On the Synthesis of Mobile Robots Algorithms: the Case of Ring Gathering

Laure Millet, Maria Potop-Butucaru, Nathalie Sznajder, Sébastien Tixeuil

On the Synthesis of Mobile Robots Algorithms: the Case of Ring Gathering

Laure Millet[1,2], Maria Potop-Butucaru[1,2], Nathalie Sznajder[1,2], and Sebastien Tixeuil[1,2,3]

[1] Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France
[2] CNRS, UMR 7606, LIP6, F-75005, Paris, France
[3] Institut Universitaire de France

Abstract. Recent advances in Distributed Computing highlight models and algorithms for autonomous swarms of mobile robots that self-organize and cooperate to solve global objectives. The overwhelming majority of works so far considers handmade algorithms and correctness proofs.
This paper is the first to propose a formal framework to automatically design distributed algorithms that are dedicated to autonomous mobile robots evolving in a discrete space. As a case study, we consider the problem of gathering all robots at a particular location, not known beforehand. Our contribution is threefold. First, we propose an encoding of the gathering problem as a reachability game. Then, we automatically generate an optimal distributed algorithm for three robots evolving on a fixed size uniform ring. Finally, we prove by induction that the generated algorithm is also correct for any ring size except when an impossibility result holds (that is, when the number of robots divides the ring size).

consists in?

1 Introduction

The Distributed Computing community, motivated by the variety of tasks that can be performed by autonomous robots and their complexity, started recently to propose formal models for these systems and to design and prove protocols in these models. The seminal paper by Suzuki & Yamashita [24] proposes a robot model, two execution models, and several algorithms (with associated correctness proofs) for gathering and scattering a set of robots. In their model, robots are identical and anonymous (they execute the same algorithm and they cannot be distinguished using their appearance), robots are oblivious (they have no memory of their past actions) and they have neither a common sense of direction, nor a common handedness (chirality). Furthermore robots do not communicate in an explicit way. However they have the ability to sense the environment and see the position of the other robots, which lets them find their way in their environment. Also, robots execute three-phase cycles: *Look*, *Compute* and *Move*. During the *Look* phase robots take a snapshot of the other robots' positions. The collected information is used in the *Compute* phase in which robots decide to move or to stay idle. In the *Move* phase, robots may move to a new position computed in the previous phase. The two execution models are denoted (using recent taxonomy [13]) FSYNC, for fully synchronous, and SSYNC, for semi-synchronous. In the SSYNC model an arbitrary

non-empty subset of robots execute the three phases synchronously and atomically. In the FSYNC model all robots execute the three phases synchronously.

A recent trend, motivated by practical applications such that exploration or surveillance, is the study of robots evolving in a discrete space with a *finite* number of locations. This discrete space is modeled by a graph, where nodes represent locations or sites, and edges represent the possibility for a robot to move from one site to the other. The discrete setting significantly increases the number of symmetric configurations when the underlying graph is also symmetric (*e.g.* a ring).

One of the benchmarking [13] problems for mobile robots evolving in a discrete space is that of *gathering*. Regardless of their initial positions, robots have to move in such a way that they are eventually located on the same location, not known beforehand, and remain there thereafter. The case of ring networks is especially intricate, since its regular structure introduces a number of possible symmetric situations, from which the limited abilities of robots make it difficult to escape. A particular disposal (or configuration) of robots in the ring is *symmetrical* if there exists an axis of symmetry, that maps single robots into single robots, multiplicities into multiplicities, and empty nodes into empty nodes. A symmetric configuration can be edge-edge, node-edge or node-node symmetrical if the axis goes through two edges, through one node and one edge, or through two nodes, respectively. A *periodic* configuration is a configuration that is invariant by non-trivial rotation.

On the negative side, it was shown [17] that gathering is impossible when the algorithm run by every robot is deterministic and there are only two robots, or if the initial configurations are periodic, or edge-edge symmetric, or if the ability for a robot to detect multiple robots on a single location (denoted as *multiplicity detection*) is not available. Running a probabilistic algorithm [20] permits to start from an arbitrary initial configuration (including periodic and edge-edge symmetric) but still requires multiplicity detection. In the deterministic setting, a number of ring gathering algorithms have been proposed in the literature [16,10,11,23,9] for the cases left open by impossibility results, focusing on the problem solvability for different initial configurations and different values for the size of the ring and the number of robots. When the robots are able to fully detect the number of robots in each location, a unified strategy was proposed [10]. When multiplicity detection is only available on the current position of each robot, more involved and specific approaches [14,15,16,9] are needed. Every aforementioned deterministic solution considers problem solvability with particular hypotheses, and does not consider performance issues (such as time needed to reach gathering, or the total number of moves before gathering is achieved). Also, only a handmade approach for both algorithm design and proof of correctness was considered in those works.

Most related to our concern are recent approaches to mechanizing the algorithm design or the correctness proof in the context of autonomous mobile robots [5,12,4,2]. Model-checking proved useful to find bugs in existing litterature [4] and formally assess published algorithms [12,4]. Proof assistants enabled the use of high order logic to certify impossibility results [2]. To our knowledge, the only previous attempt to automatically generate mobile robots algorithms (for the problem of perpetual exclusive exploration) is due to Bonnet *et al.* [5], but exhibits important limitations for studying the gathering problem. Indeed their approach is brute force (it generate every possi-

ble algorithm in a particular setting, regardless of the problem to solve) and specific to configurations where *(i)* a location can only host one robot (so, gathering cannot be expressed), and *(ii)* no symmetry appears.

