Self-stabilizing Leader Election in Population Protocols over Arbitrary Communication Graphs

Joffroy Beauquier, Peva Blanchard, Janna Burman

Self-stabilizing Leader Election in Population Protocols over Arbitrary Communication Graphs

Joffroy Beauquier[1], Peva Blanchard[1⋆], and Janna Burman[1]

LRI, Paris-South 11 University, Orsay, France, {jb,blanchard,burman}@lri.fr

Abstract. This paper considers the fundamental problem of *self-stabilizing leader election* ($\mathcal{SSLE}$) in the model of *population protocols*. In this model, an unknown number of asynchronous, anonymous and finite state mobile agents interact in pairs over a given communication graph. $\mathcal{SSLE}$ has been shown to be impossible in the original model. This impossibility can been circumvented by a modular technique augmenting the system with an *oracle* - an external module abstracting the added assumption about the system. Fischer and Jiang have proposed solutions to $\mathcal{SSLE}$, for complete communication graphs and rings, using an oracle $\Omega?$, called the *eventual leader detector*. In this work, we present a solution for arbitrary graphs, using a *composition* of two copies of $\Omega?$. We also prove that the difficulty comes from the requirement of self-stabilization, by giving a solution without oracle for arbitrary graphs, when an uniform initialization is allowed. Finally, we prove that there is no self-stabilizing *implementation* of $\Omega?$ using $\mathcal{SSLE}$, in a sense we define precisely.

Keywords: leader election, self-stabilization, population protocols, global fairness, oracles

1 Introduction

Leader election and consensus are among the most fundamental problems in distributed computing. Both have been formally proven not to admit any solution under some assumptions and especially under the presence of faults. Consensus is impossible in asynchronous message passing or shared memory systems, even with a single crash fault [14]. Leader election is impossible each time the system is completely symmetrical, involving no identifiers, or is required to be *self-stabilizing* [12], i.e., withstand state-corrupting transient failures (see, e.g., [6,4]). To circumvent these impossibilities, a lot of studies have been performed for devising and defining the (minimum) supplementary information or assumptions needed to solve these problems. Such information generally should be available or possible to retrieve in real systems, allowing practical implementations.

Devising such necessary supplementary information in a modular way can be done using *oracles*. An oracle can be viewed as a black box, which, when asked

⋆ Contact author: LRI, Bât. 650, Université Paris-Sud 11, 91405 Orsay Cedex France. tel: 33 (0)1 69 15 64 32

by the system, provides some type of information, hopefully useful to solve a given problem. A great number of studies, following Chandra and Toueg [10], have been devoted to a specific type of oracles, named *failure detectors*, and allowing to solve consensus with crashes in asynchronous networks. Generally, failure detectors provide a quite precise type of information. It is a list of process identifiers (estimated to have crashed). Obviously, the oracle that gives as few information as possible, that is the *weakest oracle*, is both of theoretical and practical interest. For instance, in their framework, Chandra et al. [9] exhibit the weakest failure detector necessary to solve consensus. This oracle is called the *eventual leader elector* and is denoted by Ω.

Fischer and Jiang [13] introduced a different type of oracles, for solving the leader election problem in the model of tiny, asynchronously mobile and pairwise communicating agents called *population protocols* [3]. In particular, this model was introduced in order to characterize what can be computed with only minimal assumptions in a network of mobile agents. The agents are assumed to be undistinguishable (no identifiers and the same algorithm for all) and memory bounded (actually, constant memory). An agent cannot know with which agent it communicates, nor if the agent it communicates with presently is the same as the agent it communicated with just before. Moreover, no knowledge or an upper bound on the number of agents is available. Such characteristics, make the classical failure detectors, or any variant involving a list or the number of identifiers, not applicable to population protocols. This is one of the reasons why Fischer and Jiang introduced a totally different type of oracle. Their oracle is able to detect the presence or the absence of (at least) one leader. It is denoted by $\Omega?$, in reference to Ω, though it is quite different from a failure detector in the sense that it provides information taken from a global configuration of a system.

Fischer and Jiang studied the possibility to solve *self-stabilizing leader election* ($\mathcal{SSLE}$) over specific communication graphs. They prove that $\Omega?$ helps to solve $\mathcal{SSLE}$ in complete graphs and on rings, while the same problem in complete graphs is proven impossible without oracles [4,7]. After the introduction of $\Omega?$, other oracles for leader election in population protocols appeared in the literature, all based on some information related to global configurations. Michail et al. [16] introduced the *absence detector*, an oracle that indicates which agent states are not present in a configuration, as well as a *covering service* which informs an agent that it has met (communicated with) all the other agents. Intuitively, both are much stronger than $\Omega?$. In [5], we solve $\mathcal{SSLE}$ in arbitrary graphs with $\Omega\$$, an oracle which distinguishes between the presence of zero, one or more leaders in a configuration (in the way that $\Omega?$ does for zero or at least one leader). Additional oracle $W\Omega?$ is introduced in [5]. It is a weaker version of $\Omega?$ that can be used to solve $\mathcal{SSLE}$ over oriented or bounded degree trees.

Our Contribution

Comparing precisely and relating all these different oracles seemed necessary. That is why the first contribution of this paper is to provide a formal framework for dealing with oracles related to $\mathcal{SSLE}$ and encompassing all the particular

oracles described above. Although it may seem complicated at a first glance, this framework is necessary for two reasons. First, it provides a unified formalism, taking into account both oracles that interact with a protocol (like $\Omega?$), and problems, which are independent of any protocol. A second important feature of the framework is a formal definition of the *implementation* of an oracle by another oracle. This step goes through the definition of *compositions* (*sequential, parallel, self*), which, e.g., allows to express that two copies of $\Omega?$, are *stronger* than a single one, or that an oracle that provides information on a three value variable is stronger than an oracle that provides only information on two. Then, based on the notion of implementation, this framework allows to classify some class of leader election oracles under the form of a double hierarchy, which leads to a lattice.

We then show that one of the elements in the lattice, $\Omega?(2,1)$ (a notation which we define in the sequel and which represents two instances of $\Omega?$, giving independently two different outputs), allows to solve $\mathcal{SSLE}$ over any connected communication graph (Sec. 6). The protocol is non trivial and, with its correctness proof, may be considered as the major contribution of this paper. On the contrary, we prove that if the property of self-stabilization is not mandatory, that is if some (uniform) initialization is allowed, leader election can be solved without oracle in any communication graph (Sec. 5). This result confirms the fact that the difficulties for solving $\mathcal{SSLE}$ do come from the tolerance to (transient) failures, modeled by the framework of self-stabilization. In addition, to the best of our knowledge, this is the first leader election population protocol over arbitrary graphs.

All the protocols proposed in the paper assume and require the original *global fairness* of population protocols. We show that, with only local fairness, leader election in arbitrary graphs is impossible even with (uniform) initialization (Sec. 4).

Finally we show that $\Omega?$ cannot be implemented using $\mathcal{SSLE}$ over the family of all graphs, even with multiple copies of $\mathcal{SSLE}$ (Sec. 7). This result is an illustration of what can be done in the proposed framework. It should be put in relation with a result in [5], stating that, over rings, $\Omega?$ and $\mathcal{SSLE}$ are *equivalent*. The paper ends with some open problems (Sec. 8).

Related Work

Self-stabilization was introduced by Dijkstra [12]. A self-stabilizing protocol does not depend on initialization of process states and converges towards a correct behavior from arbitrary starting configurations. Self-stabilization is intended to deal with transient failures, that hit a system punctually, corrupting memory and channel contents. It also deals with dynamic networks, where the topology changes during an execution.

Being an important primitive in distributed computing, leader election has been extensively studied. Below, we mention only the most relevant literature.

Since the introduction of population protocols by Angluin et al. in [2], several studies have been devoted to self-stabilizing leader election in this model. Angluin et al. [4] present a non-uniform $\mathcal{SSLE}$ algorithm for rings in the population

protocol model. They also show in the same paper that there does not exist a $\mathcal{SSLE}$ protocol for general connected networks.

Fischer and Jiang [13] propose the eventual leader detector $\Omega?$ and, using it, present uniform $\mathcal{SSLE}$ protocols for complete graphs and rings. The first protocol works under either a local or global fairness condition, whereas the second requires global fairness. It is also shown that with only local fairness, uniform self-stabilizing leader election in rings is impossible, even with the help of $\Omega?$. Canepa and Potop-Butucaru [8] propose deterministic and probabilistic protocols in arbitrary graphs, assuming $\Omega?$ and different types of local fairness conditions.

