On the Feasibility of Gathering by Autonomous Mobile Robots

Giuseppe Prencipe

Dipartimento di Informatica,
L.go Bruno Pontecorvo, 3 – 56100, Pisa, Italy
prencipe@di.unipi.it

Abstract. Given a set of n autonomous mobile robots that can freely move on a two dimensional plane, they are required to gather in a position of the plane not fixed in advance (Gathering Problem). The main research question we address in this paper is: under which conditions this task can be accomplished by the robots? The studied robots are quite simple: they are anonymous, totally asynchronous, they do not have any memory of past computations, they cannot explicitly communicate among each other. We show that this simple task cannot be in general accomplished by the considered system of robots.

Keywords: Mobile Robots, Multiplicity Detection, Distributed Coordination, Distributed Models, Computability.

1 Introduction

In this paper, we consider a distributed system populated by a set of n autonomous and anonymous mobile robots that can freely and independently move on a plane: in particular, they do not obey to any central coordinator. The behavior of these robots is quite simple: each of them execute a cycle of sensing, computing, moving and being inactive. In particular, each robot is capable of sensing the positions of other robots in its surrounding, performing local computations on the sensed data, and moving towards the computed destination. The local computation is done according to a deterministic algorithm that takes in input the sensed data (i.e., the robots' positions), and returns a destination point towards which the executing robot moves. All robots execute the same algorithm. The main research focus is to understand which are the conditions that allow these robots to complete given tasks, such as exploring the plane or forming a pattern like a circle, and design, in case the task is solvable, to design the algorithm they have to execute.

In this paper we focus on the Gathering problem: the robots are asked to meet in *finite time* in a point $\mathfrak{p}$ of the plane not determined in advance. In spite of its apparent simplicity, this problem has recently been tackled in several studies: in fact, several factors render this problem difficult to solve [3, 4, 5, 8, 10]. In particular, in all these studies, the problem has been solved only

A. Pelc and M. Raynal (Eds.): SIROCCO 2005, LNCS 3499, pp. 246–261, 2005.

making some "extra" assumption on the capability of the robots. In particular, in [4, 5, 10] the robots must be able to detect whether a given point on the plane is occupied by one or more robots. In contrast, such an assumption is not used in [3], but it is assumed an unlimited amount of memory the robots can use (the robots are said to be *non-oblivious*). In [8], the robots are assumed to have only limited visibility (i.e., they can sense only a portion of the plane) and to share a compass. Recently [1], GATHERING has also been studied in the presence of faulty robots; another study has been devoted to design convergence solutions to the problem [6].

In this paper we aim to prove that GATHERING is in general impossible, if the nature of the robots is not changed, and no "extra" assumption is made on the capabilities of the robots. The results shown here, are based on the *basic* model usually adopted in the majority of the studies present in literature. In particular, we will use the features of CORDA, first presented in [7], and of the model from Suzuki *et al.* (here referred to as ATOM): the description of these models will be the focus of the next section. In Section 3 the impossibility of the GATHERING is presented.

2 Definitions

2.1 Autonomous Mobile Robots

In this section, we describe the CORDA model, that will be used to prove the impossibility of GATHERING. The robots we consider are modeled as devices with computational capabilities[1], that are equipped with motorial capabilities – allowing them to move on the plane – and sensorial capabilities that let them to observe the positions of the other robots in the plane, and form their *local view* of the world. The set of absolute positions[2] on the plane occupied by the robots at a given time instant is called a *configuration of the robots*.

The local view of each robot includes a unit of length, an origin, and a Cartesian coordinate system defined by the *directions* of two coordinate axes, identified as the x and y axis, together with their *orientations*, identified as the positive and negative sides of the axes.

The robots are able to sense the complete plane: we say they have *Unlimited Visibility*. The robots, however, can not distinguish whether there is more than one fellow on a given positions of the plane: we say that they cannot detect *multiplicity*. The case when the robots can sense just a portion of it (*Limited Visibility*) has been studied too [8]; in particular, each robot can sense up to at most a distance V.

[1] To our knowledge, nothing is ever mentioned on the computational power of the modeled robots. For the purpose of this paper, they can be considered as Turing-equivalent machines.

[2] i.e., with respect to an inertial reference frame.

During its life, each robot cyclically executes four *states*:

i. Wait. The robot is idle. A robot cannot stay indefinitely idle (see Assumption A2 below). At the beginning all the robots are in *Wait*.

ii. Look. The robot observes the world by activating its sensors which will return a *snapshot* of the positions of all other robots within the visibility range with respect to its local coordinate system. Each robot is viewed as a point, hence its position in the plane is given by its coordinates, and the result of the snapshot (hence, of the observation) is just a set of coordinates in its local coordinate system: this set forms the *view of the world* of r. More formally, the view of the world of r at time t is defined as the last snapshot taken at a time smaller than or equal to t.

iii. Compute. The robot performs a *local computation* according to a deterministic algorithm $\mathcal{A}$ (we also say that the robot *executes* $\mathcal{A}$). The algorithm is the same for all robots, and the result of the *Compute* state is a *destination point*. Since the robots are oblivious, then $\mathcal{A}$ can access only the set of robots' positions retrieved during the last *Look*.

iv. Move. If the point computed in the previous state is the current location of r, we say that r performs a *null movement*, and it does not move; otherwise it moves towards the point computed in the previous state. The robot moves towards the computed destination of an unpredictable amount of space, which is assumed neither infinite, nor infinitesimally small (see Assumption A1 below). Hence, the robot can only go towards its goal, but it cannot know how far it will go in the current cycle, because it can stop anytime during its movement[3]. The amount of space traveled by a robot during this state is also called the *length* of the move.

