonym) reducing sequence. Then, there exists a reduced execution $C_1^{-s}e_1^{-s}C_2$ of $Name$ such that there is no s-agent in any reduced configuration in the segment $C_1^{-s}e_1^{-s}$, and C_2 is reduced and has exactly one s-agent.*

Proof. In a given execution $C_1 e_1 C_2^{-s}$ agent x does not interact, except in the very last (s-homonym) reducing sequence. Starting from C_1^{-s} we construct an execution $C_1^{-s} e_1^{-s} C_2$, by using first exactly the same prefix of the trace of transition rules of $C_1 e_1 C_2^{-s}$, until and excluding the very last (s-homonym) reducing sequence. In the given configuration, before this very last reducing sequence, two s-agents necessarily exist, one of them being till now the non-interacting agent x. However, in the sub-execution constructed at this point, x does not interact either, but is in state m, so exactly one s-agent exists at the end of the constructed sub-execution. Hence, C_2 is reached and the required $C_1^{-s} e_1^{-s} C_2$ is obtained. ◀

▶ **Theorem 11.** *Under weak fairness, without the initialization of mobile agents, there is no symmetric naming protocol with P states per agent.*

Proof. By contradiction, assume that such a protocol $Name$ exists. Consider a population of P agents. Assume an agent x that does not communicate (for long enough), while the protocol is stabilizing with only $P-1$ mobile agents. By Proposition 6, when this happens, no agent, except possibly x, is in state m. Thus, consider two possible configurations. In one, C_1, the state of x is m, and in another, x is in state $s \neq m$. In the latter case, reduce the s-state homonyms (to m) (possible by Prop. 6). The reached configuration is C_1^{-s}.

Assume that the actual obtained configuration is C_1^{-s}. By the correctness of $Name$, starting from C_1^{-s}, any execution e stabilizes to a configuration C_* where all agents are in different states and no state changes thereafter. Moreover, by Corollary 7, there is such an execution, which is reduced. Furthermore, any such e can be decomposed s.t. $e = C_1^{-s} e_1^{-s} C_2\ e_2\ C_3^{-s} e_3^{-s} C_4\ e_4\ C_5^{-s} \ldots C_k^{-s} e_k^{-s} C_* C_* \ldots$. For every odd i, no s-agent exists in any reduced configuration of $C_i^{-s} e_i^{-s}$, and C_i^{-s} is reduced. For every even i, exactly one s-state agent exists in every reduced configuration in $C_i e_i$, and C_i is reduced. By Lemma 8, for every odd i, one can choose a segment $C_i^{-s} e_i^{-s}$ such that a particular m-agent x_i never interacts in this segment.

Furthermore, e is chosen such that, for every segment $C_i e_i C_{i+1}^{-s}$ with an even i, a particular s-state agent $y_i \neq x_i$ does not interact, except in the very last (s-homonym) reducing sequence. Let us show (by induction) that such e exists. First, since in $C_i^{-s} e_i^{-s}$, for odd i, there is an m-agent $y \neq x_i$ in every reduced configuration. Thus, in the following C_{i+1} (even $i+1$) configuration, any such agent or other non-m-agent $y_i \neq x_i$ can become an s-agent. This implies that every agent in e has the opportunity to interact repeatedly. Second, by Lemma 9, $C_1 e_1 C_2^{-s}$ exists (such that there is exactly one agent in state s in every reduced configuration of $C_1 e_1$).Thus, starting from C_1, at the end of $C_1 e_1 C_2^{-s}$, no s-agent would exist. Then, by correctness of $Name$, there exists $C_2^{-s} e_2^{-s} C_3$ (where no s-agent exists in $C_2^{-s} e_2^{-s}$), and by Lemma 8, such that, in $C_2^{-s} e_2^{-s}$, a particular m-state agent does not interact, e.g., the pivot agent of C_2^{-s} and C_2. Thus, by Lemma 9, there exists e, such that, in $C_2 e_2 C_3^{-s}$, a particular s-state agent y_2 does not interact, except in the very last (s-homonym) reducing sequence (this proves the base of induction).

Now, one can repeat the same arguments, for any segment $C_i^{-s} e_i^{-s} C_{i+1} e_{i+1} C_{i+2}^{-s}$, for an odd i, in e, and show (by induction) that the chosen e exists. Recall that we assumed that $Name$ has stabilized in C^* and then, the leader has to stop renaming the agents. In addition, C_* is reduced (all agents are distinctly named). Hence, from this point, only null-transitions are possible.