Cai et al. [7] show that, in complete communication graphs, n agent-states are necessary and sufficient to solve $\mathcal{SSLE}$, where n is the population size. This result involves that an oracle is necessary for solving $\mathcal{SSLE}$ in population protocols. For the enhanced model of *mediated population protocols* - MPP (allowing an extra memory on every agent pair) [15], the work of Mizogushi et al. [17] shows that $(2/3)n$ agent states and a single bit memory on every agent pair are sufficient to solve $\mathcal{SSLE}$. They also show that there is no MPP that solves $\mathcal{SSLE}$ with any constant agent-states and any constant size memory on each agent-pair, for general n.

Michail et al. [16] introduce the *absence detector*, an oracle for population protocols that indicates which agent states are not present in a configuration, as well as a *covering service* which informs an agent that it has met (communicated with) all the other agents. Intuitively, both are much stronger than $\Omega?$.

Finally, in [5] we define $\Omega\$$ and $W\Omega?$, two oracles respectively stronger and weaker than $\Omega?$, and prove that $\mathcal{SSLE}$ can be solved with $\Omega\$$ over weakly connected communication graphs, with $W\Omega?$ over oriented trees and with $\Omega?$ over weakly connected communication graphs of bounded degree.

2 Model and Definitions

2.1 Population Protocol

We use the same definitions as in [13] with some slight modifications. A network is represented by a directed graph $G = (V, \mathcal{E})$ with n vertices and no multi-edges nor self-loops. Each vertex represents a finite-state sensing device called an *agent*, and an edge $(u, v) \in \mathcal{E}$ indicates the possibility of a communication between two distinct nodes u and v in which u plays the role of the *initiator* and v of the *responder*. The orientation of an edge corresponds to this asymmetry in roles. In this paper, we consider weakly connected networks.

A *population protocol* $\mathcal{A}(D, \mathcal{Q}, Init, X, Y, O, \delta)$ consists of a family of graphs D (the *domain* of the protocol), a finite *state space* $\mathcal{Q}$, a function $Init$ that associates every graph $G(V, \mathcal{E})$ in D with a set $Init(G)$ of *initial configurations* (see below) on G , a finite *input alphabet* X, a finite *output alphabet* Y, an *output function* $O : \mathcal{Q} \to Y$ and a transition function $\delta : (\mathcal{Q} \times X)^2 \to \mathcal{P}(\mathcal{Q}^2)$ that maps any tuple (q_1, x_1, q_2, x_2) to a non-empty (finite) subset $\delta(q_1, x_1, q_2, x_2)$ in

$\mathcal{Q}^2$. A *(transition) rule* of the protocol is a tuple $(q_1, x_1, q_2, x_2, q_1', q_2')$ such that $(q_1', q_2') \in \delta(q_1, x_1, q_2, x_2)$ and is denoted by $(q_1, x_1)(q_2, x_2) \to (q_1', q_2')$. The population protocol $\mathcal{A}$ is *deterministic* if the set $\delta(q_1, x_1, q_2, x_2)$ always has exactly one element.

Given a graph $G(V, \mathcal{E})$ in D and a set Z, an *assignment with values in Z* is a function from V to Z. A *configuration* C is an assignment with values in the state space $\mathcal{Q}$. An *input assignment* (resp. *output assignment*) is an assignment with values in the input alphabet X (resp. *output alphabet* Y). Each configuration C induces an output assignment $O \circ C$ where O is the output function of the protocol. A *trace T with values in Z on the graph $G(V, \mathcal{E})$* is an infinite sequence of assignments with values in Z, i.e., $T = \alpha_0\alpha_1 \dots$ where $\alpha_i : V \to Z$. An *input trace* (resp. *output trace*) is a trace with values in the input alphabet X (resp. the output alphabet Y). The trace $\alpha_0\alpha_1 \dots$ is *constant* if $\alpha_0 = \alpha_1 = \dots$, and it is *uniform constant* if it is constant and for every $u, v \in V$, $\alpha(u) = \alpha(v)$.

Given a graph $G(V, \mathcal{E})$ in D, an *action* is a pair $\sigma = (e, r)$ where r is a rule $(q_1, x_1)(q_2, x_2) \to (q_1', q_2')$ and $e = (u, v)$ an edge of G. Let C, C' be configurations and α be an input assignment. We say that σ *is enabled in* (C, α) if $C(u) = q_1, C(v) = q_2$ and $\alpha(u) = x_1, \alpha(v) = x_2$. We say that (C, α) *goes to* C' *via* σ *in one step*, denoted $(C, \alpha) \xrightarrow{\sigma} C'$, if σ is enabled in (C, α), $C'(u) = q_1', C'(v) = q_2'$ and $C'(w) = C(w)$ for all $w \in V - \{u, v\}$. In other words, C' is the configuration that results from C by applying the transition rule r to the node pair e. We also denote by $(C, \alpha) \to C'$ when $(C, \alpha) \xrightarrow{\sigma} C'$ for some action σ. Given an input trace $T_{in} = \alpha_0\alpha_1 \dots$, we write $C \xrightarrow{*} C'$ if there is a sequence of configurations $C_0C_1 \dots C_k$ such that $C = C_0$, $C' = C_k$ and $(C_i, \alpha_i) \to C_{i+1}$ for all $0 \leq i < k$, in which case we say that C' is *reachable* from C given the input trace T_{in}.

Given a graph G in D, a *virtual execution* E is an infinite sequence of configurations, input assignments and actions $E = (C_0, \alpha_0, \sigma_0)(C_1, \alpha_1, \sigma_1) \dots$ such that $C_0 \in Init(G)$ and for each i, $(C_i, \alpha_i) \xrightarrow{\sigma_i} C_{i+1}$. Such a virtual execution induces an output trace denoted by $O(E)$ defined as $(O \circ C_0)(O \circ C_1) \dots$ where O is the output function of the protocol. We denote by SE the (infinite) suffix of E such that each couple (C, α) (C being a configuration, and α an input assignment) in SE appears infinitely often in SE. This suffix is well-defined because the number of couples (C, α) that occurs finitely often in E is bounded.

We now define fair executions. We first recall two fairness conditions used with population protocols [13]:

(Local Fairness) a virtual execution $(C_0, \alpha_0, \sigma_0)(C_1, \alpha_1, \sigma_1) \dots$ is *locally fair* when, for every action σ, if σ is enabled in (C_i, α_i) for infinitely many i, then $(C_j, \alpha_j) \xrightarrow{\sigma} C_{j+1}$ for infinitely many j.

(Global Fairness) a virtual execution $(C_0, \alpha_0, \sigma_0)(C_1, \alpha_1, \sigma_1) \dots$ is *globally fair* when, for every C, C', α such that $(C, \alpha) \to C'$, if $(C, \alpha) = (C_i, \alpha_i)$ for infinitely many i, then $C' = C_j$ for infinitely many j.

In this paper, unless stated otherwise, an *execution* is a virtual execution that is globally fair. Finally we consider two types of population protocols. A population protocol is *uniformly initialized* if there exists a state q_0 such that every initial configuration is an assignment with values in $\{q_0\}$. In a *non-initialized*

population protocol, the set of initial configurations is the set of all possible configurations.

2.2 Run, Behaviour, Oracle and Implementation

The definitions of runs, behaviours and oracles that we give below, are different from those in [4,13] and are required to obtain a proper framework for defining oracles and establishing relations between them. For instance, in this framework, the oracles are self-implementable, in contrast with the traditional failure detectors' frameworks [11].

A *schedule* on a network $G(V, \mathcal{E})$ is a sequence of edges $S = e_1 e_2 \ldots$, i.e., $e_i \in \mathcal{E}$ for all i. The *schedule S associated with an execution E* is the sequence S of edges that appear in the sequence of actions in E; we also say that E is an *execution with schedule S*.

The following notion of *compatibility of a trace with a schedule* involves that the changes in a trace are only caused by the interactions. A trace $T = \alpha_0 \alpha_1 \ldots$ on G is said to be *compatible* with the schedule $S = (u_0, v_0)(u_1, v_1) \ldots$ on G if, for every i, for every $w \in V - \{u_i, v_i\}$, $\alpha_i(w) = \alpha_{i+1}(w)$. That is, two consecutive assignments can differ only in the assignment values of the two agents in the corresponding edge in the schedule. Note that, by definition, the output trace induced by an execution with schedule S of a population protocol on G, is compatible with S.

(Run). A *run* $R(X, Y)$ with an input alphabet X and output alphabet Y on a network $G(V, \mathcal{E})$ is a triple (T_{in}, T_{out}, S), where T_{in} is a trace with alphabet X on G, T_{out} is a trace with alphabet Y on G and S is a schedule on G such that T_{in} and T_{out} are both compatible with S. The trace T_{in} (resp. T_{out}) is referred to as the *input trace* (resp. *output trace*) *of the run*.