The sequence: *Wait - Look - Compute - Move* will be called a *computation cycle* (or briefly *cycle*) of a robot.

The (global) time that passes between two successive states of the same robot is finite but unpredictable. In addition, no time assumption within a state is made. This implies that the time that passes after the robot starts observing the positions of all others and before it starts moving is arbitrary, but finite. That is, the actual move of a robot may be based on a situation that was observed arbitrarily far in the past, and therefore it may be totally different from the current situation.

This assumption of *asynchronicity within a cycle* is important in a totally asynchronous environment, since each robot has enough time to perform its local computation; furthermore, in this way it is possible to model also different motorial speeds of the robots.

In the model, there are only two limiting assumptions about space and time. The first one refers to space.

[3] That is, a robot can stop before reaching its destination point, e.g. because of limits to the robot's motorial capabilities.

Assumption A1(Distance). *The distance traveled by a robot r in a move is not infinite. Furthermore, there exists an arbitrarily small constant $\delta_r > 0$, such that if the destination point is closer than δ_r, r will reach it; otherwise, r will move towards it of at least δ_r.*

As no other assumptions on space exist, the distance traveled by a robot in a cycle is unpredictable.

Similarly, to prove that the algorithms designed in CORDA terminate in finite time, the following assumption on the length of a computational cycle is made.

Assumption A2(Computational Cycle). *The amount of time required by a robot r to complete a computational cycle is not infinite. Furthermore, there exists a constant $\varepsilon_r > 0$ such that the cycle will require at least ε_r time.*

As no other assumption on time exists, the resulting system is *fully asynchronous* and the duration of each activity (or inactivity) is unpredictable. As a result, the robots do not have a common notion of time, robots can be seen while moving, and computations can be made based on obsolete observations.

The robots do not necessarily share the same $x-y$ coordinate system, and do not necessarily agree on the location of the origin (that we can assume, without loss of generality, to be placed in the current position of the robot), or on the unit distance. In general, there is no agreement among the robots on the chirality of the local coordinate systems (i.e., in general they do not share the same concept of where North, East, South, and West are).

The robots are totally *oblivious*; that is, the robots can only store the robots' positions retrieved in the last observation.

The robots are completely *autonomous*: no central control is needed. Furthermore they are *anonymous*, meaning that they are a priori indistinguishable by their appearance, and they do not (need to) have any kind of identifiers that can be used during the computation[4].

Moreover, there are no explicit direct means of communication: any communication occurs in a totally implicit manner. Specifically, it happens by means of observing the robots' positions in the plane, and taking a deterministic decision accordingly. In other words, the only mean for a robot to send information to some other robot is to move and let the others observe (reminiscent of bees in a bee dance).

In the following, we will discuss in detail the implications of time settings.

2.2 Activation Schedules

Before proceeding to prove the main result of this paper, we need to describe in more detail the critical feature that regards the way the robots act during

[4] Note that the non obliviousness feature does not imply the possibility for a robot to find out which robot corresponds to which position it stored, since the robots are anonymous.

the computation; that is, the timing of the operations executed by each robot during its life.

In particular, in the model described so far, the amount of time spent in observation[5], in computation, in movement, and in inaction is finite but otherwise unpredictable; then, we say that the robots are *fully asynchronous*. In particular, the robots do not (need to) have a common notion of time. Each robot executes its actions at unpredictable time instants. This setting is adopted in CORDA. If the robots move according to this time setting, we say that they move according to an *asynchronous activation schedule*. Furthermore,

Definition 1. *An algorithm* $\mathcal{A}$ solves *a problem* $\mathcal{P}$ *in* CORDA *if, by activating the robots according to* any *asynchronous activation schedule, the robots reach a configuration such that the task defined by* $\mathcal{P}$ *is accomplished.*

In contrast, if the robots execute their activities (observation, computation, movement, and waiting) in an atomic and instantaneous fashion (that is, the amount spent in each activity of each cycle is negligible), we say that the robots are *atomically synchronized*, and that they move according to an *atomic activation schedule*. This temporal setting was first introduced by Suzuki *et al.* [10]; we will refer to this setting as ATOM.

Let us denote by $\mathfrak{C}$ and $\mathfrak{Z}$ the class of problems that are solvable in the asynchronous and the atomic setting, respectively. The relationship between these two classes is expressed from the following

Theorem 1 ([9]). $\mathfrak{C} \subset \mathfrak{Z}$.

Therefore, in order to prove the impossibility of GATHERING, it is sufficient to show that the problem is unsolvable in the atomic setting.

In an atomic activation schedule, at each time instant t, every robot r_i is either *active* or *inactive*. At least one robot is active at every time instant, and every robot becomes active at infinitely many unpredictable time instants[6]. For any $t \geq 0$, if r_i is inactive, then $p_i(t+1) = p_i(t)$; otherwise $p_i(t+1) = p$, where $p_i(t)$ denotes the position of robot r_i at time instant t, and p is the point returned by $\mathcal{A}$ [10].

Thus, an active robot r_i executes its cycle *atomically* and *instantaneously*, in the sense that a robot that is active and observes at t, has already reached its destination point p at $t+1$, and no fellow robot can see it *while* it is moving (or, alternatively, the movement is *instantaneous*).

We now introduce two general properties that follow from the ATOM setting, and that are not specific to the GATHERING. The first one stresses out the fact that, if a set of robots that at a given time instant t lie on the same position of the plane are all active at time t, then they will behave like they were one robot.

[5] i.e., activating the sensors and receiving their data.