Notice that the conditions of Lemma 9 are satisfied for the segments $C_i^{-s} e_i^{-s} C_{i+1}$ with an odd i, and the conditions of Lemma 10 are satisfied for the segments $C_i e_i C_{i+1}^{-s} e_i^{-s}$ with an even i. Hence, by applying these lemmas repeatedly to the segments of e, one inductively builds the execution segment $e' = C_1 e_1 C_2^{-s} e_2^{-s} C_3 e_3 \dots C_k e_k C_*^{-s}$. However, C_*^{-s} is reduced, and far away by only one state from C_*. No agent except the pivot, can distinguish C_*^{-s} from C_*, since each one is in the same state in both configurations. In particular, $C_*^{-s}[leader] = C_*[leader]$. Thus, and by Lemma 2 and Prop. 6, the only possible transitions with the leader are null transitions, as well as the transitions involving mobile agents (there are no non-m-homonyms). Obviously, in C_*^{-s}, the protocol has not stabilized yet. But, no transition can change the configuration C_*^{-s}. This contradicts the assumption that $Name$ is correct. ◀

4 Positive Results

We start by a proposition that illustrates the power of asymmetric transition rules, compared to symmetric ones. Basically, with asymmetric rules, a leader is not necessary for breaking symmetry, even under weak fairness. Moreover, P states are sufficient and no initialization is necessary, i.e., self-stabilizing space-optimal naming is possible.

The proof is by construction of a protocol with a single asymmetric type of rule, $(s, s) \rightarrow (s, (s + 1) \mod P)$. This idea is known in the literature, e.g., [9, 5]. It aimed at solving other (than naming) problems, but provided naming as a by-product. [9] considers self-stabilizing leader election, assuming that the exact size of the population n is known (an assumption proven to be necessary). Under this assumption, the presented protocol also solves naming. In [5], a similar idea is used to count the arbitrarily initialized mobile agents, assuming an initialized leader, and realizes also naming.

The asymmetric space-optimal naming protocol presented below is proven under more general assumptions (with upper bound P, instead of exact knowledge of n, without a leader, and under both fairness assumptions). Its proof uses the novel technique of *hole* and *hole distance* in a configuration.

▶ **Proposition 12.** *Even if agents cannot be initialized, asymmetric naming (under global or weak fairness) is possible using an optimal number of states (P) per agent and without leader.*

Proof. Consider the following asymmetric protocol with P-state agents and only one type of transition rules: $(s, s) \rightarrow (s, (s + 1) \mod P)$.

To prove its correctness let us use the following definitions. A *hole in a configuration C* is an integer i such that no agent is in state i in C. The *hole distance of an agent, in state i, in a configuration C*, is the minimum positive integer j such that $i + j \mod P$ is a hole, if such an integer j exists, and 0 otherwise. The *hole distance of a configuration C* is the sum of the hole distances of the agents in C. Let f be the function mapping each configuration C to a pair of integers (number of holes in C, hole distance of C).

Let C and C' be two different configurations such that $C \rightarrow C'$. Let us show that, for the lexicographical order, $f(C) > f(C')$. First, remark that C' cannot have more holes than C. If C' has one hole less than C, we are done. If not (C' has the same holes as C), there is an agent in state i in C that has changed its state to $i + 1 \mod P$ in C'. Thus, the hole distance of C' is the hole distance of C minus 1. We are done also in this case.

Since f is upper bounded (e.g., by $(P, P(P - 1))$), there is a sequence of transitions that reaches a configuration from which only null transitions are possible, making no effect on agents' states, that are thus necessarily distinct. Then, naming is achieved. ◀

Now we consider one of the most difficult cases for symmetric protocols – assuming no leader and no initialization - impossible under weak fairness. Recall that global fairness mimics, in some sense, the behavior of randomized environments. The following proposition shows that this pseudo randomization is sufficient for breaking symmetry, and that a distinguishable agent is not needed for that. Thus, we propose below the first *symmetric space-optimal self-stabilizing* naming obtained without a leader (the complete proof is in [8]). Notice, that by Proposition 3, at least $P + 1$ states per agent have to be used in this case.

▶ **Proposition 13.** *Even if agents cannot be initialized and without a leader, symmetric (self-stabilizing) naming under global fairness, for $n > 2$, is possible using $P + 1$ states per agent.*

Proof Sketch. Consider the following symmetric protocol with state space $Q = \{0, 1, \ldots, P\}$ and defined by three types of transition rules:
1. *if* $s \neq P : (s, P) \to (s, (s+1) \mod P)$; **2.** *if* $s \neq P : (s, s) \to (P, P)$; **3.** $(P, P) \to (1, 1)$.

