Asynchronous Pattern Formation: the effects of a rigorous approach

Serafino Cicerone*, Gabriele Di Stefano*, Alfredo Navarra†

*Department of Information Engineering, Computer Science and Mathematics
University of L'Aquila, Via Vetoio, I–67100, L'Aquila, Italy.
Email: {serafino.cicerone, gabriele.distefano}@univaq.it
†Department of Mathematics and Computer Science
University of Perugia, Via Vanvitelli 1, I–06123, Perugia, Italy.
Email: alfredo.navarra@unipg.it

arXiv:1706.02474v1 [cs.DC] 8 Jun 2017

Abstract

Given a multiset F of points in the Euclidean plane and a set R of robots such that $|R| = |F|$, the Pattern Formation (PF) problem asks for a distributed algorithm that moves robots so as to reach a configuration similar to F. Similarity means that robots must be disposed as F regardless of translations, rotations, reflections, uniform scalings. Initially, each robot occupies a distinct position. When active, a robot operates in standard Look-Compute-Move cycles. Robots are asynchronous, oblivious, anonymous, silent and execute the same distributed algorithm. So far, the problem has been mainly addressed by assuming chirality, that is robots share a common left-right orientation. We are interested in removing such a restriction.

While working on the subject, we faced several issues that required close attention. We deeply investigated how such difficulties were overcome in the literature, revealing that crucial arguments for the correctness proof of the existing algorithms have been neglected.

Here we design a new deterministic distributed algorithm that solves PF for any pattern when asynchronous robots start from asymmetric configurations, without chirality. The focus on asymmetric configurations might be perceived as an over-simplification of the subject due to the common feeling with the PF problem by the scientific community. However, we demonstrate that this is not the case. The systematic lack of rigorous arguments with respect to necessary conditions required for providing correctness proofs deeply affects the validity as well as the relevance of strategies proposed in the literature. Our new methodology is characterized by the use of logical predicates in order to formally describe our algorithm as well as its correctness. In addition to the relevance of the obtained results, the new techniques might help in revisiting previous results in order to design new algorithms. In fact, it comes out that well-established results for PF like [Fujinaga et al., *SIAM J. Comp.* 44(3) 2015] or more recent approaches like [Bramas et al., *Brief Announcement PODC* 2016] revealed to be not correct. Our claim is not just based on some marginal counter-examples but we show how fundamental properties have been completely ignored, hence affecting the rationale behind the proposed strategies.

Keywords

Distributed Algorithms, Mobile Robots, Pattern Formation, Asynchrony

The work has been supported in part by the European project "Geospatial based Environment for Optimisation Systems Addressing Fire Emergencies" (GEO-SAFE), contract no. H2020-691161 and by the Italian project "RISE: un nuovo framework distribuito per data collection, monitoraggio e comunicazioni in contesti di emergency response", Fondazione Cassa Risparmio Perugia, code 2016.0104.021.

I. Introduction

In distributed computing, one of the most studied problem is certainly the *Pattern Formation* (PF) which is strictly related to *Consensus* and *Leader Election*. Given a team of robots (agents or entities) and a geometric pattern in terms of points in the plain with respect to an ideal coordinate system, the goal is to design a distributed algorithm that works for each robot to guide it so that eventually all robots together form the pattern. As the global coordinate system might be unknown to the robots, a pattern is declared formed as soon as robots form a pattern similar to the input one, that is regardless of translations, rotations, reflections, uniform scalings.

The PF problem has been largely investigated in the last years under different assumptions. Different characterizations of the environment consider whether robots are fully-synchronous, semi-synchronous (cf. [1]–[3]) or asynchronous (cf. [4]–[8]):

- **Fully-synchronous** (FSYNC): The *activation* phase (i.e. the execution of a Look-Compute-Move cycle) of all robots can be logically divided into global rounds. In each round all the robots are activated, obtain the same snapshot of the environment, compute and perform their move.
- **Semi-synchronous** (SSYNC): It coincides with the FSYNC model, with the only difference that not all robots are necessarily activated in each round.
- **Asynchronous** (ASYNC): The robots are activated independently, and the duration of each phase is finite but unpredictable. As a result, robots do not have a common notion of time. Moreover, they can be seen while moving, and computations can be made based on obsolete information about positions.

One of the latest and most important results, see [6], solves the problem for robots endowed with few capabilities. Initially, no robots occupy the same location, and they are assumed to be:

- Dimensionless: modeled as geometric points in the plane;
- Anonymous: no unique identifiers;
- Autonomous: no centralized control;
- Oblivious: no memory of past events;
- Homogeneous: they all execute the same *deterministic* algorithm;
- Silent: no means of direct communication;
- Asynchronous: there is no global clock that synchronizes their actions;
- Chiral: they share a common left-right orientation.

In particular, in ASYNC, the robots are activated independently, and the duration of each phase is finite but unpredictable. As a result, robots do not have a common notion of time. Moreover, they can be seen while moving, and computations can be made based on obsolete information about positions.

When active, a robot operates in standard *Look-Compute-Move* cycles. In one cycle a robot takes a snapshot of the current global configuration (Look) in terms of robots' positions according to its own coordinate system. Successively, in the Compute phase it decides whether to move toward a specific direction or not, and in the positive case it moves (Move).

During the Look phase, robots can perceive *multiplicities*, that is whether a same point is occupied by more than one robot, and how many robots compose a multiplicity.

Cycles are performed asynchronously, i.e., the time between Look, Compute, and Move phases is finite but unbounded, and it is decided by an adversary for each robot. Moreover, during the Look phase, a robot does not perceive whether other robots are moving or not. Hence, robots may move based on outdated perceptions. In fact, due to asynchrony, by the time a robot takes a snapshot of the configuration, this might have drastically changed when it starts moving. The scheduler determining the Look-Compute-Move cycles timing is assumed to be fair, that is, each robot becomes active and performs its cycle within finite time and infinitely often.

The distance traveled within a move is neither infinite nor infinitesimally small. More precisely, the adversary has also the power to stop a moving robot before it reaches its destination, but there exists an unknown constant $\nu > 0$ such that if the destination point is closer than ν, the robot will reach it, otherwise the robot will be closer to it of at least ν. Note that, without this assumption, an adversary would make impossible for any robot to ever reach its destination.

The main open question left in [6] within ASYNC concerns the resolution of the more general PF problem in the described setting but without chirality. So far, the only sub-problems solved within the weakest setting are the gathering problem [9], where all robots must move toward a common point, and the circle-formation problem [10], [11], where n robots must form a regular n-gon.

The contribution of this work is threefold: (1) to provide counter-examples to the correctness of algorithms for the PF problem proposed in well-established and in recent papers, (2) to solve the PF problem with asynchronous robots without chirality when the initial configurations are asymmetric, and (2) to use a rigorous approach for handling problems in the asynchronous environment able to provide accurate arguments to state the correctness of the designed algorithms.

A. *Motivation and related work*

Starting with [12], we keep on investigating about PF without chirality but we faced several issues mainly related to asynchrony that required close attention. Since the main difficulties were not depending on the lack of chirality, we deeply investigated how such problems were overcome in the literature. In particular, we closely explored [6] – that can be considered as a milestone in the advancement of PF study within ASYNC– and we found that our difficulties with the PF problem have not been addressed at all. Actually, we are able to devise fundamental counter-examples to the correctness of [6]. We contacted the authors of [6] and they confirmed us that the provided counter-example and most importantly the rationale behind it represents a main issue that requires deep investigation.

In order to catch a first idea of our findings, it is necessary to understand what should be carefully analyzed when dealing with asynchrony and robots that do not share a common coordinate system. The first problem arising when approaching PF is where robots should form the pattern F, that is how to embed F on the area occupied by robots so as each point of F can be seen as a 'landmark'. Both in [6] and in our strategy, the algorithms are logically divided into phases. Usually the first phase is devoted to move a few of robots in such a way the embedding becomes easy. Then, there is an intermediate phase where all robots but those placed in the first phase are moved in order to form F. Finally there is a third phase where the 'special' robots are moved to finalize F. While in the second phase it is relatively easy to move robots to partially form F because the embedding is well defined, this is not the case for the other two phases. For instance, if not carefully managed, it may happen that during a move the configuration changes its membership to a different phase, especially from the first phase. In order to provide the correctness proof of an algorithm under the sketched scheme, it is not sufficient to define some invariants that exclusively define the membership of a configuration to a phase (as done so far in the literature), but it is mandatory to prove that the defined moves cannot change the membership of the current configuration *while robots are moving*. In the ASYNC model a change of membership is possible, if not carefully considered, as robots can be seen while moving. If such a situation happens, other robots 'believing to be in a different phase' may start moving and then the situation becomes sometimes intractable or even they prevent the algorithm to accomplish the PF. Unfortunately this is the case for [6], where the authors neglected such situations, and we can provide counter-examples where their algorithm fails. It is worth to remark, that the systematic lack of arguments with respect to such events completely invalidates the correctness and the relevance of the strategy proposed in [6]. Moreover, it is not possible to recover the algorithm with some easy patches as it requires structural intervention.

On this basis, and in order to better understand the problem, we restricted our attention to the case where initial configurations are asymmetric. Even if solving this 'restricted' version of the PF problem seems to be an easy task, the analysis of the literature revealed the following state of art:

- A first study can be found in [8]. The authors provide a distributed algorithm that can form many patterns, subject to significant restrictions in the input configurations. One of such constraints almost coincides with asymmetry, but it is not the only one. Unfortunately, the way it is presented does not allow to exactly specify from which kind of configurations robots can start and which patterns are formable.
- A formal characterization that includes asymmetric configurations has been conducted in [13], [14] where also a nice comparison between PF and leader election shows the equivalence of the two problems under some circumstances. Still the results are based on chirality. Moreover, the patterns considered do not allow multiplicities, that is the points of any given pattern to be formed by n robots compose a set of n distinct elements. In our study, as well as in [6], patterns may allow multiplicities, and this deeply affect the design of resolution algorithms.
- Other approaches present in the literature to solve PF are probabilistic, see [15], [16]. In particular, in [15] the authors claim to solve PF with a strategy that is divided into two main phases: the first phase is probabilistic and is used to make asymmetric the input configuration; the second phase is deterministic and solves PF from asymmetric configurations even without assuming any form of multiplicity detection. An extended version of the paper can be found in [17]. Basically, the strategy in the second phase solves the problem we are investigating even without any multiplicity detection. Unfortunately, the proposed strategy suffers of similar problems revealed for [6]. In particular, the lack of rigorous arguments supporting the correctness of the proposed algorithm led to inconsistent results. We are able to provide counter-examples that affect the rationale behind the proposed strategy, and then also in this case the proposed algorithm is not correct.
- Further 'unofficial' results can be found in [18], [19]. Concerning [18], not all patterns are considered but only asymmetric ones. Concerning [19], the authors claim to solve the asymmetric case by slightly modifying the results of [14]. There are three main issues about this paper. First of all, the main proof is given by a sketchy description, whereas we show how formal arguments are extremely necessary in this context. Secondly, the patterns considered in [19] do not allow multiplicities, that is the points of any given pattern to be formed by n robots compose a set of n distinct elements. In our study, as well as in [6], patterns may allow multiplicities, and this deeply affect the design of resolution algorithms. In fact, also the gathering becomes a sub-problem of PF where it is required to solve the so-called *point formation*. It is well-known how difficult is the resolution of the gathering problem [9]. As a consequence, studying asymmetric configurations for any number of robots and for patterns including multiplicities is much harder. Finally, the minimum number of robots required by their algorithm is 5. Note that in robot based computing systems, instances with small numbers of robots usually require very different and non-trivial arguments with respect to the general algorithm. This is the case for instance for the square formation [11] and for gathering both in the Euclidean plane [9] or in discrete rings [20]–[23].

Such an analysis also motivates the third aim of this paper, that is to provide a rigorous approach for designing algorithms in ASYNC. The need of new ways of expressing algorithm in ASYNC is widely recognized. For instance, in [24]–[26] a formal model to describe mobile robot protocols under synchrony and asynchrony assumptions is provided. So far, these only concern robots operating in a discrete space i.e., with a finite set of possible robot positions.

B. Our results

We provide fundamental arguments affecting the correctness of both [6] and [15]. The purpose is to convince the community that something wrong has been systematically accepted in the literature. The relevance of our finding is given not only by the fact that we re-open the PF problem in the ASYNC context, but also that possibly other tasks may suffer of the same arguments. It follows that a problem considered closed and easy like for PF in the case of asymmetric configurations must be carefully revisited. We show that its resolution is far from being an easy task, especially for providing an 'impeccable' correctness proof.

We fully solve the PF problem when initial configurations are asymmetric, that is for any number of robots we can form any pattern, including symmetric ones and those with multiplicities. Since we do not assume chirality, symmetries to be considered for the patterns are not only rotations as in [6], [13], [14], but also reflections.

Finally, we design our algorithm according to a rigorous approach. Such an approach is based on basic predicates that composed in a Boolean logic way provides all the invariants needed to be checked during the execution of the algorithm. In contrast to previous approaches used in the literature, we use invariants that describe properties holding during the movements of robots. In turn, this implies that for each single move the algorithm may require three different invariants (to describe properties at the start, during, and at the end of the move). Hence, our algorithm is organized as a set of moves, each associated to up three invariants. Moves are grouped and associated to a phase, where a phase represents a general task of the algorithm. Summarizing, the approach leads to a greater level of detail that provides us rigorous arguments to state the correctness of the algorithm. This approach itself represents a result of this paper, as it highlights crucial properties in ASYNC contexts that so far have been underestimate in the literature.

As further remarks, it is worth to note that differently from [6], we do not require that the local coordinate system specific of a single robot remains the same among different Look-Compute-Move cycles. Moreover, the trajectories traced during a move specified by our algorithm are always well-defined either as straight lines or as rotations along specified circles.

Finally, our algorithm does not require to specify the pattern to be formed as a set of coordinates in a Cartesian system. There are two possible options. As in [8], the pattern can be specified as a list of ratios of its sides and angles between the sides. Another option is to provide the list of distances among all points. In both cases, each robot can locally evaluate a possible set of points consistent with the input and its local coordinate system. Clearly, it turns out that doing this way robots do not share the same information about the pattern but each one acquires its own representation.

C. Outline

In the next section, some further details on the considered robots' model are provided. In Section III, useful notation and definitions are introduced. Section IV provides a first description of our strategy to solve PF in ASYNC without chirality and starting from asymmetric configurations. The section also contains a description of the counter-examples we used to convince the scientific community that PF in ASYNC is back of being an open problem, even for asymmetric configurations. Section V, provides our new distributed algorithm. It is given in terms of various sub-phases where different moves are performed. The correctness of the algorithm is given in Section VI. Finally, Section VII, concludes the paper.

II. ROBOT MODEL

The robot model is mainly borrowed from [6] and [12]. We consider a system composed of n mobile *robots*. At any time, the multiset $R = \{r_1, r_2, \ldots, r_n\}$, with $r_i \in \mathbb{R}^2$, contains the *positions* of all the robots.

We arbitrarily fix an x-y coordinate system Z_0 and call it the *global coordinate system*. A robot, however, does not have access to it: it is used only for the purpose of description, including for specifying input. All actions taken by a robot are done in terms of its local x-y coordinate system, whose origin always indicates its current position. Let $r_i(t) \in \mathbb{R}^2$ be the location of robot r_i (in Z_0) at time t, where $\mathbb{R}$ is the set of real numbers. Then a multiset $R(t) = \{r_1(t), r_2(t), \ldots, r_n(t)\}$ is called the *configuration* of R at time t (and we simply write R instead of $R(t)$ when we are not interested in any specific time).

A robot is said to be stationary in a configuration $R(t)$ if at time t it is:

- inactive, or
- active, and:
  - it has not taken the snapshot yet;
  - it has taken snapshot $R(t)$;
  - it has taken snapshot $R(t')$, $t' < t$, which leads to a null movement.

A configuration R is said to be *stationary*[1] if all robots are stationary in R.

If an element in $r \in R$ occurs more than one time, then r is said to belong to (or compose) a *multiplicity*. A configuration R is said *initial* if it is stationary and all elements in R are distinct, that is, no multiplicity occurs. In this work we assume that initial configurations are asymmetric.

Each robot r_i has a local coordinate system Z_i, where the origin always points to its current location. Let $Z_i(p)$ be the coordinates of a point $p \in \mathbb{R}^2$ in Z_i. If r_i takes a time interval $[t_0, t_1]$ for performing the Look phase, then it obtains a multiset $Z_i(R(t)) = \{Z_i(r_1(t)), Z_i(r_2(t)), ..., Z_i(r_n(t))\}$ for some $t \in [t_0, t_1]$, where $Z_i(r_i(t)) = (0, 0)$. That is, r_i has the (strong) multiplicity detection ability and can count the number of robots sharing a location. More generally, if P is a multiset of points, for any x-y coordinate system Z, by $Z(P)$ we denote the list of the coordinates $Z(p)$ in Z for all $p \in P$.

Let $\{t_i : i = 0, 1, \ldots\}$ be the set of time instances at which a robot takes the snapshot $R(t_i)$ in Look. Without loss of generality, we assume $t_i = i$ for all $i = 0, 1, \ldots$. Then, an infinite sequence $\mathbb{E} : R(0), R(1), \ldots$ is called an execution with an initial configuration $I = R(0)$ that by definition is stationary and without multiplicities. Actually, depending on the algorithm, multiplicities may be created in $R(i)$, with $i > 0$.

Unlike the initial configuration, in general, not all robots are stationary in $R(i)$ when $i > 0$, but at least one robot that takes the snapshot $R(i)$ is stationary by definition. Whether or not a given configuration R is stationary (or a robot is stationary at R) depends not only on R but also on the execution history, in general. Let $\mathbb{E} : R(0), R(1), \ldots, R(f)$ and $\mathbb{E}' : R'(0), R'(1), \ldots$ be two executions, and assume $R(f) = R'(0)$. Then $\mathbb{E}\mathbb{E}' : R(0), R(1), \ldots, R(f)(= R'(0)), R'(1), \ldots$ is always a correct execution for SSYNC (and hence for FSYNC) robots since $R(f)$ is stationary by the definition of SSYNC robots. However, this is not the case for ASYNC robots, since in $\mathbb{E}'$, $R'(0)$ is assumed to be stationary, but in $\mathbb{E}$, $R(f)$ may not be; the transition from $R'(0)$ to $R'(1)$ may be caused by the Look of a robot r which is moving at $R(f)$ and hence cannot observe $R(f)$. If an algorithm can guarantee that $R(f)$ is stationary, like for SSYNC robots, then we can safely concatenate $\mathbb{E}$ and $\mathbb{E}'$ to construct a legitimate execution even for ASYNC robots. An execution fragment that starts and ends at a stationary configuration is called a *phase*.

Let P_1 and P_2 be two multisets of points: if P_2 can be obtained from P_1 by translation, rotation, reflection, and uniform scaling, P_2 is *similar* to P_1. Given a pattern F expressed as a multiset $Z_0(F)$, an algorithm A *forms* F from an initial configuration I if for any execution $\mathbb{E} : R(0)(= I), R(1), R(2), \ldots$, there exists a time instant $i > 0$ such that $R(i)$ is similar to F and no robots move after i, i.e., $R(t) = R(i)$ hold for all real numbers $t \geq i$.

[1]The definition of stationary robot provided in [6] is slightly different but also inaccurate. In fact, it does not catch the third scenario about active robots described by our definition. If removing such a case, no configuration might be declared stationary during an execution.

Definition 1. *Let R be a configuration and r be a robot moving towards a target p according to a move m dictated by algorithm A. Assume that r is the only robot moving in R. Let $[t_1, t_2]$ be any time interval in which r is moving but it has not yet reached p, and let R' be any configuration observed during $[t_1, t_2]$. We say that m is* safe *if in R' algorithm A allows only robot r to move; A is* transition-safe *if each move in A is safe.*

Note that, according to the above definition, if R' is stationary and has been obtained from R by means of move m, necessarily the adversary has stopped r. From there, if A is transition-safe then either move m is again performed by r or still r moves according to a move $m' \neq m$ (possibly toward a target $p' \neq p$).

For the sake of readability, Definition 1 is suitably designed for the case where it is possible to select a single robot to move. However, it can be generalized to the case of many robots involved by a move, when symmetries occur.

III. Notation and basic properties

Given two distinct points u and v in the plane, let $d(u, v)$ denote their distance, let $line(u, v)$ denote the straight line passing through these points, and let (u, v) ($[u, v]$, resp.) denote the open (closed, resp.) segment containing all points in $line(u, v)$ that lie between u and v. The half-line starting at point u (but excluding the point u) and passing through v is denoted by $hline(u, v)$. We denote by $\sphericalangle(u, c, v)$ the angle centered in c and with sides $hline(c, u)$ and $hline(c, v)$. The angle $\sphericalangle(u, c, v)$ is measured from u to v in clockwise or counter-clockwise direction, the measure is always positive and ranges from 0 to less than 360 degrees, and the direction in which it is taken will be clear from the context.

Given an arbitrary multiset P of points in $\mathbb{R}^2$, $C(P)$ and $c(P)$ denote the smallest enclosing circle of P and its center, respectively. Let C be any circle concentric to $C(P)$. We say that a point $p \in P$ is *on* C if and only if p is on the circumference of C; ∂C denotes all the points of P that are on C. We say that a point $p \in P$ is *inside* C if and only if p is in the area enclosed by C but not in ∂C; $int(C)$ denotes all the points inside C. The radius of C is denoted by $\delta(C)$. The smallest enclosing circle $C(P)$ is unique and can be computed in linear time [27]. A useful characterization of $C(P)$ is expressed by the following property.