(Behaviour). A *behaviour* B is given by a family D of graphs (the domain of B), an input alphabet X, an output alphabet Y and a function that maps any graph G in D to a set $B(G)$ of runs with input alphabet X and output alphabet Y. Given a population protocol $\mathcal{A}$ with domain D, input alphabet X and output alphabet Y, we define the *behaviour $Beh(\mathcal{A})$ associated with the protocol $\mathcal{A}$* as follows. The domain is D, the input alphabet is X, the output alphabet is Y, and, for any graph G in D, for any run (T_{in}, T_{out}, S) on G, $(T_{in}, T_{out}, S) \in Beh(\mathcal{A})(G)$ if and only if there exists an execution of $\mathcal{A}$ on G with the input trace T_{in}, the output trace T_{out} and the schedule S.

In the following paragraph, we define the notion of *composition* of behaviours. Informally, a *serial* composition uses the output of one behaviour as the input of another behaviour. A *parallel* composition consists in two behaviours being used independently. Finally, a *self* composition uses (a part of) the output of a behaviour as the input to the same behaviour, producing a sort of "feedback". In [13], the self composition is implicitly used, when the oracle $\Omega?$ produces a new input to a protocol based on the output of the same protocol.

Formally, consider two behaviours B_1, B_2 with (respectively) domains D_1, D_2 such that $D_1 \cap D_2 \neq \emptyset$, input alphabets X_1, X_2, and output alphabets Y_1, Y_2. We denote by T_X a trace with values in X. The *parallel composition* $B = B_1 \otimes$

B_2 is the behaviour with domain $D_1 \cap D_2$, alphabets $X_1 \times X_2$, $Y_1 \times Y_2$ such that, for every $G \in \mathcal{F}$, $B(G)$ is the set of runs $((T_{X_1}, T_{X_2}), (T_{Y_1}, T_{Y_2}), S)$ with $(T_{X_1}, T_{Y_1}, S) \in B_1(G)$ and $(T_{X_2}, T_{Y_2}, S) \in B_2(G)$. If $Y_1 = X_2 = U$, the *serial composition* $B = B_2 \circ B_1$ is the behaviour with domain $D_1 \cap D_2$ and alphabets X_1, Y_2 defined as follows. For every $G \in \mathcal{F}$, $B(G)$ is the set of runs (T_{X_1}, T_{Y_2}, S) such that there exists a trace T_U satisfying $(T_{X_1}, T_U, S) \in B_1$ and $(T_U, T_{Y_2}, S) \in B_2$. If $X_1 = U \times V$ and $Y_1 = U \times W$, the *self composition* $B = Self^U(B_1)$ *on* U is the behaviour with domain D_1, alphabets V, W, where, for every $G \in \mathcal{F}$, $B(G)$ is the set of runs $((T_U^{in}, T_V^{in}), (T_U^{out}, T_W^{out}), S) \in B$ such that $T_U^{in} = T_U^{out}$.

Given a family H of behaviours, a behaviour B is a *composition of behaviours from* H if it is a combination of serial, parallel and self composition of behaviours in H.

(Implementation, Comparison). A behaviour B_2 is an *implementation of a behaviour* B_1 *over a family* $\mathcal{F}$ *of graphs* when $\mathcal{F} \subset D_1 \cap D_2$, and for every graph $G \in \mathcal{F}$, $B_2(G) \subset B_1(G)$.

Consider a family $\mathcal{H}$ of behaviours and a family $\mathcal{F}$ of graphs. We say that a behaviour B_1 is *weaker* than a behaviour B_2 over $(\mathcal{F}, \mathcal{H})$, denoted by $B_1 \preccurlyeq B_2$ mod $(\mathcal{F}, \mathcal{H})$, when there exists a composition B involving the behaviour B_2 and behaviours from $\mathcal{H}$ that implements B_1 over $\mathcal{F}$. In other words, if we can compose behaviours from $\mathcal{H}$ with one copy of B_2 to implement B_1, then B_1 is said to be weaker than B_2. This is analogous to the definition in [10] of an oracle being weaker than another one.

The two behaviours are *equivalent* if $B_1 \preccurlyeq B_2 \mod (\mathcal{F}, \mathcal{H})$ and $B_2 \preccurlyeq B_1$ mod $(\mathcal{F}, \mathcal{H})$. We denote this case by $B_1 \simeq B_2 \mod (\mathcal{F}, \mathcal{H})$. When $\mathcal{F}$ and $\mathcal{H}$ are clear from the context, we write $B_1 \preccurlyeq B_2$ and $B_1 \simeq B_2$.

A *problem* and an *oracle* are defined as behaviours. A population protocol $\mathcal{A}$ is a *solution to a problem* P (resp. an *implementation of an oracle* Θ) *using a behaviour* B *over a family* $\mathcal{F}$ *of graphs* if there exists a composition involving the behaviours $Beh(\mathcal{A})$ and B that implements the behaviour P (resp. Θ) over $\mathcal{F}$. Note that with these definitions, if there exists a population protocol in some family $\mathcal{H}$ of protocols that solves the problem P_1 using the problem P_2 over a family $\mathcal{F}$, then P_1 is weaker than P_2 over $(\mathcal{F}, \mathcal{H}^*)$, where $\mathcal{H}^*$ is the family of the behaviours associated with the protocols in $\mathcal{H}$.

Given a behaviour B, we define *the stabilizing behaviour* B_s *associated with* B as follows. It has the same domain D, the same input and output alphabets as B, and for any graph G in D, the set of runs $B_s(G)$ comprises the runs having a suffix[1] belonging to $B(G)$. Given a problem P (resp. an oracle Θ), a population protocol $\mathcal{A}$ is a *self-stabilizing solution to* P (resp. *self-stabilizing implementation of* Θ) if it is non-initialized and it is a solution to the stabilizing problem P_s associated with P (resp. an implementation of the stabilizing oracle Θ_s associated with Θ).

Remark 1. The results in the paper concern the family $\mathcal{F}_{all}$ of all (weakly connected) graphs. Note however that in Sec. 5 and 6, we present protocols that

[1] A run can be seen as a sequence of triples $(\alpha_s, \beta_s, e_s)_{s \in \mathbb{N}}$ where α_s (resp. β_s) is an input (resp. output) assignment and e_s is an edge.

solve the leader election problem in the family of all *strongly* connected graphs. The extension of these protocols to the family of all *weakly* connected graphs is detailed in Appendix A. Roughly speaking, given a weakly connected graph G, one can simulate an execution over the symmetric closure G' of G, which is strongly connected. This can be done by performing, at each interaction, a non-deterministic choice to select which agent plays the role of the initiator and which agent plays the role of the responder. Then, it can be shown that such a non-deterministic execution on G is an execution on G'. It is possible to get a deterministic version of this simulation using the transformer in [4].

3 Specific Behaviours and Oracles

3.1 Eventual Leader Election Behaviour $\mathcal{ELE}$

The domain of the behaviour $\mathcal{ELE}$ is the family $\mathcal{F}_{all}$ of all the graphs, the input alphabet is $\{\perp\}$ (no input), the output alphabet is $\{0,1\}$ and, for any graph $G \in \mathcal{F}_{all}$, a run $(\perp, T, S)$ belongs to $\mathcal{ELE}(G)$ if and only if T has a constant suffix $T' = \alpha\alpha\alpha\ldots$ and there exists a node λ such that $\alpha(\lambda) = 1$ and $\alpha(u) = 0$ for every $u \neq \lambda$. In other words, λ is the unique leader. Note that for all our protocols, there is an implicit output function that maps a state to 1 if it is a leader state, and to 0 otherwise.

In our settings, the (informal) problem of Self-Stabilizing Leader Election ($\mathcal{SSLE}$) is reformulated as the problem of constructing a population protocol that is a self-stabilizing solution to the $\mathcal{ELE}$ problem (using some oracle, if necessary).

3.2 Oracles $\Omega?(k,d)$

We first define, for each $d \geq 1$, an oracle $\Omega?(1,d)$. Its input alphabet is $\{0,1\}$, and its output alphabet is $\{0,\ldots,d\}$. The domain of $\Omega?(1,d)$ is all the graphs. Given an assignment α, we denote by $l(\alpha)$ the number of vertices that are assigned the value 1 by α. Informally, if $l(\alpha) = c$ or $l(\alpha) \geq c$ for all α in an (infinite) execution suffix, then the oracle will eventually permanently output values in $\{c\}$ in the former case, and in $\{c,\ldots,d\}$ in the latter. When $l(\alpha) = 0$ for all α in an (infinite) execution suffix, it is only required that the oracle permanently outputs 0 at one agent at least.