From a configuration with homonyms, rule 2 can be applied repeatedly to obtain a configuration C' where there are only P-state homonyms, and possibly some other uniquely named agents. If, in C', no uniquely named agents exist, let us force transitions using rules 3 and then 1, to create at least one uniquely named agent. Then, rule 2 is applied again, to reach a configuration C with only P-state homonyms and with at least one uniquely named agent. Then, whenever there are still some P-state homonyms in C, pick an agent with a unique name s such that no agent with a unique name $(s + 1) \mod P$ exists. Make the s-agent interact with some P-agent, applying rule 1. The obtained configuration contains one more unique name than in C. If naming is not yet reached, this scenario is repeated until it is reached. Such an execution segment is possible from any configuration with homonyms. Hence, naming is reached in any globally fair execution. ◀

Up to this point, we have covered the possible positive cases assuming that no leader is present. Now, we show that the impossibility results of Section 3 can be circumvented by the assumption of a distinguishable agent. This allows to obtain three space-optimal symmetric protocols: 1) a simple P state protocol with all agents being initialized, including the leader (Prop. 14; its proof is in [8]); and two more intricate protocols: 2) a self-stabilizing one (with $P + 1$ states) under weak fairness (Prop. 16); and 3) a protocol using only P states under global fairness (Prop. 17).

▶ **Proposition 14.** *Given a unique initialized leader, and uniform initialization of mobile agents, symmetric naming is possible using only P states per agent, under weak or global fairness.*

The next two results (Prop. 16 and 17) exploit the existing space-optimal *counting* protocol from [4], which uses P states per mobile agent, and (deterministically and exactly) counts such non-initialized agents under weak fairness, assuming an initialized leader . Let us denote it by $Count_P$. It was not originally intended to be a naming protocol, neither a self-stabilizing one. However, it can be observed that, for the case of $n < P$, it performs (a non self-stabilizing) naming. This property is reflected in Theorem 15 below.

▶ **Theorem 15** ([4]). *Protocol $Count_P$ in [4] solves the counting problem, under weak fairness, for up to P mobile agents, each with P states. Moreover, for any $n < P$, the protocol names (up to $P - 1$) mobile agents with distinct names in $\{1, \ldots n\}$.*

By augmenting the mobile agents' state space to $P + 1$ and adapting $Count_P$ accordingly, one obtains a naming protocol (also correct in the case where $n = P$), though using a non-optimal $P + 1$ number of states. Let us denote the resulting protocol by $Count_{P+1}$. This protocol is adapted here further for solving the naming problem in a self-stabilizing way – Prop. 16, while the protocol of Prop. 17 is based on the original $Count_P$.

For presenting these protocols, some more details of *Count* have to be given. First, note that, in the new protocols here, an appropriate *Count* protocol is a priori executed independently, by every agent. The leader (also assumed in *Count*) manages an estimate for the population size. Let us denote it here by $Count.N$ (or just N when the particular version of *Count* is clear from the context). In the original version of *Count*, N is initialized to 0 and incremented until reaching the actual size n (N is non-decreasing). Each mobile agent x in *Count* has an arbitrary initialized variable $name_x$, which eventually contains a unique name (in $\{1, \ldots, n\}$), with $Count_{P+1}$ (for any $n \leq P$), or with $Count_P$ for any $n < P$. Only the leader can assign a new (non 0) name to an interacting agent in state 0. This state plays the role of the sink state (see definitions in Sect. 3.1). The only action of mobile agents is to reduce homonym states to the sink. Because of that, 0-agents appear along an execution until naming with names in $\{1, \ldots, n\}$ is reached. Notice that, even though naming may not be reached in $Count_P$ (0-agents may persist forever), it terminates, i.e., agents will eventually execute only null-transitions. This happens whenever a reduced configuration satisfying $Count_P.N = n$ is reached (this is used in Protocol 1, Prop. 17).

▶ **Proposition 16.** *Self-stabilizing (every agent state is initialized arbitrary) symmetric naming under weak fairness is possible using $P + 1$ states per mobile agent, given a unique (non-initialized) leader.*

Proof. By Theorem 15, for any $n \leq P$, $Count_{P+1}$ assigns unique names in $\{1, \ldots, n\}$ to mobile agents, if the leader is well initialized. However, to get a self-stabilizing version, one have to abandon this latter assumption (the leader cannot be initialized). Then, it may happen that $Count_{P+1}.N$ starts in a non 0 value and reaches $P + 1$, before the naming (and the correct count) is realized. Notice that in this case, mobile agents in state 0 exist (since a naming is not yet reached).