[6] A special case is when every robot is active at every time instant; in this case we say that the robots are *strongly synchronized*. In [2, 10], the authors refer to this case simply as *synchronous*.

Lemma 1. *Let $\mathbb{H}$ be a set of black robots that at time t lie all on the same point $p^t_{\mathbb{H}}$. If all robots in $\mathbb{H}$ are active at time t, then at time $t+1$ all robots in $\mathbb{H}$ will again lie on the same position (possibly different from $p^t_{\mathbb{H}}$).*

Proof. The lemma follows from the fact that $\mathcal{A}$ is deterministic, the robots cannot detect multiplicity, and that all robots in $\mathbb{H}$ clearly have the same view of the world at t.

The following lemma points out that, if all robots in the system take the decision to move towards a point p at the same time instant t, then, even if a subset of them is blocked, all the others will still move towards p.

Lemma 2. *Let us assume that activating all robots at time t they gather on the same point p at time $t+1$, and let $\mathbb{H}$, with $1 \leq |\mathbb{H}| < n$, be any subset of robots that are not on p at t. If all robots not in $\mathbb{H}$ were still activated at t, and all robots in $\mathbb{H}$ were inactive at t, then all r_i, $r_i \notin \mathbb{H}$, will be on p at $t+1$, and all robots in $\mathbb{H}$ will not.*

Proof. The lemma follows from the lack of multiplicity detection and from the fact that $\mathcal{A}$ is deterministic.

3 Is Gathering Possible?

To our knowledge, in all solutions proposed to solve the Gathering, the ability of the robots to detect multiplicity (i.e., if on a given point there is more than one robot) is used either implicitly (like in [10]) or explicitly (like in [4]). Moreover, as already mentioned, the only attempt to avoid use of multiplicity detection to solve the problem, produced a solution that works only for non oblivious robots [3]. In other words, all previous solutions make some extra assumption on the capabilities of the robots. In this section, we indeed prove that Gathering is impossible in general.

In particular, we first focus on Atom; by Theorem 1, the result extends to Corda. In the following we assume that the n robots in the system execute only deterministic and oblivious algorithms according to atomic activation schedules. Moreover, we assume $n \geq 3$. In fact, in [10] it has been proven that there exists no oblivious algorithm that solves the problem in a model based on Atom when $n = 2$, under the assumption that two robots never collide (since they are modeled as no-dimensional points). Therefore, by Theorem 1, this case is unsolvable in Corda too[7].

Moreover, we denote by $\mathcal{A}$ a generic deterministic and oblivious algorithm, and by $\mathcal{A}_g$ an oblivious deterministic algorithm that correctly solves the gathering problem in Atom. Recall that $\mathcal{A}_g$ solves the gathering problem if, starting

[7] In [5], however, has been proved that the problem is trivially solvable in Corda, hence in Atom, if the robots can collide: in this case, in fact, it is sufficient to move the robots against each other until they gather.

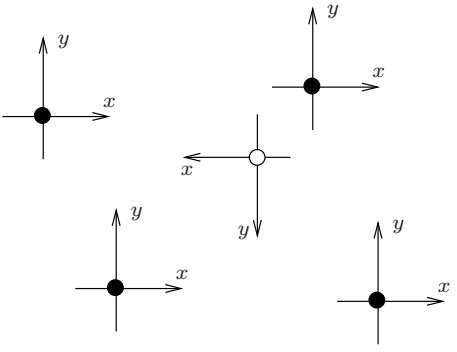

Fig. 1. Orientation of the axes of the black robots and of the white robot, in Assum3

from any valid initial configuration, it lets the robots gather on the same point $\mathfrak{p}$ in finite time: here, a valid initial configuration is a configuration where no two robots occupy the same position on the plane.

Finally, let $\mathbb{H}$ be a set of robots that at time t lie all together on the same point on the plane: in the following, we indicate such a position by $p^t_{\mathbb{H}}$, and by $|\mathbb{H}|$ the number of robots in $\mathbb{H}$.

3.1 The Proof: General Idea

The general idea to prove impossibility of GATHERING is as follows. First, we define a scenario that we will use to defeat any possible $\mathcal{A}_g$. In particular, in this scenario

Assum1. all robots have the same unit distance;

Assum2. $\delta = \delta_1 = \ldots = \delta_n$ (with δ_i as defined in Assumption A1 of Section 2.1);

Assum3. robots $r_1, \ldots, r_{n-1}$, from now on the *black* robots, have the same orientation and direction of the local coordinate system, while r_n, from now on the *white* robot, has a local coordinate system where both axes have the same direction but opposite orientation with respect to the coordinate system of the black robots (see Figure 1). In the following, we denote by p^t_w the position of the white robot at time t. The black and white coloring is used only for the sake of presentation, and this information is not used by the robots during the computation. The same applies for the indices given to the robots (they are anonymous).

We want to stress out, however, that Assum1–Assum3 are not known to the robots; hence they cannot use these information in their computations. Moreover, $\mathcal{A}_g$ correctly solves GATHERING iff the robots gather in finite time regardless their local unit measures, and the local orientation of their axes; hence, $\mathcal{A}_g$ must work also in a scenario described by Assum1–Assum3.