Games and protocols synthesis. In the formal methods community, automatically synthesizing programs that would be correct by design is a problem that raised interest early [8,18,1,22]. Actually, this problem goes back to Church [7,6]. When the program to generate is intended to work in an open system, maintaining an on-going interaction with a (partially) unknown environment, it is known since [6] that seeing the problem as a *game* between the system and the environment is a successful approach. The system and its environment are considered as opposite players that play a game on some graph, the winning condition being the specification the system should fulfill however the environment behave. Then, the classical problem in game theory of determining winning strategies for the players is equivalent to find how the system should act in any situation, in order to always satisfy its specification. The case of mobile autonomous robots that we focus on in this paper falls in this category of problems: the robots may evolve (possibly indefinitely) on the ring, making decisions based on the global state of the system at each time instant. The vertices of graph on which the players will play would then be some representation of the different global positions of the robots on the ring. The presence of an opposite player (or environment) is motivated by the absence of chirality of the robots: when a robot is on an axis of symmetry, it is unable to distinguish its two sides one from another, hence to choose exactly *where* it moves ; this decision is supposed to be taken by the opposite player.

Our contribution. In this paper, we introduce the use of formal methods for automatic synthesis of autonomous mobile robot algorithms, in the discrete space model. As a case study, we consider the problem of gathering all robots at a particular location, not known beforehand. Our contribution is threefold. First, we propose an encoding of the gathering problem as a reachability game, the players being the robot algorithm on the one side and the scheduling adversary (that is also capable for dynamically deciding robot chirality at every activation) on the other side. Our encoding is general enough to encompass classical FSYNC and SSYNC execution models for robots evolving on ring-shaped networks, including (and contrary to the existing ad hoc solution [5]) when several robots are located at the same node and when symmetric situations occurs. Then, in the FSYNC model, we automatically generate an *optimal* distributed algorithm for three robots evolving on a fixed size uniform ring. Our optimality criterion refers to the number of robot moves that are necessary to actually achieve gathering. Finally, we prove by induction that the mechanically generated algorithm is also correct for any ring size except when an impossibility result holds (that is, when the number of robots divides the ring size). Our method can be seen as a first step towards "correct by design" actual robot protocol implementations.

2 Background

In this section we present a formal model for a robot system evolving on a ring and definitions and notations for a reachability game.

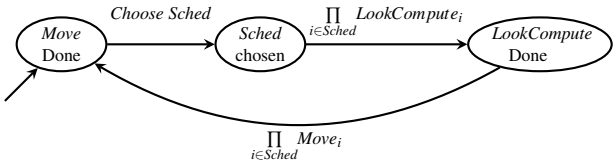


Fig. 1: The Semi Synchronous Schedulers automaton

2.1 Robot Network model

In the following we present the robots and system model using the formalism we proposed in [4]. We consider a set of robots evolving on a ring.

Robot model A robot behavior can be described by a finite automaton. Each robot executes a three-phase cycle composed of *Look*, *Compute*, *Move* phases. To start a cycle, a robot takes a snapshot of its environment, which is represented by a *Look* transition. Then it computes its future movement (*Compute* transition). Finally the robot moves according to its previous computation, this effective movement is represented by a *Move* transition, going back to its initial state. On a ring there are only three possibilities for the move: stay idle, move in the clockwise direction or in the counterclockwise direction. Note also that Look and Compute states can be merged in a single state - LookCompute.

Scheduler model. The three existing asynchrony models fully synchronous (FSYNC), semi-synchronous (SSYNC) and asynchronous (ASYNC) in robot networks are called schedulers. The scheduler can be modeled by a finite automaton. The synchronization of these schedulers with robots automata is an automaton that represents the global behavior of robots in the chosen model.

In the sequel we denote by $LookCompute_i$ (respectively $Move_i$), the *LookCompute* (resp. *Move*) phase of i^{th} robot. And for a subset *Sched* of robots, we denote by $\prod_{i \in Sched} LookCompute_i$ (resp. $\prod_{i \in Sched} Move_i$) the synchronization of all $LookCompute_i$ (resp. $Move_i$) actions of all robots in *Sched*.

In the SSYNC model, an arbitrary non-empty subset of robots is scheduled for execution at every phase, and operations are executed synchronously. In this case, the automaton consists of a cycle, where a set "*Sched*" is first chosen, then the *LookCompute* and *Move* phases are synchronized for this set. A generic automaton for SSYNC is described in Figure 1.

The FSYNC model is a particular case of the SSYNC model, where all robots are scheduled for execution at every phase, and operate synchronously thereafter.

System model. A configuration of k robots on a ring of size n encodes the position of the robots in the ring. The system is modeled by the automaton obtained by the synchronized product of k robot automata and the possible configurations. The scheduler is used to define the synchronization function. The alphabet of actions is $A = \prod_i A_i$, with

$A_i = \{LookCompute_i, Move_i, idle\}$ for each robot i. From this definition, states are of the form $s = (s_1, \ldots, s_k, c)$ where s_i is the local state of robot i, and c the configuration. A transition of the system is labeled by a tuple $a = (a_1, \ldots, a_k)$, where $a_i \in A_i$ for all $1 \leq i \leq k$ and $(s_1, \ldots, s_k, c) \xrightarrow{a} (s'_1, \ldots, s'_k, c')$ *iff* for all i, $s_i \xrightarrow{a_i} s'_i$ and c' is obtained from c by updating the positions of all robots i such that $a_i = Move_i$. To represent the scheduling, we denote by $\prod_{i \in Sched} Act_i$ the action $(a_1, \ldots, a_k)$ such that $a_i = idle$ if $i \notin Sched$ and $a_i \in \{LookCompute_i, Move_i\}$ otherwise.