Property 2. *[28] $C(P)$ passes either through two of the points of P that are on the same diameter (antipodal points), or through at least three points. $C(P)$ does not change by eliminating or adding points to $int(P)$. $C(P)$ does not change by adding points to $\partial C(P)$. However, it may be possible that $C(P)$ changes by either eliminating or changing positions of points in $\partial C(P)$.*

Given a multiset P, we say that a position $p \in P$ is *critical* if and only if $C(P) \neq C(P \setminus \{p\})$[2]. It easily follows that if $p \in P$ is a critical position, then $p \in \partial C(P)$.

Property 3. *[29] If $|\partial C(P)| \geq 4$ then there exists at least one point in $\partial C(P)$ which is not critical.*

Given a multiset P, consider all the concentric circles that are centered on $c(P)$ and with at least one point of P on them: $C^i_\uparrow(P)$ denotes the i-th of such circles, and they are ordered so that by definition $C^0_\uparrow(P) = c(P)$ is the first one, $C(P)$ is the last one, and the radius of $C^i_\uparrow(P)$ is greater than the radius of $C^j_\uparrow(P)$ if and only if $i > j$. Additionally, $C^i_\downarrow(P)$ denotes one of the same concentric circles, but now they are ordered in the opposite direction: $C^0_\downarrow(P) = C(P)$ is the first one, $c(P)$ by definition is the last one, and the radius of $C^i_\downarrow(P)$ is greater than the radius of $C^j_\downarrow(P)$ if and only if $i < j$.

The radius of three of such circles will play a special role in the remainder: $\delta_0(P) = \delta(C^0_\downarrow(P))$, $\delta_1(P) = \delta(C^1_\downarrow(P))$, and $\delta_2(P) = \delta(C^2_\downarrow(P))$ (with δ_1 and δ_2 equal to zero when the corresponding circles do not exist).

[2]Note that in this work we use operations on multisets.

Definition 4. *Let R be a configuration. We define $\delta_{0,1} = (\delta_0(R) + \delta_1(R))/2$ and $\delta_{0,2} = (\delta_0(R) + \delta_2(R))/2$, and we denote by $C^{0,1}(R)$ and $C^{0,2}(R)$ the circles centered on $c(R)$ and with radii $\delta_{0,1}$ and $\delta_{0,2}$, respectively.*

Definition 5. *Let R be a configuration and F a pattern. Assuming $C(R) = C(F)$, let $d = \delta(C^1_\uparrow(F))$. The* guard circle *$C^g(R)$ and the* teleporter circle *$C^t(R)$ are defined as the circles centered on $c(R)$ of radii equal to $d/2^i$ and $d/2^{(i-1)}$, respectively, with $i > 1$ being the minimum integer such that the following conditions hold:*

- $int(C^g(R)) \setminus c(C^g(R)) = \emptyset$*;*
- $|int(C^t(R)) \setminus int(C^g(R))| \leq 1$*;*
- $|\partial C^t(R)| \leq 1$.

Circles $C^g(R)$ and $C^t(R)$ are not defined for any configuration. Circle $C^g(R)$ will be used by our algorithm as the place where a specific robot g is moved in order to maintain asymmetry during the formation of pattern F. Circle $C^t(R)$, which is larger than $C^g(R)$, represents the place closest to $c(R)$ where a robot deviates in case its trajectory should traverse $C^g(R)$. Doing so, no robots can be confused with g.

Definition 6. *Let F be a pattern, the* reference angle *α is defined as $\alpha = \frac{1}{3} \cdot \min\{\sphericalangle(x, c(F), y) : x, y \in \partial C(F), x \neq y\}$.*

Such a reference angle will be used to correctly place robot g in order to maintain asymmetry during the formation of pattern F.

A. View of a point and symmetries

We now introduce the concept of *view* of a point in the plane; it can be used by robots to determine whether a configuration R and/or a pattern F is asymmetric or not.

Given a generic set of points P, a map $\varphi : P \to P$ is called an *isometry* or distance preserving if for any $p, q \in P$ one has $d(\varphi(p), \varphi(q)) = d(p, q)$. This can be extended to the case where P is a multiset as follows. Given $p \in P$ then the multiplicity on $\varphi(p)$ must be of the same cardinality of the multiplicity on p. If P admits only the identity isometry, then P is said *asymmetric*, otherwise it is said *symmetric*.

Given a point $p \in P$, $p \neq c(P)$, then $V^+(p)$ denotes the counter-clockwise view of P computed from p. Essentially, $V^+(p)$ is a string whose elements are the polar coordinates of all points in P. The elements in $V^+(p)$ are arranged as follows: first p, then in order and starting from $c(P)$ all the points in the ray $hline(c(P), p)$, and finally all points in the other rays, rays processed in counter-clockwise fashion. Similarly, $V^-(p)$ denotes the clockwise view of P computed from p. By assuming a lexicographic order for polar coordinates, the view of p is defined as $V(p) = \min\{V^+(p), V^-(p)\}$.

If $c(P) \in P$ then $c(P)$ is said the point in P of *minimum view*, otherwise any $p = \operatorname{argmin}\{V(p') : p' \in P\}$ is said of minimum view in P. Given P and $P' \subseteq P$, we use the notation *min_view*(P') to denote any point p with minimum view in P'.

The possible symmetries that P can admit are reflections and rotations. P admits a reflection if and only if there exist two points $p, q \in P$, $p, q \neq c(P)$, not necessarily distinct, such that $V^+(p) = V^-(q)$;[3] P admits a rotation if and only if there exist two distinct points $p, q \in P$, $p, q \neq c(P)$, such that $V^+(p) = V^+(q)$. It follows that if P is asymmetric then there exists a unique multipoint with minimum view.

We now redefine the concept of clockwise (and hence of counter-clockwise) direction for a multiset of points P so as to make it independent of a global coordinate system.[4] If P is asymmetric and $p =$

[3] When chirality can be exploited, reflections can be ignored as $V^+(p)$ can be always discriminated from $V^-(q)$.

[4] Indeed, this is not necessary when chirality is assumed.

min_view$(P \setminus c(P))$, then the direction used to compute $V(p)$ during the analysis of all the rays starting from $c(P)$ is the *clockwise direction* of P. If P is symmetric, there might be many multipoints of minimum view. Symmetries we take care of are of two types: rotations and reflections. If P is rotational (but not reflexive), again the direction used to compute $V(p)$ from any point $p \neq c(P)$ of minimum view determines the *clockwise direction* of P. If P is reflexive, any direction can be assumed as the clockwise direction of P since they are indistinguishable.

We can now apply the redefined concept of clockwise direction to a configuration R and/or a pattern F. For instance, in Fig. 4 left, the multiset R is asymmetric, hence its clockwise direction coincides with that used to compute $V(r)$, with $r =$ *min_view*(R); whereas in Fig. 4 right, the multiset F is rotational and reflexive, hence its clockwise direction does not distinguish a left-right orientation.

IV. THE STRATEGY

In this section, we provide a general description of the strategy underlying our algorithm. It is based on a functional decomposition approach: the problem is divided into three sub-problems respectively denoted as `RefSys` (Reference System), `ParForm` (Partial Formation), and `Fin` (Finalization). For each sub-problem an algorithm is provided. Each algorithm is defined in a way that its execution consists of a sequence of phases. The whole strategy is then realized by composing the algorithms of each phase.

First of all, $C(F)$ is scaled to $C(R)$, and this will never change (but for the special case of instances composed of just three robots). We now provide a high-level description of the three sub-problems.

Problem `RefSys`: It concerns the main difficulty arising when the pattern formation problem is addressed: the lack of a unique embedding of F on R that allows each robot to uniquely identifying its target (the final destination point to form the pattern).[5] In particular, `RefSys` can be described as the problem of moving some (minimal number of) robots into specific positions such that they can be used by any other robot as a common reference system. Such a reference system should imply a unique mapping from robots to targets, and should be maintained along all the movements of robots (but for the *finalization phase*, where the algorithm moves the robots forming the reference system).

Our strategy solves `RefSys` by using three robots (called *guards*). Such guards are positioned as described in Fig. 1: the *boundary guards* are denoted as g' and g'' and are two antipodal robots on $C(R)$, the *internal guard* is denoted as g and is the unique robot on $C^g(R)$ (the *guard circle*). The internal guard is placed so that the angle $\sphericalangle(g, c(R), g')$ is equal to a value α (the *reference angle*) whose value only depends on F. Once `RefSys` is solved, each robot can use the guards to univocally determine its target position in the subsequent phases.

Notice that in specific cases the presence of $C^g(R)$ implies a refinement to the strategy defined for the sub-problem `RefSys`. In fact, in case of a multiplicity on $c(F)$, all the robots on $c(R)$ (but one) must be placed before $C^g(R)$ (and hence the internal guard g) is used. Then, two distinct phases are designed for addressing `RefSys`:

- phase $\mathcal{F}1$, responsible for setting the external guards g' and g'' and, if required, for placing the multiplicity on $c(R)$;
- phase $\mathcal{F}2$, responsible for setting the internal guard g.

Problem `ParForm`: This sub-problem concerns moving all the non-guards robots (i.e., $n-3$ robots) toward the targets. In our strategy, a phase $\mathcal{F}3$ is designed to solve this problem. The difficulties in this phase are the following: (1) during the phase, the reference system must be preserved, (2) the movements must

[5] In the literature, this is sometime realized by inducing a common coordinate system. This method can be effective only if F is specified by coordinates and not by distances.

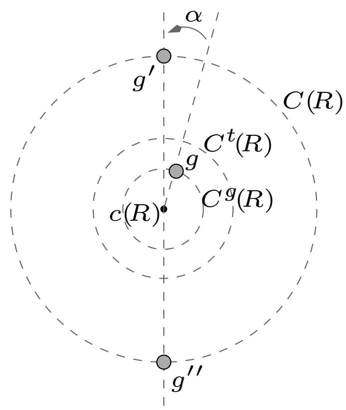

Figure 1: Basic concepts and notation in the definition of the common reference system.

be performed by avoiding undesired multiplicities (collision-free routes), and (3) the movements must be performed without entering into the guard circle (routing through the teleporter circle).

Once the guards are placed, a unique robot per time is chosen to be moved toward its target: it is the one not on a target, closest to an unoccupied target, and of minimum view in case of tie. We are ensured that always one single robot r will be selected since the configuration is maintained asymmetric by the guards. The selected robot is then moved toward one of the closest targets until it reaches such a point. All such moves must be performed so as to avoid the occurrence of undesired multiplicities; hence, it follows that sometimes the movements are not straightforward toward the target point but robots may deviate their trajectories. To this aim, the strategy makes use of a procedure called COLLISIONFREEMOVE designed ad-hoc for computing alternative trajectories. Moreover, according to the role of the internal guard g, robots cannot enter into the circle $C^g(R)$, otherwise the reference system induced by the guards is lost. The latter implies that, in case of a possible route passing through $C^g(R)$, a robot avoids entering into the guard circle by deviating along the boundary of $C^t(R)$ (the *teleporter circle*); such a circle is centered on $c(R)$ and its radius is opportunely chosen so that in its interior there is only g and in $\partial C^t(R)$ there is at most one robot.

Problem `Fin`: It refers to the so-called finalization phase, where the last three robots (the guards) must be moved to their targets to complete the formation of pattern F. In our strategy, a phase $\mathcal{F}4$ is designed to solve this problem. This phase must face a complex task, since (1) moving the guards leads to the loss of the common reference system, and (2) moving without a common reference system makes hard to finalize the pattern formation.

A. Further details on the strategy

An additional problem, that crosses different phases, is that described in the Introduction. It is the problem of avoiding that moves defined in the algorithm can change the membership of the current configuration *while robots are moving*. In fact, if such a situation happens, other robots 'believing to be in a different phase' may start moving and then the situation becomes sometimes intractable or even they prevent the algorithm to accomplish the PF. This is the case for the algorithm proposed in [6], where the authors neglected such situations: in the next sub-section we provide a possible counter-example where their algorithm fails. In contrast, our strategy correctly addresses such a general problem by making use of an ad-hoc procedure called STATIONARYMOVE. This procedure is invoked to control each move that potentially could lead to non-stationary configurations, and hence to change the membership of the current configuration in an uncontrolled manner. Among others, Procedure STATIONARYMOVE exploits two main properties of our algorithms that are ensured during the whole computation: there is at most one moving robot and $C(R(0)) = C(R(t)) = C(F)$,

$t > 0$. For instance, while solving `RefSys`, the embedding of F into R is not yet defined. Hence, in some cases StationaryMove can force moving robots crossing circles $C^i_{\downarrow}(F)$ to stop on such circles (hence potentially on a point of F). In this way, we force the configuration to become stationary in a moment where its membership may have changed, because perhaps an embedding of F that leads to a successive phase holds.

B. A counter-example to the correctness of the algorithm presented in [6]

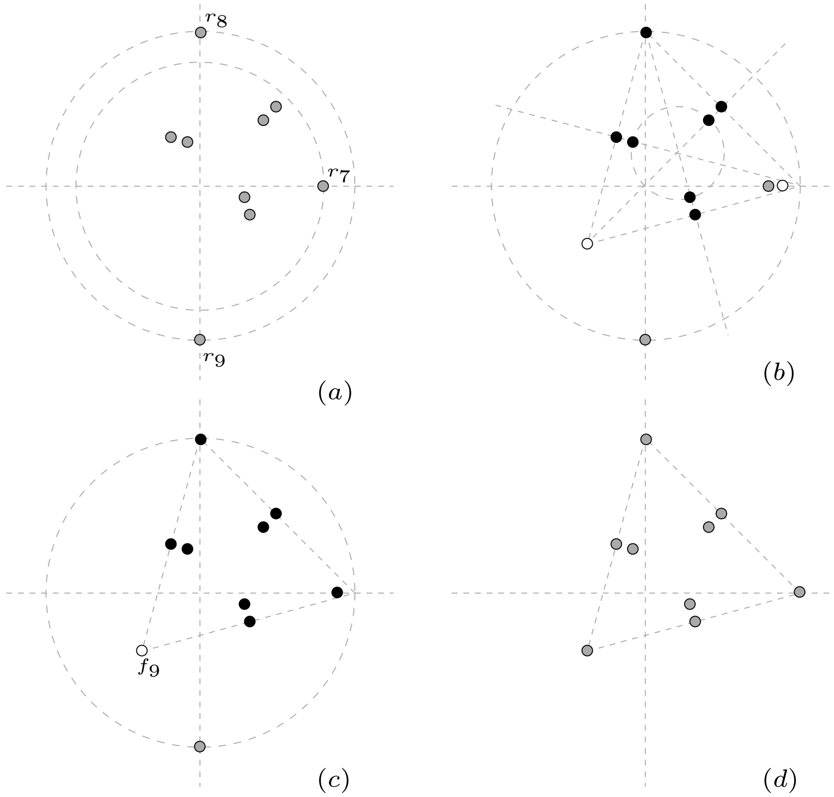

Figure 2: A counter-example to the correctness of the algorithm $FORM$ presented in [6]. Grey circles represent robots, white circles represent points in the pattern F, black circles represent both robots and points in F. (a) An initial configuration I; (b) A visualization of a pattern F to be formed along with points of I; (c) A configuration R obtainable from I during the movement of robot r_7 in the first phase (execution of the algorithm A_1 to form a T-stable configuration). From it, since invariant INV_c holds, also robot r_9 might start moving (toward f_9); (d) A symmetric configuration R' obtainable if both r_7 and r_9 complete their scheduled moves.

In this section, we provide a counter-example to the correctness of the algorithm $FORM$ presented in [6] to solve PF assuming chirality. $FORM$ is designed to form a pattern F (possibly with multiplicities) from an initial configuration I (without multiplicity) each time $\rho(I)$ divides $\rho(F)$, where $\rho(\cdot)$ is the parameter that captures the symmetricity of a multiset of points in the plane (cf. [6]). Since the algorithm assumes robots empowered with chirality, function $\rho(\cdot)$ only measures the number of possible rotations that the input set of points admits. In fact, dealing with chirality overcome managing reflections.

An execution of algorithm $FORM$ is partitioned into four phases called the pattern embedding (EMB), the embedded pattern formation (FOR), the finishing (FIN), and the gathering (GAT) phases. Phase EMB embeds a given pattern F, phase FOR forms a substantial part $\tilde{F}$ of F, and phase FIN forms the remaining

$F \setminus \tilde{F}$ of F to complete the formation. Phase GAT treats a pattern F with multiplicities. These phases occur in this order, but some of them may be skipped.

For each of the phases, the authors present an algorithm and an invariant (i.e., predicate) that every configuration in the phase satisfies. Each invariant INV is considered as a set too; a configuration R satisfies INV if and only if $R \in INV$. Authors show that the defined invariants are pairwise disjoint, so exactly one of the algorithms implementing the phases is executed.

We start by briefly recalling both the invariant INV_{EMB} and the algorithm A_{EMB} for the first phase EMB. For a formal understanding of the arguments below, the reader is invited to refer to [6] for the definition of ℓ-stable configurations, which in turn defines a set $\Lambda \subseteq \partial C(R)$:

- $INV_{EMB} = \neg(INV_{FOR} \vee INV_{FIN})$. Informally, INV_{FOR} is the set of the so called ℓ-stable configurations R such that not all robots in $R \setminus \Lambda$ are located at their final positions in F, while INV_{FIN} is the set of configurations R such that all robots in $R \setminus \Lambda$ are located at their final positions in F.
  Fig. 2.(a) shows an initial configuration I such that $I \in INV_{EMB}$. Notice that the number of robots in I is odd and $\rho(I) = 1$ (since I is asymmetric).
- A_{EMB} consists of three algorithms A_1, A_2, and A_3, that are devoted to three different cases. Such algorithms are responsible of forming a ℓ-stable configuration. In particular A_1 is responsible for forming a T-stable configuration, that is a configuration R with exactly three robots in $\partial C(R)$ such that two of them are antipodal and the third one is a midpoint of them. Actually, A_1 is invoked when $|\partial C(R)| = 2$ (and this is the case in I), and it moves an additional robot on $C(R)$ to get a T-stable configuration.
  From I, algorithm A_1 moves robot r_7 straightly toward the point $[c(I), r_7] \cap C(I)$.

Both the invariant INV_{FIN} and the algorithm A_{FIN} for the third phase FIN are also necessary to build our counter-example. The finishing phase consists of different invariants depending on the kind of ℓ-stable configuration obtained in the first phase. In particular, when the T-stable configuration has been built according to algorithm A_1, INV_{FIN} consists of three invariants INV_a, INV_b, and INV_c, each associated to an algorithm that moves one of the three robots on $C(R)$. What we need to explore for the counter-example is INV_c.

- A configuration R satisfies INV_c if and only if $R \setminus F = \{r\}$ and r is on τ_c, where τ_c is a route designed as follows:
  - Let $R = \{r_1, r_2, \ldots, r_n\}$, $F = \{f_1, f_2, \ldots, f_n\}$ be the set of robots and pattern points ordered according to their distance from $c(R)$ and $c(F)$, respectively. Assume that $c(R) = c(F)$ and that $R \setminus \{r_n\}$ is similar to $F \setminus \{f_n\}$. In such a case, pattern F can be formed from R by moving just r_n toward f_n along a route τ_c defined as any route such that, for any point p on τ_c, still r_n is the unique robot to be moved (toward f_n) to form F from R', where R' is constructed from R by replacing r_n with p.

  Concerning the algorithm executed when INV_c holds, it simply requires that robot r_n traces the path τ_c.

We have recalled all the details necessary to describe the counter-example. Assume now that algorithm A_1 is moving r_7 to form a T-stable configuration, and consider the pattern F depicted in Fig. 2.(b) composed by the black and white circles. Notice that $\rho(F) = 1$ as F is asymmetric. Since A_1 moves robot r_7 straightly toward the point $[c(I), r_7] \cap C(I)$, at a certain time it is possible that robot r_9 observes (during a Look phase) the configuration R depicted in Fig. 2.(c). This means that during the movement of r_7, robot r_9 starts moving toward f_9 according to the algorithm associated to the invariant INV_c. If this happens, the following properties hold:

1) there are two moving robots;
2) each move is due to a different phase;
3) if both moving robots reach their current targets – see Fig. 2.(d) – then the obtained configuration R' admits $\rho(R') = 3$ and from there it would be impossible to form F. In fact, as proved in [6], F is formable from R' if and only if $\rho(R')$ divides $\rho(F)$, but here $\rho(F) = 1$.

By personal communications, the authors of [6] confirmed us that the provided counter-example and most importantly the rationale behind it represents a main issue that requires deep investigation. While it is possibly easy to find a patch to the counter-example by slightly modifying algorithm A_1, it is not straightforward to provide general arguments that can ensure the correctness of the whole algorithm. The main question left is: how can be guaranteed among all the phases that the membership of a configuration to a phase does not change while a robot is moving?

As we are going to show in the correctness section, our algorithm does not suffer of such arguments as it prevents such scenarios.

C. *A counter-example to the correctness of the algorithm presented in [17]*

In this section, we show how missing arguments affect the correctness of [17].

Similarly to our approach, the algorithm in [17] selects a specific robot r_1 closest to $c(P)$ to serve as what we call guard. Such a robot is moved in a specific placement in order to be always recognized as such until the very last step that finalizes the formation of the input pattern F.

Another ingredient of the algorithm presented in [17] we need to describe for our purpose is how the authors get rid of any form of multiplicity detection. When F contains multiplicities, the robots first form a different pattern $\tilde{F}$ obtained by the robots from F as follows: for each point p of multiplicity k, $k-1$ further points $p_1, \ldots, p_{k-1}$ are added such that the distance of each p_i from $c(F)$ equals the distance of p from $c(F)$, $\sphericalangle(p, c(F), p_i) < p_i$, and $|p_i - p| = \frac{d}{4i}$, with $d = \min_{f,f' \in F} |f - f'|$.

