Deterministic Self-Stabilising Leader Election for Programmable Matter with Constant Memory

Jérémie Chalopin
Aix Marseille Univ, CNRS, LIS, Marseille, France

Shantanu Das
Aix Marseille Univ, CNRS, LIS, Marseille, France

Maria Kokkou
Aix Marseille Univ, CNRS, LIS, Marseille, France

Abstract

The problem of electing a unique leader is central to all distributed systems, including programmable matter systems where particles have constant size memory. In this paper, we present a silent self-stabilising, deterministic, stationary, election algorithm for particles having constant memory, assuming that the system is simply connected. Our algorithm is elegant and simple, and requires constant memory per particle. We prove that our algorithm always stabilises to a configuration with a unique leader, under a daemon satisfying some fairness guarantees (*Gouda fairness* [27]). We use the special geometric properties of programmable matter in 2D triangular grids to obtain the first self-stabilising algorithm for such systems. This result is surprising since it is known that silent self-stabilising algorithms for election in general distributed networks require $\Omega(\log n)$ bits of memory per node, even for ring topologies [20].

2012 ACM Subject Classification Computer systems organization → Fault-tolerant network topologies; Computing methodologies → Self-organization

Keywords and phrases Leader Election, Programmable Matter, Self-Stabilisation, Silent, Deterministic, Unique Leader, Simply Connected, Gouda fair Daemon, Constant Memory

Digital Object Identifier 10.4230/LIPIcs.DISC.2024.13

Related Version *Full Version*: `https://arxiv.org/abs/2408.08775` [11]

Funding This work has been partially supported by ANR project DUCAT (ANR-20-CE48-0006).

1 Introduction

Leader election (LE), introduced by Le Lann [32], allows to distinguish a unique process in the system as a *leader*. The leader process can then act as an initiator or a coordinator, for solving other distributed problems. Thus, election algorithms are often used as building blocks for many problems in this domain. We are interested in deterministic election algorithms that are *self-stabilising*. Since the seminal work of Dijkstra [18], the self-stabilisation paradigm has been thoroughly investigated (see [19] for a survey). A distributed algorithm is self-stabilising if when executed on a distributed system in an arbitrary global initial configuration, the system eventually reaches a legitimate configuration. Self-stabilising protocols are able to autonomously recover from transient memory failures, without external intervention. A self-stabilising algorithm is *silent* if the system always reaches a configuration where the processes no longer change their states. In the self-stabilising setting, LE is particularly important, as many self-stabilising algorithms rely on the existence of a distinguished node.

The concept of silent self-stabilising algorithms is also related to *proof-labelling* schemes [31] where each node is given a local certificate to verify certain global properties of the system (e.g., the existence of a unique leader). Each node can check its own certificate and those of its neighbours to verify it is in a correct configuration. If the global configuration is incorrect, at least one node should be able to detect an inconsistency using the local certificates. In this case, this node will change its state, leading its neighbours to change their states and so on, until the system stabilises to a correct configuration. Blin et al. [5] proved that from any proof-labelling scheme where each process has a certificate of size ℓ, one can build a silent self-stabilising algorithm using $O(\ell + \log n)$ bits of memory per process, where n is the number of processes in the network. One of the standard techniques for self-stabilising LE is to build a spanning tree rooted at the leader, with all other nodes pointing towards their parent in the tree. In order to detect cycles when the system is in an incorrect state, the local certificate at each node includes the hop-distance to the root, in addition to the pointer to the parent. So the size of the certificate depends on the size of the system.

Here, we consider *programmable matter* (PM) systems which are distributed systems consisting of small, intelligent particles that connect to each other and can autonomously change shapes according to input signals. Such systems should be scalable to arbitrary sizes, so the particles have constant size memory independent of the size of the system, similar to finite state automata. This requirement also implies that the particles are anonymous (i.e., do not have unique identifiers) and all communication is limited to $O(1)$ size messages. One well-studied model for PM is the *Amoebot* model [15] where particles operate on a triangular grid (see Section 1.2). LE is a well studied problem in this model. When the system is simply connected, there are stationary deterministic algorithms for election based on the *erosion* approach [17] where the algorithm starts by deactivating particles on the boundary and moving inwards, until the last active node becomes the leader. This approach works under the minimum assumptions on the system and is the inspiration for our work.

Tolerating faults is important for PM, however none of the existing algorithms for election in these systems are self-stabilising. The question is: given the constant memory of particles, is it still possible to obtain a self-stabilising algorithm for PM, using other properties of such systems? We answer this question in the affirmative for *simply connected* PM systems, showing that in this case, a deterministic silent self-stabilising algorithm for LE is indeed possible. We use the property of such systems that there is a unique boundary in the system that is well defined, such that any particle can determine whether it is on the boundary.

1.1 Our results

We present a silent self-stabilising, deterministic, stationary, election algorithm for constant-memory particles, in a simply connected system. We prove our algorithm always stabilises to a unique leader configuration, under a sequential scheduler with some fairness guarantees.

We first present a proof labelling scheme ensuring the existence of a unique leader. Our certificate orients the edges of the network and a configuration is valid when: every edge is oriented, outgoing edges appear consecutively around each particle and there are no directed triangles. Note that our certificate does not ensure that the global orientation of the network is acyclic. However, using the geometric properties of the configuration, we are able to show that any valid configuration has a unique sink. As we are interested in a constant memory algorithm, one cannot transform our proof labelling scheme into a self-stabilising algorithm using [5]. However, we design a very simple algorithm to orient the edges of the system. We show that under our fairness assumption, one always reaches a valid configuration and that this configuration contains a unique sink that is designated as the leader.

Following the classification of [21], our scheduler is Gouda fair [27]: for any configuration C that appears infinitely often in the execution, any successor C' of C also appears infinitely often. Since each particle has constant memory, there exists only a finite number of global configurations of the system. In this setting, Gouda fairness ensures that any configuration that is infinitely often reachable is eventually reached. Observe that a scheduler that at each step activates a particle chosen uniformly at random is a Gouda fair sequential scheduler.

We do not assume that there exists an agreement on the orientation of the grid, or even on its chirality. Observe that without simple connectivity and without agreement on orientation or chirality, it is possible to construct arbitrarily large rings of even size where all processes have the same geometric information about the system (see Figure 1). In this setting, the results of [20] show that there is no silent self-stabilising LE algorithm using constant memory. This is part of our motivation for considering simply connected systems.

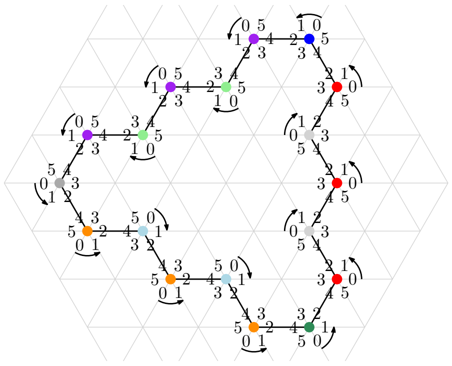

Figure 1 An 18-particle ring where for each particle, the occupied neighbours are reached through port numbers 2 and 4. Two nodes have the same colour if they agree on the grid orientation.

1.2 Related Work

In general networks, there is no self-stabilising leader election algorithm where each process has a constant memory. More precisely, Dolev et al. [20] established that any silent self-stabilising algorithm electing a leader in the class of rings requires $\Omega(\log n)$ bits of memory per process (where n is the size of the ring). This lower bound only uses the assumption that there exists a silent correct configuration and holds for any kind of scheduler. More recently, Blin et al. [4] showed that non-silent self-stabilising algorithms require $\Omega(\log \log n)$ bits of memory per process in order to elect a leader synchronously in the class of rings. Note that these lower bounds are tight in the sense that for rings, there exist silent (resp. non-silent) self-stabilising LE algorithms using $O(\log n)$ (resp., $O(\log \log n)$) bits of memory per process [12, 6]. Constant memory self stabilising algorithms for rings can be designed under special assumptions, as in [30] which gives an algorithm for prime sized rings assuming a sequential scheduler. However, [20] established that this algorithm cannot be made silent.

There exists a large literature about distributed systems where each process has finite memory. Cellular automata, introduced in the 40s in [34] are one of the best known models of this kind. More recently, numerous papers have been devoted to population protocols introduced in [1]. In this model, there is a population of finite-state agents and at each step, a scheduler picks two agents that jointly update their states according to their current states. The scheduler satisfies the same fairness condition as the one we consider in this paper: any configuration that is infinitely often reachable is eventually reached. In this setting, there exist election protocols using only two states when all agents start in the same state. However, when considering self-stabilising LE in this setting, Cai et al. [9] showed that a protocol using $n-1$ states cannot solve the problem in a population of n agents. This shows that even with a Gouda fair scheduler, it is not always possible to solve the LE problem in a self-stabilising way when processes have constant memory.