Given a graph G and a run (T_{in}, T_{out}, S) on G, $(T_{in}, T_{out}, S) \in \Omega?(1,d)(G)$ when the following conditions hold. If T_{in} has a suffix $\alpha_0\alpha_1\ldots$ such that $\forall s, l(\alpha_s) = 0$, then T_{out} has a suffix in which at least one agent is permanently assigned the value 0. For every $1 \leq r \leq d-1$, if T_{in} has a suffix $\alpha_0\alpha_1\ldots$ such that $\forall s, l(\alpha_s) = r$, then T_{out} has a suffix equal to the uniform constant trace r. For every $0 \leq r \leq d$, if T_{in} has a suffix $\alpha_0\alpha_1\ldots$ such that $\forall s, l(\alpha_s) \geq r$, then T_{out} has a suffix with values in $\{r, r+1, \ldots, d\}$. Otherwise, any T_{out} (compatible with S) is valid.

For any $k, d \geq 1$, we formally define $\Omega?(k,d) = \bigotimes^k \Omega?(1,d)$. In other words, $\Omega?(k,d)$ is the parallel composition of k copies of $\Omega?(1,d)$. Thus, the input alphabet of $\Omega?(k,d)$ is $\{0,1\}^k$, and the output alphabet is $\{0,\ldots,d\}^k$.

Note that $\Omega?(1,1)$ corresponds to the Fischer and Jiang's oracle $\Omega?$ in [13], while $\Omega?(1,2)$ corresponds to the oracle $\Omega\$$ in [5], except that in the case of absence of a leader, it is only required that at least one agent reports the fact. It is easy to see that the oracles $\Omega?(k,d)$ form a lattice, i.e., if $k \leq k'$ and $d \leq d'$, then $\Omega?(k,d) \preccurlyeq \Omega?(k',d')$ over any graph and behaviour families.

4 Impossibility of Leader Election under Local Fairness with Uniform Initialization

In this section, we show that the eventual leader election problem cannot be solved by any uniformly initialized population protocol under the local fairness assumption.

We first recall the notion of *graph covering* [1,6]. A *fibration* (resp. *opfibration*) between graphs G and B is a graph morphism $\phi : G \to B$ such that for every node b in B, for every node y satisfying $\phi(y) = b$, ϕ induces a bijection between the set of incoming (resp. outgoing) edges at y and the set of incoming (resp. outgoing) edges at b. A *covering* from G to B is a graph morphism from G to B that is both a fibration and an opfibration. The graph G is called the *total* graph, and B is the *base* graph. The *fiber* over a node b in B is the set of nodes in G that are mapped to b via ϕ, which we denote by $\phi^{-1}(b)$. A fiber is *trivial* if it is a singleton. A covering is a *k-covering* if every fiber has k elements, i.e., $\forall b, |\phi^{-1}(b)| = k$. For instance, there is a covering from a ring of size $2 \cdot n$ to a ring of size n obtained by mapping two diametrically opposite nodes to the same node.

The following theorem is inspired by the impossibility result of leader election in the family of rings under local fairness [13] and the ideas developed in [1,6]. Note that the models considered in [1,6] are different from the population protocols. Hence, the results do not directly apply to our case.

Theorem 1. *Let $\mathcal{F}$ be a family of graphs that contains graphs G and B such that there exists a k-covering $\phi : G \to B$ with $k \geq 2$. There is no uniformly initialized population protocol that solves the $\mathcal{ELE}$ problem over the family $\mathcal{F}$ under the local fairness assumption.*

Proof (Sketch). Full details are presented in Appendix B. The result is proved by contradiction. Assume that such a protocol exists, and consider a locally fair execution E_B on B with $\gamma_0\gamma_1\ldots$ being the corresponding sequence of configurations. Thanks to the property of covering, we can lift E_B to get a locally fair execution E_G on G containing configurations g_s such that $g_s = \gamma_s \circ \phi$ for every $s \in \mathbb{N}$. Hence, since ϕ is a k-covering, and since E_B has a suffix during which there is a unique leader, E_G contains infinitely many configurations with $k \geq 2$ leaders; whence a contradiction. □

5 Leader Election under Global Fairness with Uniform Initialization

We establish that, under global fairness, solving the leader election problem on arbitrary communication graphs is possible without oracle, when an uniform initialization is possible (Alg. 1). In other words, there exists a uniformly initialized population protocol that solves the $\mathcal{ELE}$ problem over the family of all graphs under the global fairness assumption. This result highlights the difference between global and local fairness. It also shows that the necessity to use an oracle comes from the requirement of self-stabilization. As explained in Remark 1, our protocol considers *strongly* connected graphs. Each agent x can be leader or non-leader (implemented with a variable $leader_x$) and can hold a white or black token (implemented with a variable $token_x$). Initially, every agent is a leader and holds a black token (uniform initialization). The tokens move through the network by swapping between two agents during an interaction. When two black tokens meet, one of them turns white. When a white token interacts with a leader x, x becomes a non-leader and the token is destroyed.

Algorithm 1: Leader Election with Uniform Initialization

variables *for every agent x:*
  $leader_x$: 0 (non-leader) or 1 (leader)
  $token_x$: $\perp$ (no token), *white* or *black*
initialization: $\forall x, (leader_x, token_x) = (1, black)$ `/* uniform */`
protocol *(initiator x, responder y):*
  if $token_x = token_y = black$ **then**
    $token_y \leftarrow white$
  if $token_x = white \wedge leader_y = 1$ **then**
    $leader_y \leftarrow 0$ `/* y becomes a non-leader */`
    $token_x \leftarrow \perp$ `/* the token is destroyed */`
  $token_x \leftrightarrow token_y$ `/* swap the tokens */`

We consider an execution E of Alg. 1 and prove that there is eventually a unique leader. Recall that SE denotes the infinite suffix of E such that each couple (C, α) in SE occurs infinitely often in SE (see Sec. 2.1). Given a configuration C, let $b(C)$ be the number of black tokens, $w(C)$ the number of white tokens and $l(C)$ the number of leaders in C. In addition, for every agent x, we denote by $C.leader_x$ (resp. $C.token_x$) the value of the variable $leader_x$ (resp. $token_x$) in the configuration C.

Lemma 1. *In each configuration C in every execution E of Alg. 1, $b(C) + w(C) = l(C)$ and $b(C) \geq 1$.*

Proof (Sketch). Full details are presented in Appendix C, Lem. A. The initial configuration satisfies this relation. During an interaction, if no leader is turned

into a non-leader, then the total number of tokens remains constant. When a leader is turned into a non-leader (by a white token), the corresponding token is also destroyed. □

Lemma 2. *For every configuration C in SE, $b(C) = 1$.*

Proof (Sketch). Full details are presented in Appendix C, Lem. B. The global fairness and the fact that two colliding black tokens yield one black token and one white token involves that eventually in E, there is always a unique black token. □

Theorem 2. *In every execution E of Alg. 1, there exists exactly one agent λ such that for every configuration C in SE, $C.leader_\lambda = 1$ and for every agent $\mu \neq \lambda$, $C.leader_\mu = 0$.*

Proof (Sketch). Full details are presented in Appendix C, Th. B. By the previous lemmas, for every configuration C in SE, $l(C) = w(C) + 1$. If a configuration C in SE has $l \geq 2$ leaders, then C also has $l - 1$ white tokens. Thus there is a sequence of steps during which each white token is moved to turn one leader into a non-leader, then reaching a configuration C' with one leader. By global fairness, C' occurs in SE. The configuration C' has exactly one leader, one black token and no white token, thus every subsequent configuration has the same unique leader. □

6 Self-Stabilizing Leader Election using $\Omega?(2,1)$ under Global Fairness

In this section, we exhibit a self-stabilizing solution to $\mathcal{ELE}$ using $\Omega?(2,1)$, i.e., two copies of the Fischer and Jiang's oracle, over the family $\mathcal{F}_{all}$ of all graphs under the global fairness assumption. Alg. 2 below, referred to as the protocol $\mathcal{A}$, is a self-stabilizing solution[2] to $\mathcal{ELE}$ using $\Omega?(2,1)$ over $\mathcal{F}_{all}$. In this protocol, each agent can be a leader or not, and a leader can be either black or white. An agent can also hold a token, and a token can be either black or white. We denote by $\Omega?^l$, resp. $\Omega?^t$, the copy of the oracle $\Omega?$ used to detect the absence of leaders, resp. tokens. As explained in Remark 1, we only consider *strongly* connected graphs.