To overcome this case, we incorporate a reset technique. When a mobile agent x in state 0 ($name_x = 0$) interacts with the leader and the estimate $Count_{P+1}.N$ is bigger than P, the leader resets its internal variables (together with N) to the initialization values of the original $Count_{P+1}$. It is clear that, whenever such a reset is executed, the required naming is eventually and correctly obtained, by the correctness of $Count_{P+1}$. This solves the only problematic case described above. Hence, the proposition follows. ◀

Now, for proving Prop. 17, a protocol using only P states per mobile (non-initialized) agent is constructed. It is done by modifying protocol $Count_P$ and by exploiting the global fairness assumption. The resulting protocol is not self-stabilizing, as the leader is initialized (and otherwise impossible by Prop. 4 - with only P states). Notice that this case is similar to the conditions of Theorem 11 (stating impossibility of naming) except for the fairness condition. In fact, it is not easy to see why a similar reasoning used to prove Th. 11 (roughly the fact that the leader can never detect whether the naming is achieved or not) does not also hold in the current case (with global fairness). Intuitively, this is because the power of global fairness allows to eventually reach a favorable (for stabilization) sub-execution even using a deterministic protocol (though a particular one), while with weak fairness unfavorable sub-executions may persist forever.

▶ **Proposition 17.** *With an initialized leader (without initialization of mobile agents), symmetric naming under global fairness is possible using only P states per mobile agent.*

Proof. The proposed protocol is a modification of $Count_P$ in the code of the leader. The mobile agents are reducing homonyms to the sink state as in the original $Count_P$. The modified code is given below - Protocol 1. For every $n < P$, the new protocol works in the same way as $Count_P$. Hence, by Theorem 15, it eventually stabilizes to naming for every $n < P$. The case of $n = P$ is treated separately, in lines 3 - 8. For this case, a variable *name_ptr* with possible values in $\{0, \ldots, P\}$ is used for indicating to the leader the name to assign to a mobile agent x interacting with it (using variable $name_x$, from the original protocol). The variable *name_ptr* is initialized to 0. Below we consider only the case of $n = P$.

Whenever $n = P$, the leader increments *name_ptr* each time it meets a mobile agent whose name is the current value of *name_ptr*. Otherwise, the agent is named by the value of *name_ptr*, and *name_ptr* is reset.

Let us consider only reduced (to 0) executions (see the definition in Sec. 3.1). From any *non-terminal* configuration, there is the following possible sequence of interactions during which the leader first resets *name_ptr* (if not 0 due to initialization), and then meets the existing $j < P$ uniquely named agents in the increasing order of their names $0, 1, 2, 3, \ldots j-1$. Variable *name_ptr* is then increased to j (≥ 1). After, the leader meets an agent in a state different from j. It names the agent by the current value of *name_ptr* ($= j$), and resets *name_ptr* again. Then, the scenario repeats, until *name_ptr* (and j) reaches P. No value can be changed thereafter, and all agents are named by names in $\{0, \ldots, P-1\}$. By global fairness, this terminal naming configuration is eventually reached. ◀

Algorithm 1 Space-Optimal Naming under Global Fairness (P states per mobile agent).

```
Variables at the leader:
  $name\_ptr$: $[0, \ldots, P]$, initialized to 0
Variable at a mobile agent $x$:
  $name_x$: non-negative integer in $[0, \ldots, P-1]$, initialized arbitrarily
1: when a mobile agent $x$ interacts with the leader do
2:   execute $Count_P$
3:   if $Count_P.N = P \wedge name\_ptr < P$ then
4:     if $name_x = name\_ptr$ then
5:       $name\_ptr \leftarrow name\_ptr + 1$
6:     else
7:       $name_x \leftarrow name\_ptr$
8:       $name\_ptr \leftarrow 0$
9: when two mobile agents $x$ and $y$ interact do
10:  execute $Count_P$
```

5 Conclusion and Perspectives

This paper studies a strong form of symmetry breaking, giving distinct names to indistinguishable agents in population protocols. It provides a comprehensive overview of the results in terms of possibility, impossibility and space optimality, and some insights on the trade-offs between criteria (global vs. weak, symmetric vs. asymmetric, need of a leader, initialization).