PM was introduced in [33] and has since gained popularity. Several models have been introduced, such as [28, 35, 24]. In this paper we consider the well studied *Amoebot* model [15, 13]. In this model, constant-memory computational entities, called *particles*, operate in a triangular grid. Each node of the grid is occupied by at most one particle and particles can determine whether nodes at distance one are occupied by particles. Each particle can communicate with its neighbours by reading their respective registers. It is usually assumed that particles do not have any global sense of direction, while some papers assume that the particles have a common sense of rotational orientation, called *chirality* (e.g., [22, 3]) or that particles agree on a common direction (e.g., [10]). In Amoebot, particles have the ability to move to neighbouring nodes (e.g., [22, 23]), which we do not use. The problem of LE has been studied in the specific context of PM in both 2D (e.g., [3, 17]) and 3D settings (e.g., [26, 8]) and both deterministic (e.g., [22]) and randomized algorithms (e.g., [16]) have been proposed. The existing algorithms for LE in PM can be categorized based on the use of two main techniques: *erosion* (e.g., [17, 25]) and message passing on boundaries (e.g., [3, 16]).

Research on self-stabilisation in the PM setting is more limited. In [16], a randomised LE algorithm is given and the authors discuss the possibility of making it self-stabilising by combining it with techniques from [2, 29]. However, it is assumed that particles have $O(\log^* n)$ memory. In the same paper, it is argued that self-stabilisation in PM is not possible for problems where movement is needed, as the system can become permanently disconnected. A self-stabilising algorithm for constructing a spanning forest was introduced in [14]. The algorithm in [14] is deterministic and particles have constant memory. However, it is assumed that at least one non-faulty special particle always remains in the system. The need to extend the Amoebot model to also address self-stabilising algorithms is also discussed in [13].

2 Model

Let G_Δ be an infinite regular triangular grid where each node has six neighbours. A connected particle system, $\mathcal{P}$, is simply connected if $G_\Delta \setminus \mathcal{P}$ is connected. We assume each node of the simply connected $\mathcal{P}$ contains exactly one particle. We call nodes that are in $\mathcal{P}$ *occupied* and those that are not in $\mathcal{P}$, *empty*. Each particle is anonymous, has constant memory and is stationary (i.e., does not move). A particle is incident to six *ports*, leading to consecutive neighbouring nodes in G_Δ. Each port is associated with a label so that ports i and $i+1$ mod 6 lead to neighbouring nodes. A particle knows if each port leads to an occupied or empty node. For each occupied neighbour q, the particle p knows the label assigned by q to qp. Each particle has a constant-size register with arbitrary initial contents. A particle can read the register of each occupied neighbour but can only write in its own register. All particles are *inactive* unless activated by the scheduler. An activated particle reads the contents of its register and the register of each of the neighbouring particles. Based on this information it updates the contents of its own register according to the given algorithm.

We call $\mathcal{P}$, the *support* of the particle system. The configuration C of the system at any time, consists of the set $\mathcal{P}$ and the contents of the registers of each particle in $\mathcal{P}$. A distributed algorithm $\mathcal{A}$ is a set of local rules that particles execute. The rules of the algorithm depend only on the content of the registers of the particles and of its neighbours and they modify only the register of the particle. For an algorithm $\mathcal{A}$, a configuration C, and a particle p, we say that p is *activable* in C, if the execution of $\mathcal{A}$ modifies the contents of the register of p. For two configurations, C and C' that have the same support, we say that C' is a successor of C if there exists an activable particle p in C such that, when p executes $\mathcal{A}$, C' is obtained. An execution $\mathcal{S}$ is an infinite sequence of configurations $\mathcal{S} = C_0, C_1, \ldots$ such that for any i, C_i and C_{i+1} have the same support and either there exists an activable particle p_i such that when p_i executes $\mathcal{A}$ in C_i, C_{i+1} is obtained, or there is no activable particle and $C_{i+1} = C_i$. If there exists a step where $C_{i+1} = C_i$, we call C_i a *final configuration*. An execution is *Gouda fair* [21, 27] if for any configuration C that appears infinitely often in the execution, any successor C' of C also appears infinitely often. An algorithm $\mathcal{A}$ is silent self-stabilising under a *Gouda fair* scheduler, if any such execution of the algorithm contains a final configuration C^* that is valid. The notion of valid configurations depends on the algorithm. In the next section, we define the valid configurations we consider in this paper.

We now present some notations and observations about the geometry of the system. Let v and v' be two neighbouring nodes in $\mathcal{P}$. We say that an edge that is oriented from v to a neighbouring node v' is *outgoing* for v and *incoming* for v'. We write $\overrightarrow{vv'}$ to denote an edge directed from v to v' and vv' to denote an undirected edge or an edge whose orientation is not known. Particles with at least one neighbour that is not in $\mathcal{P}$ are on the *boundary*. Since $\mathcal{P}$ is simply connected, there exists only one boundary in the system. Let p be a particle on the boundary. We say that p is *pending* if p has a unique neighbouring particle in $\mathcal{P}$. We say that p is an *articulation point* if the removal of p disconnects $\mathcal{P}$. If p is neither pending, nor an articulation point, then p is incident to two distinct edges pq, pr on the boundary of $\mathcal{P}$. In this case, since $\mathcal{P}$ is simply connected, there is a path of particles in the 1-neighbourhood of p from q to r. We say that p is on a $\theta \in \{60°, 120°, 180°, 240°\}$ angle to denote the angle that is formed when moving from q to r around p and no empty nodes are encountered. By slight abuse of notation, we also call a particle on a θ angle a θ *particle*. It is easy to see that a particle on the boundary cannot be on a $300°$ angle, otherwise q and r are adjacent and p is not on the boundary, a contradiction. $\mathcal{P}$ is 2–connected if it does not contain any articulation point. Notice that in systems with at least three particles, a system with no articulation point does not contain any pending particle. In a 2–connected particle system, the following observation implies that there should be a $60°$ or a $120°$ particle.

▶ **Observation 1.** *If $\mathcal{P}$ is 2–connected and $|\mathcal{P}| \geq 3$, particles on the boundary satisfy the formula $2n_{60} + n_{120} - n_{240} = 6$, where n_θ is the number of θ particles on the boundary.*

Proof. If $\mathcal{P}$ is 2–connected, it forms a simple polygon. The sum of internal angles of a simple polygon is $(n-2)\pi$, where n is the number of vertices of the polygon. So $(n_{60} + n_{120} + n_{180} + n_{240} - 2)\pi = n_{60}\frac{\pi}{3} + n_{120}\frac{2\pi}{3} + n_{180}\pi + n_{240}\frac{4\pi}{3}$, that is, $2n_{60} + n_{120} - n_{240} = 6$. ◀

▶ **Lemma 2.** *In any simply connected particle system $\mathcal{P}$ with at least two particles, the boundary of $\mathcal{P}$ contains one of following:*

1. *a pending particle, or*
2. *a $60°$ particle, or*
3. *two $120°$ particles that are connected by a path of $180°$ particles on the boundary.*

Proof. A block is a 2–connected component of $\mathcal{P}$. As $\mathcal{P}$ contains at least two particles, each block is either an edge or it contains at least three particles. The block tree of $\mathcal{P}$ is a tree where each vertex is a block and there is an edge between two blocks if they share a vertex (i.e., an articulation point of $\mathcal{P}$). A leaf, $\mathcal{P}'$, of the block tree is a 2–connected component of $\mathcal{P}$ and contains a unique articulation point p' of $\mathcal{P}$. If $\mathcal{P}'$ contains precisely two particles p' and q', then p' is the unique neighbour of q' in $\mathcal{P}$ and q' is a pending particle, as in Case 1.

Suppose $\mathcal{P}'$ contains at least three particles. Since $\mathcal{P}'$ is 2–connected, every particle on the boundary of $\mathcal{P}'$ is a $\theta \in \{60°, 120°, 180°, 240°\}$ particle. Any θ particle $p \neq p'$ of $\mathcal{P}'$ is also a θ particle of $\mathcal{P}$. So a 60° particle $p \neq p'$ in $\mathcal{P}'$, is a 60° particle in $\mathcal{P}$, which is Case 2.

Suppose now that in $\mathcal{P}'$, any boundary particle p different from p' is a θ particle with $\theta \in \{120°, 180°, 240°\}$. Let n'_{120}, n'_{180}, n'_{240} be respectively the number of 120°, 180°, 240° particles in $\mathcal{P}'$ that are different from p'. Since p' is an articulation point, p' cannot have more than three consecutive particle neighbours. Consequently, in $\mathcal{P}'$, p' is either a 60° or a 120° particle. If p' is a 60° particle in $\mathcal{P}'$, from Observation 1, we have $2 + n'_{120} - n'_{240} = 6$. If p' is a 120° particle in $\mathcal{P}'$, from Observation 1, we have $n'_{120} + 1 - n'_{240} = 6$. In both cases, we then have $n'_{120} \geq n'_{240} + 4$. Let $p_1, \ldots, p_{n'_{120}}$ be the 120° particles of $\mathcal{P}'$ in the order in which they appear when we move on the boundary of $\mathcal{P}'$ starting from p' (i.e., p' appears between $p_{n'_{120}}$ and p_1). Since $n'_{120} \geq n'_{240} + 4 > n'_{240} + 1$, there exists an index $1 \leq i \leq n'_{120} - 1$ such that only 180° particles appear on the boundary of $\mathcal{P}'$ between p_i and p_{i+1}. Since all these 180° particles are also 180° particles on the boundary of $\mathcal{P}$, we are in Case 3. ◀

We explain how two adjacent particles in a triangle detect each other's chirality. The label $\lambda(\Pi)$ of a path $\Pi = (p_1, p_2, \ldots, p_k)$ in the graph induced by the particles is a sequence of pairs of labels $(\mathsf{a}_1, \mathsf{b}_2), (\mathsf{a}_2, \mathsf{b}_3), \ldots, (\mathsf{a}_{k-1}, \mathsf{b}_k)$ where for each i, a_i (resp. b_i) is the port connecting p_i to p_{i+1} (resp. p_{i-1}).