Whenever the oracle $\Omega?^l$, resp. $\Omega?^t$, outputs 0, a black leader, resp. a black token, is created. The tokens keep moving through the network by swapping between two agents during an interaction. When a black token interacts with a white leader, the leader becomes a non-leader. When a white token interacts with a black leader, the leader becomes white. When a token interacts with a leader having the same color, then both the token and the leader turn into the opposite color.

[2] More formally, the behaviour $Self(\Omega?(2,1) \circ Beh(\mathcal{A}))$ implements the behaviour $\mathcal{ELE}$ (see Sec. 2).

Algorithm 2: Self-Stabilizing Leader Election

variables *agent* x
  $\Omega?^l_x$: input (read-only) from the leader detector
  $\Omega?^t_x$: input (read-only) from the token detector
  $leader_x : \bot$ (non-leader), *white* or *black*
  $token_x : \bot$ (no token), *white* or *black*
protocol *(initiator x, responder y)*
  if $\Omega?^l_x = 0$ **then** $leader_x \leftarrow black$
  if $\Omega?^t_x = 0$ **then** $token_x \leftarrow black$
  if $token_x = black \wedge leader_y = white$ **then** $leader_y \leftarrow \bot$
  if $token_x = white \wedge leader_y = black$ **then** $leader_y \leftarrow white$
  if $token_x = leader_y = black$ **then** $token_x \leftarrow leader_y \leftarrow white$
  if $token_x = leader_y = white$ **then** $token_x \leftarrow leader_y \leftarrow black$
  if $token_x \neq \bot \wedge token_y \neq \bot$ **then** $token_x \leftarrow \bot$
  $token_x \leftrightarrow token_y$

Given an input assignment α for the Alg. 2, we denote by $\alpha.\Omega?^l_x$ (resp. $\alpha.\Omega?^t_x$) the value assigned by α to the (read-only) variable $\Omega?^l_x$ (resp. $\Omega?^t_x$). Similarly, given a configuration C, for every agent x, we denote by $C.leader_x$ (resp. $C.token_x$) the value of the variable $leader_x$ (resp. $token_x$) in the configuration C.

Given a configuration C, let $t(C)$ (resp. $l(C)$) be the total number of tokens (resp. leaders) in C. In C, if an agent x is a leader and an agent y holds a token (x and y not necessarily neighbours), we say that the leader at x and the token at y are *synchronized* if they have the same color. Then, we say that the configuration C *contains a synchronized pair of leader and token*. We consider an execution E of Alg. 2 and its infinite suffix SE (each couple (C, α) in SE occurs infinitely often in SE).

Lemma 3. *For every (C, α) in SE, there is a unique token in C and α assigns 1 to every variable $\Omega?^t_x$, i.e. $t(C) = 1$ and $\forall x, \alpha.\Omega?^t_x = 1$.*

Proof (Sketch). Full details are presented in Appendix D, Lem. C. The oracle $\Omega?^t$ ensures that eventually there is at least one token. Since the number of tokens decreases only when two tokens merge, there is eventually always at least one token; whence eventually $\Omega?^t$ always outputs 1 everywhere. Finally, by global fairness, all the tokens eventually merge, and from that point there is exactly one (circulating) token[3]. □

Lemma 4. *Consider a configuration C that contains a synchronized pair of leader and token such that $l(C) \geq t(C) = 1$. Consider an input assignment α that assigns 1 to every variable $\Omega?^t_x$, i.e., for all x, $\alpha.\Omega?^t_x = 1$. Then for any configuration C' such that $(C, \alpha) \rightarrow C'$, C' contains a synchronized pair of leader and token and $l(C') \geq t(C') = 1$.*

[3] Note that this token may change its color.

Proof. Full details are presented in Appendix D, Lem. D. The assumption on α ensures that no token is created during the step $(C, \alpha) \to C'$. If the unique token meets a leader with which it is synchronized, the leader remains a leader, and both flip their colors. Hence, C' still contains a unique token and some leader synchronized with this token. □

Lemma 5. *There exists a configuration C in SE that contains a synchronized pair of leader and token such that $l(C) \geq t(C) = 1$.*

Proof (Sketch). Full details are presented in Appendix D, Lem. E. We already know that every configuration in SE has a unique token. By contradiction, assume that no configuration in SE satisfies the condition. This means that in every configuration C in SE, every leader (if any) has a color opposite to the color of the unique token. Thanks to $\Omega?^l$, there is a configuration C in SE that has at least one leader, thus $l(C) \geq t(C) = 1$. If the token is white, all the leaders are black, and it is possible to move the token to whiten one of the leaders. The resulting configuration C' contains a synchronized pair of leader and token, and $l(C') \geq t(C') = 1$. By global fairness, C' occurs in SE. On the other hand, if the token is black, it is possible to turn all the white leaders into non-leaders and keep a black token. By global fairness, the resulting configuration C' occurs in SE. Since C' has no leader, thanks to the oracle $\Omega?^l$, a black leader is created at some point in SE. Hence, the corresponding configuration C'' has a synchronized pair of leader and token, and $l(C'') \geq t(C'') = 1$. □

Lemma 6. *For every (C, α) in SE, C contains a synchronized pair of leader and token, $l(C) \geq t(C) = 1$ and for every agent x, $\alpha.\Omega?^l_x = \alpha.\Omega?^t_x = 1$.*

Proof (Sketch). Full details are presented in Appendix D, Lem. F. The result follows from Lemmas 3, 4, 5 and the definition of $\Omega?^l$. □

Theorem 3. *Alg. 2 is a self-stabilizing solution to $\mathcal{ELE}$ using $\Omega?(2,1)$. Precisely, in any execution, there exists exactly one agent λ such that for every configuration C in SE, $C.leader_\lambda \neq \bot$ and for every agent $\mu \neq \lambda$, $C.leader_\mu = \bot$.*

Proof (Sketch). Full details are presented in Appendix D, Th. C. Thanks to Lem. 6, no leader is ever created during SE. In addition, in every configuration in SE, there is a leader synchronized with the token. On one hand, if the token is white, it can whiten all the black leaders, interact with one white leader, become black and turn all the white leaders into non-leaders. On the other hand, if the token is black, it can interact with a black leader (the leader with which it is synchronized) and become white; the next steps are the same as before. In both cases, the resulting configuration has exactly one leader. By global fairness, this configuration occurs in SE. Since no leader is created, there is actually a unique and permanent leader in SE. □

7 Impossibility of Self-Stabilizing Implementation of $\Omega?$ using $\mathcal{ELE}$ under Global Fairness

We show that there is no self-stabilizing implementation of $\Omega?$ (i.e. $\Omega?(1,1)$) using $\mathcal{ELE}$, even if we are allowed to use many copies of $\mathcal{ELE}$, under the global fairness assumption.

Theorem 4. *There is no non-initialized population protocol A such that, for some $k \geq 1$, the composition $B = (\mathcal{ELE} \otimes \cdots \otimes \mathcal{ELE}) \circ Beh(A)$, using k copies of $\mathcal{ELE}$, implements the behaviour $\Omega?$ over the family of all graphs under the global fairness assumption.*

Proof (Sketch). Full details are presented in Appendix E, Th. D. The result is proved by contradiction. Assume that such a protocol A exists. We consider a complete graph G of size $n \geq k+1$. We consider a run of the composition B on G, with a constant input trace $\alpha\alpha\ldots$ that assigns permanently 1 to a unique agent μ. In the corresponding execution E of A, at some point in SE, the output of the different $\mathcal{ELE}$ oracles have stabilized, and all the agents permanently output the value 1. However, by looking at the subgraph obtained by excluding μ, thanks to the assumption on A and the global fairness, there is a configuration in SE in which all the agents but μ output 0; whence a contradiction. □

8 Discussion and Open Problems

Although an abundant literature has been devoted to leader election in the population protocol model, some problems remain open. One of the most challenging is maybe to decide whether or not an oracle is necessary for self-stabilizing solutions to $\mathcal{ELE}$ over rings. Angluin et al. [4], who raised first the issue, present non-uniform solutions (solutions depending on the size of the ring), but the question of an uniform solution has been open for several years. In [13], Fischer and Jiang tackle this issue, provided that the oracle $\Omega?$ is available.

The general framework we proposed allows to express several natural questions. We list some of them here.