2.2 Reachability Games

In the following we revisit the reachability games. We present here classical notions on this subject. For more details, the interested reader can fruitfully consult the survey [19]. If A is a set of symbols, A^* is the set of finite sequences of elements of A (also called *words*), and A^ω the set of infinite such sequences, with ε the empty sequence. We note $A^+ = A^* \setminus \{\varepsilon\}$, and $A^\infty = A^* \cup A^\omega$. For a sequence $w \in A^\infty$, we denote its *length* by $|w|$. If $w \in A^*$, $|w|$ is equal to its number of elements. If $w \in A^\omega$, $|w| = \infty$. For all words $w = a_1 \cdots a_k \in A^*$, $w' = a'_1 \cdots \in A^\infty$, we define the *concatenation* of w and w' by the word noted $w \cdot w' = a_1 \cdots a_k a'_1 \cdots$. We sometimes omit the symbol and simply write ww'. If $L \subseteq A^*$ and $L' \subseteq A^\infty$, we define $L \cdot L' = \{w \cdot w' \mid w \in L, w' \in L'\}$.

A game is composed of an *arena* and *winning conditions*.

Arena An arena is a graph $\mathcal{A} = (V, E)$ in which the set of vertices $V = V_p \uplus V_o$ is partitioned into V_p, the vertices of the protagonist, and V_o the vertices of the opponent. The set of edges $E \subseteq V \times V$ allows to define the set of successors of some given vertex v, noted $vE = \{v' \in V \mid (v, v') \in E\}$. In the following, we will only consider finite arenas.

Plays To play on an arena, a token is positioned on an initial vertex. Then the token is moved by the players from one vertex to one of its successors. Each player can move the token only if it is on one of her own vertices. Formally, a play is a path in the graph, i.e., a finite or infinite sequence of vertices $\pi = v_0 v_1 \cdots \in V^\infty$, where for all $0 < i < |\pi|$, $v_i \in v_{i-1}E$. Moreover, a play is finite only if the token has been taken to a position without any successor (where it is impossible to continue the game): if π is finite with $|\pi| = n$, then $v_{n-1}E = \emptyset$.

Strategies A strategy for the protagonist determines to which position she will bring the token whenever it is her turn to play. To do so, the player takes into account the history of the play, and the current vertex. Formally, a strategy for the protagonist is a (partial) function $\sigma : V^* \cdot V_p \to V$ such that, for all sequence (representing the current history) $w \in V^*$, all $v \in V_p$, $\sigma(w \cdot v) \in vE$ (i.e. the move is possible with respect to the arena). A strategy σ is *memoryless* if it does not depend on the history. Formally, it means that for all $w, w' \in V^*$, for all $v \in V_p$, $\sigma(w \cdot v) = \sigma(w' \cdot v)$. In that case, we may simply see the strategy as a function $\sigma : V_p \to V$.

Given a strategy σ for the protagonist, a play $\pi = v_0 v_1 \cdots \in V^\infty$ is said to be σ-*consistent* if for all $0 < i < |\pi|$, if $v_{i-1} \in V_p$, then $v_i = \sigma(v_0 \cdots v_{i-1})$. Given an initial vertex v_0, the *outcome* of a strategy σ is the set of plays starting in v_0 that are σ-consistent. Formally, given an arena $A = (V, E)$, an intial vertex v_0 and a strategy $\sigma : V^* V_p \to V$, we let $Outcome(A, v_0, \sigma) = \{v_0\pi \in V^\infty \mid v_0\pi \text{ is a play and is } \sigma\text{-consistent}\}$.

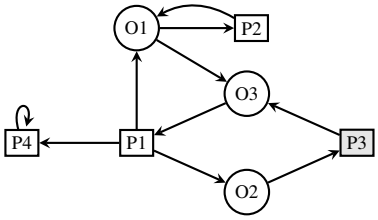


Fig. 2: A two-player game. In this figure protagonist vertices are represented by rectangles and antagonist vertices by circles. The winning condition is $Reach(\{P3\})$. Any path in the graph is a play. From P2 the protagonist has no winning strategy. From P1 a (memoryless) winning strategy is to go to O2. Winning positions are $\{P1, P3\}$.

Winning conditions, winning plays, winning strategies We define the *winning condition* for the protagonist as a subset of the plays $Win \subseteq V^\infty$. Then, a play π is *winning* for the protagonist if $\pi \in Win$. In this work, we focus on the simple case of reachability games: the winning condition is then expressed according to a subset of vertices $T \subseteq V$ by $Reach(T) = \{\pi = v_0 v_1 \cdots \in V^\infty \mid \exists 0 \leq i < |\pi| : v_i \in T\}$. This means that the protagonist wins a play whenever the token is brought on a vertex belonging to the set T. Once it has happened, the play is winning, regardless of the following actions of the players.