Following [36], we define the *view* of depth k of a particle p, denoted by $\texttt{view}_k(p)$, to be the set of labels $\lambda(\Pi)$ of paths Π starting at p of length at most k. Note that for each $1 \leq j \leq k$, if both $(\mathsf{a}_1, \mathsf{b}_2), (\mathsf{a}_2, \mathsf{b}_3), \ldots, (\mathsf{a}_j, \mathsf{b}_{j+1})$ and $(\mathsf{a}_1, \mathsf{b}_2), (\mathsf{a}_2, \mathsf{b}_3), \ldots, (\mathsf{a}_j, \mathsf{b}'_{j+1})$ belong to $\texttt{view}_k(p)$, then $\mathsf{b}_{j+1} = \mathsf{b}'_{j+1}$. From [7], for any constant k, each particle p can construct $\texttt{view}_k(p)$ in a self stabilising way with constant memory.

▶ **Lemma 3.** *For any triangle of particles pqr, p can infer the chirality of q from* $\mathrm{view}_3(p)$.

Proof. In the following, for a particle p, we let $\{\mathsf{p}_i \mid 0 \leq i \leq 5\}$ be the set of ports incident to p and we assume that either $\mathsf{p}_{i+1} = \mathsf{p}_i + 1$ for each $0 \leq i \leq 5$, or $\mathsf{p}_{i+1} = \mathsf{p}_i - 1$ for each $0 \leq i \leq 5$ (where additions are made modulo 6). We will use the following observation.

▷ Claim 4. If pqr is a triangle, then the ports connecting r to p and q are consecutive.

Consider a triangle pqr. Let p_1 (resp. q_1) be the port through which p (resp. q) is connected to q (resp. p). Further, let p (resp. r) be connected to r (resp. p) through p_0 (resp. r_1). Observe that if p learns the port through which q is connected to r, it also learns the chirality of q. Note that by Claim 4, this port is either q_0 or q_2 and the port from r to q is either r_0 or r_2. Notice that if r is the only common neighbour of p and q, then only one of $\{(\mathsf{p}_1, \mathsf{q}_1), (\mathsf{q}_0, \mathsf{x}) \mid 0 \leq \mathsf{x} \leq 5\} \cup \{(\mathsf{p}_1, \mathsf{q}_1), (\mathsf{q}_2, \mathsf{x}) \mid 0 \leq \mathsf{x} \leq 5\}$ is in $\texttt{view}_3(p)$ and p can then infer the chirality of q. Suppose now that p and q have two common neighbours r and r'.

▷ Claim 5. The edge qr is labelled $(\mathsf{q}_2, \mathsf{r}_0)$ if and only if the following formula holds:

$$\begin{aligned}(\mathsf{p}_1, \mathsf{q}_1), (\mathsf{q}_2, \mathsf{r}_0), (\mathsf{r}_1, \mathsf{p}_0) \in \texttt{view}_3(p) \ \wedge \ (\mathsf{p}_0, \mathsf{r}_1), (\mathsf{r}_0, \mathsf{q}_2), (\mathsf{q}_1, \mathsf{p}_1) \in \texttt{view}_3(p) \\ \wedge \ \Big[(\mathsf{p}_1, \mathsf{q}_1), (\mathsf{q}_0, \mathsf{r}_2) \notin \texttt{view}_3(p) \ \vee \ (\mathsf{p}_2, \mathsf{r}_5) \notin \texttt{view}_3(p) \ \Big]\end{aligned}$$

Proof. First let us suppose that the edge qr is labelled $(\mathsf{q}_2, \mathsf{r}_0)$. Then, the first two expressions of the formula are satisfied. Let us suppose $(\mathsf{p}_1, \mathsf{q}_1), (\mathsf{q}_0, \mathsf{r}_2) \in \texttt{view}_3(p)$. Then from Claim 4, qr' is labelled $(\mathsf{q}_0, \mathsf{r}_2)$ and pr' is either labelled $(\mathsf{p}_2, \mathsf{r}_1)$ or $(\mathsf{p}_2, \mathsf{r}_3)$. In either case, $(\mathsf{p}_2, \mathsf{r}_5) \notin \texttt{view}_3(p)$ and the formula is satisfied.

Let us now suppose that the formula is satisfied and assume that qr is not labelled $(\mathsf{q}_2, \mathsf{r}_0)$. Then by Claim 4, qr is labelled either $(\mathsf{q}_2, \mathsf{r}_2)$, or $(\mathsf{q}_0, \mathsf{r}_0)$, or $(\mathsf{q}_0, \mathsf{r}_2)$. Note that the first two cases are impossible since $(\mathsf{p}_1, \mathsf{q}_1), (\mathsf{q}_2, \mathsf{r}_0)$ and $(\mathsf{p}_0, \mathsf{r}_1), (\mathsf{r}_0, \mathsf{q}_2)$ belong to $\texttt{view}_3(p)$. Consequently, qr is labelled $(\mathsf{q}_0, \mathsf{r}_2)$ and since $(\mathsf{p}_1, \mathsf{q}_1)(\mathsf{q}_2, \mathsf{r}_0) \in \texttt{view}_3(p)$, by Claim 4, qr' is labelled $(\mathsf{q}_2, \mathsf{r}_0)$, and the label of pr' is either $(\mathsf{p}_2, \mathsf{r}_5)$ or $(\mathsf{p}_2, \mathsf{r}_1)$. Note that we are necessarily in the second case since we assumed that the formula holds and since $(\mathsf{p}_1, \mathsf{q}_1)(\mathsf{q}_0, \mathsf{r}_2) \in \texttt{view}_3(p)$. This implies that $(\mathsf{p}_1, \mathsf{q}_1)(\mathsf{q}_2, \mathsf{r}_0)(\mathsf{r}_1, \mathsf{p}_2) \in \texttt{view}_3(p)$, and thus $(\mathsf{p}_1, \mathsf{q}_1)(\mathsf{q}_2, \mathsf{r}_0)(\mathsf{r}_1, \mathsf{p}_0) \notin \texttt{view}_3(p)$, contradicting the fact that the formula holds. ◁

From Claim 4, qr must be labelled $(\mathsf{q}_0, \mathsf{r}_0)$, $(\mathsf{q}_0, \mathsf{r}_2)$, $(\mathsf{q}_2, \mathsf{r}_0)$ or $(\mathsf{q}_2, \mathsf{r}_2)$. Applying Claim 5 to each possibility, p can detect the label of qr and thus infer the chirality of q. ◀

3 A Proof Labelling Scheme for Leader Election

Our aim is to orient all edges so that a unique sink particle (i.e., particle with no outgoing edges) that we define to be the leader exists. The certificate given to each particle consists of a direction for each edge incident to the particle. The orientation of the edges is chosen so that particles that are reached by an outgoing edge of some particle p, induce a connected graph of size at most three. In general we cannot avoid the existence of cycles in the orientation, but we will show that the existence of a unique sink is always guaranteed. Each particle p checks that the following rules are locally satisfied or detects an error.

R1 Each edge is oriented and both particles agree on the direction of the edge.
R2 Particle p has at most three outgoing edges. We consider edges between p and empty nodes to be incoming for p.
R3 When looking at the ports of p cyclically, all outgoing edges of p are consecutive.
R4 For every 3-particle triangle p belongs in, the triangle is not a cycle.

We call a configuration where every particle satisfies R1–R4 a *valid configuration*. Note that R4 does not guarantee an acyclic orientation (i.e., that larger cycles do not exist in the configuration). We do not forbid global cycles, but we will prove that even if cycles of size larger than three are formed by the incoming and outgoing edges, the remaining rules guarantee that there exists a unique sink in the system that we define to be the leader.

▶ **Theorem 6.** *If all rules R1–R4 are satisfied, then there exists a unique sink in the system.*

A valid configuration that does not contain any oriented cycle is a *valid acyclic orientation*. Observe that any particle system admits a valid acyclic orientation, as it can be constructed from any erosion based Leader Election algorithms for programmable matter (e.g., [17, 22]). Indeed, consider an execution of an erosion algorithm on a system, and orient any edge pq from p to q if p is eroded before q in the execution. This orientation is acyclic and it thus obviously satisfies R1 and R4. Since an erosion based algorithm erodes only a particle that does not disconnect its neighbourhood and that is strictly convex (i.e., that has at most three non-eroded neighbours), the orientation also satisfies R2 and R3.

4 A Self-Stabilising Algorithm for Leader Election

In Section 3, we claimed that *if* all rules are locally satisfied a unique sink exists in the system. Here, we show *how* a valid configuration is reached from a configuration containing errors. Our algorithm is simple: when a particle, p, is incident to an undirected edge e, p orients e as outgoing. If the orientation of the edges incident to p violates a rule, p undirects all its outgoing edges. Each activated particle always executes both lines of Algorithm 1.