For all algorithm phases except *Termination*, $\tilde{F}$ is used instead of F. To execute the Termination phase, the configuration must be totally ordered (using the set of points, excluding multiplicity information), and at least one robot must be located at each point of F (except maybe the smallest one). If there exists a robot $r \neq r_1$ not located at a point in F, then r chooses the closest point in F as its destination and rotates toward it while remaining in its circle. The global coordinate system remains unchanged because r_1 does not move. Eventually, r_1 becomes the only robot not located at its destination, then it moves toward it, and the pattern F is formed.

As shown in the description of our strategy, and in particular in phase $\mathcal{F}2$, before creating our internal guard g, we ensure to move $k-1$ robots on $c(P)$ if F admits a multiplicity of k elements in $c(F)$. Such type of patterns (with a multiplicity in $c(F)$) are completely ignored in [17]. In such a case, the general description provided above does not apply. First of all, $\tilde{F}$ is not well-defined. Secondly, once r_1 is correctly placed, k robots should be moved close or perhaps on $c(P)$. This is not clear but in any case there will be robots getting closer to $c(P)$ with respect to r_1, hence affecting the global coordinate system.

The lack of details as well as of a rigorous methodology lead the authors to design a partial algorithm that cannot cope with all possible input. It is worth to remark that in our strategy, the case of a multiplicity in $c(F)$ required a separate phase due to the arisen difficulties. The design of such a phase is not an easy task since, in general, several robots must be moved at or close to $c(P)$ *before* the global coordinate system is established.

V. THE ALGORITHM

In this section, a robot always means an oblivious ASYNC robot. Recall that an initial configuration is asymmetric and does not contain multiplicities by definition. Concerning the number of robots n, for $n = 1$

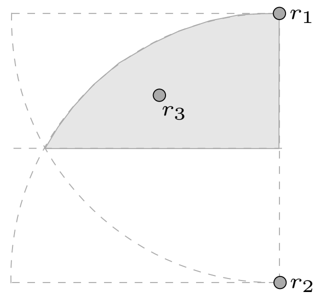

Figure 3: A picture showing arguments for the proof of Theorem 1.

the problem is trivial. When $n = 2$ the problem is either trivial (if F is composed of two distinct points) or unsolvable (gathering of two robots, see [9]).

Concerning the pattern to form, it might contain multiplicities. The case of point formation (Gathering) is delegated to [9], so we do not consider such a case as input for our algorithm.

Similarly to [6], for $n = 3$, we design an ad-hoc algorithm.

Theorem 1. *Let $R = \{r_1, r_2, r_3\}$ be an asymmetric stationary configuration with three robots, and let $F = \{f_1, f_2, f_3\}$ be a pattern. Then, there exists an algorithm able to form F from R.*

Proof: We assume that F does not contain a point of multiplicity 3, otherwise the algorithm in [9] for the Gathering problem can be used. Without loss of generality, we assume $dist(r_1, r_2) \geq dist(r_2, r_3) \geq dist(r_3, r_1)$. Since R is asymmetric, we get $dist(r_1, r_2) > dist(r_2, r_3) > dist(r_3, r_1)$. It follows that, without loss of generality, we can assume r_3 inside (and not on the boundary) of the shaded area shown in Fig. 3. Assuming $dist(f_1, f_2) \geq dist(f_2, f_3) \geq dist(f_3, f_1)$, the algorithm is based on the following steps: (1) robots embeds f_1 and f_2 on r_1 and r_2, respectively, (2) robots elect r_3 as the only robot allowed to move, (3) r_3 embeds f_3 as closer as possible to itself, and (4) r_3 moves straightly toward f_3 (in multiple steps if necessary). According to step 3, it follows that f_3 is inside the shaded area of Fig. 3 if F is asymmetric, or on the boundary of that area if F is symmetric without multiplicities, or on $f_1 = r_1$ if there is a 2 multiplicity in F. It is clear that, during the movements, r_3 will be always elected as the only robot allowed to move and hence it eventually reaches the target to complete the pattern formation. ∎

In the remainder, we will provide an algorithm able to form a pattern F (which is not a single point) starting from any asymmetric stationary configuration with $n \geq 4$. A possible input for the algorithm is shown in Fig. 4. We construct our algorithm in such a way that the execution consists of a sequence of phases $\mathcal{F}1$, $\mathcal{F}2$, $\mathcal{F}3$, and $\mathcal{F}4$ (some of which might be skipped). To each phase, we assign an invariant such that every configuration satisfies exactly one of the invariants (hence robots can correctly recognize the phase in which they are). Since each algorithm associated to a phase is composed of different kinds of moves, each phase is divided into sub-phases. Each sub-phase is characterized by a single move. Moreover, each time robots switch to a different phase/sub-phase, the current configuration is stationary, that is the move performed in a sub-phase is initiated from a stationary configuration (this property is crucial to prove the correctness of our algorithm). Basically, whenever a robot becomes active, it can deduce from the acquired snapshot to which phase and sub-phase the observed configuration belongs to, and whether it is the robot designated to move. In case it is its turn to move, it applies the move associated to the sub-phase detected. As it will be shown in the proof of correctness, our algorithm always maintains the current configuration asymmetric until the pattern is formed, hence it assures there will always be at most one robot moving and the moves are all safe. In turn we prove the algorithm is transition-safe.

phase	*var*	*definition*
$*$	$\mathtt{a}$	R is asymmetric
$\mathcal{F}1$	$\mathtt{s}_2$	$\lvert\partial C(R)\rvert = 2$
	$\mathtt{s}_3$	$\lvert\partial C(R)\rvert = 3$
	$\mathtt{s}_+$	$\lvert\partial C(R)\rvert > 3$
	$\mathtt{t}_0$	$\mathtt{s}_3 \wedge$ points in $\partial C(R)$ form a triangle of angles all different from $90°$
	$\mathtt{t}_1$	$\mathtt{s}_3 \wedge$ points in $\partial C(R)$ form a triangle of angles equal to $30°$, $60°$, and $90°$
	$\mathtt{l}$	$\lvert\partial C^1_\downarrow(R)\rvert \geq 2 \vee \delta_1 \leq \delta_{0,2}$
	$\mathtt{m}_0$	$\lvert\{c(F)\} \cap F\rvert = k,\, k > 1$
	$\mathtt{m}_1$	$\mathtt{m}_0 \wedge \lvert\{c(R)\} \cap R\rvert = k - 1$
$\mathcal{F}2$	$\mathtt{c}$	$\lvert\{c(R)\} \cap R\rvert = 1$
	$\mathtt{g}_0$	$\lvert\partial C^g(R)\rvert = 1$
	$\mathtt{g}_1$	$\mathtt{g}_0 \wedge \exists! g' \in \partial C(R) : \sphericalangle(g, c(R), g') = \alpha$
	$\mathtt{g}_2$	$\mathtt{g}_1 \wedge \exists g'' \in \partial C(R)$ antipodal to g'
	$\mathtt{f}_2$	$(\mathtt{m}_0 \Rightarrow \mathtt{m}_1) \wedge \mathtt{s}_2 \wedge \mathtt{l}$
$\mathcal{F}3$	$\mathtt{d}_0$	$\partial C^t(R) = \{r\}$
	$\mathtt{d}_1$	$\mathtt{d}_0 \wedge \exists f^* \in F^* :\ r = [c(F), f^*] \cap C^t(R)$
	$\mathtt{d}_2$	$C^t(R) \cap (r, \mu(r)] \neq \emptyset$, where $r = \mathit{min\_view}(R_\eta^{\neg m})$
	$\mathtt{f}_3$	$(\mathtt{m}_0 \Rightarrow \mathtt{m}_1) \wedge \mathtt{g}_2$
$\mathcal{F}4$	$\mathtt{q}$	$\partial C(F) = F$, F without multiplicities, and $\lvert F\rvert - 1$ points of F are on the same semi-circle
	$\mathtt{i}_1, \ldots, \mathtt{i}_6$	guards g, g' and g'' are detectable $\wedge\ R \setminus \{g, g''\}$ is similar to $F \setminus \{\mu(g), \mu(g'')\}$
	$\mathtt{f}_4$	$\neg\mathtt{q} \wedge (\mathtt{i}_1 \vee \mathtt{i}_2)\ \vee\ \mathtt{q} \wedge (\mathtt{i}_3 \vee \mathtt{i}_4 \vee \mathtt{i}_5 \vee \mathtt{i}_6)$
$*$	$\mathtt{w}$	R is similar to F

Table I: The basic Boolean variables used to define all the phases' invariants. In the first column, the phase in which the corresponding variables are mainly used. Missing notations can be found in the corresponding sections. About predicates $\mathtt{q}$ and $\mathtt{i}_1, \ldots, \mathtt{i}_6$, they are used to recognize the guards in different scenarios where guards have to be moved in the finalization phase; they are formally defined in Section V-D.

Table I contains all predicates required by our algorithm, whereas Table II describes in which phase a configuration is according to the specified properties. Notice that for each arbitrary phase/sub-phase $\mathcal{X}$ we need three predicates $\mathcal{X}_s$, $\mathcal{X}_d$, and $\mathcal{X}_e$ to distinguish between the invariant that the configuration satisfies at the beginning of the phase (start), once a robot has started to move (during), and once the moving robot has terminated to apply the same move (end), respectively. In the most cases, we have $\mathcal{X}_d = \mathcal{X}_s$; in the remaining cases, as it will be clarified in the correctness section, when $\mathcal{X}_d \neq \mathcal{X}_s$, $\mathcal{X}_d$ and $\mathcal{X}_e$ always coincide. For this reason we omit $\mathcal{X}_d$ in the presented tables.

About moves, Table III–VI contains a description of all moves applied by our algorithm for each phase. As described in Section IV, sometimes the trajectories defined by the proposed moves are opportunely manipulated so as to guarantee stationarity (by means of Procedure STATIONARYMOVE) and to avoid collisions (by means of Procedure COLLISIONFREEMOVE).

It is important to keep in mind that during the whole algorithm it is assumed $C(R) = C(F)$. We are now ready to consider each phase separately to see how the desired behavior is obtained.

A. Phase $\mathcal{F}1$

As described in Secction IV, this phase is responsible for setting the external guards g' and g'' and, if required, for placing robots on $c(F)$. Due to its complexity, the algorithm for this phase is composed of

phase	*start*	*end*
$\mathcal{F}1$	$\mathtt{a} \wedge \neg\mathtt{f}_2 \wedge \neg\mathtt{f}_3 \wedge \neg\mathtt{f}_4 \wedge \neg\mathtt{w}$	$\mathcal{F}2_s \vee \mathcal{F}3_s \vee \mathcal{F}4_s \vee \mathtt{w}$
$\mathcal{F}2$	$\mathtt{a} \wedge \mathtt{f}_2 \wedge \neg\mathtt{f}_3 \wedge \neg\mathtt{f}_4 \wedge \neg\mathtt{w}$	$\mathcal{F}3_s \vee \mathcal{F}4_s \vee \mathtt{w}$
$\mathcal{F}3$	$\mathtt{a} \wedge \mathtt{f}_3 \wedge \neg\mathtt{f}_4 \wedge \neg\mathtt{w}$	$\mathcal{F}4_s$
$\mathcal{F}4$	$\mathtt{a} \wedge \mathtt{f}_4 \wedge \neg\mathtt{w}$	$\mathtt{w}$

Table II: Each label on the first column specifies a different phase. In column 'start', for each phase it is specified the invariant that holds while a configuration belongs to the corresponding phase. In column 'end', it is specified the possible phases outside the considered one that can be reached or whether $\mathtt{w}$ may hold.

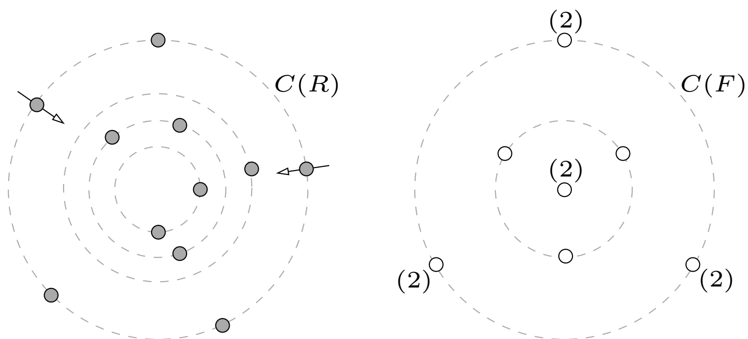

Figure 4: An example of input for the PF problem: robots R (left) and the pattern F (right). The number closed to a point denotes a multiplicity. Notice that R belongs to the sub-phase $\mathcal{A}1$ (i.e., predicates $\mathcal{F}1$ and $\mathcal{A}1_s = \mathtt{s}_+ \wedge \mathtt{l}$ hold), hence the algorithm applies move m_1 (cf Table III).

many different moves, and each move is referred to a sub-phase. Table III describes all such sub-phases, and also the corresponding invariants and moves.

The first sub-phase $\mathcal{A}$ (logically divided into $\mathcal{A}1$ and $\mathcal{A}2$) leaves exactly three robots on $C(R)$ when more than three robots are there. This is realized by selecting any robots but three that are non-*critical* for maintaining $C(R)$. We remind that a robot is critical for $C(R)$ if its removal makes $C(R)$ changing. The selected robots are moved one by one inside $C(R)$ so as to maintain asymmetry. This is realized by moving each of such robots toward a specific new circle $C^1_{\downarrow}(R)$ (hence concentric to $C(R)$ and inside it).

As an example, the configuration R in Fig. 4 belongs to the sub-phase $\mathcal{A}1$ (i.e., predicate $\mathcal{A}1_s = \mathtt{s}_+ \wedge \mathtt{l}$ holds), and hence move m_1 (cf Table III) is applied. As soon as a robot r leaves $\partial C(R)$, the obtained configuration switches to $\mathcal{A}2$ and hence move m_2 is applied until r reaches the desired circle. In fact, if r ends its movement before reaching its target, it is selected again by the algorithm and again move m_2 is applied. Once r reaches its target, predicate $\mathcal{A}1_s = \mathtt{s}_+ \wedge \mathtt{l}$ holds. Such moves are repeated so that the configuration with $|\partial C(R)| = 3$ in Fig. 6 left is obtained: this configuration belongs to $\mathcal{C}1$ since $\mathcal{C}1_s = \mathtt{s}_3 \wedge \mathtt{t}_0 \wedge \mathtt{l}$ holds.

Actually, both m_1 and m_2 are performed by invoking Procedure StationaryMove. This is done because while moving, a robot r may incur along the way in a point p that makes the current configuration belonging to the finalization phase $\mathcal{F}4$ or even final, that is predicate $\mathtt{w}$ holds. Such situations may happen when according to some embedding of F on R, difficult to detect at this stage, p coincides with a point in F. If other robots perform their Look phase while r is on p, they may start moving according to a different rule specified by our strategy, hence violating the desired property to maintain stationarity among phases. In order to avoid

phase	*start*	*end*
$\mathcal{A}1$	$\mathtt{s}_+ \wedge \mathtt{l}$	$\mathcal{A}_s \vee \mathcal{C}_s \vee \mathcal{D}_s \vee \mathcal{E}1_s$
	m_1	$\mathcal{F}4_s \vee \mathtt{w}$
$\mathcal{A}2$	$(\mathtt{s}_+ \vee \mathtt{s}_3) \wedge \neg\mathtt{l}$	$\mathcal{A}_s \vee \mathcal{C}_s \vee \mathcal{D}_s \vee \mathcal{E}1_s$
	m_2	$\mathcal{F}4_s \vee \mathtt{w}$
$\mathcal{B}$	$\mathtt{s}_2 \wedge \mathtt{m}_0 \wedge \neg\mathtt{m}_1$	$\mathcal{C}_s \vee \mathcal{D}_s$
	m_3	
$\mathcal{C}1$	$\mathtt{s}_3 \wedge \mathtt{t}_0 \wedge \mathtt{l}$	$\mathcal{C}2_s \vee \mathcal{D}_s \vee \mathcal{E}1_s$
	m_4	$\mathcal{F}4_s \vee \mathtt{w}$
$\mathcal{C}2$	$\mathtt{s}_3 \wedge \neg\mathtt{t}_0 \wedge \neg\mathtt{t}_1 \wedge \mathtt{l} \wedge \mathtt{m}_0 \wedge \neg\mathtt{m}_1$	$\mathcal{D}_s$
	m_5	
$\mathcal{D}$	$\mathtt{s}_3 \wedge \mathtt{t}_1 \wedge \mathtt{l} \wedge \mathtt{m}_0 \wedge \neg\mathtt{m}_1$	$\mathcal{E}1_s$
	m_6	$\mathcal{F}3_s$
$\mathcal{E}1$	$\mathtt{s}_3 \wedge \neg\mathtt{t}_0 \wedge \mathtt{l} \wedge (\mathtt{m}_0 \Rightarrow \mathtt{m}_1)$	$\mathcal{E}2$
	m_1	$\mathcal{F}2_s \vee \mathcal{F}4_s \vee \mathtt{w}$
$\mathcal{E}2$	$\mathtt{s}_2 \wedge \neg\mathtt{l} \wedge (\mathtt{m}_0 \Rightarrow \mathtt{m}_1)$	
	m_2	$\mathcal{F}2_s \vee \mathcal{F}4_s \vee \mathtt{w}$

name	*description*
m_1	Let r be the non-critical robot on $C(R)$ with minimum view. r moves according to StationaryMove toward $t = [r, c(R)] \cap \partial C^{0,1}(R)$.
m_2	Let r be the robot on $C^1_\downarrow(R)$. r moves according to StationaryMove toward $t = [r, c(R)] \cap \partial C^{0,2}(R)$
m_3	Let r_1 and r_2 be the two robots on $C(R)$ and $r \in C^1_\downarrow(R)$ of minimum view. r moves linearly increasing it distance from $c(R)$ toward a point t on $C(R)$ such that the angle β between $[r, t]$ and $[r_1, r_2]$ satisfies $0° < \beta < 90°$
m_4	The three robots on $C(R)$ form a triangle with angles $\alpha_1 \geq \alpha_2 \geq \alpha_3$ and let r_1, r_2 and r_3 be the three corresponding robots. For equal angles, the role of the robot is selected according to the view, i. e. if $\alpha_1 = \alpha_2$ then the view of r_1 is smaller than that of r_2. r_2 rotates according to StationaryMove toward the closest point t such that α_1 equals $90°$.
m_5	Let r be the robot on $C(R)$ such that its antipodal point is not in R. r rotates toward the closest point t such that the triangle formed by t and the two antipodal robots admits angles of $30°$, $60°$ and $90°$.
m_6	The robot closest to $c(R)$ but not on $c(R)$, of minimum view, moves toward $c(R)$.

Table III: (left) Invariants and moves for all the sub-phases of $\mathcal{F}1$. Each label on the first column specifies a different sub-phase to which a configuration belongs to. Then, with respect to each sub-phase, in the upper (shaded) side it is specified the invariant that the configuration satisfies at the beginning of the phase (start), and which sub-phase within $\mathcal{F}1$ can be reached (end). In the lower side, it is specified the corresponding move performed by the algorithm, and on the last column the possible phases outside $\mathcal{F}1$ that can be reached or whether $\mathtt{w}$ may hold. (right) Description of the moves performed by the algorithm in $\mathcal{F}1$.

such a behavior, we simply force r to stop on points that potentially may cause the described situations. For m_1 and m_2, such points belong to some $C^i_\downarrow(F)$, see Lines 4-5 of StationaryMove. If after a stop, still the configuration belongs to $\mathcal{F}1$, then the same robot r will be selected again to keep on moving.

The sub-phase $\mathcal{B}$ is concerned with the case of just two robots in $\partial C(R)$ and F admits a multiplicity on $c(F)$. In such a case, a third robot is moved from $int(C(R))$ to $C(R)$ (see move m_3 of Table III).

Sub-class $\mathcal{C}$ (logically divided into $\mathcal{C}1$ and $\mathcal{C}2$) processes configurations with exactly three robots in $\partial C(R)$, and moves them so that they form a triangle with angles of $30°$, $60°$ and $90°$. Now, assume that the three robots form a triangle with angles $\alpha_1 \geq \alpha_2 \geq \alpha_3$ and let r_1, r_2 and r_3 be the three corresponding robots. $\mathcal{C}1$ takes care of the case in which all the angles are different from $90°$ (see Fig. 6, left side): by move m_4, robot r_2 rotates on $C(R)$ in such a way that α_1 becomes of $90°$ (see Fig. 6, right side).

Similarly to m_1 and m_2, move m_4 is performed by invoking Procedure StationaryMove. The critical points on which the moving robot r must stop are now a bit different than before, as m_4 rotates r along $C(R)$ rather than moving straightly. This may cause different situations: still incurring in points in F (Lines 7-8), or creating an unexpected reference angle of α degrees (Lines 9-10).

If r_2 completes move m_4 by making $\alpha_1 = 90°$ like in Fig. 6, then there are two antipodal robots on $C(R)$ that will be detected as g' and g'' in the subsequent phases. The third robot r on $C(R)$ is now moved either along $C(R)$ or toward a position inside $C(R)$, depending on whether F admits a multiplicity in $c(F)$ of $k > 1$ elements or not. In the former case (see Fig. 6, right side), the algorithm is in sub-phase $\mathcal{C}2$ and by move m_5 it rotates r on $C(R)$ along the shortest path in such a way the composed triangle admits the required angles of $30°$, $60°$ and $90°$ (see Fig. 7, left side). In the latter case r is moved inside $C(R)$ by means of sub-phase $\mathcal{E}$.