Given an arena $\mathcal{A} = (V, E)$, an initial vertex $v_0 \in V$ and a winning condition Win, a *winning strategy* σ for the protagonist is a strategy such that any σ-consistent play is winning. In other words, a strategy σ is winning if $Outcome(\mathcal{A}, v_0, \sigma) \subseteq Win$. The protagonist wins the game $(\mathcal{A}, v_0, Win)$ if she has a winning strategy for $(\mathcal{A}, v_0, Win)$. We say that σ is winning on a subset $U \subseteq V$ if it is winning starting from any vertex in U: if $Outcome(\mathcal{A}, v_0, \sigma) \subseteq Win$ for all $v_0 \in U$. A subset $U \subseteq V$ of the vertices is *winning* if there exists a strategy σ that is winning on U.

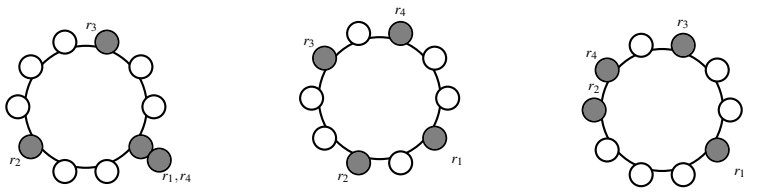
Solving a reachability game Given an arena $\mathcal{A} = (V, E)$, a subset $T \subseteq V$, one wants to determine the set $U \subseteq V$ of winning positions for the protagonist, and a strategy $\sigma : V^* V_p \to V$ for the protagonist, that is winning on U for $Reach(T)$.

Figure 2 represents a reachability 2-player game. We recall now a well-known result on reachability games:

Theorem 1. *The set of winning positions for the protagonist in a reachability game can be computed in linear time in the size of the arena. Moreover, from any position, the protagonist has a winning strategy if and only if she has a memoryless winning strategy.*

3 Encoding the gathering problem into a game

As we have claimed in the introduction, the gathering problem for synchronous robots is actually a game between the robots, that have an objective (winning condition) and evolve on a graph encoding the different configurations, and an opponent that can decide the actual movement of a disoriented robot, i.e. a robot whose observation of the



(a) A disoriented tower
$obs_{r_1}=\{(2,3,2,-1),(-1,2,3,2)\}$
$view_{r_1}=\{(2,3,2)\}$

(b) A periodic configuration
$obs_{r_1}=\{(1,2,1,2),(2,1,2,1)\}$
$view_{r_1}=obs_{r_1}$

(c) A rigid configuration
$obs_{r_1}=\{(3,0,1,2),(2,1,0,3)\}$
$view_{r_1}=obs_{r_1}$

Fig. 3: Robot observations and Views

ring is symmetrical, hence is unable to distinguish its two sides from one another. It may seem at first that the model actually needed is the one of *distributed games*, in which each robot represents a distinct player, all of them cooperating against a hostile environment. In distributed games, existence of a winning strategy for the team of players is undecidable [21]. However, the fact that the system is synchronous or semi-synchronous, and that the robots are able to sense their global environment, and thus to always know the global state of the system, allows us to stay in the framework of 2-player games, and to encode the set of robots as a single player. Of course, the strategy obtained will be centralized, but we will design the game in order to obtain only strategies that can be distributed amongst anonymous, memoryless robots without chirality. In the rest of the paper, we focus on the synchronous semantics for the system. With minor modifications, the game can be modified to handle the semi-synchronous semantics.

3.1 Encoding robots configurations: symmetries and equivalences

Consider a robot system consisting of k robots and n nodes ($k < n$). The configuration of such a system is represented by the tuple $(d_1, \cdots, d_k)$, such that $\Sigma_{i=1}^{k} d_i = n-k$, and $d_i \in \{-1, 0, \cdots, n-1\}$. Each value d_i represents the number of free nodes between the i^{th} robot and the next robot in the clockwise direction. When the two robots occupy adjacent nodes, $d_i = 0$, and when these two robots occupy the same node, $d_i = -1$. Let $\mathcal{C} = \{(d_1, \cdots, d_k) \mid \Sigma_{i=1}^{k} d_i = n-k \text{ and } d_i \in \{-1, 0, \cdots, n-1\}\}$ the set of all configurations. In a configuration, each robot can observe the entire ring, centered in its own position. Since the robots have no chirality, given a configuration $C = (d_1, \cdots, d_k)$, the *observation of robot i* is $obs_i(C) = \{(d_i, d_{i+1}, \cdots d_k, d_1, \cdots d_{i-1}), (d_{i-1}, \cdots, d_1, d_k, \cdots d_i)\}$. Let $Obs = \{obs_i(C) \mid C \in \mathcal{C}, 1 \leq i \leq k\}$ be the set of all possible observations.

Several types of configurations can be distinguished (see Figure 3): *periodic*: if there are several axis of symmetry, *symmetric*: if there is only one axis of symmetry (edge-edge, node-edge, node-node), *rigid configurations*: all other configurations.

A configuration is called *tower configuration* if there are several robots on the same node. Robots constituting this tower are the ones such that at least one tuple of their observation begins with -1.