Algorithm 1 Self Stabilising LE.

```
If   ¬R1              :  Mark all undirected edges as outgoing
If   ¬R2 ∨ ¬R3 ∨ ¬R4  :  Mark all outgoing edges as undirected
```

The directed edges incident to particles are encoded by each particle p having a variable $link_p[v'] \in \{in, out\}$ for each neighbouring node v'. For particle $p \in \mathcal{P}$ and $v' \in V_{G_\Delta \setminus \mathcal{P}}$, $link_p[v'] = in$. Any particle p can locally detect whether it is incident to an empty node, so we assume that edges between occupied and unoccupied nodes are always marked correctly. For two adjacent particles $p, p' \in \mathcal{P}$, if $link_p[p'] = in$ and $link_{p'}[p] = out$, pp' is directed from p' to p. We encode an *undirected* edge between two particles p and p' as $link_p[p'] = link_{p'}[p] = in$. We address $link_p[p'] = link_{p'}[p] = out$, as a special case. The endpoint that is activated first (say p) marks $link_p[p']$ as in. Notice that this is only possible during the first activation of p. In the remainder of this paper we only use the orientation of the edges without referencing their encoding. That is, we say that an edge between two particles p and q is: directed from p to q (i.e., $\overrightarrow{pq}$), directed from q to p (i.e., $\overrightarrow{qp}$) or undirected (i.e., pq or qp). From Lemma 3 particles in a common triangle detect each other's chirality. Since particles know both labels assigned to an edge, particles can compute the orientation of edges in triangles they belong in and check R4. From now on we only refer to particles detecting cyclic triangles. We prove that when executing our algorithm, any particle system reaches a valid configuration that contains a unique sink.

▶ **Theorem 7.** *Starting from an arbitrary simply connected configuration any Gouda fair execution of Algorithm 1 eventually reaches a configuration satisfying R1–R4 in which no rules can be applied and there exists a unique sink.*

5 Proof of Theorem 6 and Theorem 7

Here, we prove Theorem 7. Notice the second statement of Theorem 7 is precisely Theorem 6. A configuration of a particle system executing Algorithm 1 is described by the direction of each edge pq (i.e., $\overrightarrow{pq}$, $\overrightarrow{qp}$ or undirected). We make a few observations on how to change the orientation of some edges of a valid configuration and maintain a valid configuration.

▶ **Observation 8.** *Let p be a particle such that R3 is satisfied at p. Let e be an incoming edge to p and e' be an outgoing edge of p, s.t. when moving cyclically around p, e and e' are consecutive. If e becomes outgoing (resp. e' becomes incoming), R3 is not violated at p.*

▶ **Observation 9.** *Let C be a configuration and let p be a particle so that R1 (resp., R2, R4) is satisfied at some particle $q \neq p$ in C. Then, R1 (resp., R2, R4) is satisfied at q in $C \setminus \{p\}$.*

Let $\mathcal{S} = C_0, C_1, \ldots$ be an execution of Algorithm 1 starting from a configuration C_0. Notice a particle p is activable in a configuration C if when it executes Algorithm 1, one of its undirected edges becomes outgoing or one of its outgoing edges becomes undirected. If there exists a configuration C_f where no node is activable, then $C_f = C_j$ for all $j > f$, and we say that the execution stabilises to a *final* configuration. If all rules are satisfied in this final configuration, then this configuration is valid and we say it is a *final directed* configuration. If a configuration C_i is not final, we can assume that there exists an activable particle p_i such that we obtain C_{i+1} by activating p_i in C_i. Since each particle has constant memory, the number of possible configurations is finite. Hence there exists an index i_0 in $\mathcal{S}$ such that any configuration C_i with $i \geq i_0$ appears infinitely often in $\mathcal{S}$. We write $\mathcal{S}_{i_0} = C_{i_0}, C_{i_0+1}, \ldots$

to denote the part of the execution starting at C_{i_0} and in the following we consider only $\mathcal{S}_{i_0}$ and configurations C_i with $i \geq i_0$. We call the edges that are never undirected in $\mathcal{S}_{i_0}$, *stable edges.* Observe that by the definition of i_0, each edge is either stable or undirected infinitely often. Notice that any edge pq directed from p to q in C_i with $i \geq i_0$, is directed from p to q infinitely often, regardless of whether it is stable. We establish some properties in $\mathcal{S}_{i_0}$.

▶ **Lemma 10.** *In any configuration C_i with $i \geq i_0$, Rules R2, R3 and R4 are always satisfied.*

Proof. Let n_i be the number of particles in C_i that do not satisfy R2, R3 or R4. If a particle p is activated in C_i, then R2, R3 and R4 are satisfied at p in C_{i+1}. Moreover, if R2, R3, and R4 are satisfied at some particle p in C_i that is not activated at step i, then they are still satisfied at step C_{i+1}. Consequently, $n_{i+1} \leq n_i$. Since for $i \geq i_0$, C_i appears infinitely often, we get that for every $i \geq i_0$, we have $n_i = n_{i_0}$. If $n_i > 0$, there exists a particle p that always violates one of the rule R2, R3 or R4. Thus, p is eventually activated at some step i and in C_{i+1}, p satisfies the rules, a contradiction. Consequently, for any $i \geq i_0$, R2, R3 and R4 are satisfied at every particle in C_i. ◀

▶ **Lemma 11.** *If a particle p is incident to a stable outgoing edge, p is never activable in $\mathcal{S}_{i_0}$ and all edges incident to p are stable edges.*

Proof. In a configuration C_i, if a particle p is incident to an outgoing edge and an undirected edge, then p is activable in C_i. After its activation, either all the undirected edges incident to p have become outgoing edges, or all outgoing edges of p have become undirected.

Let $\overrightarrow{pq}$ be a stable edge, hence, p never marks $\overrightarrow{pq}$ as undirected. Let us suppose that in addition to $\overrightarrow{pq}$, p is also incident to an unstable edge pr. Then infinitely often, pr is undirected and thus there exists a step where p is activated and pr is undirected. At this step, p marks $\overrightarrow{pr}$ as outgoing. Then if $\overrightarrow{pr}$ becomes undirected at a later step, $\overrightarrow{pq}$ must also become undirected, which is a contradiction. Hence, all edges incident to p are stable. ◀

▶ **Lemma 12.** *If a particle p is incident to an unstable edge in $\mathcal{S}_{i_0}$, the unstable edges incident to p are at least two and do not appear consecutively around p, or there are at least four unstable edges incident to p.*

Proof. Suppose the lemma does not hold and that there exists a particle p incident to $1 \leq k \leq 3$ unstable edges that appear consecutively around p. Then, there exists an unstable edge pq incident to p such that for every unstable edge pr incident to p, either $r = q$ or r is adjacent to q. Note that by Lemma 11, all stable edges incident to p are incoming to p.

▷ Claim 13. Consider an unstable edge pr with $r \neq q$ and let s be the common neighbour of p and r that is distinct from q. Then $\overrightarrow{sp}$ and $\overrightarrow{sr}$ are stable.

Proof. By the definition of q, sp is stable, and by Lemma 11, sp is oriented from s to p. By Lemma 11 applied at s and r, sr is also stable and it is directed from s to r. ◁

Suppose first there is a configuration C_i with $i \geq i_0$ such that pq is directed from p to q in C_i and undirected in C_{i+1}. This implies that p is activated at step i. By Lemma 10 there exists at least an undirected edge pr in C_i, and when orienting all undirected edges incident to p in C_i as outgoing edges, one of R2, R3, R4 is violated. By the definition of q, this cannot be R2 or R3. If R4 is violated, it implies that in C_i, there exists an undirected edge pr and directed edges $\overrightarrow{rs}$ and $\overrightarrow{sp}$. By Claim 13, $s = q$ but this is impossible since $\overrightarrow{pq}$ is in C_i.

Then, at each step $i \geq i_0$, either qp is undirected or it is $\overrightarrow{qp}$. If there is no step $i \geq i_0$ where q is activated, then q never has any outgoing edge, qp is always undirected and we let $i_1 = i_0$. Otherwise, consider a step $i_1 - 1$ where q is activated such that in C_{i_1}, qp is undirected. Then at step i_1, all edges incident to p are either incoming or undirected. We claim that if we activate p at step i_1, it orients pq from p to q. Indeed by the definition of q, rules R2 and R3 are satisfied when pq is oriented from p to q. By Claim 13, any triangle violating R4 should contain q, but this is impossible since q has no outgoing edges in C_{i_1}. So, by the fairness condition, there exists a configuration C_i containing $\overrightarrow{pq}$, a contradiction. ◀

We now prove Theorem 7 using the structure of the boundary of $\mathcal{P}$ given by Lemma 2. Informally, the proof has the following structure. We assume that it is possible that the system does not stabilise and arrive at a contradiction. Out of the particle systems that do not stabilise to a configuration that satisfies all rules and has a unique sink, we take a system with the minimum number of particles. On the boundary of that system there exists a particle p satisfying one of the cases of Lemma 2. For each orientation of the edges incident to p we show that the edges incident to p are stable. Then we take a smaller system containing exactly one less particle, p. We show that the execution in both systems for particles that are not p is the same. Hence, if the system that contains p does not satisfy all rules and does not have a unique sink, the same is true for the system that does not contain p. Since we had assumed that the system containing p is the minimum size system that does not stabilise to a valid configuration, a smaller system not stabilising is a contradiction.