A continuation of this work could be the study of the time complexity aspects of naming and, overall, of the trade-offs between time and space. Another perspective would be to consider other forms of symmetry breaking (compact naming, leader election, coloring, two-hop coloring, majority, etc.), under constraints of optimal memory space and requirements of fault-tolerance.

References

1 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. doi:10.1007/s00446-005-0138-3.

2 D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007. doi:10.1007/s00446-007-0040-2.

3 D. Angluin, J. Aspnes, M. J. Fischer, and H. Jiang. Self-stabilizing population protocols. *ACM Trans. Auton. Adapt. Syst.*, 3(4), 2008. doi:10.1145/1452001.1452003.

4 J. Beauquier, J. Burman, S. Clavière, and D. Sohier. Space-Optimal Counting in Population Protocols. In *DISC*, pages 631–646, 2015. doi:10.1007/978-3-662-48653-5_42.

5 J. Beauquier, J. Clement, S. Messika, L. Rosaz, and B. Rozoy. Self-Stabilizing Counting in Mobile Sensor Networks with a base station. In *DISC*, pages 63–76, 2007.

6 O. Bournez, J. Chalopin, J. Cohen, X. Koegler, and M. Rabie. Population Protocols that Correspond to Symmetric Games. *IJUC*, 9(1-2):5–36, 2013. URL: http://www.oldcitypublishing.com/journals/ijuc-home/ijuc-issue-contents/ijuc-volume-9-number-1-2-2013/ijuc-9-1-2-p-5-36/.

7 O. Bournez, J. Cohen, and M. Rabie. Homonym Population Protocols. *Theory Comput. Syst.*, 62(5):1318–1346, 2018. doi:10.1007/s00224-017-9833-2.

8 J. Burman, J. Beauquier, and D. Sohier. Space-Optimal Naming in Population Protocols. Research report, LRI, Université Paris Sud, 2019. URL: https://hal.inria.fr/hal-01790517.

9 S. Cai, T. Izumi, and K. Wada. How to Prove Impossibility Under Global Fairness: On Space Complexity of Self-Stabilizing Leader Election on a Population Protocol Model. *Theory Comput. Syst.*, 50(3):433–445, 2012. doi:10.1007/s00224-011-9313-z.

10 I. Chatzigiannakis, O. Michail, S. Nikolaou, A. Pavlogiannis, and P. G. Spirakis. Passively mobile communicating machines that use restricted space. *Theor. Comput. Sci.*, 412(46):6469–6483, 2011. doi:10.1016/j.tcs.2011.07.001.

11 C. Cooper, A. Lamani, G.i Viglietta, M. Yamashita, and Y. Yamauchi. Constructing self-stabilizing oscillators in population protocols. *Inf. Comput.*, 255:336–351, 2017. doi:10.1016/j.ic.2016.12.002.

12 C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and E. Ruppert. When Birds Die: Making Population Protocols Fault-Tolerant. In *DCOSS*, pages 51–66, 2006. doi:10.1007/11776178_4.

13 E. W. Dijkstra. Self-stabilizing Systems in Spite of Distributed Control. *Commun. of the ACM*, 17(11):643–644, November 1974.

14 S. Dolev, A. Israeli, and S. Moran. Self-Stabilization of Dynamic Systems Assuming Only Read/Write Atomicity. *DC*, 7(1):3–16, 1993.

15 R. Guerraoui and E. Ruppert. Names Trump Malice: Tiny Mobile Agents Can Tolerate Byzantine Failures. In *ICALP (2)*, pages 484–495, 2009. doi:10.1007/978-3-642-02930-1_40.

16 T. Izumi, K. Kinpara, T. Izumi, and K. Wada. Space-efficient self-stabilizing counting population protocols on mobile sensor networks. *Theor. Comput. Sci.*, 552:99–108, 2014. doi:10.1016/j.tcs.2014.07.028.

17 H. Jiang. *Distributed Systems of Simple Interacting Agents*. PhD thesis, Yale University, 2007.

18 Y. Sudo, T. Masuzawa, A. K. Datta, and L. L. Larmore. The Same Speed Timer in Population Protocols. In *ICDCS*, pages 252–261, 2016. doi:10.1109/ICDCS.2016.82.

19 G. Tel. *Introduction to Distributed Algorithms (2nd ed.)*. Cambridge University Press, 2000.

20 H. Yasumi, F. Ooshita, K. Yamaguchi, and M. Inoue. Constant-Space Population Protocols for Uniform Bipartition. In *OPODIS*, pages 19:1–19:17, 2017. doi:10.4230/LIPIcs.OPODIS.2017.19.