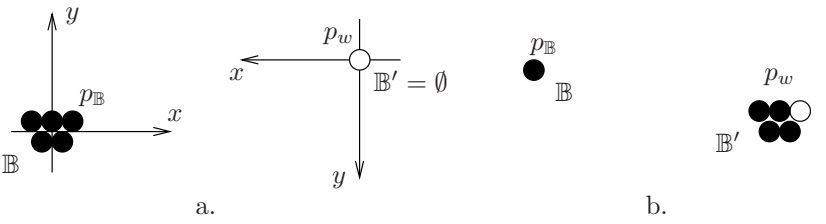

Fig. 2. In (a) a $\mathbb{E}_1$-configuration is depicted, while in (b) a $\mathbb{E}_2$-configuration. By Assum3 and since the robots cannot detect multiplicity, in both configurations (and in general in any $\mathbb{E}$-configuration) the white robot has the *same* view of the world as the robots in $\mathbb{B}$. In fact, both r_n and the robots in $\mathbb{B}$ see only one other robot on the point of coordinate (z, z'), with respect to their local coordinate systems

Second, we indeed show that there exists no $\mathcal{A}_g$ that can be executed in such a scenario according to an atomic activation schedule and that allows the robots to gather in a point in finite time. More specifically, we first show that, given $\mathcal{A}_g$, there exists always an atomic activation schedule that brings the robots, in a finite number of cycles, in a particular configuration, called $\mathbb{E}$-configuration, and defined as follows.

Definition 2 ($\mathbb{E}$-configuration). *An $\mathbb{E}$-configuration is a configuration of the robots where (i) the black robots are partitioned in two groups $\mathbb{B}$ and $\mathbb{B}'$, with $\mathbb{B}'$ possibly empty; (ii) the robots in $\mathbb{B}'$ and the white robot r_n lie on the same position p_w, and (iii) the robots in $\mathbb{B}$ lie on a position $p_{\mathbb{B}} \neq p_w$. Moreover, $\mathbb{E}_1$-configuration (shortly $\mathbb{E}_1$) is the $\mathbb{E}$-configuration where $\mathbb{B}' = \emptyset$ (see Figure 2.a), and $\mathbb{E}_2$-configuration (shortly $\mathbb{E}_2$) is the $\mathbb{E}$-configuration where $|\mathbb{B}| = 1$ and $|\mathbb{B}'| = n - 2$ (see Figure 2.b).*

Then, we prove that there exists an atomic activation schedule for $\mathcal{A}_g$ that, starting from a $\mathbb{E}$-configuration, lets the robots loop between $\mathbb{E}$-configurations, always avoiding the gathering.

Assume for a moment that at a given time t robots are in a $\mathbb{E}$-configuration; furthermore, let the robots in $\mathbb{B}$ (resp. the white robot) be active at t, and the robots in $\mathbb{B}'$ inactive for all $t' \geq t$. Then, since the robots cannot detect multiplicity, the robots in $\mathbb{B}$ and the white robot have the same view of the world at time t. Hence, since $\mathcal{A}_g$ is deterministic, we have that

Lemma 3. *If no robot changes position at time $t + 1$, then no robot will ever move, independently from their activation sequences (given that the robots in $\mathbb{B}'$ stay inactive).*

3.2 The Proof

As already outlined in Section 3.1, we first show that a $\mathbb{E}$-configuration can be reached by executing $\mathcal{A}_g$ according to a specific atomic activation schedule,

	t_s	t_s+1	$\cdots$	$t_{\mathbb{E}}-1$	$t_{\mathbb{E}}$
r_1	A	A	$\cdots$	A	
r_k	A	A	$\cdots$	I	
r_n	A	A	$\cdots$	A	

Fig. 3. The synchronous activation schedule $Sync\mathcal{F}_{\mathbb{E}}$ described in Lemma 4

say $Sync\mathcal{F}_{\mathbb{E}}$. Such a schedule is built as follows: at each cycle, if the robots, all activated, do not compute all the same destination point (according to the definition of $\mathcal{A}_g$), then they are activated and moved towards the destination point they compute. Otherwise, one of them, say r_k, is kept inactive, while all others are activated. In this way, the $n-1$ robots that are active will gather on the same point $\tilde{p}$, while r_k does not; hence, the robots are in a $\mathbb{E}$-configuration. More formally,

Lemma 4. *Given $\mathcal{A}_g$, there exists an atomic activation schedule $Sync\mathcal{F}_{\mathbb{E}}$ for $\mathcal{A}_g$, and a time $t_{\mathbb{E}} > 0$ such that, if the robots do not all occupy the same position on the plane when the execution of $\mathcal{A}_g$ starts, the robots are in $\mathbb{E}_1$ or $\mathbb{E}_2$ at time $t_{\mathbb{E}}$, if the computation is done according to $Sync\mathcal{F}_{\mathbb{E}}$.*

Proof. Let t_s be the time when the computation starts, and $pos_1, \ldots, pos_n$ be the positions occupied by the robots at this time. By hypothesis, there exist at least two positions pos_i and pos_j, $i \neq j$, such that $pos_i \neq pos_j$. $Sync\mathcal{F}_{\mathbb{E}}$ is reported in Schedule 1 (refer to Figure 3 for a pictorial representation).

Schedule 1 Build$_{\mathbb{E}}(t_s, pos_1, \ldots, pos_n)$.

Init. At the beginning, all robots are inactive. Set $t = t_s$, and go to Rule1.

Rule1. If normally activating all robots at time t they are not on the same point $\tilde{p}$ at time $t+1$, then in $Sync\mathcal{F}_{\mathbb{E}}$ all r_i are active at t. Set $t = t+1$, and go to Rule1. Otherwise,

Rule2. let r_k be a robot that is not on $\tilde{p}$ at time t. Then, in $Sync\mathcal{F}_{\mathbb{E}}$ all r_i, $i \neq k$, are active at t, while r_k is inactive at t.