Proof of Theorem 7. Let us suppose that there exists a fair execution $\mathcal{S} = C_0, C_1, \ldots$ on a particle configuration $C = C_0$ that does not stabilise to a final directed configuration containing a unique sink. Consider such an execution $\mathcal{S}$ with a support $\mathcal{P}$ of minimum size. As defined above, consider a fair execution $\mathcal{S}_{i_0} = C_{i_0}, C_{i_0+1}, \ldots$ containing only configurations appearing infinitely often. By Lemma 2, we can assume that the boundary of $\mathcal{P}$ contains either a pending particle, or a 60° particle, or two 120° particles that are connected by a path of 180° particles on the boundary. In the following, we show that each of these cases cannot occur. We first consider the case where $\mathcal{P}$ contains a pending particle.

▶ **Lemma 14.** *If $\mathcal{P}$ contains a pending particle p (i.e., a particle with only one neighbouring particle w), all edges are stable in $\mathcal{S}_{i_0}$ and there is a unique sink in the final configuration.*

Proof. By Lemma 12, the edge pw is stable. Suppose first that pw is directed from p to w in $\mathcal{S}_{i_0}$. For each $i \geq i_0$, let $C'_i = C_i \setminus \{p\}$ and consider the sequence of configurations $\mathcal{S}'_{i_0} = C'_{i_0}, C'_{i_0+1}, \ldots, C'_i, \ldots$. Observe that for each $i \geq i_0$, either $C_{i+1} = C_i$ or there exists p_i such that C_{i+1} is obtained from C_i by activating p_i and thus modifying the orientations of edges incident to p_i. Since $\overrightarrow{pw}$ is stable, by Lemma 11, for any $i \geq i_0$, $p_i \neq p$. Moreover, for each $i \geq i_0$, the edges of C'_i have the same orientation as in C_i. So, $p' \neq p$ is activable in C'_i if and only if it is activable in C_i. Furthermore, the configuration obtained by activating p_i in C'_i is precisely C'_{i+1} since the edges of C'_i have the same orientation as in C_i. Hence, $\mathcal{S}'_{i_0}$ is a fair execution of Algorithm 1 on $\mathcal{P} \setminus \{p\}$. By the minimality of the size of $\mathcal{P}$, there exists a step $i_1 \geq i_0$ such that C'_{i_1} is a final directed configuration that contains a unique sink p''. Since the edges incident to p are stable, C_{i_1} is a final directed configuration. By our definition of i_0, this implies that $i_1 = i_0$. Since p has an outgoing edge, $\overrightarrow{pw}$, in $C_{i_1} = C_{i_0}$, p is not a sink of C_{i_0} and p'' is the unique sink in C_{i_0}.

Suppose now that $\overrightarrow{wp}$ is stable. Notice that in this case, p is a sink in C_i for each $i \geq i_0$. Moreover, since $\overrightarrow{wp}$ is stable, by Lemma 11, w is never activated. Since the two common neighbours of p and w are empty, by R3, p is the only outgoing neighbor of w in C_i for any $i \geq i_0$. Consequently, w is a sink in $C_i \setminus \{p\}$ for any $i \geq i_0$. For each $i \geq i_0$, let $C'_i = C_i \setminus \{p\}$ and consider the sequence of configurations $\mathcal{S}'_{i_0} = C'_{i_0}, C'_{i_0+1}, \ldots, C'_i, \ldots$ Observe that for each $i \geq i_0$, either $C_{i+1} = C_i$ or there exists p_i such that C_{i+1} is obtained from C_i by activating p_i and thus modifying the orientations of edges incident to p_i. Since p has only incoming edges in C_i, $p_i \neq p$. Moreover, since $\overrightarrow{wp}$ is stable, $p_i \neq w$. Moreover, for each $i \geq i_0$, the edges of C'_i have the same orientation as in C_i. Consequently, $p' \neq p$ is activable in C'_i if and only if it is activable in C_i. Furthermore, the configuration obtained by activating p_i in C'_i is precisely C'_{i+1} since the edges of C'_i have the same orientation as in C_i. Consequently, $\mathcal{S}'_{i_0}$ is a fair execution of Algorithm 1 on $\mathcal{P} \setminus \{p\}$. By the minimality of the size of $\mathcal{P}$, there exists a step $i_1 \geq i_0$ such that C'_{i_1} is a final directed configuration that contains a unique sink $p'' = w$ in C'_{i_1}. Since $\overrightarrow{wp}$ is stable, C_{i_1} is a final directed configuration. By our definition of i_0, this implies that $i_1 = i_0$. Since any $p' \in C_{i_0} \setminus \{p, w\}$ is not a sink in C'_{i_0}, and since $\overrightarrow{wp}$ is in C_{i_0}, p is the unique sink in the valid configuration C_{i_0}. ◀

We now consider the case where the boundary of $\mathcal{P}$ contains a $60°$ particle p, and we let q and r be the two neighbours of p on the boundary of $\mathcal{P}$.

▶ **Lemma 15.** *If $\mathcal{P}$ contains a $60°$ particle p, all edges are stable and there is a unique sink in the final configuration.*

Proof. By Lemma 12, pq and pr are stable. Consequently, pq and pr are always directed in the same way all along $\mathcal{S}_{i_0}$ and we can talk about the orientation of pq and pr in $\mathcal{S}_{i_0}$. For each $i \geq i_0$, let $C'_i = C_i \setminus \{p\}$ and consider the sequence of configurations $\mathcal{S}'_{i_0} = C'_{i_0}, C'_{i_0+1}, \ldots, C'_i, \ldots$. For each $i \geq i_0$, either $C_{i+1} = C_i$ or there exists p_i such that C_{i+1} is obtained from C_i by activating p_i and thus modifying the orientations of edges incident to p_i. Since all edges incident to p are stable in $\mathcal{S}_{i_0}$, we can assume $p_i \neq p$, for any $i \geq i_0$.

We distinguish three cases, depending on the orientation of pq and pr in $\mathcal{S}_{i_0}$.

▷ Case 1. The edges incident to p are $\overrightarrow{pq}$ and $\overrightarrow{pr}$.

Proof. For each $i \geq i_0$, the edges of C'_i have the same orientation as in C_i. Hence, a particle $p' \neq p$ is activable in C'_i if and only if it is activable in C_i. The configuration obtained by activating p_i in C'_i is precisely C'_{i+1} since the edges of C'_i have the same orientation as in C_i. So, $\mathcal{S}'_{i_0}$ is a fair execution of Algorithm 1 on $\mathcal{P} \setminus \{p\}$. By the minimality of the size of $\mathcal{P}$, there exists a step $i_1 \geq i_0$ such that C'_{i_1} is a final directed configuration with a unique sink p''. Since the edges incident to p are stable, C_{i_1} is a final directed configuration. By our definition of i_0, this implies that $i_1 = i_0$. Since p has only outgoing edges in $C_{i_1} = C_{i_0}$, p is not a sink of C_{i_0} and p'' is the unique sink in C_{i_0}. ◁

▷ Case 2. The edges incident to p are $\overrightarrow{qp}$ and $\overrightarrow{pr}$.

Proof. Since $\overrightarrow{qp}$ is stable, qr is also stable by Lemma 11. By R4, qr is directed from q to r in $\mathcal{S}_{i_0}$. Notice that since q and p are incident to outgoing stable edges, from Lemma 11, p and q are incident only to stable edges and are never activable. The edges of $C'_{i \geq i_0} \setminus \{p\}$ have the same orientation as in $C_{i \geq i_0}$. Consequently, for any $p' \notin \{p, q\}$, p' is activable in C'_i if and only if it is activable in C_i. Let us consider q. In C_i, $\overrightarrow{qp}$ is stable and directed and in C'_i, q has an incoming edge from the respective empty node. Furthermore, q is incident to an incoming edge from the empty common neighbour of p and q. Hence, R3 is satisfied for q in C'_i from Observation 8, and the remaining rules are satisfied for q in C'_i from Observation 9. Therefore, q is never activable in $C'_{i \geq i_0}$ and thus a particle $p' \neq p$ is activable in $C'_{i \geq i_0}$ if and only if it is activable in $C_{i \geq i_0}$. So, $\mathcal{S}'_{i_0}$ is a fair execution of Algorithm 1 on $\mathcal{P} \setminus \{p\}$. By the minimality of the size of $\mathcal{P}$, there exists a step $i_1 \geq i_0$ such that C'_{i_1} is a final directed configuration that contains a unique sink p''. Note that $p'' \neq q$ since $\overrightarrow{qr} \in C'_{i_1}$. Since the edges incident to p are stable and since q is not activable, C_{i_1} is a final directed configuration. By our definition of i_0, this implies that $i_1 = i_0$. Since p has an outgoing edge in $C_{i_1} = C_{i_0}$, p is not a sink of C_{i_0} and p'' is therefore the unique sink in C_{i_0}. ◁

▷ Case 3. The edges incident to p are $\overrightarrow{qp}$ and $\overrightarrow{rp}$.