Procedure: STATIONARYMOVE
Input: target t.

Let m be the current move,
$tmp = t$;
if $m \in \{m_1, m_2, m_7\}$ **then**
  if *$\exists p \in (r,t)$ being the closest point to r intersecting a circle $C^j_\downarrow(F)$* **then**
    $tmp = p$
if $m = m_4$ **then**
  if *the number of distinct points in $\partial C(F)$ is 3 and $\exists p \in (r, tmp)$ s. t. $(\partial C(R) \setminus \{r\}) \cup \{p\}$ form a triangle similar to that formed by $\partial C(F)$* **then**
    $tmp = p$
  if *$\exists r' \in \partial C^g(R) \wedge \exists p \in (r, tmp)$ s. t. $\sphericalangle(p, c, r') = \alpha$* **then**
    $tmp = p$
move toward tmp;

Figure 5: Procedure STATIONARYMOVE performed by any robot r when moves m_1, m_2, m_4 or m_7 are executed.

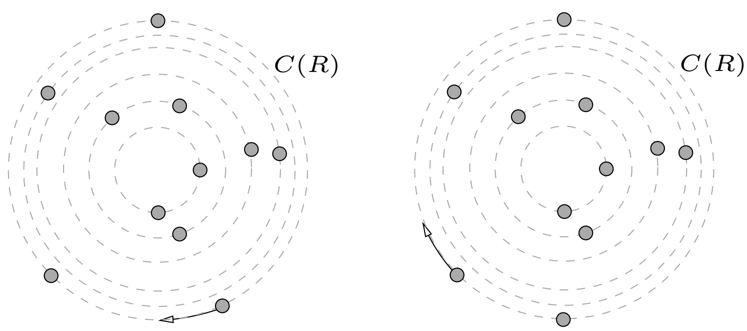

Figure 6: Configurations obtained from R as described in Fig. 4 after sub-phases $\mathcal{A}$ (left) and $\mathcal{C}1$ (right), respectively. The configuration on the right side is in phase $\mathcal{C}2$.

Once the robots in $\partial C(R)$ have formed the required triangle (case in which F admits a multiplicity of k elements in $c(F)$), sub-phase $\mathcal{D}$ can move the $k-1$ robots closest to $c(R)$ toward it (cf move m_6 of Table III and Fig. 7). The specific triangle formed by the robots in $\partial C(R)$ assure that during sub-phase $\mathcal{D}$ the configuration remains always asymmetric.

The last sub-phase of $\mathcal{F}1$ (sub-phase $\mathcal{E}$) is applied after having created a multiplicity of $k-1$ robots in $c(R)$ if F requires k elements in $c(F)$, and when exactly three robots are in $\partial C(R)$. Among such robots, there are two antipodal ones g' and g'' plus a third one r. Robot r is moved inside $C(R)$. Sub-phase $\mathcal{E}$ is logically divided into $\mathcal{E}1$ and $\mathcal{E}2$; this is because the same moves of $\mathcal{A}1$ and $\mathcal{A}2$ are used to perform the required task (see cf Fig. 7 right side with Fig. 8 left side).

B. Phase $\mathcal{F}2$

This phase is responsible for setting the internal guard g. In particular, g is initially identified as the closest robot to $c(R)$ (excluding possible robots in $c(R)$ required to compose a multiplicity), and hence moved on the guard circle $C^g(R)$ (cf Def. 5). Such a move is performed either by sub-phase $\mathcal{G}$ (logically divided into $\mathcal{G}1$ and $\mathcal{G}2$) or by sub-phase $\mathcal{H}$. Table IV describes all such sub-phases, and also the corresponding

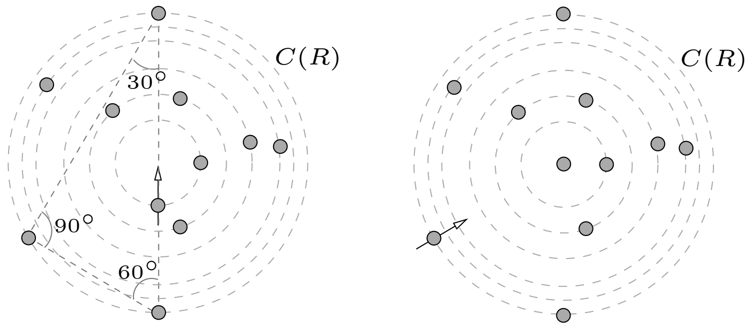

Figure 7: Configurations obtained from R as described in (the right side of) Fig. 6 after sub-phases $\mathcal{C}2$ (left) and $\mathcal{D}$ (right), respectively.

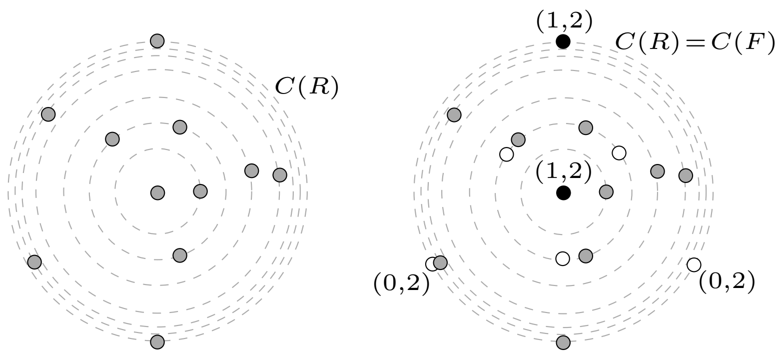

Figure 8: Configurations obtained from R as described in (the right side of) Fig. 7 after sub-phases $\mathcal{E}$ (left); such a configuration will be processed by phase $\mathcal{F}2$ (in particular, by the sub-phase $\mathcal{G}1$). On the right, a possible embedding of F onto R: black points denote robots on targets (points of F), a pair of numbers denotes multiplicities of robots and targets, respectively. The real embedding will be fixed only when the internal guard g will be correctly placed (cf Fig. 10, left side).

invariants and moves.

Sub-phase $\mathcal{G}$ starts when the invariant $\mathcal{G}1_s = \neg \mathtt{g}_0 \wedge (\mathtt{c} \Rightarrow \mathtt{m}_1)$ holds, that is when $\mathtt{g}_0$ is false (i.e., the guard g is not yet on $C^g(R)$) and $\mathtt{c} \Rightarrow \mathtt{m}_1$ is true (i.e., if there is one robot on the center $\mathtt{c}(R) = c(F)$ then this is due to the presence of a multiplicity on $c(F)$). An example of such a case is described in Fig. 8 (right

phase	*start*	*end*
$\mathcal{G}1$	$\neg \mathtt{g}_0 \wedge (\mathtt{c} \Rightarrow \mathtt{m}_1)$	$\mathcal{G}2_s$
	m_7	$\mathcal{F}3_s \vee \mathcal{F}4_s \vee \mathtt{w}$
$\mathcal{G}2$	$\mathtt{g}_0 \wedge \neg \mathtt{g}_1 \wedge (\mathtt{c} \Rightarrow \mathtt{m}_1)$	
	m_8	$\mathcal{F}3_s \vee \mathcal{F}4_s$
$\mathcal{H}$	$\mathtt{c} \wedge \neg \mathtt{m}_0$	$\mathcal{G}2_s$
	m_9	

name	*description*
m_7	Let r be the robot on $C^1_{\uparrow}(R)$ of minimum view. r moves according to StationaryMove toward $t = (r, c(R)] \cap C^g(R)$
m_8	Let r be the robot on $C^g(R)$. r rotates along $C^g(R)$ toward the closest point t such that $\sphericalangle(t, c, g') = \alpha$
m_9	Let g' be the robot on $C(R)$ of minimum view. The robot on $c(R)$ moves toward a point t on $C^g(R)$ such that $\sphericalangle(t, c, g') = \alpha/2$.

Table IV: Invariants and moves for all the sub-phases of $\mathcal{F}2$.

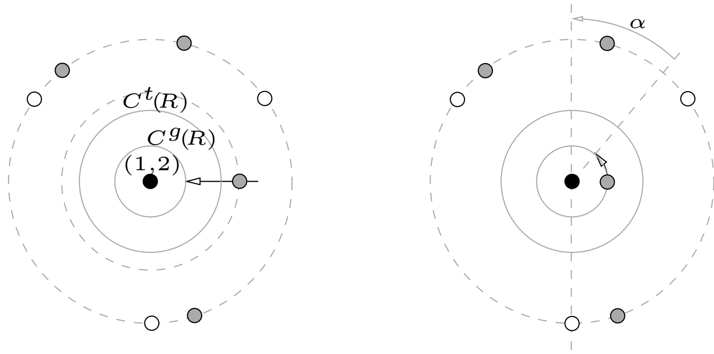

Figure 9: Zooming on the circles $C^i_\uparrow(R)$, $0 \leq i \leq 2$, of the configuration shown in Fig. 8 (right side). On the left side, the guard circle $C^g(R)$ and the teleporter circle $C^t(R)$ are shown. On the right side, the configuration obtained at the end of sub-phase $\mathcal{G}1$ after move m_7 lead a robot on the guard circle. The obtained configuration will be processed by the sub-phase $\mathcal{G}2$ to place the internal guard at the reference angle α.

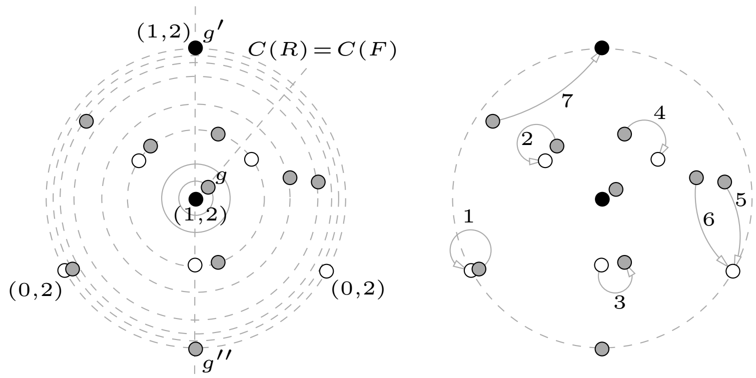

Figure 10: On the left side, the whole configuration of Fig. 9 (right side). On the right side, the mapping between non-guard robots and targets produced during phase $\mathcal{F}3$ (the numbers show the order in which the mapping is produced according to distances).

side). Sub-phase $\mathcal{G}1$ then repeatedly applies move m_7 (cf Table IV) to move the closest robot to $c(R)$ on $C^g(R)$ (cf Fig. 9). Note that, similarly to moves m_1 and m_2, move m_7 is performed by invoking Procedure STATIONARYMOVE.

When $\mathcal{G}1$ is terminated, the sub-phase $\mathcal{G}2$ starts with the aim of rotating the robot g placed on $C^g(R)$ so that g, $c(R)$, and one of the antipodal robots on $C(R)$, now detected as g', form an angle equal to the reference angle α (cf Def. 6). Move m_8 performs this task. At the end of $\mathcal{G}2$, the three guards g, g' and g'' compose the required common reference system for all robots.

Sub-phase $\mathcal{H}$ handles the specific configurations in which the invariant $\mathcal{H}_s = \mathtt{c} \wedge \neg\mathtt{m}_0$ holds, that is the cases where there is one robot on $c(R) = c(F)$ and there is no multiplicity on $c(F)$. In such a cases, m_9 moves the robot away from the center $c(R)$, so that the obtained configuration is subsequently managed by $\mathcal{G}$.

C. Phase $\mathcal{F}3$

This phase is responsible for moving all the non-guards robots (i.e., $n - 3$ robots) toward the targets. It is composed of three sub-phases, as described in Table V.

We remark that at the end of $\mathcal{F}2$, the required common reference system for all robots has been established (based on the three guards g, g' and g''). This implies that all robots can now embed F on $C(R)$. This embedding is obtained as follows:

- as already observed, $C(F)$ is superimposed on $C(R)$;
- the counter-clockwise direction for R is assumed to be the one such that g becomes collinear with g' and $c(R)$ by rotating of α degrees;
- the counter-clockwise direction for F is that defined in Section III for any multiset of points;
- let f' be a point in $\partial C(F)$ such that f' is the first point appearing in $V(f)$, being f any point in *min_view*$(\partial C(F))$; f' is superimposed on g', and any other point in F is superimposed such that the clockwise direction of F coincides with that of R.

This embedding is shown in Fig. 10 (left side). According to this embedding, each robot uses the following mapping $\mu : \{g', g'', g\} \to F$ for determining the final target of each guard:

- $\mu(g') = f'$;
- $\mu(g'') = f''$, where f'' is the point in $\partial C(F)$ closest to g'' (in case of tie, that reachable from g'' in the counter-clockwise direction);
- $\mu(g) = f$, where $f \in F \setminus \{f', f''\}$ and if $c(F) \in F$ then $f = c(F)$, else f is the point in $\partial C^1_\uparrow(F)$ closest to g (in case of tie, the first of such points reached by $hline(c(R), g)$ when this half-line is turned counter-clockwise around the center).

This embedding is maintained along all phase $\mathcal{F}3$; we remark that g, g' and g'' are not moved during this phase. Any other robot, one by one, is moved toward its closest point of $F \setminus \{\mu(g), \mu(g'), \mu(g'')\}$ (see Fig. 10, right side). At any time, each robot must determine (1) whether it is already on its target or not (i.e., whether it is *matched* or not), (2) if it is not matched, which is its target, and (3) whether it is its turn to move or not. To this aim, each robot computes the following data:

- the sets of matched robots and matched targets, that is $R^m = R \cap (F \setminus \{\mu(g), \mu(g'), \mu(g'')\})$ and $F^m = F \cap R^m$;
- the sets of unmatched robots and unmatched targets, that is $R^{\neg m} = R \setminus (R^m \cup \{g, g', g''\})$ and $F^{\neg m} = F \setminus (F^m \cup \{\mu(g), \mu(g'), \mu(g'')\})$;
- the minimum distance between unmatched robots and unmatched targets, that is $\eta = \min\{d(r, f) : \; r \in R^{\neg m}, \; f \in F^{\neg m}\}$;
- the set of unmatched robots at minimum distance from unmatched targets, that is $R^{\neg m}_\eta = \{r \in R^{\neg m} : \; d(r, F^{\neg m}) = \eta\}$;
- in a given turn, which is the robot r that has to move toward its target, that is $r =$ *min_view*$(R^{\neg m}_\eta)$, and which is the corresponding target $\mu(r)$, that is any point in $\{f \in F^{\neg m} : \; d(r, f) = \eta\}$.

According to the strategy described in Section IV, the robot that moves toward its target has two constraints: (1) avoiding undesired collisions (in case the trajectory meets an already matched robot), and (2) avoiding entering into the guard circle (to preserve the common reference system).

For the former constraint, a Procedure COLLISIONFREEMOVE is designed. The procedure is given in Fig. 12 while its description can be found in the corresponding correctness proof provided in Lemma 7.

For the latter, in case the segment $[r, \mu(r)]$ meets the teleporter circle $C^t(R)$, then an alternative trajectory is computed. For this computation, robots also need the following data: the set $F^* = \{f \in F^{\neg m} \; : \; d(f, c(F))$ is minimum$\}$. The new trajectory is divided into three parts (see Fig. 11):

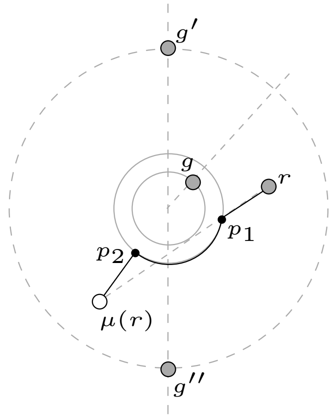

Figure 11: Visualization of a robot trajectory through the teleporter circle. The robot traces the path represented by the black polygonal curve consisting of two segments and one arc.

phase	*start*	*end*
$\mathcal{M}$	$\mathtt{d}_0 \wedge \neg\mathtt{d}_1$	$\mathcal{O}_s$
	m_{10}	
$\mathcal{N}$	$(\mathtt{d}_0 \Rightarrow \mathtt{d}_1) \wedge \mathtt{d}_2$	$\mathcal{M}_s \vee \mathcal{O}_s$
	m_{11}	
$\mathcal{O}$	$(\mathtt{d}_0 \Rightarrow \mathtt{d}_1) \wedge \neg\mathtt{d}_2$	$\mathcal{N}_s \vee \mathcal{O}_s$
	m_{12}	$\mathcal{F}4_s$

name	*description*
m_{10}	The unique robot r on $C^t(R)$ rotates toward the closest point $p_2 = [c(F), f^*] \cap C^t(R)$, where $f^* \in F^*$.
m_{11}	Robot r = *min_view*$(R_\eta^{\neg m})$ moves according to CollisionFreeMove toward the closest point p_1 on the teleporter circle, that is $p_1 = (r, \mu(r)] \cap C^t(R)$;.
m_{12}	Robot r = *min_view*$(R_\eta^{\neg m})$ moves according to CollisionFreeMove toward $\mu(r)$.

Table V: Invariants and moves for all the sub-phases of $\mathcal{F}3$.

(a) a collision free trajectory toward the closest point p_1 on the teleporter circle, that is $p_1 = (r, \mu(r)] \cap C^t(R)$;
(b) a rotation along $C^t(R)$ toward the closest point $p_2 = [c(F), f^*] \cap C^t(R)$, where $f^* \in F^*$;
(c) a collision free trajectory toward f^*.

Note that in case (c), the destination f^* may differ from the original destination $\mu(r)$ computed in case (a), but this is not a problem since, as shown in the correctness section, no other robot is moved until r reaches its final destination.

The algorithm (see Table V) performs such a new trajectory by moving robots along the teleporter circle as follows. Sub-phase $\mathcal{M}$ concerns case (b) above, that is configurations fulfilling invariant $\mathcal{M}_s = \mathtt{d}_0 \wedge \neg\mathtt{d}_1$ (i.e., configurations where there is a unique robot r on the $C^t(R)$ but r does not coincide with a point $p = [c(F), f^*] \cap C^t(R)$, where $f^* \in F^*$). Then, move m_{10} rotates r toward such a point p. Sub-phase $\mathcal{N}$ concerns case (a) above. Via move m_{11} (that invokes CollisionFreeMove), $\mathcal{N}$ is in charge of leading r on a point on $C^t(R)$ if needed (cf $\mathtt{d}_2$ in $\mathcal{N}_s = (\mathtt{d}_0 \Rightarrow \mathtt{d}_1) \wedge \mathtt{d}_2$). Finally, sub-phase $\mathcal{O}$ is concerned with case (c) above, that is when the trajectory from r to its target $\mu(r)$ does not meet the teleporter circle. Move m_{12} (that invokes CollisionFreeMove) performs this final task.

Lemma 7. *Procedure* CollisionFreeMove *performed by a robot r with input a target f always moves r avoiding collisions with other robots either toward f, if there are no robots between r and f, or toward a point p fulfilling the following conditions:*

1) *p is inside both $C(R)$ and the cell D_f of the Voronoi diagram induced by $F^{\neg m}$ where f lies;*

Procedure: COLLISIONFREEMOVE
Input: A target f.

1 **if** *there are no robots between* r *and* f **then**
2   move toward f
3 **else**
4     $\bar{r} = \mathrm{argmin}_x\{d(r,x) : \ x \in R \cap [r,f]\}$;
5   let ℓ be one of the half-lines starting from $\bar{r}$, perpendicular to $[r,f]$, and on the half-plane that does not include $c(R)$ if any;
6     $P = \{p' = \ell \cap hline(r,x) : \ p' \neq \bar{r}$ and $x \in R \setminus \{r\}\}$;
7     $p' = \ell \cap C(R)$;
8   let p'' be the intersection between ℓ and the circle centered in f of radius $[f,r]$;
9   let p''' be the intersection, if it exists, between ℓ and a side of the cell of the Voronoi diagram induced by $F^{\neg m}$ where f lies;
10     $\bar{p} = \mathrm{argmin}_x\{d(\bar{r},x) : \ x \in P \cup \{p', p'', p'''\}\}$;
11   let p be the median point in $[\bar{r},\bar{p}]$;
12   move toward p ;

Figure 12: Procedure COLLISIONFREEMOVE performed by any robot r when moves m_{11} or m_{12} must be executed.

2) *there is no robot between* p *and* f*;*
3) $d(f,p) < d(f,r)$.

Moreover, all the points x *reached by* r *during its movement share the same properties of* p*.*

Proof: At Line 2, Procedure COLLISIONFREEMOVE moves r toward f when there are no robots between r and f. As the movement is straightforward and since both D_f and the disk enclosed by $C(R)$ are convex, all the points x reached by r during its movement are inside $C(R)$ and D_f.

If there are robots between r and f, among such robots the procedure, at Line 4, identifies as $\bar{r}$ the closest to r. The point p is calculated on one of the two half-lines perpendicular to $[r,f]$ in $\bar{r}$, in accordance to the position of $c(R)$, see Line 5. On ℓ, a set $P = \{p' = \ell \cap hline(r,x) : \ p' \neq \bar{r}$ and $x \in R\}$ is calculated at Line 6. The target p is different by any point in P: being these points on the lines between r and any another robot, this will assure that the movement will be free by further collisions. To set the exact position of p on ℓ, three other points are calculated at Lines 7, 8, and 9. The first one is p', that is the intersection of ℓ and $C(R)$. The target p is such that $d(\bar{r},p) < d(\bar{r},p')$, then all the points x reached by r during its movement are inside $C(R)$. The second one is p'', that is the intersection between ℓ and the circle centered in f of radius $[f,r]$. The target p is such that $d(\bar{r},p) < d(\bar{r},p'')$, then all the points x reached by r during its movement are closer and closer to f. The third one, if exists, is p''', that is the intersection of ℓ and a side of cell D_f. The target p is such that $d(\bar{r},p) < d(\bar{r},p''')$, then all the points x reached by r during its movement are inside this cell. This assures that the points x reached by r are closer to f than any other target. Moreover, there is no robot between x and f. In fact, if such robot r' exists, the line $hline(r,x)$ would intersect ℓ in a point closest to $\bar{r}$ than each point in P, a contradiction as the target p is the closer one.