In the following we will show that, starting the execution of $\mathcal{A}_g$ at time t_s according to $Sync\mathcal{F}_{\mathbb{E}}$, all robots are in a $\mathbb{E}_1$-configuration or $\mathbb{E}_2$-configuration at time $t_{\mathbb{E}} > t_s$. In fact, since by hypothesis $\mathcal{A}_g$ solves the problem, after finite time Rule2. is executed; hence $t_{\mathbb{E}}$ is finite. Moreover, until $t_{\mathbb{E}} - 1$ all robots are always active, and at this time, r_k is the only robot to be inactive.

By construction, $t_{\mathbb{E}}$ is the first time such that, if all the robots were normally activated at time $t_{\mathbb{E}} - 1$, they would be on the same position $\tilde{p}$ at time $t_{\mathbb{E}}$. Therefore, since there exists at least two positions pos_i and pos_j at time t_s such

that $pos_i \neq pos_j$, there must exist at least one robot r_k that is not on $\tilde{p}$ at time $t_{\mathbb{E}} - 1$. According to Rule2., r_k is inactive at $t_{\mathbb{E}} - 1$. By Lemma 2, at time $t_{\mathbb{E}}$ all robots r_i, $i \neq k$, are on $\tilde{p}$, and r_k is on a position different from $\tilde{p}$, and the lemma follows.

In the following two lemmas, we show that there is no algorithm that, starting from $\mathbb{E}_1$ or $\mathbb{E}_2$, lets the robots gather in a point.

Lemma 5. *There exists no deterministic oblivious algorithm that, starting from a $\mathbb{E}_1$-configuration, solves the gathering problem in a finite number of cycles for a set of $n \geq 3$ robots that can not detect multiplicity.*

Proof. By contradiction, let $\mathcal{A}_g$ be a deterministic oblivious algorithm that, starting from a $\mathbb{E}_1$-configuration, lets the robots gather in a point in finite time when they cannot detect multiplicity. In the following, we will describe an atomic activation schedule $Sync\mathcal{F}_{\mathbb{E}_1}$ for $\mathcal{A}_g$ such that, if the robots are in a $\mathbb{E}_1$-configuration at a given time t_s and the computation is done according to $Sync\mathcal{F}_{\mathbb{E}_1}$, the robots never gather in the same point $\mathfrak{p}$.

Schedule 2 Build$_{\mathbb{E}_1}(t_s, pos_1, \ldots, pos_n)$.

Init. At the beginning, all robots are inactive. Set $t = t_s$, and go to RuleB1.

RuleB1. If activating one of the black robots at time t, it is not on p_w^t at time $t+1$, then in $Sync\mathcal{F}_{\mathbb{E}_1}$ all black robots are activated at t and moved to the destination point they compute. The white robot is inactive at t. Set $t = t+1$, and go to RuleW1.

RuleB2. Otherwise,

- RuleB2.1 In $Sync\mathcal{F}_{\mathbb{E}_1}$, the black robots $r_1, \ldots, r_{n-2}$ are active at t and moved to the destination point they compute. The black robot r_{n-1} and the white robot r_n are inactive at t. Set $t = t+1$.
- RuleB2.2 In $Sync\mathcal{F}_{\mathbb{E}_1}$, the white robot is active at t and moved to the destination point it computes. All black robots are inactive at t. Set $t = t+1$.
- RuleB2.3 In $Sync\mathcal{F}_{\mathbb{E}_1}$, the black robot r_{n-1} is active at t and moved to the destination point it computes. The black robots $r_1, \ldots, r_{n-2}$ and the white robot r_n are inactive at t. Set $t = t+1$, and go to RuleW1.

RuleW1. If activating the white robot at time t, it is not on $p_{\mathbb{B}}^t$ at time $t+1$, then in $Sync\mathcal{F}_{\mathbb{E}_1}$ the white robot is activated at t and moved to the destination point it computes. The black robots are inactive at t. Set $t = t+1$, and go to RuleB1.

RuleW2. Otherwise,

- RuleW2.1 As in RuleB2.1.
- RuleW2.2 As in RuleB2.2.
- RuleW2.3 As in RuleB2.3, except that at the end of this step go to RuleB1.

Proof. Let $pos_1 = \ldots = pos_{n-1} = p_{\mathbb{B}}^{t_s}$, and $pos_n = p_w^{t_s}$. $Sync\mathcal{F}_{\mathbb{E}_1}$ is reported in Schedule 2 (refer to Figure 5 for a pictorial representation).

It follows from the definition of $\mathbb{E}_1$ that, at t_s, $p_{\mathbb{B}}^{t_s} \neq p_w^{t_s}$. $Sync\mathcal{F}_{\mathbb{E}_1}$ moves alternatively the black robots (as a group) and the white robot, until at time

t either the black robots compute as destination point p_w^t, or the white robot computes as destination point $p_{\mathbb{B}}^t$. When this happens, the gathering is avoided

1. by first moving all the black robots but one on p_w^t; then,
2. by moving the white robot on $p_{\mathbb{B}}^t$; and finally,
3. by moving the last black robot (still on $p_{\mathbb{B}}^t$) on p_w^t;

that is the black robots and the white robot are forced to switch their positions.