Proof. Since $\overrightarrow{qp}$ and $\overrightarrow{rp}$ are stable, from Lemma 11, q and r are never activable and all edges incident to q and r are stable. Hence qr is stable and we assume without loss of generality that qr is directed as $\overrightarrow{qr}$ in $C_{i\geq i_0}$. Notice that p is a sink in C and that from R3 all edges incident to r except $\overrightarrow{rp}$ are incoming to r in C_i. Edges in $C'_i \setminus \{p\}$ have the same orientation as in C_i. Using the arguments from Case 2, R1–R4 are satisfied for q and for r in C'_i. So, q and r are never activable in C'_i. For the reasons in Case 2, $\mathcal{S}'_{i_0}$ is a fair execution of Algorithm 1 on $\mathcal{P} \setminus \{p\}$. Since $\mathcal{P}$ is of minimum size, there exists a step $i_1 \geq i_0$ such that C'_{i_1} is a final directed configuration that contains a unique sink p''. Since the only outgoing edge of r in C_{i_0} is $\overrightarrow{rp}$, $r = p''$ is the unique sink of C'_{i_1}. Furthermore, since the edges incident to p, q, r are stable, C_{i_1} is a final directed configuration. By our definition of i_0, this implies that $i_1 = i_0$. Since p has only incoming edges in $C_{i_1} = C_{i_0}$, p is a sink in C_{i_0} and r is not a sink in C_{i_0} due to $\overrightarrow{rp}$, so p is the unique sink in C_{i_0}. ◁

Therefore, for any orientation of the edges incident to p in $\mathcal{S}_{i_0}$, C_{i_0} is a directed final configuration containing a unique sink. ◀

Now assume there exist two 120° particles connected by a path of 180° particles on the boundary of $\mathcal{P}$.

▶ **Lemma 16.** *If $\mathcal{P}$ contains two 120° particle p, p^* connected by a path of 180° particles on the boundary, all edges are stable and there is a unique sink in the final configuration.*

Proof. Let q and r be the neighbours of p on the boundary and let s be the common neighbour of p, q and r. By Lemma 12, we know that ps is stable in $\mathcal{S}_{i_0}$. We split the proof of the lemma in different cases, depending on the orientation of ps in $C_{i\geq i_0}$. Due to space constraints, we omit some details, which can be found in [11], in the proofs of Cases 1.1 – 2.3.

▷ Case 1. The edge between p and s is $\overrightarrow{ps}$.

For each $i \geq i_0$, let $C'_i = C_i \setminus \{p\}$ and consider the sequence of configurations $\mathcal{S}'_{i_0} = C'_{i_0}, C'_{i_0+1}, \ldots, C'_i, \ldots$ For each $i \geq i_0$, either $C_{i+1} = C_i$ or there exists p_i such that C_{i+1} is obtained from C_i by activating p_i and thus modifying the orientations of edges incident to p_i. From Lemma 11, since $\overrightarrow{ps}$ is stable, p is never activable and the edges pq and pr are stable. Consequently, $p_i \neq p$ for any $i \geq i_0$. The orientations of pq and pr lead to the following cases.

▷ Case 1.1. Particle p is incident to $\overrightarrow{pq}$, $\overrightarrow{ps}$ and $\overrightarrow{pr}$.

Proof. The proof for this case follows the same argumentation as Case 1 of Lemma 15. ◁

▷ Case 1.2. Particle p is incident to $\overrightarrow{pq}$, $\overrightarrow{ps}$ and $\overrightarrow{rp}$.

Proof. Since $\overrightarrow{rp}$ is stable, from Lemma 11, r is not activable in C and rs is stable. From R4, $\overrightarrow{rs}$ is in $C_{i\geq i_0}$. The edges of $C'_{i\geq i_0} \setminus \{p\}$ have the same orientation as in $C_{i\geq i_0}$. So for any $p' \notin \{p, r\}$, p' is activable in C'_i if and only if p' is activable in C_i. Let us consider r. From Observation 8 and the incoming edge from the empty common neighbour of p and r to r, R3 is satisfied for r in C'_i. R1, R2 and R4 are satisfied for r in C'_i from Observation 9. Hence, r is never activable in C'_i. So, $\mathcal{S}'_{i_0}$ is a fair execution of Algorithm 1 on $\mathcal{P} \setminus \{p\}$. The unique sink in C'_{i_0} is $p'' \neq r$, since r has an outgoing edge $\overrightarrow{rs}$ in C'_i. Since p has outgoing edges in C_{i_0}, p is not a sink of C_{i_0} and p'' is the unique sink in C_{i_0}. ◁

▷ Case 1.3. Particle p is incident to $\overrightarrow{qp}$, $\overrightarrow{ps}$ and $\overrightarrow{rp}$.

Proof. Since $\overrightarrow{qp}$ and $\overrightarrow{rp}$ are stable, from Lemma 11, r and q are not activable in C and the edges rs and qs are stable. By R4, $\overrightarrow{rs}$ and $\overrightarrow{qs}$ belong to $C_{i\geq i_0}$. The edges of $C'_{i\geq i_0} \setminus \{p\}$ have the same orientation as in $C_{i\geq i_0}$. Consequently, for any $p' \notin \{p, q, r\}$, p' is activable in C'_i if and only if p' is activable in C_i. Let us consider q and r. Using the same arguments as in Case 1.2, R1–R4 are satisfied for q and for r in C'_i. Consequently, q and r are never activable in C'_i. Therefore, C'_i satisfies all rules and $\mathcal{S}'_{i_0}$ is a fair execution of Algorithm 1 on $\mathcal{P} \setminus \{p\}$. Since q and r each have an outgoing edge to s in C'_i, $p'' \notin \{q, r\}$. Since p has outgoing edges in C_{i_0}, p is not a sink of C_{i_0} and p'' is therefore the unique sink in C_{i_0}. ◁

We now consider the case where the edge between p and s is $\overrightarrow{sp}$. Observe that by Lemma 11, sq and sr are stable edges. Note that by Lemma 12, pq is stable if and only if pr is stable. We distinguish two cases depending on whether these two edges are stable.

▷ Case 2. The edge between p and s is $\overrightarrow{sp}$, and the edges pq and pr are stable.

The possible orientations of the stable edges sq and sr in $\mathcal{S}_{i_0}$ give the following subcases.

▷ Case 2.1. Particle s is incident to $\overrightarrow{qs}$ and $\overrightarrow{rs}$.

Proof. By Lemma 11, r and q are never activable and rp and qp are stable. By R4, $\overrightarrow{qp}$ and $\overrightarrow{rp}$ are in $C_{i\geq i_0}$. The edges of $C'_{i\geq i_0} \setminus \{p\}$ have the same orientation as in $C_{i\geq i_0}$ and thus for any $p' \notin \{q, r, s\}$, p' is activable in C'_i if and only if it is activable in C_i. Let us consider q, r, s. Using the same arguments as in Case 1.2, R1–R4 are satisfied for q and r in $C \setminus \{p\}$. Since sp is between two incoming edges, from Observation 8, R3 is satisfied for s in C'_i. R1, R2 and R4 are satisfied for s in C'_i from Observation 9. So, q, r, s are never activable in C'_i. Furthermore, the configuration obtained by activating $p_i \neq q, s, r$ in C'_i is precisely C'_{i+1} since the edges of C'_i have the same orientation as in C_i. Hence, $\mathcal{S}'_{i_0}$ is a fair execution of Algorithm 1 on $\mathcal{P} \setminus \{p\}$. By R3, the only outgoing edge of s in C_{i_0} is $\overrightarrow{sp}$, so $p'' = s$ is the unique sink of C'_{i_0}. Since s has an outgoing edge $\overrightarrow{sp}$ in C_{i_0}, s is not a sink of C_{i_0} and p is the unique sink in C_{i_0}. ◁

Observe that if s is incident to $\overrightarrow{qs}$ (resp., $\overrightarrow{rs}$), then since qp (resp., rp) is stable, by R4, p is incident to $\overrightarrow{qp}$ (resp., $\overrightarrow{rp}$). When among qs and rs, there is one outgoing and one incoming edge, we consider two cases depending on whether p is a sink in $\mathcal{S}_{i_0}$.

▷ Case 2.2. Particle s is incident to $\overrightarrow{qs}$ and $\overrightarrow{sr}$ and p is incident to $\overrightarrow{rp}$.

Proof. Since $\overrightarrow{qs}$ and $\overrightarrow{sp}$ are stable, necessarily $\overrightarrow{qp}$ is stable and p is a sink in $C_{i\geq i_0}$. So, p is never activable in $\mathcal{S}_{i_0}$ and from Lemma 11, q, r, s are never activable in $\mathcal{S}_{i_0}$ either. The edges of $C'_{i\geq i_0} \setminus \{p\}$ have the same orientation as in $C_{i\geq i_0}$ and thus any particle $p' \notin \{p, q, r, s\}$ is activable in C'_i if and only if it is activable in C_i. Let us consider q, r, s. Since $\overrightarrow{sp}$ is between an incoming and an outgoing edge, by Observation 8, R3 is satisfied for s in C'_i. The remaining rules are satisfied for s in C'_i from Observation 9. The arguments for q are the same as in Case 2.1. Due to the incoming edge from the common empty neighbour of r and p and from Observation 8, R3 is satisfied for r in C'_i. The remaining rules are satisfied for r in C'_i from Observation 9. So, q, r, s are never activable in $C'_{i\geq i_0}$. Hence, all rules are satisfied in C'_i and $\mathcal{S}'_{i_0}$ is a fair execution of Algorithm 1 on $\mathcal{P} \setminus \{p\}$. By R3, the only outgoing edge of r in C_i is $\overrightarrow{rp}$, so $p'' = r$ is the unique sink of C'_i. Since p does not have outgoing edges in $C_{i_1} = C_{i_0}$, r is not a sink of C_{i_0} due to $\overrightarrow{rp}$ and p is the unique sink in C_{i_0}. ◁

▷ Case 2.3. Particle s is incident to $\overrightarrow{qs}$ and $\overrightarrow{sr}$ and p is incident to $\overrightarrow{pr}$.