Finally, at Line 11, the point p is set at a position fulfilling all the above constraints. In addition, notice that the choice of half-line ℓ ensures that p cannot be on or inside $C^t(R)$; this implies that the procedure can be safely applied. In turn, all such properties prove that the claim holds. ■

phase	*start*	*end*
$\mathcal{P}1$	$\neg\mathtt{q} \wedge \mathtt{i}_1$	$\mathcal{P}2_s$
	m_{13}	
$\mathcal{P}2$	$\neg\mathtt{q} \wedge \mathtt{i}_2$	
	m_{14}	$\mathtt{w}$
$\mathcal{Q}1$	$\mathtt{q} \wedge \mathtt{i}_3$	$\mathcal{Q}2_s \vee \mathcal{Q}3_s$
	m_{15}	
$\mathcal{Q}2$	$\mathtt{q} \wedge \mathtt{i}_4$	$\mathcal{Q}3_s$
	m_{16}	
$\mathcal{Q}3$	$\mathtt{q} \wedge \mathtt{i}_5$	$\mathcal{Q}4_s$
	m_{14}	$\mathtt{w}$
$\mathcal{Q}4$	$\mathtt{q} \wedge \mathtt{i}_6$	
	m_{13}	$\mathtt{w}$

name	*description*
m_{13}	g'' rotates toward $\mu(g'')$.
m_{14}	g moves toward $\mu(g)$.
m_{15}	g moves toward $C(R) \cap hline(c(R), g)$.
m_{16}	g'' rotates along $C(R)$ toward the closest point among $\mu(g'')$ and the antipodal point to g.

Table VI: Invariants and moves for all the sub-phases of $\mathcal{F}4$.

D. Phase $\mathcal{F}4$

This phase concerns the finalization steps, where the last three robots (the guards) must be moved to their targets to complete the formation of pattern F. In particular, since in this phase we use the embedding defined in phase $\mathcal{F}3$, g' is already matched and hence at most g and g'' remain to be moved. Moving the guards leads to the loss of the common reference system, and hence ad-hoc moves must be designed to complete the pattern.

The algorithm for this phase is composed of two sub-phases denoted as $\mathcal{P}$ and $\mathcal{Q}$ (see Table VI). The former handles the majority of configurations by moving first g'' and then g toward the respective targets. The latter manages some special cases where g'' is critical for $C(R)$. In particular, predicate $\mathtt{q}$ (informally introduced in Table I) is used to characterize such special cases. Formally:

Definition 8. *Let $F = \{f_1, f_2, \ldots, f_n\}$ be a pattern to be formed. We say that the predicate* $\mathtt{q}$ *holds if F fulfills the following conditions:*

1) *$\partial C(F) = F$;*
2) *F does not contain multiplicities;*
3) *assuming $f_1 = min\_view(F)$ and $(f_1, f_2, \ldots, f_n)$ as the counter-clockwise sequence of points on $C(F)$, then $\sphericalangle(f_n, c(F), f_2) > 180°$, where such an angle is obtained by rotating $hline(c(F), f_n)$ counter-clockwise.*

Fig. 13.(a) shows an embedding of a pattern F in which $\mathtt{q}$ holds. Sub-phase $\mathcal{P}$ is divided into $\mathcal{P}1$ and $\mathcal{P}2$ to manage the moves of g'' and g, respectively. We now describe each sub-phase: its invariant, the task it performs, and the corresponding move.

In $\mathcal{P}1$, g'' rotates along $C(R)$ toward $\mu(g'')$. Each robot can recognize this phase by performing, in order, the following steps:

1) test whether $\mathtt{g}_1$ holds;
2) if the previous test is passed, it uses the same embedding defined in phase $\mathcal{F}3$: this embedding allows to recognize the guard g', and, in turn, to determine g'' as the robot on $C(R)$ closest to the antipodal point of g';

3) finally, it tests whether such an embedding makes $R \setminus \{g, g''\}$ similar to $F \setminus \{\mu(g), \mu(g'')\}$ and $g'' \neq \mu(g'')$, where also the targets $\mu(g)$ and $\mu(g'')$ are those defined in phase $\mathcal{F}3$.

The result of test at Item 3 above can be seen as the value of an invariant $\mathtt{i}_1$ (cf phase $\mathcal{P}1$ at Table VI). When a robot checks that $\mathtt{i}_1$ holds and recognizes itself as g'', it simply applies move m_{13} to complete the rotation along $C(R)$ to reach its target $\mu(g'')$. For instance, referring to Fig. 10 right side, once all non-guard robots are correctly placed during $\mathcal{F}3$, the configuration belongs to $\mathcal{P}1$, and g'' rotates in the clockwise direction along $C(R)$ to compose the multiplicity on the left. This is in fact the closest point on $C(R)$ to g'' (in the clockwise direction as there is a tie to break) with respect to the defined embedding of F. Once also g'' is correctly positioned on its target, as we are going to see, $\mathtt{i}_2$ holds and only g remains to move toward $\mu(g)$ to finalize F. In the specific example of Fig. 10, $\mu(g) = c(R)$.

The movement of g is realized in $\mathcal{P}2$ via a straight move of g toward $\mu(g)$. Each robot can recognize this phase by performing, in order, the following steps:

1) compute the set $E = \{(r, f) : \ (\{r\} = c(R) \vee \{r\} = \partial C^1_\uparrow(R)) \wedge R \setminus \{r\}$ is similar to $F \setminus \{f\}\}$;
2) test whether there exists a pair $(r, f) \in E$ such that $f = c(F) \vee (c(F) \notin F) \wedge f \in C^1_\uparrow(F) \wedge d(c(R), r) < d(c(F), f)$;
3) if there are many pairs (r, f) that fulfill the previous test, it selects one for which $d(r, f)$ is minimum;
4) the pair (r, f) selected in the previous step allows robots to recognizes the internal guard g (i.e., g coincides with r) and the target of g (i.e., the point f).

The result of test at Item 2 above can be seen as the value of an invariant $\mathtt{i}_2$. When a robot checks that $\mathtt{i}_2$ holds and recognizes itself as g, it simply applies move m_{14} to complete the movement toward the remaining unmatched target f.

To ensure a correct finalization even when $\mathtt{q}$ holds and hence when g'' might be critical for $C(R)$, the sub-phase $\mathcal{Q}$ is divided into four sub-phases. They are responsible for:

- $\mathcal{Q}1$: moving g radially toward $C(R)$;
- $\mathcal{Q}2$: rotating g'' of at most α along $C(R)$;
- $\mathcal{Q}3$: rotating g from the position acquired at $\mathcal{Q}1$ along $C(R)$ toward its target;
- $\mathcal{Q}4$: rotate again g'' if necessary along $C(R)$ toward its target.

The movement of g toward its target performed in two steps guarantees to avoid possible symmetries. We now describe each sub-phase: its invariant, the task it performs, and the corresponding move. To recognize each phase among $\mathcal{Q}1, \ldots, \mathcal{Q}4$, robots perform the following test:

1) test whether there exists an embedding of F such that $r_2, r_3, \ldots, r_{n-1}$ are matched with $f_2, f_3, \ldots, f_{n-1}$, respectively. We recall that here predicate $\mathtt{q}$ holds, hence points in F fulfill all the conditions in Definition 8 (see Fig. 13.(a)).

Since R is asymmetric, the above test always determines a unique ordering for the robots. After, robots perform two additional tests, according to the specific sub-phases to be recognized. Concerning $\mathcal{Q}1$, the following additional tests are needed:

2) test whether both $\partial C^1_\uparrow(R) = \{r_1\}$ and $\sphericalangle(r_1, c(R), f_2) = \alpha$ hold;
3) test whether r_n is antipodal to r_2 (which coincides with f_2).

If all the previous tests are passed, robots recognize g as r_1, g' as r_2, and g'' as the antipodal robot to g'. The result of such a process can be seen as the value of an invariant $\mathtt{i}_3$. When a robot recognizes itself as g and checks that $\mathtt{q} \wedge \mathtt{i}_3$ holds, it simply applies move m_{15} to move radially toward $C(R)$ (cf cases (a) and (b) of Fig. 13).

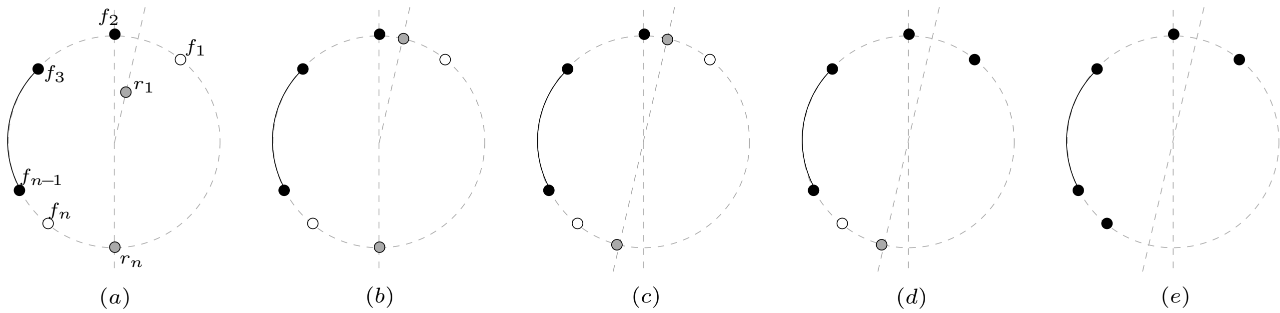

Figure 13: Visualization of some configurations belonging to $\mathcal{Q}$ (points $f_4, \ldots, f_{n-1}$, if any, all lie on the black arc). In (a), a configuration where $\mathcal{Q}1_s$ holds, in (b) $\mathcal{Q}2_s$ holds, in (c) $\mathcal{Q}3_s$ holds, in (d) $\mathcal{Q}4_s$ holds, and finally in (e) $\mathtt{w}$ holds.

Concerning $\mathcal{Q}2$, the following additional tests are needed:

2) test whether r_1 is on $C(R)$ between f_1 and f_2 such that $\sphericalangle(r_1, c(R), f_2) = \alpha$;
3) test whether r_n is on $C(R)$ such that $r_n \neq f_n$ and $\sphericalangle(r_n, c(R), p) < \alpha$, where p is the antipodal point to f_2.

If all the previous tests are passed, robots recognize g as r_1, g' as r_2, and g'' as the closest robot to p. The result of such a process can be seen as the value of an invariant $\mathtt{i}_4$. When a robot checks that $\mathtt{q} \wedge \mathtt{i}_4$ holds and recognizes itself as g'', it applies move m_{16} to rotate of at most an angle α from p or to reach its target f_n if residing before p. This is done to maintain $C(R)$, being g'' critical (cf cases (b) and (c) of Fig. 13).

Now, notice that according to the definition of view of F given in Section III-A and according to Definition 8, $\sphericalangle(f_1, c(F), f_2) = 3\alpha$. Hence, for sub-phase $\mathcal{Q}3$ the following additional tests are needed:

2) test whether r_1 is on $C(R)$ between f_1 and f_2 such that $\alpha \leq \sphericalangle(r_1, c(R), f_2) < 3\alpha$;
3) test whether r_n is on $C(R)$ such that $r_n = f_n$ or $\sphericalangle(r_n, c(R), p) = \alpha$, where p is the antipodal point to f_2.

If all the previous tests are passed, robots recognize g as r_1, g' as r_2, and g'' as the closest robot to p. The result of such a process can be seen as the value of an invariant $\mathtt{i}_5$. When a robot checks that $\mathtt{q} \wedge \mathtt{i}_5$ holds and recognizes itself as g, it applies move m_{14} to complete the rotation along $C(R)$ to reach its target f_1 (cf cases (c) and (d) of Fig. 13).

Finally, for $\mathcal{Q}4$ the following additional tests are needed:

2) test whether r_1 is matched with f_1, according to the embedding defined at test 1;
3) test whether r_n is on $C(R)$ such that $r_n \neq f_n$ and $\alpha \leq \sphericalangle(r_n, c(R), p) < \sphericalangle(f_n, c(R), p)$, where p is the antipodal point to f_2.

If all the previous tests are passed, robots recognize g as r_1, g' as r_2, and g'' as the closest robot to p. The result of such a process can be seen as the value of an invariant $\mathtt{i}_6$. When a robot checks that $\mathtt{q} \wedge \mathtt{i}_6$ holds and recognizes itself as g'', it applies move m_{13} to reach f_n (cf cases (d) and (e) of Fig. 13).

In the next section we show the correctness of the algorithm.

VI. Correctness

In this section, we provide all the results necessary to assess the correctness of our algorithm. To this end, we have to show that for each non-final configuration (that is configurations not satisfying $\mathtt{w}$) exactly one of the start predicates (predicates for stationary configurations) in Table II is true. We will show that a stationary configuration satisfying a starting predicate in Table II will be transformed by robots' moves into a stationary configuration of a subsequent phase or in a stationary configuration satisfying $\mathtt{w}$. To this end, for each phase (that is for each configuration that satisfies a starting predicate of Table II), we have to show that exactly one of the starting predicates in the corresponding table (one among Table III, IV, V, and VI) is true. This property along with the defined moves assure that, given a stationary configuration, exactly one robot moves. For each applied move, we have to show that during its implementation asymmetry is maintained, no undesired multiplicity is created, and all robots but the one involved by the move remain stationary. This assures that at the end of the move the configuration is necessarily stationary. Finally, we will show that during a move the starting predicate of Table II indicating the phase remains unchanged, with the exception of few remarked situations which do not affect the correctness of the algorithm.

Lemma 9. *Given an asymmetric configuration R and a pattern F, if $\mathtt{w}$ does not hold then exactly one of the predicates defining a starting phase of Table II is true.*

Proof: First, we show that each phase manages a different set of configurations, that is the logical conjunction of any two predicates among those defining the four starting phases in Table II is false. Then, we show that the logical disjunction of all the predicates defining the four phases of Table II is equivalent to predicate $\mathtt{a} \wedge \neg\mathtt{w}$, that is our algorithm deals with any asymmetric configuration not similar to the pattern F.

The conjunction of $\mathcal{F}1_s$ with $\mathcal{F}2_s$, $\mathcal{F}3_s$, and $\mathcal{F}4_s$ is false because of variables $\mathtt{f}_2$, $\mathtt{f}_3$, and $\mathtt{f}_4$, respectively. Similarly, the conjunction of $\mathcal{F}2_s$ with $\mathcal{F}3_s$ and $\mathcal{F}4_s$ is false because of variables $\mathtt{f}_3$ and $\mathtt{f}_4$, respectively. Finally, the conjunction of $\mathcal{F}3_s$ and $\mathcal{F}4_s$ is false because of $\mathtt{f}_4$.

For the second part of the proof, let us consider $\mathcal{F}1_s \vee \mathcal{F}2_s \vee \mathcal{F}3_s \vee \mathcal{F}4_s$. This expression is equivalent to $\mathtt{a} \wedge \neg\mathtt{w} \wedge [(\neg\mathtt{f}_2 \wedge \neg\mathtt{f}_3 \wedge \neg\mathtt{f}_4) \vee (\mathtt{f}_2 \wedge \neg\mathtt{f}_3 \wedge \neg\mathtt{f}_4) \vee (\mathtt{f}_3 \wedge \neg\mathtt{f}_4) \vee (\mathtt{f}_4)]$. Since the sub-expression in square brackets is true, the whole expression is equivalent to $\mathtt{a} \wedge \neg\mathtt{w}$. ■

Lemma 10. *Given a configuration R and a pattern F, if $\mathcal{F}1_s$ is true then exactly one of the predicates for the starting phases in Table III is true.*

Proof: We first show that at least one predicate among $\mathcal{A}1_s$, $\mathcal{A}2_s$, $\mathcal{B}_s$, $\mathcal{C}1_s$, $\mathcal{C}2_s$, $\mathcal{D}_s$, $\mathcal{E}1_s$, and $\mathcal{E}2_s$ is true. By simple algebraic transformations, we obtain $\mathcal{A}1_s \vee \mathcal{A}2_s \vee \mathcal{B}_s \vee \mathcal{C}1_s \vee \mathcal{C}2_s \vee \mathcal{D}_s \vee \mathcal{E}1_s \vee \mathcal{E}2_s = \mathtt{s}_+ \vee \mathtt{s}_3 \vee (\mathtt{s}_2 \wedge \neg((\mathtt{m}_0 \Rightarrow \mathtt{m}_1) \wedge \mathtt{l}))$. Note that $\mathtt{s}_+ \vee \mathtt{s}_3 \vee \mathtt{s}_2$ is true for each configuration, since that expression is referred to all the possibilities about the number of robots on $C(R)$ as we assumed $|R| \geq 3$. So it is sufficient to show that $\mathcal{F}1_s \Rightarrow \neg((\mathtt{m}_0 \Rightarrow \mathtt{m}_1) \wedge \mathtt{l})$ when $\mathtt{s}_2$ is true. As $\mathcal{F}1_s$ implies $\neg\mathtt{f}_2 = \neg((\mathtt{m}_0 \Rightarrow \mathtt{m}_1) \wedge \mathtt{s}_2 \wedge \mathtt{l})$ (see Table II), when $\mathtt{s}_2$ holds we trivially get that $\neg(\mathtt{l} \wedge (\mathtt{m}_0 \Rightarrow \mathtt{m}_1))$ holds.

We now show that at most one of the predicates for starting phases in Table III is true. To this end, it is sufficient to show that the logical conjunction of any two predicates is false. In most cases, this is obtained by showing that both the predicates imply the same variable, but with opposite logical values.

- Concerning $\mathcal{A}1_s = \mathtt{s}_+ \wedge \mathtt{l}$, it is disjoint with $\mathcal{A}2_s$ because of $\mathtt{l}$. Since $\mathtt{s}_+$ implies $\neg(\mathtt{s}_2 \vee \mathtt{s}_3)$, then $\mathcal{A}1_s$ is disjoint with any of the remaining predicates, as for them either $\mathtt{s}_2$ or $\mathtt{s}_3$ is true.
- Concerning $\mathcal{A}2_s = (\mathtt{s}_+ \vee \mathtt{s}_3) \wedge \neg\mathtt{l}$, either it implies $\mathtt{s}_+$ (and then, as above, it differs from all the remaining predicates) or it implies $\mathtt{s}_3 \wedge \neg\mathtt{l}$. However, $\mathtt{l}$ is positive in all the remaining predicates where $\mathtt{s}_3$ holds; predicates $\mathcal{B}_s$ and $\mathcal{E}2_s$ are both disjoint with $\mathcal{A}2_s$ because of $\mathtt{s}_2$.
- Predicate $\mathcal{B}_s$ is disjoint with all the remaining predicate but $\mathcal{E}2_s$ because of $\mathtt{s}_2$ (the others require $\mathtt{s}_3$). $\mathcal{B}_s$ and $\mathcal{E}2_s$ are disjoint because of $\mathtt{m}_0 \Rightarrow \mathtt{m}_1$.
- Concerning $\mathcal{C}1_s = \mathtt{s}_3 \wedge \mathtt{t}_0 \wedge \mathtt{l}$, it is disjoint with both $\mathcal{C}2_s$ and $\mathcal{E}1_s$ because of $\mathtt{t}_0$, with $\mathcal{D}_s$ because of $\mathtt{t}_1$ (since $\mathtt{t}_0$ implies $\neg\mathtt{t}_1$), and with $\mathcal{E}2_s$ because $\mathtt{s}_3$ implies $\neg\mathtt{s}_2$.

- Predicate $\mathcal{C}2_s$ is disjoint with $\mathcal{D}_s$ because of $\mathtt{t}_1$, with $\mathcal{E}1_s$ because of $\mathtt{m}_0 \Rightarrow \mathtt{m}_1$, and with $\mathcal{E}2_s$ because $\mathtt{s}_3$ implies $\neg\mathtt{s}_2$.
- Predicate $\mathcal{D}_s$ is disjoint with $\mathcal{E}1_s$ because of $\mathtt{m}_0 \Rightarrow \mathtt{m}_1$, and with $\mathcal{E}2_s$ because $\mathtt{s}_3$ implies $\neg\mathtt{s}_2$.
- Predicate $\mathcal{E}1_s$ is disjoint with $\mathcal{E}2_s$ because $\mathtt{s}_3$ implies $\neg\mathtt{s}_2$.

Summarizing, we get that exactly one of the predicates for the starting phases in Table III is true when $\mathcal{F}1_s$ holds. ■

Lemma 11. *Given a configuration R and a pattern F, if $\mathcal{F}2_s$ is true then exactly one of the predicates for the starting phases in Table IV is true.*

Proof: We first show that at most one of the predicates for starting phases in Table IV is true. To this end, it is sufficient to show that the logical conjunction of any two predicates is false. This is obtained by showing that both the predicates imply the same variable, but with opposite logical values. In particular, $\mathcal{G}1_s$ and $\mathcal{G}2_s$ are disjoint because of $\mathtt{g}_0$. Since $\mathtt{m}_1 \Rightarrow \mathtt{m}_0$, we can assume that both $\mathcal{G}1_s$ and $\mathcal{G}2_s$ imply $\mathtt{c} \Rightarrow \mathtt{m}_0 = \neg\mathtt{c} \vee \mathtt{m}_0 = \neg(\mathtt{c} \wedge \neg\mathtt{m}_0)$. Then, both $\mathcal{G}1_s$ and $\mathcal{G}2_s$ are disjoint with $\mathcal{H}_s$ because of $(\mathtt{c} \wedge \neg\mathtt{m}_0)$.