Since the robots take snapshots of the configuration, and their decisions are based on this information, the states of the arena must represent the different configurations of the ring. The robots are anonymous, hence, different rotations of a similar ring in fact represent the same configuration. We define the rotation relation $\circlearrowleft \subseteq \mathcal{C} \times \mathcal{C}$ as follows: for all configurations C, $C' \in \mathcal{C}$, $C \circlearrowleft C'$ if and only if $C = (d_i, d_{i+1}, \cdots, d_{i+k-1})$ and $C' = (d_{i+1}, d_{i+2}, \cdots d_{i+k})$, where the addition symbol $+$ means sum modulo k. Since the robots have no chirality, one can easily observe that, for two configurations C and C', if $C = (d_1, \cdots, d_k)$ and $C' = (d_k, \cdots, d_1)$, then, for all robot i, $obs_i(C) = obs_i(C')$. We then define the mirror relation $\sim \subseteq \mathcal{C} \times \mathcal{C}$ by $C \sim C'$ if and only if $C = (d_1, \cdots, d_k)$ and $C' = (d_k, \cdots, d_1)$. From these two relations, we define an equivalence relation $\equiv \subseteq \mathcal{C} \times \mathcal{C}$ on the configurations, that identify all the configurations on which the robots should behave the same way: we let $\equiv \stackrel{\text{def}}{=} (\circlearrowleft \cup \sim)^*$.

The following lemma states that our equivalence relation is correct with respect to robots behavior.

Lemma 2. *For all $C \in \mathcal{C}$, $\bigcup_{1 \leq i \leq k} obs_i(C) = [C]_{\equiv}$.*

Then, an equivalence class of configurations can be seen as the set of observations for the robots in such a configuration.

We let $[C]_{\equiv}$ be the equivalence class of a configuration $C \in \mathcal{C}$, and we define an application $rep : \mathcal{C}/\equiv \rightarrow \mathcal{C}$, such that $rep([C]_{\equiv}) \in [C]_{\equiv}$ for all $C \in \mathcal{C}$, that associates to each equivalence class a unique representative in this class, say the smallest w.r.t lexicographic order on tuples. For the rest of the paper, when we use the sum symbol on indexes of elements of a configuration, it means sum modulo k.

3.2 Encoding the moves of the robots and transitions between configurations

To define precisely the transitions between the configurations, we need the following auxiliary notations. We let $\mathbb{M} = \{\curvearrowright, \curvearrowleft, \uparrow\}$ be the different possible moves for a robot, where, as one easily guesses, $\curvearrowright$ means that the robot moves in the clockwise direction, $\curvearrowleft$ means that it moves in the counter-clockwise direction, and $\uparrow$ means that the robot does not move. We will use the fact that, for robots on a tower, a deterministic algorithm will either make them all move, or none of them. However, if they are disoriented, they can move in different directions. When a robot i moves, it modifies the distances d_i and d_{i-1} (increasing one of these two distances by one, and decreasing by one the other). We can encode this by an algebraic notations, adding the configuration and one vector of movement for each robot: the effect on the configuration of the move $\curvearrowleft$ of robot i will be represented by the k-tuple $m^{i,\curvearrowleft}$, the effect of the move $\curvearrowright$ will be represented by $m^{i,\curvearrowright}$ and if the robot does not move, it will be represented by m^0. These tuples are defined as follows: for a robot $1 \leq i \leq k$, $m_i^{i,\curvearrowleft} = 1$ and $m_{i-1}^{i,\curvearrowleft} = -1$ and $m_j^{i,\curvearrowleft} = 0$ for all other $1 \leq j \leq k$. Similarly, $m_i^{i,\curvearrowright} = -1$ and $m_{i-1}^{i,\curvearrowright} = 1$ and $m_j^{i,\curvearrowright} = 0$ for all other $1 \leq j \leq k$. The last tuple is $m_j^0 = 0$ for all $1 \leq j \leq k$.

The idea is to add (in an element-by-element fashion) the current configuration to all the tuples representing the movements of the robots to obtain the next configuration. However, when the movements of two adjacent robots imply that they switch their positions in the ring, some absurd values (-2 or -3) may appear in the obtained configuration,

if the sum is naively effected, so a careful treatment of these particular cases must be done. To obtain the correct configuration, one should recall that robots are anonymous, hence if two robots switch their positions, it has the same effect as if none of them has moved. Also, if in a tower, some robots want to move clockwise, and the others want to move counterclockwise, the exact robots that will move are of no importance: the only important thing is the number of robots that move. We will then reorganize the movements between the robots, in order to keep correct values in our configurations: in a tower, we will assume that the robots that will move in the counterclockwise direction will always be the bottom ones, and when a robot moves right and joins a tower, we will assume that it will be placed at the bottom of the tower, and when it moves left and joins a tower, it will be placed at the top of the tower. These conventions will ensure that when adding the configuration and the different movements, we will not obtain aberrant values.

Formally, given a configuration $C = (d_1, \dots, d_k)$, we define $PosTower(C) = \{(i,j) \mid d_j \neq -1 \text{ and } \forall i \leq \ell < j, d_\ell = -1\}$ that contains the positions of the towers, encoded by the position of the first and the last robot in it. We then define $Pos(C) = PosTower(C) \cup \{(i,i) \mid 1 \leq i \leq k, \forall 1 \leq \ell \leq k, (i,\ell), (\ell,i) \notin PosTower(C)\}$, that contains the positions of the towers, and the positions of the isolated robots. Given a tuple of movements $(m_i)_{1\leq i\leq k}$, given $(i,j) \in Pos(C)$, $N^{\curvearrowright}_{(i,j)} = |\{m^{\curvearrowright}_\ell \mid i \leq \ell \leq j\}|$ and $N^{\curvearrowleft}_{(i,j)} = |\{m^{\curvearrowleft}_\ell \mid i \leq \ell \leq j\}|$. We first reorganize the movements of the robots in the towers: for all $(i,j) \in PosTower(C)$, we let $m'_\ell = m^{\ell,\curvearrowleft}$ for all $i \leq \ell \leq \left(N^{\curvearrowleft}_{(i,j)} + i - 1\right)$ and $m'_\ell = m^{\ell,\curvearrowright}$ for all $\left(N^{\curvearrowleft}_{(i,j)} + i\right) \leq \ell \leq j$. For all $(i,i) \in Pos(C) \setminus PosTower(C)$, $m'_i = m_i$. Now, we iteratively modify the tuple m'. Let $(i,j) \in Pos(C)$ be the element of $Pos(C)$ considered at the t^{th} iteration and let m^t be the current tuple encoding the moves.