Proof. As noted before the stable edge qp is oriented as $\overrightarrow{qp}$ by R4. By Lemma 11, p, q and s are never activable in $\mathcal{S}_{i_0}$ and the edges pr and sr are stable. The edges of $C'_{i\geq i_0} \setminus \{p\}$ have the same orientation as in $C_{i\geq i_0}$. So, for any $p' \notin \{p, q, s\}$, p' is activable in C'_i if and only if it is activable in C_i. Let us consider q and s. Using the same arguments for both q and s as in Case 2.2, we obtain that R1–R4 are always satisfied at q and s, and that they are never activable in $C'_{i\geq i_0}$. Hence, $\mathcal{S}'_{i_0}$ is a fair execution of Algorithm 1 on $\mathcal{P} \setminus \{p\}$. The unique sink of C'_{i_0}, $p'' \notin \{q, s\}$, since q and s have outgoing edges in $C \setminus \{p\}$. Since the edges incident to p are stable, C_{i_0} is a final directed configuration. Since p is incident to $\overrightarrow{pr}$ in C_{i_0}, p is not a sink in C_i, hence p'' is the unique sink in C_{i_0}. ◁

▷ Case 2.4. Particle s is incident to $\overrightarrow{sq}$ and $\overrightarrow{sr}$.

Proof. Since pq, pr, ps are stable and since ps is directed as $\overrightarrow{sp}$ in $C_{i\geq i_0}$, p is incident to at most one outgoing edge. Without loss of generality, we can thus assume that $\overrightarrow{qp}$ is in $C_{i\geq i_0}$. Observe that by R3, $\overrightarrow{qp}$ is the only outgoing edge at q and no neighbour of q is activable in $C_{i\geq i_0}$ by Lemma 11. Note also that s has three outgoing edges $\overrightarrow{sq}, \overrightarrow{sp}, \overrightarrow{sr}$ in C_i and since s is not activable in C_i, by R2, all other edges incident to s are incoming. Again, this implies that all neighbours of s different from r are not activable in $C_{i\geq i_0}$. For each $i \geq i_0$, let C^*_i be the configuration obtained from C_i by replacing $\overrightarrow{sq}$ by $\overrightarrow{qs}$.

For any particle $p' \notin \{s, q\}$, R2 and R3 are satisfied at p' in C^*_i since they are satisfied at p' in C_i by Lemma 10 and the orientation of the edges incident to p' in C^*_i is the same as in C_i. Since q only has one outgoing edge in C_i, R2 and R3 are satisfied at q in C^*_i. Since s has three outgoing edges $\overrightarrow{sq}, \overrightarrow{sp}, \overrightarrow{sr}$ in C_i and since all other edges incident to s are incoming, R2 directly holds at s in C^*_i and R3 holds at s in C^*_i since p, q and r are reached through consecutive ports of s by definition. If R4 is not satisfied at some particle p' in C^*_i, there is a directed triangle made of the edges $\overrightarrow{qs}, \overrightarrow{sp'}, \overrightarrow{p'q}$ in C^*_i. Since the only out-neighbours of s in C^*_i are p and r, necessarily, $p' = p$, but this is impossible since pq is oriented from q to p. So, R2, R3, R4 are always satisfied in $C^*_{i\geq i_0}$. Since all edges incident to q and s are stable in C_i, R1 is also satisfied at s and q in C_i and in C^*_i. Hence, q and s are never activable in C^*_i. For any $p' \notin \{q, s\}$, p' is activable in C^*_i if and only if it is activable in C_i. Therefore, $\mathcal{S}^*_i$ is a fair execution of Algorithm 1 on $\mathcal{P}$. Note that when considering $C^*_{i\geq i_0}$, we are in Case 2.2 or 2.3. So, we know that $C^*_{i_0}$ is a final directed configuration that contains a unique sink p'' different from q and s. Since a particle p' is activable in C_{i_0} if and only if it is activable in $C^*_{i_0}$, C_{i_0} is also a final directed configuration, and p'' is the unique sink of C_{i_0}. ◁

Finally, we consider the case where the edges pq and pr are not stable. We remind the reader that from Lemma 12, pq and pr are either both stable or both unstable.

▷ Case 3. The edge between p and s is $\overrightarrow{sp}$ and the edges pq and pr are not stable.

Proof. We will prove that this case is not possible.

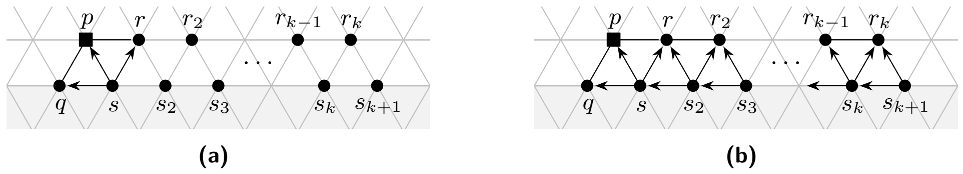

Figure 2 *Left:* The setting in Case 3, that is, a 120° particle p (square) with $\{pq, pr\}$ unstable and s incident to the directed edges sq, sp and sr. *Right:* The final orientation of edges in Case 3.

Without loss of generality, assume that r is on the path connecting p to p^* via 180° particles. Note that it is possible that $p^* = r$. Let $(r_0 = p, r_1 = r, r_2, \ldots, r_k = p^*)$ be the path on the boundary from p to p^* whose inner particles are all 180° particles (if $r = p^*$, then $k = 1$). Let s_{j+1} be the common neighbour of any pair of consecutive particles r_j and r_{j+1} with $0 \leq j \leq k-1$, and observe that $s_1 = s$. Let $s_0 = q$ and let s_{k+1} be the neighbour of r_k on the boundary that is distinct from r_{k-1}. This setting is also shown in Figure 2a.

Since s is incident to a stable edge, $\overrightarrow{sp}$, from Lemma 11 all edges incident to s are stable. By Lemma 11 applied at q and r and since pq and pr are not stable, necessarily $\overrightarrow{sq}$ and $\overrightarrow{sr}$ are stable in $\mathcal{S}_{i_0}$. Since in this setting s has three outgoing edges, sq, sp and sr, from R2 ss_2 is incoming to s. So, s_2 is incident to the stable outgoing edge $\overrightarrow{s_2s}$. From Lemma 11, s_2 is not activable and all edges incident to s_2 are stable and directed. From R4, s_2r is oriented from s_2 to r. If $r = p^*$ (i.e., if $k = 1$), r is incident to only one edge that is not stable, which is impossible from Lemma 12. Hence, $k \geq 2$ and rr_2 is not stable. As s_2 is not activable, from Lemma 11, s_2r_2 is stable. If s_2r_2 is directed from r_2 to s_2, r_2 has a stable outgoing edge and from Lemma 11, rr_2 is stable, which is a contradiction for r. So s_2r_2 is directed from s_2 to r_2.

Generalising, for $i \geq 1$ each particle s_i is incident to the stable edges $\overrightarrow{s_is_{i-1}}$, $\overrightarrow{s_ir_{i-1}}$ and $\overrightarrow{s_ir_i}$ and the edge $r_{i-1}r_i$ is not stable. Then, for $i = k$ the edge $r_{k-1}r_k$ should be the only unstable edge incident to r_k which is impossible from Lemma 12, a contradiction. ◁

This ends the proof of Lemma 16. ◀

The proof of Theorem 7 follows from Lemmas 2, 14, 15 and 16. ◀

6 Further Remarks

We showed that our algorithm works assuming that the scheduler is sequential and Gouda fair. The execution presented in Figure 3 shows that if we consider a sequential unfair scheduler (i.e., we only ask that the scheduler activates an activable particle at each step), there exist periodic executions that never reach a valid configuration. It would thus be interesting to understand if we can design a self-stabilising leader election algorithm for simply connected configurations that is correct even with an unfair scheduler. In the case where this is possible, we believe that this would lead to a much more complex algorithm than our algorithm.

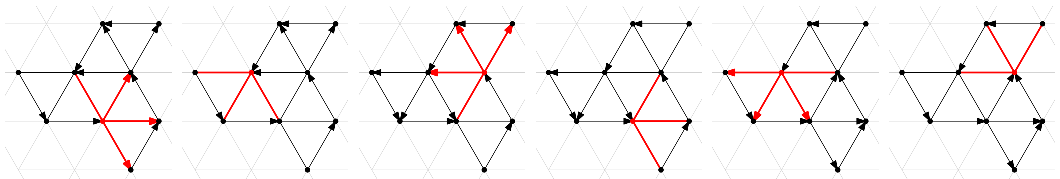

Figure 3 A periodic unfair execution of our algorithm. At each step, the red vertex is activated, and it modifies the status of its incident red edges (i.e., the ones that are not incoming).