In Sec. 3.2, we generalize Fischer and Jiang's oracle and present a lattice of oracles $\{\Omega?(k,d)\}_{k,d\geq 1}$ such that $\Omega?$ coincides with $\Omega?(1,1)$. Analyzing the relations among oracles, which are strong enough to solve leader election, is an interesting way to assess the hardness of this problem. For instance, in a previous work [5], the authors complement Fischer and Jiang's approach by showing that $\mathcal{ELE}$ is equivalent to $\Omega?$ over rings, for non-initialized protocols' behaviours, i.e., each problem is as hard as the other. It seems that the same technique as in [5] would show that all the oracles $\Omega?(k,d)$ are equivalent to $\mathcal{ELE}$ over rings, for non-initialized protocols' behaviours.

In addition, in this paper, we address the issue of comparing $\mathcal{ELE}$ with the oracles $\Omega?(k,d)$ over the family $\mathcal{F}_{all}$ of all graphs, for the family, denoted by PP_{NI}, of non-inialized protocols' behaviours. In Sec. 6, we show that $\mathcal{ELE} \preccurlyeq \Omega?(2,1)$, and in Sec. 7, we show that $\Omega? \not\preccurlyeq \mathcal{ELE}$. Since $\Omega? \preccurlyeq \Omega?(2,1)$, we have the strict

relation $\mathcal{ELE} \prec \Omega?(2,1)$. In addition, it has been shown in [5] that $\Omega?(1,1)$ is sufficient to solve $\mathcal{ELE}$ over the family $BDeg(d)$ of d-bounded degree graphs (for any d), i.e. $\mathcal{ELE} \preccurlyeq \Omega?(1,1) \mod (BDeg(d), PP_{NI})$. It is an open problem to determine whether there exists a self-stabilizing implementation of $\mathcal{ELE}$ using $\Omega?(1,1)$ over $\mathcal{F}_{all}$ and if the relations $\Omega?(k,d) \preccurlyeq \Omega?(k',d') \mod (\mathcal{F}_{all}, PP_{NI})$ ($k \leq k'$ and $d \leq d'$) are strict when $k < k'$ or $d < d'$.

References

1. D. Angluin. Local and global properties in networks of processors. In *12th Symposium on the Theory of Computing*, pages 82–93. ACM, 1980.
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. *Distributed Computing*, 18(4):235–253, 2006.
4. D. Angluin, J. Aspnes, M. J. Fischer, and H. Jiang. Self-stabilizing population protocols. *ACM Trans. Auton. Adapt. Syst.*, 3(4), 2008.
5. J. Beauquier, P. Blanchard, J. Burman, and O. Denysyuk. Oracles for self-stabilizing leader election in population protocols. Technical report, INRIA, 2013. http://hal.archives-ouvertes.fr/hal-00839759.
6. P. Boldi, S. Shammah, S. Vigna, B. Codenotti, P. Gemmell, and J. Simon. Symmetry breaking in anonymous networks: Characterizations. In *ISTCS*, pages 16–26, 1996.
7. 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.
8. D. Canepa and M. G. Potop-Butucaru. Self-stabilizing tiny interaction protocols. In *WRAS*, pages 10:1–10:6, 2010.
9. T. D. Chandra, V. Hadzilacos, and S. Toueg. The weakest failure detector for solving consensus. *J. ACM*, 43(4):685–722, 1996.
10. T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, 1996.
11. B. Charron-Bost, M. Hutle, and J. Widder. In search of lost time. *Inf. Process. Lett.*, 110(21):928–933, 2010.
12. E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. of the ACM*, 17(11):643–644, Nov. 1974.
13. M. Fischer and H. Jiang. Self-stabilizing leader election in networks of finite-state anonymous agents. In *OPODIS*, pages 395–409, 2006.
14. M. H. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, Apr. 1985.
15. O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Mediated population protocols. *Theor. Comput. Sci.*, 412(22):2434–2450, 2011.
16. O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Terminating population protocols via some minimal global knowledge assumptions. In *SSS*, pages 77–89, 2012.
17. R. Mizoguchi, H. Ono, S. Kijima, and M. Yamashita. On space complexity of self-stabilizing leader election in mediated population protocol. *Distributed Computing*, 25(6):451–460, 2012.

Appendix

A From Strongly Connected to Weakly Connected Graphs

In Sec. 5 and 6, we present protocols that solve the leader election problem in the family of all *strongly* connected graphs. We now show how to extend these results to the family $\mathcal{F}_{all}$ of *weakly* connected graphs. Consider a population protocol $\mathcal{A}$. First, we define the non-deterministic protocol $\mathcal{A}^{ND}$ with the same state space and input alphabet as $\mathcal{A}$, and the following transition rules. The rule $(p,x)(q,y) \to (p',q')$ is a rule of $\mathcal{A}^{ND}$ if and only if $(p,x)(q,y) \to (p',q')$ is a rule of $\mathcal{A}$ or $(q,y)(p,x) \to (q',p')$ is a rule of $\mathcal{A}$. In other words, there is a non-deterministic choice that selects which agent is the initiator, and which is the responder, in a rule of $\mathcal{A}$. Given a weakly connected graph G, the symmetric closure G_{sym} of G is necessarily a strongly connected graph. If $E = (C_0,\alpha_0,\sigma_0)(C_1,\alpha_1,\sigma_1)\dots$ is a globally fair execution of $\mathcal{A}^{ND}$ on G, then there is a sequence[4] of actions σ'_i, $i \in \mathbb{N}$, such that the sequence $E' = (C_0,\alpha_0,\sigma'_0)(C_1,\alpha_1,\sigma'_1)\dots$ is a globally fair execution of $\mathcal{A}$ on G_{sym}. Hence if $\mathcal{A}$ solves $\mathcal{ELE}$ on G_{sym} using an oracle Θ such that $\Theta(G) = \Theta(G_{sym})$, then $\mathcal{A}^{ND}$ solves $\mathcal{ELE}$ on G using the oracle Θ. It is then possible to transform $\mathcal{A}^{ND}$ into a deterministic protocol that implements $\mathcal{ELE}$ using Θ over G. It can be done, for instance, by using the general deterministic transformer in [4], since in terms of [4], $\mathcal{A}^{ND}$ implements an *elastic behaviour*.

B Impossibility of Leader Election under Local Fairness with Uniform Initialization

Theorem A *Let $\mathcal{F}$ be a family of graphs that contains graphs G and B such that there exists a k-covering $\phi : G \to B$ with $k \geq 2$. There is no uniformly initialized population protocol that solves the $\mathcal{ELE}$ problem over the family $\mathcal{F}$ under the local fairness assumption.*

Proof. We prove the result by contradiction. Assume that there exists a protocol A that solves the leader election problem with uniform initialization (all agents are initially in the same state q) under local fairness. We first show how to simulate a step of A on B with a specific sequence of steps on G. Then we show how to lift any locally fair execution on B to a locally fair execution on G, and finally we prove the contradiction.

(Simulation). Consider configurations γ, γ' on B and an action $\sigma = ((a,b),(p,q) \to (p',q'))$ enabled in γ such that $\gamma \xrightarrow{\sigma} \gamma'$. Since ϕ is an opfibration, we know that for each node x_i in $\phi^{-1}(a)$ $(1 \leq i \leq k)$, there is a unique edge (x_i, y_i) that is

[4] If $\sigma_i = (u_i, v_i, (q,y)(p,x) \to (q',p'))$ with $(p,x)(q,y) \to (p',q')$ a rule of $\mathcal{A}$, then define $\sigma'_i = (v_i, u_i, (p,x)(q,y) \to (p',q'))$. If $(q,y)(p,x) \to (q',p')$ is a rule of $\mathcal{A}$, then define $\sigma'_i = \sigma_i$.

mapped to (a,b); then let $s_i = ((x_i,y_i),(p,q) \to (p',q'))$ be an action (on G). If there were indices $i \neq j$ such that $y_i = y_j = y$, then y would have two incoming edges that are both mapped to the edge (a,b); whence a contradiction with the fact that ϕ is a fibration. Hence, the y_i's are pairwise distinct (as well as the x_i's by definition).

We denote by u_0 the configuration on G such that $u_0(\phi^{-1}(c)) = \{\gamma(c)\}$ for every c in B. The action s_1 is enabled in u_0 since $(u_0(x_1),u_0(y_1)) = (\gamma(a),\gamma(b)) = (p,q)$. Thus the configuration u_1 such that $u_0 \xrightarrow{s_1} u_1$ is well-defined, and we have $(u_1(x_0),u_1(y_0)) = (p',q')$. The action s_2 is enabled in u_1 since $x_1 \neq x_2$, $y_1 \neq y_2$ and (thus) $(u_1(x_1),u_1(y_1)) = (u_0(x_1),u_0(y_1)) = (p,q)$. Hence, the configuration u_2 such that $u_1 \xrightarrow{s_2} u_2$ is well defined. We can iterate the construction until $i = k$. In the last configuration we have $(u_k(x_i),u_k(y_i)) = (p',q')$ for every $1 \leq i \leq k$. Actually, $u_k(\phi^{-1}(b)) = \{\gamma'(b)\}$ for every agent b in B. In other words, we have simulated the step $\gamma \to \gamma'$ in B by a sequence of steps $u_0 \xrightarrow{*} u_k$ in G.