- If $d_j \neq 0$, $m^{t+1} = m^t$.
- Otherwise, let r such that $(j+1,r) \in Pos(C)$ (if $r = j+1$, the next robot is isolated, otherwise it is a tower).
  - If $N^{\curvearrowright}_{(i,j)} \geq N^{\curvearrowleft}_{(j+1,r)}$, then $m^{t+1}_\ell = m^{\ell,\curvearrowright}$ for all $j - N^{\curvearrowright}_{(i,j)} + N^{\curvearrowleft}_{(j+1,r)} + 1 \leq \ell \leq j$, $m^{t+1}_\ell = m^{\ell,0}$ for all $j - N^{\curvearrowright}_{(i,j)} \leq \ell \leq j - N^{\curvearrowright}_{(i,j)} + N^{\curvearrowleft}_{(j+1,r)}$ and for all $j+1 \leq \ell \leq j + N^{\curvearrowleft}_{(j+1,r)} - 1$, and $m^{t+1}_\ell = m^t_\ell$ for all other ℓ.
  - If $N^{\curvearrowright}_{(i,j)} < N^{\curvearrowleft}_{(j+1,r)}$, then the modification is symmetrical.

When all the elements of $Pos(C)$ have been visited, we obtain a tuple $(m^f_i)_{1\leq i\leq k}$.

Proposition 3. *For all configurations $C \in \mathcal{C}$, for all tuples $(m_i)_{1\leq i\leq k}$, $C + \sum_{i=1}^{k} m^f_i \in \mathcal{C}$, where $(m^f_i)_{1\leq i\leq k}$ has been obtained as described above.*

Proof (sketch). Let $C = (d_1, \dots, d_k)$. For all $1 \leq i \leq k$, if $d_i = 0$, then if the robot i wants to move in the clockwise direction, and the robot $i+1$ wants to move in the counterclockwise direction, then by our construction, $m^f_i = m^{i,0}$ and $m^f_{i+1} = m^{i+1,0}$, and the resulting distance will stay 0. For all other decisions of the robots, the distance obtained will be positive. If $d_i = -1$, by the reorganization of the robots on a tower, it is impossible that robot i wants to move in the clockwise direction and that the robot $i+1$

wants to move in the counterclockwise direction. Hence, the distance obtained is never less than -1. In all other cases, the obtained distance is necessarily positive. □

Definition 4 (successor of a configuration). *Given a configuration $C \in \mathcal{C}$ and a tuple of moves for the different robots $(m_i)_{i\in\{1,\dots,k\}} \in \mathbb{M}^k$, the successor configuration, noted $C \oplus (m_i)_{i\in\{1,\dots,k\}}$ is obtained by $C + \sum_{i=1}^{k} m_i^f \in \mathcal{C}$, where $(m_i^f)_{1\leq i\leq k}$ has been obtained as described above.*

3.3 The Gathering Game

We build an arena for a reachability game, such that the protagonist has a winning strategy if and only if one can design an algorithm for the robots to gather on a single node, starting from any configuration. The possible decisions of movements taken by the robots will be noted by $\Delta = \{\curvearrowright, \curvearrowleft, \uparrow, ?\}$, which is the set $\mathbb{M}$ of possible movements, added by a special decision ?, taken by a disoriented robot that nevertheless wants to move. We will note $\overline{\curvearrowright} = \curvearrowleft$, $\overline{\curvearrowleft} = \curvearrowright$, $\overline{\uparrow} = \uparrow$ and $\overline{?} = ?$. We consider the arena $\mathcal{A}_{\text{gather}} = (V_p \uplus V_o, E)$, where the set of protagonist states is $V_p = (\mathcal{C}/\equiv)$, the set of antagonist states is $V_o = \mathcal{C} \times (\Delta^k)$, and the edge relation E will ensure a strict alternance between the two players: $E \subseteq (V_p \times V_o) \cup (V_o \times V_p)$ and will be detailed in the rest of the subsection.

From V_p to V_o From a protagonist position, representing an equivalence class of configurations, the play continues on an antagonist position memorizing the different movements decided by each robot. Such a move is possible if, in a given equivalence class of configurations, the robots with the same observation take the same decision. However, our definition of observation does not capture what happens when several robots are stacked to form a tower: consider two robots on a tower, in a configuration of the form $C = (-1, d_2, \cdots, d_k)$. Using our definition of observation, we obtain $obs_1(C) = \{(-1, d_2, \cdots, d_k), (d_k, \cdots, d_2, -1)\}$ and $obs_2(C) = \{(d_2, \cdots, d_k, -1), (-1, d_k, \cdots, d_2)\}$, hence $obs_1(C) \neq obs_2(C)$ whereas in reality they observe the same thing. Thus, we will use the notion of *view* for a robot, where, if a robot is part of a tower, the distance from other robots in the tower is removed from its observation. Formally, we define the *view* of the robot i as follows:

Definition 5 (view). *Let $C \in \mathcal{C}$ and $1 \leq i \leq k$ be a robot. Let $(d_1, \dots, d_k) \in obs_i(C)$ be the smallest observation of C, with respect to the lexicographic order. We define the* view of robot *i by $view_i(C) = \{(d_i, \dots, d_j), (d_j, \dots, d_i)\}$, where $i < j$ are respectively the smallest and greatest index such that $d_i \neq -1$ (respectively $d_j \neq -1$).*

We let $\mathcal{V} = \{view_i(C) \mid C \in \mathcal{C}, 1 \leq i \leq k\}$ be the set of all possible views.