Our algorithm heavily uses the geometry of the system and relies on the support being simply connected. For particles that agree on the orientation of the grid, the impossibility results of Dolev et al. [20] no longer hold. One can thus wonder if it is possible to design a silent self-stabilising LE algorithm using constant memory for arbitrary connected configurations if particles agree on the orientation of the grid. Again, the geometry of the grid should be used to overcome the impossibility results of [20], but it seems very challenging.

References

1 Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Comput.*, 20(4):279–304, 2007. doi:10.1007/s00446-007-0040-2.

2 Baruch Awerbuch and Rafail Ostrovsky. Memory-efficient and self-stabilizing network reset. In *PODC 1994*, pages 254–263. ACM, 1994. doi:10.1145/197917.198104.

3 Rida A. Bazzi and Joseph L. Briones. Stationary and deterministic leader election in self-organizing particle systems. In *SSS 2019*, volume 11914 of *Lecture Notes in Comput. Sci.*, pages 22–37. Springer, 2019. doi:10.1007/978-3-030-34992-9_3.

4 Lélia Blin, Laurent Feuilloley, and Gabriel Le Bouder. Optimal space lower bound for deterministic self-stabilizing leader election algorithms. *Discret. Math. Theor. Comput. Sci.*, 25, 2023. doi:10.46298/dmtcs.9335.

5 Lélia Blin, Pierre Fraigniaud, and Boaz Patt-Shamir. On proof-labeling schemes versus silent self-stabilizing algorithms. In *SSS 2014*, volume 8756 of *Lecture Notes in Comput. Sci.*, pages 18–32. Springer, 2014. doi:10.1007/978-3-319-11764-5_2.

6 Lélia Blin and Sébastien Tixeuil. Compact deterministic self-stabilizing leader election on a ring: the exponential advantage of being talkative. *Distributed Comput.*, 31(2):139–166, 2018. doi:10.1007/s00446-017-0294-2.

7 Paolo Boldi and Sebastiano Vigna. Universal dynamic synchronous self–stabilization. *Distributed Comput.*, 15:137–153, 2002. doi:10.1007/s004460100062.

8 Joseph L. Briones, Tishya Chhabra, Joshua J. Daymude, and Andréa W. Richa. Invited paper: Asynchronous deterministic leader election in three-dimensional programmable matter. In *ICDCN 2023*, pages 38–47. ACM, 2023. doi:10.1145/3571306.3571389.

9 Shukai Cai, Taisuke Izumi, and Koichi 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 Jérémie Chalopin, Shantanu Das, and Maria Kokkou. Deterministic leader election for stationary programmable matter with common direction. In *SIROCCO 2024*, volume 14662 of *Lecture Notes in Comput. Sci.*, pages 174–191. Springer, 2024. doi:10.1007/978-3-031-60603-8_10.

11 Jérémie Chalopin, Shantanu Das, and Maria Kokkou. Deterministic self-stabilising leader election for programmable matter with constant memory. *arXiv preprint*, 2024. doi:10.48550/arXiv.2408.08775.

12 Ajoy Kumar Datta, Lawrence L. Larmore, and Priyanka Vemula. Self-stabilizing leader election in optimal space under an arbitrary scheduler. *Theor. Comput. Sci.*, 412(40):5541–5561, 2011. doi:10.1016/j.tcs.2010.05.001.

13 Joshua J. Daymude, Andréa W. Richa, and Christian Scheideler. The canonical Amoebot model: Algorithms and concurrency control. *Distributed Comput.*, 36(2):159–192, 2023. doi:10.1007/s00446-023-00443-3.

14 Joshua J. Daymude, Andréa W. Richa, and Jamison W. Weber. Bio-inspired energy distribution for programmable matter. In *ICDCN 2021*, pages 86–95. ACM, 2021. doi:10.1145/3427796.3427835.

15 Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W Richa, Christian Scheideler, and Thim Strothmann. Amoebot – A new model for programmable matter. In *SPAA 2014*, pages 220–222. ACM, 2014. doi:10.1145/2612669.2612712.

16 Zahra Derakhshandeh, Robert Gmyr, Thim Strothmann, Rida A. Bazzi, Andréa W. Richa, and Christian Scheideler. Leader election and shape formation with self-organizing programmable matter. In *DNA 2015*, volume 9211 of *Lecture Notes in Comput. Sci.*, pages 117–132. Springer, 2015. doi:10.1007/978-3-319-21999-8_8.

17 Giuseppe Antonio Di Luna, Paola Flocchini, Nicola Santoro, Giovanni Viglietta, and Yukiko Yamauchi. Shape formation by programmable particles. *Distributed Comput.*, 33(1):69–101, 2020. doi:10.1007/s00446-019-00350-6.

18 Edsger W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. ACM*, 17(11):643–644, 1974. doi:10.1145/361179.361202.

19 Shlomi Dolev. *Self-Stabilization*. MIT Press, 2000. doi:10.7551/mitpress/6156.001.0001.

20 Shlomi Dolev, Mohamed G. Gouda, and Marco Schneider. Memory requirements for silent stabilization. *Acta Inf.*, 36(6):447–462, 1999. doi:10.1007/s002360050180.

21 Swan Dubois and Sébastien Tixeuil. A taxonomy of daemons in self-stabilization. *arXiv preprint*, 2011. doi:10.48550/arXiv.1110.0334.

22 Fabien Dufoulon, Shay Kutten, and William K. Moses Jr. Efficient deterministic leader election for programmable matter. In *PODC 2021*, pages 103–113. ACM, 2021. doi:10.1145/3465084.3467900.

23 Yuval Emek, Shay Kutten, Ron Lavi, and William K Moses Jr. Deterministic leader election in programmable matter. In *ICALP 2019*, volume 132 of *LIPIcs Leibniz Int. Proc. Inform.*, pages 140:1–140:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.140.

24 Sándor P. Fekete, Robert Gmyr, Sabrina Hugo, Phillip Keldenich, Christian Scheffer, and Arne Schmidt. Cadbots: Algorithmic aspects of manipulating programmable matter with finite automata. *Algorithmica*, 83(1):387–412, 2021. doi:10.1007/s00453-020-00761-z.

25 Nicolas Gastineau, Wahabou Abdou, Nader Mbarek, and Olivier Togni. Distributed leader election and computation of local identifiers for programmable matter. In *ALGOSENSORS 2018*, volume 11410 of *Lecture Notes in Comput. Sci.*, pages 159–179. Springer, 2018. doi:10.1007/978-3-030-14094-6_11.

26 Nicolas Gastineau, Wahabou Abdou, Nader Mbarek, and Olivier Togni. Leader election and local identifiers for three-dimensional programmable matter. *Concurr. Comput. Pract. Exp.*, 34(7), 2022. doi:10.1002/cpe.6067.

27 Mohamed G. Gouda. The theory of weak stabilization. In *WSS 2001*, volume 2194 of *Lecture Notes in Comput. Sci.*, pages 114–123. Springer, 2001. doi:10.1007/3-540-45438-1_8.

28 Elliot Hawkes, Byoungkwon An, Nadia M. Benbernou, H. Tanaka, Sangbae Kim, Erik D. Demaine, Daniela Rus, and Robert J. Wood. Programmable matter by folding. *Proc. Natl. Acad. Sci.*, 107(28):12441–12445, 2010. doi:10.1073/pnas.0914069107.

29 Gene Itkis and Leonid Levin. Fast and lean self-stabilizing asynchronous protocols. In *FOCS 1994*, pages 226–239. IEEE Computer Society, 1994. doi:10.1109/SFCS.1994.365691.

30 Gene Itkis, Chengdian Lin, and Janos Simon. Deterministic, constant space, self-stabilizing leader election on uniform rings. In *WDAG 1995*, volume 972 of *Lecture Notes in Comput. Sci.*, pages 288–302. Springer, 1995. doi:10.1007/BFb0022154.

31 Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Comput.*, 22(4):215–233, 2010. doi:10.1007/s00446-010-0095-3.

32 Gérard Le Lann. Distributed systems - towards a formal approach. In *IFIP 1977*, pages 155–160. North-Holland, 1977. URL: https://inria.hal.science/hal-03504338.

33 Tommaso Toffoli and Norman Margolus. Programmable matter: Concepts and realization. *Int. J. High Speed Comput.*, 5(2):155–170, 1993. doi:10.1016/0167-2789(91)90296-L.

34 John von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, 1966. URL: https://dl.acm.org/doi/book/10.5555/1102024.

35 Damien Woods, Ho-Lin Chen, Scott Goodfriend, Nadine Dabby, Erik Winfree, and Peng Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *ITCS 2013*, pages 353–354. ACM, 2013. doi:10.1145/2422436.2422476.

36 Masafumi Yamashita and Tiko Kameda. Computing on an anonymous network. In *PODC 1988*, pages 117–130. ACM, 1988. doi:10.1145/62546.62568.