First note that, after every execution of RuleB1. all black robots *must change* position, and move *all together* towards the new destination (different from the position occupied by the white robot). In fact, let $t^* \geq t_s$ be a time instant when RuleB1. starts being executed, and such that all robots in $\mathbb{B}$ are on the same position $p_{\mathbb{B}}^{t^*} \neq p_w^{t^*}$. It follows from the definition of $Sync\mathcal{F}_{\mathbb{E}_1}$ that at t^* all black robots are active. If all these robots are still on $p_{\mathbb{B}}^{t^*}$ at the end of RuleB1. (that is at time $t^* + 1$), then by Lemma 3 no robot would ever move, hence the robots would never gather on the same point. Therefore, the robots in $\mathbb{B}$ cannot be on $p_{\mathbb{B}}^{t^*}$ at the end of RuleB1., and they *must change* position. Furthermore, since there is a black robot that, if active at t^*, would reach a position $p \neq p_w^{t^*}$ at time $t^* + 1$, by Lemma 1 the black robots will reach *all together* p at time $t^* + 1$, with $p \neq p_w^{t^*}$ and $p \neq p_{\mathbb{B}}^{t^*}$. Symmetrically, it follows that, if RuleW1. starts at time t^*, the white robot will be on a position $p \neq p_w^{t^*}$ and $p \neq p_{\mathbb{B}}^{t^*}$ at time $t^* + 1$, while all black robots are inactive at t^* (hence they are still on $p_{\mathbb{B}}^{t^*}$ at time $t^* + 1$). Therefore, as long as RuleB1. or RuleW1. are executed, the robots are in $\mathbb{E}_1$-configurations.

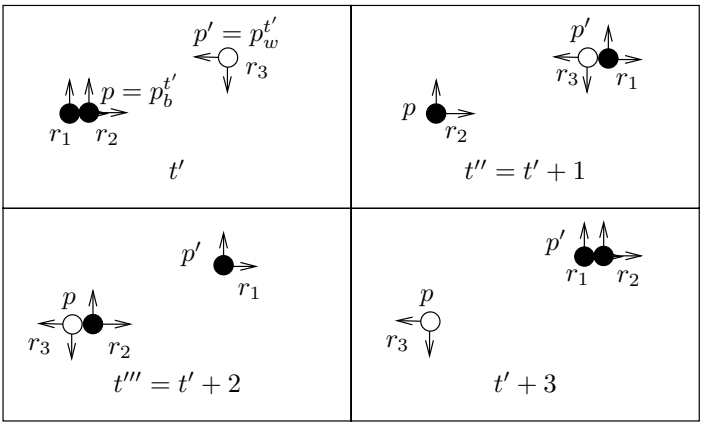

Fig. 4. Execution of RuleB2. in schedule $\mathbf{Build}_{\mathbb{E}_1}()$ in Lemma 5, with $n = 3$. At time t' each robot sees only one other robot; in particular, r_1 and r_2 see one robot on the point of coordinate (z, z') (with respect to their local coordinate system), and r_3 sees one robot on the point of coordinate (z, z') (with respect to its local coordinate system). That is, all the robots have the *same* view of the world. This view of the world is observed also by r_3 at time t'', and by r_2 at time t'''

	t_s	B1.	W1.		B2.1	B2.2	B2.3	W1.
r_1		A	I	$\cdots$	A	I	I	$\cdots$
$\vdots$		$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$
r_{n-2}		A	I	$\cdots$	A	I	I	$\cdots$
r_{n-1}		A	I	$\cdots$	I	I	A	$\cdots$
r_n		I	A	$\cdots$	I	A	I	$\cdots$

Fig. 5. The synchronous activation schedule $Sync\mathcal{F}_{\mathbb{E}_1}$ described in Lemma 5. Here is depicted the case when RuleB2. is invoked first

Since, by hypothesis, $\mathcal{A}_g$ solves the problem, after a finite number of cycles either RuleB2. or RuleW2. is executed. Without loss of generality, let us assume that RuleB2. is executed first, say at time $t' > t_s$ (the case when RuleW2. is executed first can be handled similarly). Thus, according to $Sync\mathcal{F}_{\mathbb{E}_1}$, $n-2$ black robots are active at time t', while r_{n-1} and r_n are inactive (RuleB2.1). This rule is chosen because there is a black robot that, if normally activated at t', would compute $p_w^{t'}$ as destination point. Hence, by Lemma 1, the $n-2$ active robots will leave $p = p_{\mathbb{B}}^{t'}$ and reach $p' = p_w^{t'}$ (Figure 4).

At this point, RuleB2.2 is invoked at time $t'' = t' + 1$: the white robot is active at t'', while all black robots are inactive. By Assum1–Assum3 and since multiplicity cannot be detected, r_n has the same view of the world that the black robots that moved in RuleB2.1 had at time t' (refer to Figure 4); specifically, the white robot sees only one robot, that is the last black robot r_{n-1} that at this time is still on p (r_{n-1} is inactive at t' and t''). As a consequence, since $\mathcal{A}_g$ is oblivious and deterministic, the result of the *Compute* state of r_n at t'' is the same as the result of the *Compute* state that the black robots performed at time t' (in RuleB2.1): that is, r_n decides to reach the only other robot it sees (r_{n-1}), hence r_n computes p as destination point. Therefore, at time $t'' + 1$ the white robot reaches r_{n-1} on p.

Finally, RuleB2.3 is started at time $t''' = t'' + 1$: the last black robot r_{n-1} (still on p) is active at t''', while all the other black robots (at this time on p') and r_n (on p) are inactive. At time t''', r_{n-1} has the same view of the world that the black robots that moved in RuleB2.1 had at time t'; specifically, since it can not distinguish multiplicity, it sees all other black robots (on p') as one robot. Therefore it computes p' as destination point, and reaches all the other black robots at time $t''' + 1$.

In conclusion, if RuleB2.1 is started at time t', at time $t'''+1 = t'+3$ all black robots are on p', and the white robot is on p. That is, the black and white robots simply switched positions, and at time $t'+3$ they are again in a $\mathbb{E}_1$-configuration. Therefore, by executing $\mathcal{A}_g$ according to $Sync\mathcal{F}_{\mathbb{E}_1}$, the robots will never gather on the same point. This leads to a contradiction, and the lemma follows.