We now show that exactly one of the predicates for starting phases in Table IV is true. To this end, we show that $\mathcal{G}1_s \vee \mathcal{G}2_s \vee \mathcal{H}_s$ is true when $\mathcal{F}2_s$ holds. We first analyze $\mathcal{G}1_s \vee \mathcal{G}2_s$; it corresponds to $[\neg\mathtt{g}_0 \wedge (\mathtt{c} \Rightarrow \mathtt{m}_1)] \vee [\mathtt{g}_0 \wedge \neg\mathtt{g}_1 \wedge (\mathtt{c} \Rightarrow \mathtt{m}_1)] = [\neg\mathtt{g}_0 \vee (\mathtt{g}_0 \wedge \neg\mathtt{g}_1)] \wedge (\mathtt{c} \Rightarrow \mathtt{m}_1) = [\neg\mathtt{g}_0 \vee \neg\mathtt{g}_1] \wedge (\mathtt{c} \Rightarrow \mathtt{m}_1)$. Since $\neg\mathtt{g}_0 \Rightarrow \neg\mathtt{g}_1$, the last expression can be simplified into

$$\mathcal{G}1_s \vee \mathcal{G}2_s = \neg\mathtt{g}_1 \wedge (\mathtt{c} \Rightarrow \mathtt{m}_1). \tag{1}$$

Now, observe that $\mathcal{F}2_s$ implies both $\mathtt{f}_2 = (\mathtt{m}_0 \Rightarrow \mathtt{m}_1) \wedge \mathtt{s}_2 \wedge \mathtt{l}$ and $\neg\mathtt{f}_3 = \neg[(\mathtt{m}_0 \Rightarrow \mathtt{m}_1) \wedge \mathtt{g}_2] = \neg(\mathtt{m}_0 \Rightarrow \mathtt{m}_1) \vee \neg\mathtt{g}_2$. In turn, it follows that $\mathcal{F}2_s$ implies $(\mathtt{m}_0 \Rightarrow \mathtt{m}_1)$, $\mathtt{s}_2$, and $\neg\mathtt{g}_2$. According to the definition of $\mathtt{g}_2$, it follows that either $\mathtt{g}_1$ is false or $\nexists g'' \in \partial C(R)$ antipodal to g'. The latter condition cannot hold since $\mathcal{F}2_s$ implies $\mathtt{s}_2$, and hence $\mathtt{g}_1$ is false. Concluding, Eq. 1 is equivalent to $\mathcal{G}1_s \vee \mathcal{G}2_s = \mathtt{c} \Rightarrow \mathtt{m}_1$.

Finally, we get $\mathcal{G}1_s \vee \mathcal{G}2_s \vee \mathcal{H}_s = (\mathtt{c} \Rightarrow \mathtt{m}_1) \vee (\mathtt{c} \wedge \neg\mathtt{m}_0) = \neg\mathtt{c} \vee (\mathtt{m}_0 \Rightarrow \mathtt{m}_1)$. Since in $\mathcal{F}2_s$ predicate $\mathtt{m}_0 \Rightarrow \mathtt{m}_1$ holds, then the claim follows.

Summarizing, we get that exactly one of the predicates for the starting phases in Table IV is true when $\mathcal{F}2_s$ holds. ■

Lemma 12. *Given a configuration R and a pattern F, if $\mathcal{F}3_s$ is true then exactly one of the predicates for the starting phases in Table V is true.*

Proof: By observing that $\mathtt{d}_0 \wedge \neg\mathtt{d}_1$ is equivalent to $\neg(\mathtt{d}_0 \Rightarrow \mathtt{d}_1)$, it is easy to see that the logical conjunction of any two predicates among $\mathcal{M}_s$, $\mathcal{N}_s$, and $\mathcal{O}_s$ is false. Then at most one of these predicates is true for R. On the other hand the logical disjunction of predicates $\mathcal{M}_s$, $\mathcal{N}_s$, and $\mathcal{O}_s$ is also trivially true. Then, exactly one of the predicates for the starting phases in Table V is true. ■

Lemma 13. *Given a configuration R and a pattern F, if $\mathcal{F}4_s$ is true then exactly one of the predicates for the starting phases in Table VI is true.*

Proof: We first show that at least one predicate among $\mathcal{P}1_s$, $\mathcal{P}2_s$, $\mathcal{Q}1_s$, $\mathcal{Q}2_s$, $\mathcal{Q}3_s$, and $\mathcal{Q}4_s$ is true. Since $\mathcal{F}4_s = \mathtt{a} \wedge \mathtt{f}_4 \wedge \neg\mathtt{w}$ holds, then $\mathtt{f}_4$ holds as well. Since $\mathtt{f}_4 = \neg\mathtt{q} \wedge (\mathtt{i}_1 \vee \mathtt{i}_2) \ \vee \ \mathtt{q} \wedge (\mathtt{i}_3 \vee \mathtt{i}_4 \vee \mathtt{i}_5 \vee \mathtt{i}_6)$, then the logical disjunction $\mathcal{P}1_s \vee \mathcal{P}2_s \vee \mathcal{Q}1_s \vee \mathcal{Q}2_s \vee \mathcal{Q}3_s \vee \mathcal{Q}4_s$ is true.

We now show that R is processed by exactly one sub-phase of $\mathcal{F}4$. $\mathcal{P}1_s$ is disjoint with $\mathcal{P}2_s$ since $\mathtt{i}_1$ implies that exactly two robots are unmatched, while $\mathtt{i}_2$ implies that exactly one robot is unmatched. $\mathcal{P}1_s$ is disjoint with (any sub-phase of) $\mathcal{Q}$ because of $\mathtt{q}$. Similarly for $\mathcal{P}2_s$. According to the formal definitions of predicates $\mathtt{i}_3, \ldots, \mathtt{i}_6$, it follows that $\mathcal{Q}1_s$ is disjoint with any other sub-phase of $\mathcal{Q}$, since $\mathtt{i}_3$ implies r_1 inside $C(R)$ while $\mathtt{i}_4, \ldots, \mathtt{i}_6$ all imply $R = \partial C(R)$. $\mathcal{Q}4_s$ is disjoint with both $\mathcal{Q}2_s$ and $\mathcal{Q}3_s$ since $\mathtt{i}_6$ implies that exactly one robot is unmatched, while both $\mathtt{i}_4$ and $\mathtt{i}_5$ imply that exactly two robots are unmatched. Finally, $\mathcal{Q}2_s$ and $\mathcal{Q}3_s$ are disjoint because of the third items in the definitions of $\mathtt{i}_4$ and $\mathtt{i}_5$. ■

We are now ready to provide the correctness proof of our algorithm for each phase, and then we combine all phases by means of the final theorem that provides the correctness of the whole algorithm. For each phase we consider all possible sub-phases. For each sub-phase we show all the possible scenarios where the corresponding moves lead. In particular, for each move m defined in the algorithm, we need to show several properties that guarantee to our algorithm to safely evolve until pattern F is formed:

- H_0: at the beginning, m involves only one robot;
- H_1: while a robot is moving according to m, the configuration is asymmetric;
- H_2: m is safe, and in particular that while a robot is moving according to m, all other robots are stationary;
- H_3: while a robot is moving according to m, no collisions are created;
- H_4: if m is associated to any phase $\mathcal{X}$, then the predicate $\mathcal{X}_e$ holds once a robot has terminated to apply m;
- H_5: m preserves stationarity.

About Property H_4, the stop of a robot r is due to three events. First, the adversary may stop r before reaching its target. Second, the move might be subject to Procedures STATIONARYMOVE or COLLISIONFREEMOVE, hence r reaches an intermediate target. Third, r reaches the real target imposed by the current move. From the proofs, we omit the analysis of the first condition because the situation obtained once the adversary stops the moving robot r always equals what can happen while r is moving, that is the analysis of property H_2 holds.

About Property H_5, we always omit this property form our proofs because it comes for free from the other properties once we have shown that there is always only robot r moving. So whenever r stops moving, the configuration is stationary.

Lemma 14. *Let R be a stationary configuration in $\mathcal{F}1$. From R the algorithm eventually leads to a stationary configuration belonging to $\mathcal{F}2$, $\mathcal{F}3$, $\mathcal{F}4$ or where $\mathtt{w}$ holds.*

Proof: By Lemma 10, exactly one of the predicates for the starting phases in Table III is true. In turn, this implies that exactly one of the moves associated to the sub-phasess of $\mathcal{F}1$ is applied to R. We show that the properties $H_0, \ldots, H_4$ holds for each possible move applied to R.

Let us consider sub-phase $\mathcal{A}1$ where move m_1 is performed.

- H_0: Move m_1 only concerns the non-critical robot r on $C(R)$ of minimum view.
- H_1: During the movement of r, the configuration remains asymmetric as r cannot participate to neither a rotation, being the only robot on $C^1_\downarrow(R)$, or a reflection as the axis of symmetry should pass throw r, but then the starting configuration R was symmetric, a contradiction.
- H_2: We show that m_1 is safe. As soon as the robot moves, predicate $\mathcal{A}2 = (\mathtt{s}_+ \vee \mathtt{s}_3) \wedge \neg \mathtt{l}$ holds, since from $\mathtt{s}_+$ by moving one robot either $\mathtt{s}_+$ or $\mathtt{s}_3$ holds and $\neg \mathtt{l}$ holds as the robot has not yet reached the target.
  The configurations observed during the move of r cannot belong to $\mathcal{F}2$ as $\mathtt{s}_2$ does not hold. It cannot belong to $\mathcal{F}3$ as well because $\mathtt{f}_3$ does not hold. In fact, $\mathtt{f}_3$ does not hold in $\mathcal{A}1_s$ and m_1 cannot change this status. Possibly, the configuration falls in $\mathcal{F}4$, in particular sub-phases $\mathcal{P}2$ and $\mathcal{Q}1$. In fact, in $\mathcal{P}1$ predicate $\mathtt{g}_1$ should hold, but r is certainly not on $C^g(R)$; in $\mathcal{Q}2$, $\mathcal{Q}3$, and $\mathcal{Q}4$ all robots should belong to $\partial C(R)$, but r does not. If $\mathcal{P}2$ holds, then only r can be

the remaining unmatched robot that moves in $\mathcal{P}2$ since r is guaranteed to not meet a point in F according to the use of Procedure StationaryMove. It follows that during the movement, in case r is stopped by the adversary, it will be selected again by the algorithm as the unique robot that performs move m_{14}. Similar arguments can be applied if $\mathcal{Q}1$ holds, where there is only one robot inside $C(R)$.
The above arguments also ensure that no other robot than r can move from the reached configurations.

H_3: Move m_1 guarantees there are no robots between r and its target.

H_4: Assume that r stops moving because it reaches an intermediate target dictated by Procedure StationaryMove. In this case, predicates $\mathtt{w}$, $\mathcal{P}1_s$, or $\mathcal{P}2_s$ might hold because $\mathtt{i}_1$ or $\mathtt{i}_2$ become true (clearly, $\mathtt{i}_3, \ldots, \mathtt{i}_6$ cannot become true as F should equal ∂F). If both $\mathtt{i}_1$ and $\mathtt{i}_2$ are still false, r is unmatched and the same considerations given for H_2 hold, that is the configuration belongs to phases $\mathcal{A}2$ or $\mathcal{P}2$ or $\mathcal{Q}1$.
Assume that r reaches the target $t = [r, c(R)] \cap \partial C^{0,1}(R)$. If the configuration remains in $\mathcal{F}1$, then $\mathtt{l}$ holds and the configuration is either in $\mathcal{A}1$ if $\mathtt{s}_+$ holds, or in $\mathcal{A}2$, $\mathcal{C}$, $\mathcal{D}$, or $\mathcal{E}1$ if $\mathtt{s}_3$ holds. If the configuration is not in $\mathcal{F}1$, then by the above analysis it belongs to $\mathcal{F}4$ or $\mathtt{w}$ holds.

Sub-phase $\mathcal{A}2$, where move m_2 is performed, is the continuation of sub-phase $\mathcal{A}1$ in case the moving robot r stops before reaching its target on $C^{0,1}(R)$ to make predicate $\mathtt{l}$ newly true. Then the same analysis of move m_1 applies. Hence moves m_1 and m_2 are repeatedly applied in order to remove non-critical robots from $\partial C(R)$ until there remain exactly three robots on $C(R)$ (unless the configuration reaches phase $\mathcal{F}4$ or satisfies $\mathtt{w}$ before). This can be done according to Properties 2 and 3. Once the resulting configuration satisfies $\mathtt{s}_3 \wedge \mathtt{l}$ it does not belong to sub-phase $\mathcal{A}$ anymore. This situation occurs in a finite number of steps. In fact, as shown above, move m_1 along with move m_2 bring non-critical robots one by one to their designed targets. Since by assumption a robot is guaranteed to traverse at least distance ν each time it moves, then a finite number of steps suffices to reach the desired configuration.

Three robots on $\mathcal{C}(R)$ are necessary to maintain the asymmetry in case a multiplicity should be formed in $c(R)$, that is when predicate $\mathtt{m}_0 \wedge \neg\mathtt{m}_1$ is true. If there are only two robots in $\partial C(R)$, a third one from $int(C(R))$ is moved on $C(R)$. This is done in sub-phase $\mathcal{B}$ by means of move m_3.

H_0: The move only concerns robot r on $C^1_\downarrow(R)$ with minimum view.

H_1: During the movement of r the configuration remains asymmetric as r cannot participate to neither a rotation, being the only robot on $C^1_\downarrow(R)$, or a reflection as the axis of symmetry should pass throw r, but then the starting configuration R was symmetric too, a contradiction.

H_2: We show that m_3 is safe. Assuming that the configuration observed during the move of r still belongs to $\mathcal{F}1$, then r is always detected as the unique robot on $C^1_\downarrow(R)$ and hence $\mathcal{B}_s$ still holds. The configuration observed while r moves cannot belong neither to $\mathcal{F}2$ nor to $\mathcal{F}3$ as $\mathtt{m}_0 \Rightarrow \mathtt{m}_1$ does not hold (we have $\mathtt{m}_0 \wedge \neg\mathtt{m}_1$ by hypothesis). Similarly, it cannot belong to $\mathcal{F}4_s$ as a multiplicity in $c(R)$ must be created. The above arguments ensure that during the movement all robots but r are stationary as the configuration remains in $\mathcal{B}$.

H_3: No collision is possible as, by definition, there are no robots between $C^1_\downarrow(R)$ and $C(R)$, and the target t on $C(R)$ cannot coincide with one of the positions of the two robots on $C(R)$.

H_4: Once r reaches its target, $\mathtt{s}_3$ holds and the configuration remains in $\mathcal{F}1$. In particular, the configuration is in $\mathcal{C}_1$, $\mathcal{C}_2$ or $\mathcal{D}$, depending on the kind of triangle formed by the robots on $C(R)$, that is the status of predicates $\mathtt{t}_0$ and $\mathtt{t}_1$.

When $|\partial C(R)| = 3$, we have to guarantee that two robots on $C(R)$ are antipodal before removing the third one, otherwise $C(R)$ could change its radius. This is done in sub-phase $\mathcal{C}1$ by means of move m_4. The move involving one of the three robots on $C(R)$ makes the triangle they form containing a 90° angle, and hence two antipodal robots.

H_0: As specified by the definition of m_4, the only robot r involved in this move is that on $C(R)$ corresponding to angle α_2 if the triangle has angles $\alpha_1 \geq \alpha_2 \geq \alpha_3$ (in case of ties, the uniqueness of r is guaranteed by using the view of robots).

H_1: The configuration remains asymmetric, because as soon as r starts moving we get $\alpha_1 > \alpha_2 > \alpha_3$, that is the triangle formed by the three robots on $C(R)$ is asymmetric. Moreover the triangle remains asymmetric as during the movement and until the end, α_1 increases, α_2 maintains its value and α_3 decreases.

H_2: We show that m_4 is safe. Assuming that the configuration observed during the move of r still belongs to $\mathcal{F}1$, as $\mathtt{s}_3 \wedge \mathtt{t}_0$ remains true during the movement, then the configuration remains in $\mathcal{C}1$. While r moves the observed configuration cannot belong to $\mathcal{F}2$ as $\mathtt{s}_2$ does not hold. The same for $\mathcal{F}3$, as there are no two antipodal robots and then $\mathtt{g}_2$ is false. Predicates $\mathcal{F}4_s$ cannot hold as the movement is controlled by StationaryMove which preventively calculates the possible points that could make $\mathcal{F}4_s$ true at the end of the movement. So, during the movement, $\mathcal{F}4_s$ remains false. As the above defined angles are such that $\alpha_1 > \alpha_2 > \alpha_3$ during the move, then the robot r on α_2 is always uniquely determined;

H_3: No collisions are created as the robot moves on $C(R)$ having as target a point antipodal to one of the other two robots. The third robot cannot be on its trajectory as, otherwise, the smallest enclosing circle would be different from $C(R)$.

H_4: If r stops on a target specified by Procedure StationaryMove, the configuration can remain in $\mathcal{C}1$. As in case H_2, the configuration cannot belong to $\mathcal{F}2$ as $\mathtt{s}_2$ does not hold. The same for $\mathcal{F}3$, as there are no two antipodal robots and then $\mathtt{g}_2$ is false. If Procedure StationaryMove computes a target specified at line 8, it could be possible that there exists an embedding satisfying $\mathtt{i}_1$. Then the configuration can belong to sub-phase $\mathcal{P}1$ of $\mathcal{F}4$. Moreover, predicate $\mathtt{w}$ might hold. If the computed target comes from line 9 of Procedure StationaryMove, an angle of α degrees is formed with a robot on $C^g(R)$. In that case the configuration may belong to either sub-phase $\mathcal{P}1$ or sub-phase $\mathcal{P}2$ of $\mathcal{F}4$.
If the robot stops reaching the target of move m_4, forming an angle of 90°, t_0 is false and then the configuration can satisfy $\mathcal{C}2_s \vee \mathcal{D}_s \vee \mathcal{E}1_s$. If the configuration is not in $\mathcal{F}1$, as above, it can satisfy either $\mathtt{w}$ or $\mathcal{F}4_s$.

At the end of sub-phase $\mathcal{C}1$ or when $R = R(0)$, the three robots on $C(R)$ could form a symmetric triangle, even though the whole configuration is asymmetric. In case a multiplicity in $c(R)$ must be formed, in order to maintain asymmetry, the algorithm makes the triangle asymmetric. To guarantee the stationarity of the configuration, we impose that the triangle has angles equal to 30°, 60°, and 90° degrees. This is done in sub-phase $\mathcal{C}2$ by means of move m_5.

H_0: Among the three robots on $C(R)$, only the one that does not admit an antipodal robot is moved (notice that the status of $\mathtt{t}_0$ in $\mathcal{C}2_s$ implies two antipodal robots on $C(R)$).

H_1: The configuration is always asymmetric because the triangle formed by the three robots can be symmetric only at the beginning.

H_2: We show that m_5 is safe. As predicate $\mathtt{m}_0 \wedge \neg\mathtt{m}_1$ holds during the whole movement, that is a multiplicity in $c(R)$ must be formed, then the configuration observed during the movement remains in sub-phase $\mathcal{C}2$ until $\mathtt{t}_1$ becomes true. For the same reason, the configuration cannot belong to $\mathcal{F}2$, $\mathcal{F}3$, and $\mathcal{F}4$. During the movement all other robots are stationary, for the same reason as in H_0.

H_3: No collisions are created as the target of the moving robot is always between the antipodal robots on $C(R)$.

H_4: As discussed in H_2, the configuration remains in $\mathcal{F}1$ and in particular in $\mathcal{C}2$ or $\mathcal{D}$ depending on $\mathtt{t}_1$.

Once the three robots on $C(R)$ form a triangle having angles equal to 30°, 60°, and 90° degrees, the multiplicity in $c(R)$ can be safely formed with respect to symmetries. This is done in sub-phase $\mathcal{D}$ by means of move m_6.

- H_0: The only moving robot r is the closest to $c(R)$, not on $c(R)$, and of minimum view.
- H_1: The asymmetry of the configuration is guarantee by predicate $\mathtt{t}_1$ which remains true during the movement. In fact, none of the three robots on $C(R)$ is moved because at least two points in F are on $C(F)$, so the multiplicity to be formed is at most of $|F|-2$ elements. However the algorithm moves at most $|F|-3$ robots on $c(R)$ (see predicate $\mathtt{m}_1$).
- H_2: We show that m_6 is safe. Predicate $\mathtt{t}_1 \wedge \mathtt{m}_0 \wedge \neg\mathtt{m}_1$ holds during the whole movement, then the configuration remains in $\mathcal{D}$ until $\mathtt{m}_1$ becomes true. As long as $\mathtt{m}_1$ is false, the configuration cannot be in $\mathcal{F}2$, $\mathcal{F}3$, and $\mathcal{F}4$. During the movement all other robots are stationary, for the same reason as in H_0.
- H_3: No collisions are created as the moving robot is the closest to the target.
- H_4: Once the robot reaches $c(R)$, either another robot must be moved on $c(R)$ because $\mathtt{m}_1$ is still false and then the configuration is still in $\mathcal{D}$, or $\mathtt{m}_1$ holds. In the latter case the configuration can be in $\mathcal{E}1$. It cannot be in $\mathcal{F}2$, as $\mathtt{s}_3$ holds, and $\mathtt{w}$ cannot be satisfied as at least one robot must be still moved on $c(R)$. The configuration can be in any sub-phase of $\mathcal{F}3$ as, by chance, $\mathtt{g}_2$ might hold. Moreover the configuration can belong to $\mathcal{F}4$, sub-phase $\mathcal{P}$, but not $\mathcal{Q}$ as predicate $\mathtt{q}$ is false.