Note that if robot i does not belong to a tower then $view_i(C) = obs_i(C)$. Also, when $|view_i(C)| = 1$, the robot is disoriented (see Figure 3). For $o \in Obs$ an observation, we let $p(o) \in \mathcal{V}$ be the projection from an observation to obtain a view.

A *decision function* is a function that suggests a movement to a robot, according to its view.

Definition 6 (decision function). *A* decision function *is a function* $f : \mathcal{V} \to \Delta$ *such that, for all* $V \in \mathcal{V}$*, if* $|V| = 1$*, then* $f(V) \in \{\uparrow, ?\}$ *and if* $f(V) = ?$ *then* $|V| = 1$.

Given a configuration $C = (d_1, \ldots, d_k) \in \mathcal{C}$, we translate a decision function f into a real movement of each robot. For all $1 \leq i \leq k$, let $f(C,i)$ be defined as follows. If $(d_i, \cdots, d_k, d_1, \cdots d_{i-1})$ is the smallest element of $view_i(C) = \{(d_i, \cdots, d_k, d_1, \cdots d_{i-1}), (d_{i-1}, \cdots, d_1, d_k, \cdots d_i)\}$ in the lexicographic order, then $f(C,i) = f(view_i(C))$. Otherwise, $f(C,i) = \overline{f(view_i(C))}$. This is so because, when applying the real movements on a real configuration, the game (that makes the robots move) must be coherent on a common direction.

We are able to determine now the edge relation from a protagonist state to an antagonist state: for all $v \in V_p, v' \in V_o$, $(v,v') \in E$ if and only if there exists a decision function f such that $v' = \big(C, (a_1, \ldots, a_k)\big)$ defined as follows: $C = rep(v) = (d_1, \ldots, d_k)$ and, for all $1 \leq i \leq k$, $a_i = f(C,i)$.

From V_o to V_p The moves of the antagonist lead the game into the following configuration of the system resulting of the application of the decisions of all the robots. If one robot decides to move, but is disoriented, then the antagonist chooses the actual move ($\curvearrowright$ or $\curvearrowleft$) the robot will make. The next configuration reached by the robots is then determined by the actions chosen and by the decisions taken by the antagonist.

Definition 7. *For a state* $v' = (C, (a_1, \ldots, a_k))$*, we say that a tuple* $(m_i)_{i \in \{1,\ldots,k\}}$ *is* v'-compatible *if,*

- *for all* $1 \leq i \leq k$ *such that* $a_i \neq ?$*,* $m_i = a_i$*,*
- *for all* $1 \leq i \leq k$ *such that* $a_i = ?$*,* $m_i \neq \uparrow$.

A v'-compatible tuple is then a tuple in which the antagonist has chosen in which directions disoriented robots will move.

Then, we can formally define the edge relation from an antagonist state to a protagonist state: for all $v \in V_p$, $v' = (C, (a_1, \ldots, a_k)) \in V_o$, $(v', v) \in E$ if and only if there exists a v'-compatible tuple $(m_i)_{i \in \{1,\ldots,k\}}$ such that $v = [C \oplus (m_i)_{i \in \{1,\ldots,k\}}]_{\equiv}$.

To sum up, in $\mathcal{A}_{\text{gather}}$[4],

$$
\begin{aligned}
E = \{&(v,v') \in V_p \times V_o \mid \\
&\text{there exists a decision function } f \text{ such that } v' = \big(rep(v), (f(C,1), \ldots, f(C,k))\big)\} \\
&\cup \{(v',v) \in V_o \times V_p \mid v' = (C, (a_i, \ldots, a_k)) \\
&\text{and there exists a } v'\text{-compatible tuple } m(_i)_{i \in \{1,\ldots,k\}}, v = [C \oplus (m_i)_{i \in \{1,\ldots,k\}}]_{\equiv}\}.
\end{aligned}
$$

We now state the result that validates the construction: solving the reachability game that we have just defined amounts to automatically synthesizing an algorithm achieving the gathering for this system. Let $W = [(-1, \cdots, -1, n-1)]_{\equiv} \in V_p$ be the equivalence class of all the configurations representing the case where all the robots are positioned on a single node.

[4] To handle the semi synchronous semantics, the antagonist should also choose at each step the subset of robots that will be activated.