(Locally Fair Lift). Consider a locally fair execution $E_B = \gamma_0\gamma_1\dots$ of A on the graph B; we have $\forall b, \gamma_0(b) = q$. Thanks to the simulation above, we can build a virtual execution $E_G = g_0 \dots g_1 \dots g_2 \dots$ of A on G such that for every $t \in \mathbb{N}$, for every node $b \in B$, $g_t(\phi^{-1}(b)) = \{\gamma_t(b)\}$. Note that g_0 maps every node in G to q, so E_G is uniformly initialized.

We show that E_G is locally fair. Assume that an action $s = ((x,y),(p,q) \to (p',q'))$ is enabled infinitely often in E_G. The construction of E_G involves that s is enabled in g_i for infinitely many i. But, since $(g_i(x),g_i(y)) = (p,q) = (\gamma_i(\phi(x)),\gamma_i(\phi(y)))$, the action $\sigma = ((\phi(x),\phi(y)),(p,q) \to (p',q'))$ is enabled infinitely many times in E_B. Hence, by local fairness, there are infinitely many i such that $\gamma_i \xrightarrow{\sigma} \gamma_{i+1}$. Then, for infinitely many i, the construction of the sequence $g_i \xrightarrow{*} g_{i+1}$ involves that the action s is triggered during it. Whence E_G is locally fair.

(Contradiction). If A solves the leader election problem, there exists some $i_0 \in \mathbb{N}$ such that for every $i \geq i_0$, the configuration γ_i on B outputs a unique leader at λ. By construction, for every $l \in \phi^{-1}(\lambda)$, $g_i(l) = \gamma_i(\lambda)$. This involves that g_i outputs a leader at k agents (since $|\phi^{-1}(\lambda)| = k$) for infinitely many i. This contradicts the fact that any locally fair execution of A solves leader election on G.

Note that imposing only that ϕ is a fibration (or an opfibration) is not enough to lift a locally fair execution on the base graph to a locally fair execution on the total graph. □

C Leader Election with Uniform Initialization under Global Fairness

We consider an execution E and prove that there eventually is a unique leader. Recall that SE denotes the infinite suffix of E such that each couple (C,α) in SE occurs infinitely often in SE (see Sec. 2). Given a configuration C, let $b(C)$ be the number of black tokens, $w(C)$ the number of white tokens and $l(C)$ the number of leaders in C. In addition, for every agent x, we denote by $C.leader_x$

(resp. $C.token_x$) the value of the variable $leader_x$ (resp. $token_x$) in configuration C.

Lemma A *In each configuration C in an execution E of Alg. 1, $b(C)+w(C) = l(C)$ and $b(C) \geq 1$.*

Proof. In the initial configuration, $b(C) = l(C) = n$ the number of agents, and $w(C) = 0$. We show that for any configuration C satisfying the property, any configuration C' such that $C \to C'$, C' satisfies the property. In the algorithm, the swapping of tokens (line 11) does not modify the number of tokens nor the number of leaders. If line 7 is executed, then $b(C') = b(C)-1 \geq 1$ (the condition in the if statement implies $b(C) \geq 2$), $w(C') = w(C)+1$ and $l(C') = l(C)$; whence $b(C') + w(C') = l(C')$. If lines 9 and 10 are executed, then $b(C') = b(C) \geq 1$, $w(C') = w(C) - 1$ and $l(C') = l(C) - 1$; whence $b(C') + w(C') = l(C')$. Hence, in all cases, C' also satisfies the property. □

Lemma B *For every configuration C in SE, $b(C) = 1$.*

Proof. First note that, since no black token is ever created in Alg. 1, if $C \to C'$, then $b(C) \geq b(C')$. Hence, the number of black tokens cannot increase during SE. Assume that there is a configuration C in SE such that $b(C) = t \geq 2$. By global fairness, there is a configuration in SE where two black tokens are in two neighboring nodes. From this configuration, there is a reachable configuration C' resulting from the interaction of these two neighbors. In C', $b(C') \leq t-1 < b(C)$. The global fairness ensures that C' is in SE. By the first remark, C cannot occur in SE after the first occurrence of C'. This is a contradiction with the definition of SE. □

Theorem B *In any execution E of Alg. 1, there exists an agent λ such that for every configuration C in SE, $C.leader_\lambda = 1$ and for every agent $\mu \neq \lambda$, $C.leader_\mu = 0$.*

Proof. We show by contradiction that for every C in SE, $w(C) = 0$. Assume that there exists a C such that $w(C) \geq 1$. Since $b(C) = 1$, $l(C) = w(C) + b(C) = w(C) + 1 \geq 1$. By global fairness, there is a configuration in SE where a white token and a leader are in two neighbouring nodes. From this configuration, there is a reachable configuration C' resulting from the interaction of these two neighbours such that $l(C') < l(C)$. The global fairness ensures that C' is in SE. Since C is also in SE, there must be a sequence of steps $C' \xrightarrow{*} C$. During this sequence, a leader must be created. This is impossible since no leader is ever created. Then, $w(C) = 0$ for every C in SE. This implies that $l(C) = w(C) + b(C) = 0 + 1 = 1$ for every C in SE. Since the variables $leader_x$'s are never swapped, there exists an agent λ such that for every configuration C in SE, $C.leader_\lambda = 1$ and for every agent $\mu \neq \lambda$, $C.leader_\mu = 0$. □

D Self-Stabilizing Leader Election using $\Omega?(2,1)$ under Global Fairness

Given an input assignment α for the Alg. 2, we denote by $\alpha.\Omega?^l_x$ (resp. $\alpha.\Omega?^t_x$) the value assigned by α to the (read-only) variable $\Omega?^l_x$ (resp. $\Omega?^t_x$). Similarly, given a configuration C, for every agent x, we denote by $C.leader_x$ (resp. $C.token_x$) the value of the variable $leader_x$ (resp. $token_x$) in configuration C.

Given a configuration C, let $t(C)$ and $l(C)$ be the total number of tokens and leaders respectively in the configuration C. In C, if an agent x is a leader and agent y holds a token (x and y not necessarily neighbours), we say that the leader at x and the token at y are *synchronized* if they have the same color. We say that the configuration C *contains a synchronized pair of leader and token* if there exist a leader at some agent and a token at another agent that are synchronized. We consider an execution E of Alg. 2. Recall that SE denotes the infinite suffix of E such that each couple (C,α) in SE occurs infinitely often in SE (see Sec. 2).

Lemma C *For every (C,α) occurring in SE, there is a unique token in C and α assigns 1 to every $\Omega?^t_x$ variable, i.e. $t(C)=1$ and $\forall x, \alpha.\Omega?^t_x = 1$.*

Proof. Assume first that for every (C,α) in SE, $t(C)=0$. Then by the definition of $\Omega?^t$, for every (C,α) in SE, $\alpha.\Omega?^t_x = 0$ for every agent x. By line 8, a token is created at some point during SE; whence a contradiction. Hence, there exists (C',α') in SE such that $t(C') \geq 1$. Since the only way to reduce the number of tokens is by mergin two existing tokens (line 13), for every configuration C such that $(C',\alpha') \to C$, $t(C) \geq 1$. Hence, for every couple (C,α) in SE, $t(C) \geq 1$. The definition of $\Omega?^t$ involves that for every (C,α) in SE, $\alpha.\Omega?^t_x = 1$ for every agent x. This disables the creation of token during SE. Thus, the number of tokens cannot increase during SE. Actually, since each couple (C,α) occurs infinitely often in SE, the number of tokens during SE is constant, say t_0. The previous argument shows that $t_0 \geq 1$. Assume that $t_0 \geq 2$. Then, by global fairness, there is a configuration in SE in which two tokens are located at two neighbouring nodes. From this configuration, there is a reachable configuration C' resulting from the interaction of these two neighbours, such that $t(C') \leq t_0 - 1$. The global fairness ensures that C' is in SE; whence a contradiction. Hence, $t_0 = 1$, i.e., there is a unique token during SE. □

Lemma D *Consider a configuration C that contains a synchronized pair of leader and token, and such that $l(C) \geq t(C) = 1$. Consider also an input assignment α that assigns 1 to every variable $\Omega?^t_x$, i.e., for all x, $\alpha.\Omega?^t_x = 1$. Then for any configuration C' such that $(C,\alpha) \to C'$, C' contains a synchronized pair of leader and token and $l(C') \geq t(C') = 1$.*

Proof. In Alg. 2, if line 7 is executed, then the number of leader increases. Line 8 is not executed since $\alpha.\Omega?^t_x = 1$ for every x.