As soon as phase $\mathcal{D}$ terminates and the configuration is in $\mathcal{E}1$, move m_1 is applied to remove one robot from $C(R)$ leaving only two antipodal robots.

- H_0: The only the non-critical robot r on $C(R)$ is moved.
- H_1: During the movement of r the configuration remains asymmetric as r cannot participate to neither a rotation, being the only robot on $C^1_{\downarrow}(R)$, nor a reflection as the axis of symmetry should pass throw r, but then the starting configuration R was symmetric too, a contradiction.
- H_2: We prove that m_1 in $\mathcal{E}1$ is safe. Assuming that the configuration observed during the movement of r still belongs to $\mathcal{F}1$, then as soon as r starts moving, predicate $\mathcal{E}2_s = \mathtt{s}_2 \wedge \neg\mathtt{l} \wedge (\mathtt{m}_0 \Rightarrow \mathtt{m}_1)$ holds. In fact, in $\mathcal{E}1$ predicate $\mathtt{s}_3$ holds and by moving one robot, $\mathtt{s}_2$ becomes true while $\mathtt{l}$ becomes false as the robot has to reach its target.
  The configuration while r moves cannot belong to $\mathcal{F}2$ as $\mathtt{l}$ does not hold. It cannot belong to $\mathcal{F}3$ as well because $\mathtt{f}_3$ does not hold in $\mathcal{E}1_s$ and m_1 cannot change this status. Possibly, the configuration falls in $\mathcal{F}4$, in particular only in sub-phase $\mathcal{P}2$. In fact, in $\mathcal{P}1$ predicate $\mathtt{i}_1$ should hold, but since $\mathtt{s}_2$ holds, g'' should be on $\mu(g'')$. In $\mathcal{Q}$, there must be at least three robots on $C(R)$. If $\mathcal{P}2$ holds then only r can be the remaining unmatched robot that moves in $\mathcal{P}2$ since r is guaranteed to not meet a point in F according to the use of Procedure StationaryMove. It follows that during the movement and once r is stopped by the adversary, it will be selected again by the algorithm as the unique robot that performs move m_{14}. The above arguments also ensure that no other robot than r can move from the reached configurations.
- H_3: Move m_1 guarantees there are no robots between r and its target.
- H_4: If r stops because it reaches an intermediate target dictated by Procedure StationaryMove, then $\mathtt{w}$ or $\mathcal{P}2$ might hold, because $\mathtt{i}_2$ becomes true. For the same reasons as above, $\mathcal{P}1$ and $\mathcal{Q}$ cannot be reached. If r reaches the target $t = [r, c(R)] \cap \partial C^{0,1}(R)$, the obtained configuration is not in $\mathcal{F}1$ anymore, as $\mathtt{f}_2$ holds. Then, it can be in $\mathcal{F}2$ as $\mathtt{f}_2$ is now true, and as above it can be in $\mathcal{F}4$ or $\mathtt{w}$ holds.

Similarly to sub-phases $\mathcal{A}1$ and $\mathcal{A}2$, sub-phase $\mathcal{E}2$ is the continuation of sub-phase $\mathcal{E}1$ in case the moving robot r stops before reaching its target on $\partial C^{0,1}(R)$ to make predicate $\mathtt{l}$ newly true. Then the same analysis of move m_1 for $\mathcal{E}1$ applies. Once r reaches its target, predicate $\mathtt{f}_2$ holds, and the configuration can be in $\mathcal{F}2$, $\mathcal{F}4$ or $\mathtt{w}$ holds. ■

Lemma 15. *Let R be a stationary configuration in $\mathcal{F}2$. From R the algorithm eventually leads to a stationary configuration belonging to $\mathcal{F}3$, $\mathcal{F}4$ or where* $\mathtt{w}$ *holds.*

Proof: Recall that the aim of $\mathcal{F}2$ is the formation of a configuration satisfying predicate $\mathtt{g}_2$ (cf definition of predicate $\mathtt{f}_3$ in $\mathcal{F}3_s$). That is, the obtained configuration has three robots acting as guards such that two of them, g' and g'', are antipodal on $C(R)$ and the third one g is on $C^g(R)$ forming an angle of α degree with g'.

By Lemma 11, exactly one of the predicates for the starting phases in Table IV is true. In turn, this implies that exactly one of the moves associated to the sub-phasess of $\mathcal{F}2$ is applied to R. We show that the properties $H_0, \ldots, H_4$ holds for each possible move applied to R.

We analyze move m_7: it is performed in sub-phase $\mathcal{G}1$ to bring a robot r on $C^g(R)$.

- H_0: The move selects only one robot: the robot r on $C^1_\uparrow(R)$ of minimum view.
- H_1: During the movement of r, the configuration remains asymmetric as r cannot participate to neither a rotation, being the only robot on $C^1_\uparrow(R)$, or a reflection as the axis of symmetry should pass throw r, but then the starting configuration R was symmetric too, a contradiction.
- H_2: We show that m_7 is safe. During the movement of r, as $\mathtt{f}_2$ remains true, the observed configuration cannot belong to $\mathcal{F}1$. Assuming the observed configuration still belongs to $\mathcal{F}2$, predicate $\mathcal{G}1_s$ remains true. Since r has not yet reached the target, $\mathtt{g}_2$ is still false and hence the observed configuration cannot belong to $\mathcal{F}3$. Possibly, the configuration falls in $\mathcal{F}4$, in particular sub-phase $\mathcal{P}2$. In fact, in $\mathcal{P}1$ predicate $\mathtt{g}_1$ should hold, but r is certainly not on $C^g(R)$ (target of the current move). In $\mathcal{Q}$ there should be at least three robots in $\partial C(R)$, but $\mathtt{s}_2$ holds. If $\mathcal{P}2$ holds then only r can be the remaining unmatched robot that moves in $\mathcal{P}2$ since r is guaranteed to not meet a point in F according to the use of Procedure STATIONARYMOVE. It follows that during the movement and once r is stopped by the adversary, it will be selected again by the algorithm as the unique robot that performs move m_{14}.
  Summarizing, while r is moving the configuration can be in sub-phase $\mathcal{G}1$ of $\mathcal{F}2$ or in sub-phase $\mathcal{P}2$ of $\mathcal{F}4$. In both cases the r is always recognized as the only one allowed to move.
- H_3: no collisions are created as there is no robot between r and $C^g(R)$.
- H_4: Assume that r stops moving because it reaches an intermediate target dictated by Procedure STATIONARYMOVE. Then, $\mathtt{w}$ can hold or the configuration is still in $\mathcal{G}1$. In fact, by the analysis in H_2, the configuration might be in $\mathcal{P}2$, but this is excluded as the robot is on $C^i_\uparrow(F)$, for some $i > 0$, while i_2 does not hold because it requires that $d(c(R), r) < d(c(F), f)$, where f is a point on $C^1_\uparrow(F) \cap F$.
  Assume r reaches its target on $C^g(R)$. Then, $\mathtt{w}$ cannot hold because there are no points of F on $C^g(R)$ by definition. The configuration is not in $\mathcal{F}1$ as $\mathtt{f}_2$ holds. As $\mathtt{g}_0$ holds, the configuration can be in $\mathcal{G}2$ and in $\mathcal{F}3$ in case also $\mathtt{g}_1$ holds. The configuration can be in $\mathcal{F}4$, in particular in $\mathcal{P}1$ or $\mathcal{P}2$ depending on $\mathtt{i}_1$ or $\mathtt{i}_2$. It cannot be in $\mathcal{Q}$ because $\mathtt{s}_2$ holds.

Once there is a robot r on $C^g(R)$, to make $\mathtt{g}_1$ true, that is to correctly place guard g, it should be rotated on $C^g(R)$. This is done in sub-phase $\mathcal{G}2$ by move m_8.

- H_0: r is the unique robot on $C^g(R)$;
- H_1: During the movement of r, the configuration remains asymmetric as r cannot participate to neither a rotation, being the only robot on $C^g(R)$, or a reflection as the axis of symmetry should pass throw r, and throw the antipodal robots g' and g'' or between them. These cases can happen only if r is collinear with g' and g'' or if it lies on the line perpendicular to the segment $[g', g'']$. These cases are only possible at the beginning of the move, where the configuration is asymmetric, but not during the movement.

H_2: We show that m_8 is safe. During the movement of r, as $\mathtt{f}_2$ remains true, the observed configuration cannot belong to $\mathcal{F}1$. Assuming the configuration still belongs to $\mathcal{F}2$, predicate $\mathcal{G}2_s$ remains true. As $\mathtt{g}_2$ is still not true, the configuration cannot belong to $\mathcal{F}3$. The observed configuration does not fall in $\mathcal{F}4$. In particular: as $\mathtt{g}_1$ is false, it is not in $\mathcal{P}1$; it is not in $\mathcal{P}2$ otherwise $\mathtt{i}_2$ true during the movement of r implies $\mathtt{i}_2$ true in the starting configuration R too, a contradiction; it is not in $\mathcal{Q}$ as $\mathtt{s}_2$ still holds during the movement of r, while $\mathcal{Q}$ handles configurations with at least $n-1$ robots on $C(R)$. As the configuration remains in $\mathcal{G}2$ by the above analysis, robot r is always detected as the only moving one.

H_3: No collisions are created as there is only r on $C^g(R)$.

H_4: If r reaches its target on $C^g(G)$, $\mathtt{w}$ cannot hold because there are no points of F on $C^g(R)$ by definition. The configuration is not in $\mathcal{F}1$ as $\mathtt{f}_2$ holds. As $\mathtt{g}_2$ holds, the configuration can be in $\mathcal{F}3$. The configuration can be in $\mathcal{F}4$, in particular in $\mathcal{P}1$ as $\mathtt{i}_1$ might hold. It cannot be in $\mathcal{P}2$ or in $\mathcal{Q}$ by the same analysis in H_2.

In case a robot r is on $c(R)$, but there is no multiplicity in $c(F)$, then r will be moved on $C^g(R)$ in sub-phase $\mathcal{H}$ by means of move m_9.

H_0: r is clearly the only robot to move;

H_1: the configuration is always asymmetric by the same analysis provided for move m_8 in sub-phase $\mathcal{G}2$.

H_2: We show that m_9 is safe. As soon as r starts moving, the observed configuration can only belong in $\mathcal{G}$. In fact, during the movement of r, $C^g(R)$ changes, but r is recognized as the unique robot on $C^1_\uparrow(R)$. It follows that the configuration is in $\mathcal{G}2$ or $\mathcal{G}1$ depending whether r is on the current circle $C^g(R)$ or not. The configuration cannot be in $\mathcal{F}1$ as $\mathtt{f}_2$ holds. It cannot be in $\mathcal{H}$ as $\mathtt{c}$ is false, and it cannot be in $\mathcal{F}3$ as $\mathtt{g}_2$ is false. It cannot be in $\mathcal{F}4$ as $\mathtt{g}_1$ remains false which excludes $\mathcal{P}1$, $\mathtt{i}_2$ remains false which excludes $\mathcal{P}2$, and $\mathcal{Q}$ is excluded by $\mathtt{s}_2$. In any of the reachable sub-phases described, r is the only moving robot.

H_3: No collisions are created as there are no further robots between $c(R)$ and $C^g(R)$.

H_4: Once r reaches $C^g(R)$, $\mathtt{w}$ cannot hold because there are no points of F on $C^g(R)$ by definition. The obtained configuration is not in $\mathcal{F}1$ as $\mathtt{f}_2$ holds, and is not in $\mathcal{F}3$ as $\mathtt{g}_1$ is false. The obtained configuration is not in $\mathcal{F}4$ as both $\mathtt{f}_4$ and $\mathtt{q}$ remain false. It means the configuration can only be in phase $\mathcal{F}2$, in particular it cannot be in $\mathcal{G}1$ as $\mathtt{g}_0$ is true, and it cannot be in H as $\mathtt{c}$ is false. So it can only be in $\mathcal{G}2$.

■

Lemma 16. *Let R be a stationary configuration in $\mathcal{F}3$. From R the algorithm eventually leads to a stationary configuration belonging to $\mathcal{F}4$.*

Proof: By Lemma 12, exactly one of the predicates for the starting phases in Table V is true. In turn, this implies that exactly one of the moves associated to the sub-phases of $\mathcal{F}3$ is applied to R. We show that the properties $H_0, \ldots, H_4$ holds for each possible move applied to R.

Let $P^* = \{[c(F), f^*] \cap C^t(R) \mid f^* \in F^*\}$, and let d be the distance from any point in P^* to any target in F^*. Let us start the analysis of moves m_{10}, m_{11}, and m_{12} by assuming there is exactly one robot r on $C^t(R)$ and that r is not on a point of P^*, that is $\mathtt{d}_0$ holds while $\mathtt{d}_1$ is false. In this case, the configuration is in sub-phase $\mathcal{M}$ and move m_{10} is applied.

H_0: The only moved robot is that on $C^t(R)$, that by assumption is r.

H_1: The configuration is maintained asymmetric by the position of the three guards g, g', and g''. In fact, as g is the only robot on $C^g(R)$, the configuration cannot be rotational. Moreover, the only possible reflexion axis should pass throw g, but there is no robot that can be reflected to g' as, by predicate $\mathtt{g}_1$, g' is the only robot such that $\sphericalangle(g, c(R), g') = \alpha$.

H_2: We show that m_{10} is safe. As $\mathtt{f}_3$ holds during the movement of r, the observed configuration cannot be in $\mathcal{F}1$ and $\mathcal{F}2$. Moreover, during the move, both robots r and g do not stay neither on a target of F nor on $C(R)$. As each predicate among $\mathtt{i}_1, \ldots, \mathtt{i}_6$ requires that at most two robots are not on target and at least one of them is on $C(R)$, they are all false and then the observed configuration is not in $\mathcal{F}4$. Hence, as $\mathtt{d}_0 \wedge \neg\mathtt{d}_1$ holds, the observed configuration remains in $\mathcal{F}3$, sub-phase $\mathcal{M}$. Robot r remains the only robot on $C^t(R)$, then all the other robots are stationary.

H_3: Collisions are impossible as r rotates on $C^t(R)$ and it is the only robot on it.

H_4: Once r reaches the target, by the analysis done in H_2, the configuration remains in $\mathcal{F}3$, but now $\mathtt{d}_1$ holds. If still $r = \mathit{min\_view}(R_\eta^{\neg m})$, then $\mathtt{d}_2$ is false as the current target $\mu(r) \in F^*$ is reachable without intersecting $C^t(R)$. If another robot $r' = \mathit{min\_view}(R_\eta^{\neg m})$ (this can happen only the first time phase $\mathcal{F}3$ is applied, and there was already r on $C^t(R)$), then again $\mathtt{d}_2$ is false because otherwise the distance from r' to $\mu(r')$ would be greater than d, a contradiction. Hence, the obtained configuration is in sub-phase $\mathcal{O}$.

When $\mathtt{d}_0 \Rightarrow \mathtt{d}_1$ holds, the configuration is in sub-phase $\mathcal{N}$ or $\mathcal{O}$ depending on whether $\mathtt{d}_2$ is true or not. Let us assume that $\mathtt{d}_2$ is true, that is $C^t(R) \cap (r, \mu(r)] \neq \emptyset$, where $r = \mathit{min\_view}(R_\eta^{\neg m})$. Then, move m_{11} is applied and r is moved toward the closest point p_1 in $C^t(R) \cap (r, \mu(r)]$ according to Procedure COLLISIONFREEMOVE.

H_0: The only moved robot is r that is the one with minimum view in $R_\eta^{\neg m}$. It is unique as the configuration is asymmetric and hence there cannot be two robots with the same view.

H_1: As in the analysis done for move m_{10}, the configuration is maintained asymmetric by the guards.

H_2: We show that m_{11} is safe. During the movement of r, the configuration remains in $\mathcal{F}3$ by the same analysis done for move m_{10}. In particular, it can still belong to sub-phase $\mathcal{N}$ or to sub-phase $\mathcal{O}$. In fact, if there are robots between r and p_1, then r receives an intermediate target p by procedure COLLISIONFREEMOVE and hence $\mathtt{d}_2$ may remain true or not. By Lemma 7, condition 3), robot r reduces its distance to p_1 and then to $\mu(r)$, being p_1 an intermediate point between the initial position of r and $\mu(r)$. Then the distance η' of r to $\mu(r)$ is such that $\eta' < \eta$, and hence it remains the only robot in $R_{\eta'}^{\neg m}$. All the other robots remain stationary as their distance to any target is at least η.

H_3: Collisions are impossible as r is moved according to Procedure COLLISIONFREEMOVE.

H_4: Here there are three possible cases: (i) r reaches target p_1; (ii) r has reached the new target p (or it has been stopped before by the adversary) but now the trajectory toward $\mu(r)$ does not intersect $C^t(R)$ anymore; (iii) r has reached the new target p (or it has been stopped before by the adversary) and still the trajectory toward $\mu(r)$ intersects $C^t(R)$. In case (i), by the analysis provided for m_{10}, r is now the only robot on $C^t(R)$, and the configuration is in sub-phase $\mathcal{M}$. In case (ii), $\mathtt{d}_2$ is false and the configuration is in $\mathcal{O}$, and by Lemma 7, r will be selected again to move. In case (iii), the configuration is still in $\mathcal{N}$ and robot r will be selected again by the algorithm. In fact, by condition 3) of Lemma 7, $r = \mathit{min\_view}(R_{\eta'}^{\neg m})$, with $\eta' < \eta$. Moreover, by condition 1) of Lemma 7, $C(R)$ and the target $\mu(r)$ do not change. Let $p' \neq p$ be the closest point to r on $(r, \mu(r)] \cap C^t(R)$. By condition 2) of Lemma 7, there are no robots between r and p', that is there will not be a deviation by means of COLLISIONFREEMOVE when r applies again m_{11}. Note that, in case (iii), still r applies m_{11}, so predicate $\mathcal{N}_e$ does not hold, but now p' is assured to be reached within a finite number of steps because in each step r moves of at least ν.

Let us assume that $\mathtt{d}_2$ is false, that is $C^t(R) \cap (r, \mu(r)] = \emptyset$, where $r = \mathit{min\_view}(R_\eta^{\neg m})$. Then, the configuration is in $\mathcal{O}$, move m_{12} is applied and r is moved toward $\mu(r)$ according to Procedure COLLISIONFREEMOVE.

H_0: As in move m_{11}, the only moved robot is that with minimum view $r = R_\eta^{\neg m}$. Robot r is unique as the configuration is asymmetric.

H_1: As in the analysis done for move m_{10}, the configuration is maintained asymmetric by the guards.

H_2: During the movement of r, the configuration remains in $\mathcal{F}3$ by the same analysis done for move m_{10}. In particular, it can only belong to $\mathcal{O}$ as the trajectory from r to $\mu(r)$ cannot intersect $C^t(R)$, even if Procedure COLLISIONFREEMOVE assigns a new target. By Lemma 7 condition 3), robot r reduces its distance to $\mu(r)$. Then the distance η' of r to $\mu(r)$ is such that $\eta' < \eta$, and hence it remains the only robot in $R_{\eta'}^{\neg m}$. All the other robots remain stationary as their distance to any target is at least η.

H_3: Collisions are impossible as r is moved according to Procedure COLLISIONFREEMOVE.

H_4: If r reaches a new target p assigned by COLLISIONFREEMOVE (or it has been stopped before by the adversary), the configuration is still in $\mathcal{O}$ but now, by condition 2) of Lemma 7, there are no robots between r and $\mu(r)$. By condition 1) of the same lemma both $C(R)$ and $\mu(r)$ do not change, and by condition 3), robot r reduces its distance to $\mu(r)$. So r applies again m_{12}, predicate $\mathcal{O}_e$ does not hold, but now $\mu(r)$ is assured to be reached within a finite number of steps because in each step r moves of at least ν.
If r reaches $\mu(r)$ and there are still unmatched points in $F \setminus \{\mu(g), \mu(g'')\}$ then the configuration remains in $\mathcal{F}3$ as $\mathtt{a} \wedge \mathtt{f}_3 \wedge \neg\mathtt{f}_4 \wedge \neg\mathtt{w}$ holds. In particular, it belongs either to $\mathcal{N}$ or to $\mathcal{O}$ depending on $\mathtt{d}_2$. It cannot belong to $\mathcal{M}$ as a robot on $C^t(R)$ would move before r.
If r reaches $\mu(r)$ and all points in $F \setminus \{\mu(g), \mu(g'')\}$ are matched, then $\mathtt{w}$ does not hold as g is not on $\mu(g)$ and hence the configuration is in $\mathcal{F}4$. In particular it is in sub-phases $\mathcal{P}1$, $\mathcal{P}2$, or $\mathcal{Q}1$, as g is not on $C(R)$.

Clearly, the fact that from $\mathcal{O}$ the configuration can go back to $\mathcal{N}$ or to $\mathcal{O}$ can happen only a finite number of times, until all points in $F \setminus \{\mu(g), \mu(g'')\}$ become matched.