Lemma 6. *In* Corda *there exists no deterministic oblivious algorithm that, starting from a $\mathbb{E}_2$-configuration, solves the gathering problem in a finite number of cycles for a set of $n \geq 3$ robots that can not detect multiplicity.*

Proof. By contradiction, let $\mathcal{A}_g$ be a deterministic oblivious algorithm that, starting from a $\mathbb{E}_2$-configuration, lets the robots gather in a point in finite time when they cannot detect multiplicity. Similarly to the previous lemma, we will describe a synchronous activation schedule $Sync\mathcal{F}_{\mathbb{E}_2}$ for $\mathcal{A}_g$ such that, if the robots are at a given time t_s in a $\mathbb{E}_2$-configuration and the computation is done according to $Sync\mathcal{F}_{\mathbb{E}_2}$, the robots never gather in the same point $\mathfrak{p}$. By Lemma 1,

Schedule 3 Build$_{\mathbb{E}_2}(t_s, pos_1, \ldots, pos_n)$.

Init. At the beginning, all robots are inactive. Set $t = t_s$, and go to Rule1.

Rule1. If activating all robots at time t, they are not on the same position $\tilde{p}$ at time $t+1$, then in $Sync\mathcal{F}_{\mathbb{E}_2}$ all robots are normally activated. Set $t = t+1$, and go to Rule1.

Rule2. Otherwise,

- Rule2.1 If no robot is on $\tilde{p}$ at time t, then in $Sync\mathcal{F}_{\mathbb{E}_2}$ all robots in $\mathbb{B}'$ and r_{n-1} are active at t and moved to the destination point they compute. The white robot r_n is inactive at t. Set $t = t+1$, and go to RuleB1. defined in Lemma 5.
- Rule2.2 If r_n is on $\tilde{p}$ at time t, then all robots in $\mathbb{B}'$ are active at t, while r_{n-1} and r_n are inactive at t. Set $t = t+1$, and go to Rule1.
- Rule2.3 If r_{n-1} is on $\tilde{p}$ at time t, then all robots in $\mathbb{B}'$ are active at t, while r_n and r_{n-1} are inactive at t. Set $t = t+1$, and go to RuleB1. in Schedule 2.
- Rule2.4 If all robots in $\mathbb{B}'$ are on $\tilde{p}$ at time t, then r_{n-1} is active at t, while the robots in $\mathbb{B}'$ and r_n are inactive. Set $t = t+1$, and go to RuleB1. in Schedule 2.

as long as Rule1. is executed, all robots in $\mathbb{B}'$ move always all together; hence, at any time, they always occupy the same position on the plane. Since by hypothesis $\mathcal{A}_g$ solves the problem, after a finite number of cycles Rule2. is executed, say at time t', and let $\tilde{p}$ as defined in Rule2., that is the point where the robots would gather if all active at t'.

It follows from the definition of $\mathbb{E}_2$ that at the beginning $p_{\mathbb{B}}^{t_s} \neq p_w^{t_s}$. Without loss of generality, let us assume that $r_1, \ldots, r_{n-2}$ are the black robots in $\mathbb{B}'$ (at t_s they lie on $p_w^{t_s}$), and that r_{n-1} is the only robot in $\mathbb{B}$.

$Sync\mathcal{F}_{\mathbb{E}_2}$ moves all robots until they decide to gather on the same point (eventually this happens, since by hypothesis $\mathcal{A}_g$ solves the problem); in particular, all robots in $\mathbb{B}'$ are forced to move together, hence to lie always on the same point. When this happens, the robots are forced to reach either a $\mathbb{E}_1$ or a $\mathbb{E}_2$-configuration. At this point, $Sync\mathcal{F}_{\mathbb{E}_2}$ behaves exactly like $Sync\mathcal{F}_{\mathbb{E}_1}$ described in the previous lemma; hence it avoids the gathering. Let $pos_1, \ldots, pos_{n-2}, pos_n = p_w^{t_s}$, and $pos_{n-1} = p_{\mathbb{B}}^{t_s}$. $Sync\mathcal{F}_{\mathbb{E}_2}$ is reported in Schedule 3 (refer to Figure 6 for a pictorial representation).

First, note that it is impossible that at time t' the robots in $\mathbb{B}'$ and r_n are already on $\tilde{p}$, while the only robot in $\mathbb{B}$ is not. In fact, let us assume that r_n and the robots in $\mathbb{B}'$ are already on $\tilde{p}$ at time t'; thus, the robots are in a $\mathbb{E}$-configuration at t'. Rule2. is executed at t' because, if all the robots were active at t', they would

be on $\tilde{p}$ at time $t' + 1$; hence, since by hypothesis r_n and the robots in $\mathbb{B}'$ are already on $\tilde{p}$ at time t', these robots would not move between time t' and $t' + 1$. Therefore, is like the robots in $\mathbb{B}'$ are inactive at t'. Hence, by Lemma 3, no robot would change position between time t' and $t' + 1$, hence they would not gather on $\tilde{p}$ at time $t' + 1$, and Rule2. would not have been executed at time t'. Similarly, it can be proven that

> it is impossible that at time t' the robot in $\mathbb{B}$ and r_n are already on $\tilde{p}$, while the robots in $\mathbb{B}'$ are not (it is sufficient to switch the roles of $\mathbb{B}$ and $\mathbb{B}'$ in Lemma 3); and
>
> it is impossible that at time t' the robot in $\mathbb{B}$ and those in $\mathbb{B}'$ are already on $\tilde{p}$, while r_n is not.