If line 9 is executed, then $l(C') = l(C) - 1$ and $t(C') = t(C) = 1$. Since C contains a synchronized pair of leader and token and since the unique token is

black in C, there must be a black leader in C (not involved in the interaction). Thus $l(C) \geq 2$, $l(C') \geq t(C') = 1$ and C' also contains a synchronized pair of leader and token.

If line 10 is executed, then $l(C') = l(C)$ and $t(C') = t(C) = 1$, whence $l(C') \geq t(C') = 1$. Since C contains a synchronized pair of leader and token and since the unique token is white in C, there must be a white leader in C (not involved in the interaction). Hence, C' also contains a synchronized pair of leader and token.

If line 11 is executed, then $l(C') = l(C)$ and $t(C') = t(C) = 1$, whence $l(C') \geq t(C') = 1$. The interaction involves a synchronized pair of leader and token, and since both the leader and the token flip their color, C' also contains the same synchronized pair of leader and token. The same argument applies for line 12.

Finally, line 13 cannot be executed since $t(C) = 1$, and line 14 just swap the token values. Therefore, in all cases, C' contains a synchronized pair of leader and token and $l(C') \geq t(C') = 1$. □

Lemma E *There exists a configuration C occurring in SE that contains a synchronized pair of leader and token, and such that $l(C) \geq t(C) = 1$.*

Proof. We prove the result by contradiction. By Lem. C, we already know that every configuration in SE contains a unique token. Hence, assume that, for every configuration C in SE, any leader in C (if any) does not have the same color as the (unique) token in C. Note that, if every configuration C in SE has no leader, then the definition of $\Omega?^l$, the global fairness and the rules of the protocol involve that a (black) leader is created at some point in SE; whence a contradiction. Hence, there exists a configuration C in SE which has at least one leader, $l(C) \geq t(C) = 1$.

By our hypothesis, every leader in C has the same color, opposite to the color of the token. Consider the case where the token is white. Thus all the leaders in C are black. Whatever the sequence of input assignment is, it is possible to reach from C a configuration C' with one white leader and one white token, simply by moving the white token towards one of the black leaders, and apply the rule of the protocol that turns this leader white. The configuration C' has a synchronized pair of leader and token, and $l(C') \geq t(C') = 1$. By the global fairness, C' must belong to SE; whence a contradiction.

Consider the case where where the token is black. Thus all the leaders in C are white. By moving the token, it is possible to turn all the leaders into non-leaders. Hence, there exists a configuration C' occurring in SE with no leaders and one black token. Now since C occurs in SE, it occurs infinitely many times in SE, and there is a sequence of steps $(C', \alpha') \dots (C, \alpha)$ in SE. During this sequence, a leader is created. Before this creation, the unique token stays black since it interacts with no leader. The rules of the protocol involve that the first created leader is black. Hence, there exists a configuration C'' in SE which contains a synchronized pair of leader and token, and such that $l(C'') \geq t(C'') = 1$; whence a contradiction. □

Lemma F *For every (C, α) in SE, C contains a synchronized pair of leader and token, $l(C) \geq t(C) = 1$ and for every agent x, $\alpha.\Omega?^l_x = \alpha.\Omega?^t_x = 1$.*

Proof. By Lem. C, we already know that for every (C, α) in SE, $t(C) = 1$ and for every agent x, $\alpha.\Omega?^t_x = 1$. Also by Lem. E, we know that there exists a (C, α) in SE, such that C contains a synchronized pair of leader and token, and $l(C) \geq t(C) = 1$. These two results, and Lem. D ensure that every (C, α) in SE contains a synchronized pair of leader and token, and $l(C) \geq t(C) = 1$. Then, the definition of $\Omega?^l$ involves that every input assignment α occurring in SE is such that for all x, $\alpha.\Omega?^l_x = 1$. □

Theorem C *Alg. 2 is a self-stabilizing solution to $\mathcal{ELE}$ using $\Omega?(2, 1)$. Precisely, there exists an agent λ such that for every configuration C in SE, $C.leader_\lambda \neq \perp$ and for every agent $\mu \neq \lambda$, $C.leader_\mu = \perp$.*

Proof. By Lem. F, we know that during SE, the leader detector $\Omega?^l$ outputs 1 everywhere. Hence, no leader is ever created during SE. This involves that the number of leaders (greater than or equal to 1) cannot increase during SE. Actually, since each (C, α) in SE occurs infinitely often in SE, the number of leaders is constant during SE. We denote by c this constant; we already know that $c \geq 1$.

Assume that $c \geq 2$. Consider a configuration C occurring in SE. We know that C contains a synchronized pair of leader and token and that $l(C) = c \geq 2$, $t(C) = 1$. We now describe scenarios that produce a configuration C' out of C, such that C' contains a unique leader (synchronized with the unique token).

Case (a). The unique token in C is black. There must be a black leader since C contains a synchronized pair of leader and token. By global fairness, it is possible to move the token near this leader, and to turn them both white. Then we come down to case (b).

Case (b). The unique token in C is white. By moving the token to meet every black leaders, we can turn all the black leaders white. Then by global fairness, we can assume that there are no black leaders in C. Still by global fairness, the following sequence of moves is possible. First, the white token meets a white leader and they both turn black. Then the black token successively meets the white leaders and turn them into non-leaders. The resulting configuration has a unique (black) leader (and a unique black token). The global fairness ensures that this configuration occurs in SE; whence a contradiction with the fact that the number of leaders is $c \geq 2$.

Therefore, $c = 1$, i.e., there is a unique leader in every configuration during SE. Since every configuration in SE contains a synchronized pair of leader and token, in each configuration, the unique leader must be synchronized with the unique token. Since a leader cannot be turned into a non-leader by a token with which it is synchronized, the unique leader is the same for every configuration in SE. Precisely, there exists an agent λ such that for every configuration C in SE, $C.leader_\lambda \neq \perp$ and for every agent $\mu \neq \lambda$, $C.leader_\mu = \perp$. □

E Impossibility of Self-Stabilizing Implementation of $\Omega?$ using $\mathcal{ELE}$ under Global Fairness

Theorem D *There is no non-initialized population protocol A such that for some $k \geq 1$, the composition $B = (\mathcal{ELE} \otimes \cdots \otimes \mathcal{ELE}) \circ Beh(A)$ with k copies of $\mathcal{ELE}$ implements the behaviour $\Omega?$ over the family of all graphs under the global fairness assumption.*

Proof. We prove the result by contradiction. Assume that protocol A exists and consider a complete graph G of size $n \geq k+1$. Let (T_{in}, T_{out}, S) be a run of the behaviour B on G. By definition, there exist traces $T_1, \ldots, T_k$ such that $(\perp, T_i, S)$ is a run of the i-th copy of $\mathcal{ELE}$ on G, and an execution E of A on G with an input trace $IT = (T_1, \ldots, T_k, T_{in})$ and a schedule S that induces the output trace T_{out}.

By the definition of $\mathcal{ELE}$, each T_i eventually permanently assigns 1 to a unique agent λ_i and 0 to every other; we denote by β_i this assignment. Note that the λ_i's are not necessarily distinct. We choose the trace T_{in} to be the constant trace $\alpha\alpha\ldots$ where α assigns 1 to some agent $\mu \notin \{\lambda_1, \ldots, \lambda_k\}$ and 0 to every other. By the assumption on A, the output trace T_{out} has a suffix equal to the uniform constant trace 1. Thus, for every couple (C, β) in SE, $\beta = (\beta_1, \ldots, \beta_k, \alpha)$ and the output associated with C assigns 1 to every agent. Now, consider such a couple (C, β) in SE. If we restrict (C, β) to the network G^c, obtained from G by eliminating the node μ, we obtain a configuration and input assignment (C^c, β^c). The agents λ_i are still the unique agents to be assigned the value 1 by β_i^c respectively, and α^c assigns 0 to every agents in G^c. Since the protocol must be self-stabilizing, there is a sequence of steps, involving all the agents but μ and having the input assignment β^c at each step. This leads to a configuration C'^c that outputs 0 on every agent in G^c.

This involves that there is a sequence of steps $(C, \beta)(C_1, \beta)(C_2, \beta)\ldots(C', \beta)$ such that C' outputs 1 on agent μ and 0 on every other agent. The global fairness ensures that C' occurs in SE; whence a contradiction. □