In conclusion, moves m_{10}, m_{11}, and m_{12} can be applied only a finite number of times, then eventually the configuration leaves phase $\mathcal{F}3$ and, following the above analysis, phase $\mathcal{F}4$ is reached. ■

Lemma 17. *Let R be a stationary configuration in $\mathcal{F}4$. From R the algorithm eventually leads to a stationary configuration where* $\mathtt{w}$ *holds.*

Proof: By Lemma 12, exactly one of the predicates for the starting phases in Table VI is true. In turn, this implies that exactly one of the moves associated to the sub-phasess of $\mathcal{F}4$ is applied to R. We show that the properties $H_0, \ldots, H_4$ holds for each possible move applied to R.

We recall that in this phase only guards g and g'' need to be moved to complete the pattern formation. As guards move, the embedding exploited in phase $\mathcal{F}3$ cannot be always recognized, at least not straightforwardly. Each sub-phase refers in fact to a different embedding that tries to reconstruct where the configuration comes from.

Sub-phases $\mathcal{P}1$ and $\mathcal{P}2$ manage the case where $\mathtt{q}$ is false. In sub-phase $\mathcal{P}1$, guard g'' rotates along $C(R)$ in order to reach $\mu(g'')$, performing move m_{13}. Since $\mathtt{q}$ is false, we are guaranteed that $C(R)$ does not change while g'' moves. In fact, let p be the antipodal point to $\mu(g'')$. If $\mathtt{q}$ is false because of the first condition that is $\partial C(F) \neq F$, then all points in $\partial C(F) \setminus \mu(g'')$ are occupied by robots as $\mu(g)$ is inside $C(R)$. Without loss of generality, let us assume that g'' needs to rotate in the clockwise direction to reach $\mu(g'')$. Since $C(F) = C(R)$, there must exist a point $f \in \partial C(F)$, which is occupied, that either coincides with p or it can be met in the clockwise direction by rotating on $C(R)$ from p. Robot g'' moves in between $\mu(g'')$ and f, and hence it is not critical. Similar arguments hold if $\mathtt{q}$ is false because of the second condition that is F contains multiplicities since $\mu(g)$ is either inside $C(R)$ or on a multiplicity. If $\mathtt{q}$ is false because of the third condition, then $\sphericalangle(f_2, c(R), f_n) \leq 180°$. If $\mu(g'')$ is reached by g'' in the counter-clockwise direction, then by the disposal of the points on $C(F)$ and the minimality of f_1, g'' can safely move without affecting $C(R)$. If $\mu(g'')$ is reached by g'' in the clockwise direction, then the condition on $\mathtt{q}$ assures that f_n lies in the counter-clockwise direction from g'', it is occupied, and it is closer than p to $\mu(g'')$.

- H_0: Only g'' is involved in the move. Even though g'' moves, $\mathtt{i}_1$ ensures to always recognize the same robot as g''.
- H_1: As in $\mathcal{F}3$, the configuration is maintained always asymmetric by the presence of g on $C^g(R)$.
- H_2: We show that m_{13} is safe. As $\mathtt{q}$ only depends on F, it is always false. During the movement of g'', $\mathtt{i}_1$ remains true hence, by Lemma 13, the observed configuration always belongs to $\mathcal{P}1$ and it cannot belong to any other sub-phase of $\mathcal{F}4$. Moreover, it cannot belong to any other phase because $\mathtt{f}_4$ holds. It follows that g'' is the only moving robot.
- H_3: No collisions are created as there are no robots between g'' and $\mu(g'')$ on $C(R)$.
- H_4: Once g'' reaches the target, predicate $\mathtt{i}_1$ does not hold anymore as $g'' = \mu(g'')$ but predicate $\mathtt{i}_2$ holds, that is by Lemma 13 the configuration is in $\mathcal{P}2$. The configuration cannot satisfy $\mathtt{w}$ as g is on $C^g(R)$. It is not in any other phase because $\mathtt{f}_4$ holds.

From $\mathcal{P}1$ only guard g remains to be positioned in order to form F. Now the embedding of F on R is more difficult to detect and g moves according to move m_{14} in phase $\mathcal{P}2$.

- H_0: Only g is involved in the move. Even though g moves, $\mathtt{i}_2$ ensures to always recognize the same robot as g since the distance to the target always decreases.
- H_1: The configuration is maintained always asymmetric during the movement because g is the only robot on $\mathcal{C}^1_\uparrow(R)$. So it cannot participate to a rotation. Moreover, the configuration cannot admit a reflection as the axis of symmetry should pass through g. This means that there exists another embedding of F such that the target of g would be closer than that it has currently calculated, but this is in contradiction with predicate $\mathtt{i}_2$ that chooses the one that minimizes the distance.
- H_2: We show that m_{14} is safe. As $\mathtt{q}$ only depends on F, it is always false. During the movement $\mathtt{i}_2$ remains true, hence, by Lemma 13, the configuration always belongs to $\mathcal{P}2$ and it cannot belong to any other sub-phase of $\mathcal{F}4$. Moreover, it cannot belong to any other phase because $\mathtt{f}_4$ holds. It follows that g is the only moving robot.
- H_3: No collisions are created as there are no robots between g and its target since g always moves inside $C^1_\uparrow(F)$ toward its border or toward $c(F)$. In any case no further robot is met as all of them are already positioned according to F.
- H_4: Once g reaches the target, predicate $\mathtt{w}$ holds, that is F has been formed and the configuration does not belong to any phase.

Sub-phases $\mathcal{Q}1$–$\mathcal{Q}4$ manage the case where $\mathtt{q}$ is true. The main difficult here is to maintain $C(R)$ unchanged while guards are moving. In fact, if g'' rotates toward $\mu(g'')$ as in sub-phase $\mathcal{P}1$, $C(R)$ could change.

As first move, in $\mathcal{Q}1$ guard g is moved radially on $C(R)$ by means of move m_{15}.

- H_0: Only g is involved in the move. As it moves radially toward $C(R)$, angle α is maintained along all the movement, hence g is easily recognizable.
- H_1: The configuration is maintained always asymmetric as g is the only robot inside $C(R)$. So it cannot participate to neither a rotation, nor a reflection as the axis of symmetry should pass throw g, but then the starting configuration R was symmetric, a contradiction.
- H_2: We show that m_{15} is safe. As $\mathtt{q}$ only depends on F, it is always true. During the movement of g, $\mathtt{i}_3$ remains true and hence the observed configuration always belongs to $\mathcal{Q}1$. By Lemma 13,

it cannot belong to any other sub-phase of $\mathcal{F}4$ and it cannot belong to any other phase because $\mathtt{f}_4$ holds. It follows that g is the only moving robot.

H_3: No collisions are created as g is the only robot inside $C(R)$ and there are no robots forming an angle of α degrees on $C(R)$ as they are all well positioned according to F but for g'' that by construction is not on the way of g.

H_4: Once g reaches the target, predicate $\mathtt{i}_3$ becomes false while either $\mathtt{i}_4$ or $\mathtt{i}_5$ become true, depending whether g'' is already on target or not. By Lemma 13, this means the configuration may belong to $\mathcal{Q}2$ or $\mathcal{Q}3$ and it cannot belong to any other sub-phase of $\mathcal{F}4$. Moreover, it cannot belong to any other phase because $\mathtt{f}_4$ holds.

Sub-phase $\mathcal{Q}2$ is applied if g'' is not yet on its target. Since now all robots are in $\partial C(R)$, g'' cannot freely move toward $\mu(g'')$ as this could change $C(R)$. A safe place to reach is the antipodal point p to g. Move m_{16} rotates g'' on $C(R)$ toward the closest point among p and $\mu(g'')$.

H_0: Only g'' is involved in the move. The angle α between g and g' maintains g'' easily recognizable along all the movement.

H_1: The configuration is maintained always asymmetric as the angle α between g and g' guarantees no rotations. Moreover, the only axis of reflection should cut α. Since $\mathtt{q}$ holds, $|F|-1$ points occupy a semi-circle. As all robots but g and g'' are not yet positioned according to F, it follows that g is the only robot in the semi-circle between g' and g'' in the clockwise direction. Since by assumption there are at least four robots, this situation cannot hold as there cannot be a robot specular to one which is not a guard.

H_2: We show that m_{16} is safe. As $\mathtt{q}$ only depends on F, it is always true. During the movement of g'', $\mathtt{i}_4$ remains true. As a consequence, by Lemma 13, the observed configuration belongs to $\mathcal{Q}2$ and it cannot belong to any other sub-phase of $\mathcal{F}4$. Moreover, it cannot belong to any other phase because $\mathtt{f}_4$ holds. It follows that g'' is the only moving robot.

H_3: No collisions are created as by construction the path between g'' and its target along $C(R)$ does not contain further robots.

H_4: Once g'' reaches the target, predicate $\mathtt{i}_4$ becomes false while predicate $\mathtt{i}_5$ becomes true. By Lemma 13, this means the configuration may belong to $\mathcal{Q}3$ and it cannot belong to any other sub-phase of $\mathcal{F}4$. Moreover, it cannot belong to any other phase because $\mathtt{f}_4$ holds.

In sub-phase $\mathcal{Q}3$, guard g can freely move toward its final target by means of move m_{14} In fact, because of predicate $\mathtt{q}$, the move does not affect $C(R)$.

H_0: Only g is involved in the move. Robot g is always recognizable as the only robot of minimum view.

H_1: The configuration is maintained always asymmetric as g is the only one with minimum view and its clockwise view is different from its anti-clockwise view.

H_2: We show that m_{14} in phase $\mathcal{Q}3$ is safe. As $\mathtt{q}$ only depends on F, it is always true. During the movement of g, $\mathtt{i}_5$ remains true. Hence, by Lemma 13, the observed configuration belongs to $\mathcal{Q}3$ and it cannot belong to any other sub-phase of $\mathcal{F}4$. Moreover, it cannot belong to any other phase because $\mathtt{f}_4$ holds. It follows that g is the only moving robot.

H_3: No collisions are created as by construction the path between g and its target along $C(R)$ does not contain further robots.

H_4: Once g reaches the target, predicate $\mathtt{i}_5$ becomes false while either $\mathtt{i}_6$ or $\mathtt{w}$ become true, depending whether g'' is already on target or not. This means the configuration either belongs to $\mathcal{Q}4$ according to Lemma 13 or F is formed. It cannot belong to any other phase as either $\mathtt{f}_4$ or $\mathtt{w}$ holds.

In sub-phase $\mathcal{Q}4$, guard g'' can freely move toward its final target by means of move m_{13}. In fact, predicate $\mathtt{q}$ guarantees that the target of g'' does not overcome the antipodal point to g, hence the movement does not affect $C(R)$.

H_0: Only g'' is involved in the move. Robot g'' is always recognizable as $\mathtt{i}_6$ holds and g'' is the only robot not on target.

H_1: During the movement, the configuration cannot admit a rotation as the arc from g' to g'' in the clockwise direction is greater than half of $C(R)$. There cannot be an axis of symmetry that makes g specular to g'. In fact, g'' cannot admit a specular robot with respect to such an axis as it is closer to the axis than g which contradicts the property of g being the robot of minimum view. There cannot be an axis making specular g to g''. In fact, g cannot admit a specular robot r with respect to such an axis as $\sphericalangle(r, c(R), g'')$ should be equal to $\sphericalangle(g, c(R), g') = 3\alpha$, but this is possible only once g'' has reached its target being g the robot of minimum view.

H_2: We show that m_{13} in phase $\mathcal{Q}4$ is safe. As $\mathtt{q}$ only depends on F, it is always true. During the movement of g'', $\mathtt{i}_6$ remains true. Hence, by Lemma 13, the configuration belongs to $\mathcal{Q}3$ and it cannot belong to any other sub-phase of $\mathcal{F}4$. Moreover, it cannot belong to any other phase because $\mathtt{f}_4$ holds. It follows that g'' is the only moving robot.

H_3: No collisions are created as by construction the path between g'' and its target along $C(R)$ does not contain further robots.

H_4: Once g'' reaches the target, predicate $\mathtt{w}$ becomes true. This means the configuration does not belong to any phase.

■

Theorem 2. *Let R be an initial asymmetric configuration of* ASYNC *robots without chirality, and F any pattern (possibly with multiplicities) with $|F| = |R|$. Then, there exists an algorithm able to form F from R.*

Proof: As remarked in Section V, the cases of $|R| \leq 2$ are either trivial or unsolvable, and hence are not required to be managed by our algorithm. The case of F being a single point is delegated to [9], whereas when $|R| = 3$ Theorem 1 holds. When $|R| > 3$, the claim simply follows by Lemmata 9, 14, 15, 16 and 17. In fact, Lemma 9 shows that R belongs exactly to one phase among $\mathcal{F}1$, $\mathcal{F}2$, $\mathcal{F}3$, and $\mathcal{F}4$, while the Lemmata 14–17 show that from a given phase only subsequent phases can be reached, or $\mathtt{w}$ eventually holds. The only possible cycles among transitions can occur in phase $\mathcal{F}1$ among sub-phases $\mathcal{A}1$ and $\mathcal{A}2$, or in $\mathcal{F}3$ among sub-phases $\mathcal{M}$, $\mathcal{N}$ and $\mathcal{O}$. However, the corresponding Lemmata 14 and 16 also show that such cycles can be performed only a finite number of times. ■

VII. Conclusion

We considered the Pattern Formation problem in the well-known asynchronous Look-Compute-Move model. So far the problem has been mainly investigated with the further assumption on the capability of robots to share a common left-right orientation (chirality).

Our study try to remove any assumption concerning the orientation of the robots. We shown that starting from asymmetric configurations, robots can deterministically form any pattern including symmetric ones and those containing multiplicities. This extends the previously known results in terms of required number of robots, multiplicities, and formalisms. In fact, our algorithm does not rely on any assumption of the number of robots, allows the formation of multiplicities if required by the given pattern, and it is provided in terms of logical predicates that facilitate to check its correctness.

Also, the relevance of our results is shown in light of the consequences obtained with respect to [6] and [15]. Our analysis re-opens the case of ASYNC robots endowed with chirality. In fact, the arguments in [6] and [15] still need to be fixed and it is not clear at the moment whether a completely new approach is required or even if the problem is solvable.

The main open question left asks to provide a deterministic algorithm that solves the Pattern Formation from any initial configuration, including symmetric ones. Potentially, robots should be able to form any pattern if starting from configurations characterized by symmetries that are included in the final pattern. The main difficulty in designing an algorithm for such cases is that in symmetric configurations many robots may move simultaneously, all those that look equivalent with respect to the symmetry. The adversary can decide to move any subset of such robots, and all of them may traverse different distances as the adversary can stop them in different moments. Hence, during a Look phase, it becomes very difficult to provide a mean to guess how the current configuration has been originated.

References

[1] I. Suzuki and M. Yamashita, "Distributed anonymous mobile robots: Formation of geometric patterns," *SIAM J. Comput.*, vol. 28, no. 4, pp. 1347–1363, 1999.

[2] M. Yamashita and I. Suzuki, "Characterizing geometric patterns formable by oblivious anonymous mobile robots," *Theor. Comput. Sci.*, vol. 411, no. 26-28, pp. 2433–2453, 2010.

[3] Y. Yamauchi, T. Uehara, S. Kijima, and M. Yamashita, "Plane formation by synchronous mobile robots in the three dimensional euclidean space," in *Proc. 29th Int.'l Symp. on Distributed Computing (DISC)*, ser. LNCS, vol. 9363. Springer, 2015, pp. 92–106.

[4] S. Bhagat, S. G. Chaudhuri, and K. Mukhopadhyaya, "Formation of general position by asynchronous mobile robots under one-axis agreement," in *Proc. 10th Int.'l WS on Algorithms and Computation (WALCOM)*, ser. LNCS, vol. 9627. Springer, 2016, pp. 80–91.

[5] S. Das, P. Flocchini, N. Santoro, and M. Yamashita, "Forming sequences of geometric patterns with oblivious mobile robots," *Distributed Computing*, vol. 28, no. 2, pp. 131–145, 2015.

[6] N. Fujinaga, Y. Yamauchi, H. Ono, S. Kijima, and M. Yamashita, "Pattern formation by oblivious asynchronous mobile robots," *SIAM J. Computing*, vol. 44, no. 3, pp. 740–785, 2015.

[7] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer, "Arbitrary pattern formation by asynchronous, anonymous, oblivious robots," *Theor. Comput. Sci.*, vol. 407, no. 1-3, pp. 412–447, 2008.

[8] S. Ghike and K. Mukhopadhyaya, "A distributed algorithm for pattern formation by autonomous robots, with no agreement on coordinate compass," in *Proc. 6th Int.'l Conf. on Distributed Computing and Internet Technology, (ICDCIT)*, ser. LNCS, vol. 5966. Springer, 2010, pp. 157–169.

[9] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro, "Distributed computing by mobile robots: Gathering," *SIAM J. on Computing*, vol. 41, no. 4, pp. 829–879, 2012.

[10] P. Flocchini, G. Prencipe, N. Santoro, and G. Viglietta, "Distributed computing by mobile robots: Solving the uniform circle formation problem," in *Proc. 18th Int.'l Conf. on Principles of Distributed Systems (OPODIS)*, ser. LNCSe, vol. 8878. Springer, 2014, pp. 217–232.

[11] M. Mamino and G. Viglietta, "Square formation by asynchronous oblivious robots," in *Proc. of the 28th Canadian Conference on Computational Geometry (CCCG)*, 2016, pp. 1–6.

[12] S. Cicerone, G. Di Stefano, and A. Navarra, "Asynchronous embedded pattern formation without orientation," in *Proc. 30th Int.'l Symp. on Distributed Computing (DISC)*, ser. LNCS, vol. 9888. Springer, 2016, pp. 85–98.

[13] Y. Dieudonné, F. Petit, and V. Villain, "Brief announcement: leader election vs pattern formation," in *Proc. 29th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, 2010, pp. 404–405.

[14] ——, "Leader election problem versus pattern formation problem," in *Proc. 24th Int.'l Symp. on Distributed Computing (DISC)*, ser. LNCS, vol. 6343. Springer, 2010, pp. 267–281.

[15] Q. Bramas and S. Tixeuil, "Brief Announcement: Probabilistic asynchronous arbitrary pattern formation," in *Proc. 35th ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing (PODC)*, 2016.

[16] Y. Yamauchi and M. Yamashita, "Randomized pattern formation algorithm for asynchronous oblivious mobile robots," in *Proc. 28th Int.'l Symp. on Distributed Computing, (DISC)*, ser. LNCS, vol. 8784. Springer, 2014, pp. 137–151.

[17] Q. Bramas and S. Tixeuil, "Probabilistic asynchronous arbitrary pattern formation," *CoRR*, vol. abs/1508.03714, 2016. [Online]. Available: https://arxiv.org/abs/1508.03714

[18] S. G. Chaudhuri, S. Ghike, S. Jain, and K. Mukhopadhyaya, "Pattern formation for asynchronous robots without agreement in chirality," *CoRR*, vol. abs/1403.2625, 2014. [Online]. Available: http://arxiv.org/abs/1403.2625

[19] Y. Dieudonné, F. Petit, and V. Villain, "Leader election problem versus pattern formation problem," *CoRR*, vol. abs/0902.2851, 2009. [Online]. Available: http://arxiv.org/abs/0902.2851

[20] G. D'Angelo, G. Di Stefano, and A. Navarra, "Gathering on rings under the look-compute-move model," *Distributed Computing*, vol. 27, no. 4, pp. 255–285, 2014.

[21] G. D'Angelo, G. D. Stefano, and A. Navarra, "Gathering six oblivious robots on anonymous symmetric rings," *J. Discrete Algorithms*, vol. 26, pp. 16–27, 2014.

[22] G. D. Stefano, P. Montanari, and A. Navarra, "About ungatherability of oblivious and asynchronous robots on anonymous rings," in *Proc. 26th Int.'l WS on Combinatorial Algorithms (IWOCA)*, ser. LNCS, vol. 9538. Springer, 2016, pp. 136–147.

[23] F. Bonnet, M. Potop-Butucaru, and S. Tixeuil, "Asynchronous gathering in rings with 4 robots," in *Proc. 5th Int.'l Conf. on Ad-hoc, Mobile, and Wireless Networks (ADHOC-NOW)*, ser. LNCS, vol. 9724. Springer, 2016, pp. 311–324.

[24] B. Bérard, P. Lafourcade, L. Millet, M. Potop-Butucaru, Y. Thierry-Mieg, and S. Tixeuil, "Formal verification of mobile robot protocols," *Distributed Computing*, vol. 29, no. 6, pp. 459–487, 2016.

[25] H. T. T. Doan, F. Bonnet, and K. Ogata, "Model checking of a mobile robots perpetual exploration algorithm," in *Proc. 6th Int.'l Work. on Structured Object-Oriented Formal Language and Method (SOFL+MSVL)*, ser. Lecture Notes in Computer Science, vol. 10189, 2017, pp. 201–219.

[26] L. Millet, M. Potop-Butucaru, N. Sznajder, and S. Tixeuil, "On the synthesis of mobile robots algorithms: The case of ring gathering," in *Proc. 16th Int.'l Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, ser. LNCS, vol. 8756. Springer, 2014, pp. 237–251.

[27] N. Megiddo, "Linear-time algorithms for linear programming in R^3 and related problems," *SIAM J. Comput.*, vol. 12, no. 4, pp. 759–776, 1983.

[28] E. Welzl, "Smallest enclosing disks (balls and ellipsoids)," in *Results and New Trends in Computer Science*. Springer-Verlag, 1991, pp. 359–370.

[29] M. Cieliebak and G. Prencipe, "Gathering autonomous mobile robots," in *Proceedings of the 9th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, vol. 13. Carleton Scientific, 2002, pp. 57–72.