Moreover, since by hypothesis t' is the first time such that activating all robots, they would gather on the same point, it can not be that all robots are already on $\tilde{p}$ at t'. In the following, we analyze the remaining possible cases.

1. *No robot is on $\tilde{p}$ at time t'.* In this case, Rule2.1 is executed, and r_n is inactive at t'. Hence, by Lemma 2, at time $t' + 1$ all robots but r_n are on $\tilde{p}$; that is, the robots are in a $\mathbb{E}_1$-configuration.
2. *Only r_n is already on $\tilde{p}$ at time t'.* In this case, Rule2.2 is executed, and the robots in $\mathbb{B}'$ are active at t', while r_{n-1} and r_n are inactive. Hence, by Lemma 2, at time $t' + 1$ all robots in $\mathbb{B}'$ and r_n are on $\tilde{p}$, while r_{n-1} is not. That is, the robots do not gather in $\tilde{p}$ at $t' + 1$, and they are again in a $\mathbb{E}_2$-configuration.
3. *Only r_{n-1} is already on $\tilde{p}$ at time t'.* In this case, Rule2.3 is executed: at t', r_{n-1} and r_n are inactive, while the robots in $\mathbb{B}'$ are active. By Lemma 2, at time $t' + 1$ all robots but r_n are on $\tilde{p}$; that is, the robots are in a $\mathbb{E}_1$-configuration.
4. *Only the robots in $\mathbb{B}'$ are already on $\tilde{p}$ at time t'.* Rule2.4 is executed. Using an argument similar to the one used in the previous case, it follows that also in this case the robots are in a $\mathbb{E}_1$-configuration at time $t' + 1$.

In conclusion, at time $t' + 1$, either the robots are in a $\mathbb{E}_1$-configuration or again in a $\mathbb{E}_2$-configuration. In the first case, the lemma follows by Lemma 5. In the second case, either Rule2.2 is never executed again after $t' + 1$, or every time it is executed the robots are once again in a $\mathbb{E}_2$-configuration. In both cases, the lemma follows.

To summarize, thus far we proved that,

> given any algorithm $\mathcal{A}_g$, there exists an atomic activation schedule that, starting from any valid configuration for the gathering problem, brings the robots either in a $\mathbb{E}_1$ or $\mathbb{E}_2$-configuration in a finite number of cycles (Schedule 1);

	t_s Rule1.	Rule1.		Rule2.	B1.
r_1	A	A	$\cdots$	A	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$
r_{n-2}	A	A	$\cdots$	A	$\cdots$
r_{n-1}	A	A	$\cdots$	A	$\cdots$
r_n	A	A	$\cdots$	I	$\cdots$

Fig. 6. The synchronous activation schedule $Sync\mathcal{F}_{\mathbb{E}_2}$ described in Lemma 6. The case when Rule2.1 is executed first is depicted

there exists no deterministic oblivious algorithm that, starting form a $\mathbb{E}_1$ or $\mathbb{E}_2$-configuration, solves the gathering problem in a finite number of cycles (Schedules 2 and 3 in the Appendix).

Hence, by Lemmas 4–6, and by Theorem 1, it follows that

Theorem 2. *In* CORDA *and* ATOM*, there exists no deterministic oblivious algorithm that solves the* GATHERING *problem in a finite number of cycles, hence in finite time, for a set of* $n \geq 2$ *robots.*

References

1. N. Agmon and D. Peleg. Fault Tolerant Gathering Algorithms for Autonomous Mobile Robots. In *Proc.* 15^{th} *Symposium on Discrete Algorithms (SODA 2004)*, pages 1063–1071, 2004.
2. H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. A Distributed Memoryless Point Convergence Algorithm for Mobile Robots with Limited Visibility. *IEEE Transactions on Robotics and Automation*, 15(5):818–828, 1999.
3. M. Cieliebak. Gathering non-oblivious mobile robots. In *Proc. Latin American Conf. on Theoretical Informatics (LATIN '04)*, LNCS 2976, pages 577–588, 2004.
4. M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Solving The Gathering Problem. In *Proc.* 30^{th} *International Colloquium on Automata, Languages and Programming (ICALP '03)*, pages 1181–1196, 2003.
5. M. Cieliebak and G. Prencipe. Gathering Autonomous Mobile Robots. In *Proc. of* 9^{th} *International Colloquium On Structural Information And Communication Complexity (SIROCCO 9)*, pages 57–72, June 2002.
6. R. Cohen and D. Peleg. Robot Convergence via Center-of-gravity Algorithms. In *Proc.* 11^{th} *International Colloquium On Structural Information And Communication Complexity (SIROCCO 11)*, pages 79–88, 2004.
7. P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Hard Tasks for Weak Robots: The Role of Common Knowledge in Pattern Formation by Autonomous Mobile Robots. In *Proc.* 10^{th} *Annual International Symposium on Algorithms and Computation (ISAAC '99)*, LNCS 1741, pages 93–102, December 1999.
8. P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of Autonomous Mobile Robots With Limited Visibility. In *Proc.* 18^{th} *International Symposium on Theoretical Aspects of Computer Science (STACS 2001)*, LNCS 2010, pages 247–258, February 2001.

9. G. Prencipe. The Effect of Synchronicity on the Behavior of Autonomous Mobile Robots. *Theory of Computing Systems*, 2004. (accepted for publication).
10. I. Suzuki and M. Yamashita. Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. *Siam Journal of Computing*, 28(4):1347–1363, 1999.