Theorem 8. *The winning region for the game $(\mathcal{A}_{gather}, W)$ corresponds exactly to the set of configurations from which the robots can achieve the gathering.*

Proof (Sketch). An algorithm $\mathcal{F}$ can be turned into a decision function $f : \mathcal{V} \to \Delta$ as follows: let $\{view_1, view_2\} \in \mathcal{V}$, and assume that $view_1 < view_2$ with $<$ being the lexicographic order. Let $o \in p^{-1}(view_1)$ be an observation compatible with the view $view_1$ (we recall that p is the projection of an observation for a robot in a tower to its view that removes the elements equal to -1). Then $f(\{view_1, view_2\}) = \mathcal{F}(o)$. Since the algorithm $\mathcal{F}$ takes the same decision for all the robots in a tower, hence for all $o \in p^{-1}(view_1)$, this definition indeed translates the algorithm into a decision function. The strategy that chooses this decision function will visit the same configurations as the algorithm on the real ring. Reciprocally, a winning strategy from a configuration class gives a decision function. To turn the decision functions for each configuration class into a distributed algorithm, we remark, thanks to Lemma 2, that one observation for a robot belongs to exactly one equivalence class of configurations. To determine the movement a robot takes according to its observation of the ring, it suffices to translate the decision function associated to the corresponding equivalence class into a movement in the ring. Then one can show that any sequence of configurations obtained by the algorithm corresponds to a play in the game, visiting the same configurations. □

4 Synthesis of 3-robots gathering protocol

In the case of a system with three robots, there are 6 distinct types of configuration classes:

- The 3-robots tower configuration, which is the configuration to reach: $[(-1,-1,n-1)]_{\equiv}$. From this class of configuration the edge leads to $(C,(a_1,a_1,a_1)$ with $a_1 \in \{\uparrow, ?\}$. However, this edge is not of interest for us since the gathering property is verified.
- The disoriented tower is a configuration where there is an axis of symmetry passing through the tower and the isolated robot. This configuration belongs to the class $[(-1, \frac{n-1}{2}, \frac{n-1}{2})]_{\equiv}$ and occurs only when n is odd. In this case, all robots are disoriented and thus the outgoing edges lead to all the states $\{(-1, \frac{n-1}{2}, \frac{n-1}{2}), (a_1,a_1,a_2)\}$ with $a_1, a_2 \in \{\uparrow, ?\}$.
- The tower configurations are the configurations of the classes $[(-1,d_2,d_3)]_{\equiv}$, with $rep([(-1,d_2,d_3)]_{\equiv}) = (-1,d_2,d_3)$ and $-1 < d_2 < d_3 \in \mathbb{N}$. The edges lead to all the states $\{(-1,d_2,d_3), (a_1,a_1,a_2\}$ with $a_1, a_2 \in \{\curvearrowleft, \curvearrowright, \uparrow\}$.
- The symmetrical configurations, which is in $[(d_1,d_1,d_2)]_{\equiv}$ with $-1 \neq d_1 \neq d_2$ and $-1 \neq d_2$. Recall that when k is odd and there is an axis of symmetry, the axis goes through an occupied node. If $d_1 < d_2$, the edges lead to $(C,(a_1,a_2,a_1)$ with $a_1 \in \{\curvearrowright, \curvearrowleft, \uparrow\}$ and $a_2 \in \{\uparrow, ?\}$, otherwise edges lead to $(C,(a_1,a_1,a_2)$ with $a_1 \in \{\curvearrowright, \curvearrowleft, \uparrow\}$ and $a_2 \in \{\uparrow, ?\}$.
- The rigid configurations are all other configurations. For a class $\mathbb{C}$ such that $rep(\mathbb{C}) = C$ does not fall into any of the above categories, the outgoing edges go to states $(C,(a_1,a_2,a_3))$ with $a_1, a_2, a_3 \in \{\curvearrowright, \curvearrowleft, \uparrow\}$.

We implemented the arena for three robots and different ring sizes, in the game-solver tool UPPAAL TIGA [3]. We verified the impossibility of the gathering from periodic configurations. Moreover we obtained that there is a winning strategy from all protagonist vertices except from the periodic configurations, and we identified in the edges relation that the edges that lead to $\{(C,(a,a,a))\}$ with $a \in \mathbb{M}$ are not part of any winning strategy.

The arena without the periodic class of configuration $\{[(d,d,d)]_{\equiv}\}$, and the edges that lead to $\{(C,(a,a,a))\}$ with $a \in \mathbb{M}$ from a protagonist vertex $[C]_{\equiv}$, is the graph such that all protagonist vertices are winning. In order to find the best winning strategies, weights are added on the edges. In order to minimize the number of robot moves, each edge is weighed by the number of robots that move. A strategy is a shortest path algorithm on this graph such that the protagonist vertices and opponent vertices are handled differently. The distance between a protagonist vertex and the configuration to reach is the minimum distance, and the distance between an opponent vertex and this configuration is the maximal distance between them.

We obtained all the optimal strategies, for each class of configurations $[(d_1,d_2,d_3)]_{\equiv}$, the edge relation is restricted. From these strategies we outline the following pattern of strategy.

- If all robots form a tower nobody moves. From $[(-1,-1,n-1)]_{\equiv}$ the edge relation leads to $((-1,-1,n-1),(\uparrow,\uparrow,\uparrow))$.
- If 2 robots form a tower the last robot takes the shortest path to the tower. From $[(-1,d_1,d_2)]_{\equiv}$with $-1 < d_1 < d_2$, the edge relation leads to $((-1,d_1,d_2),(\uparrow,\uparrow,\curvearrowleft))$. And from $[(-1,\frac{n-1}{2},\frac{n-1}{2})]_{\equiv}$ the edge relation leads to $((-1,\frac{n-1}{2},\frac{n-1}{2}),(\uparrow,\uparrow,?))$.
- If the configuration is symmetrical, in$[(d_1,d_1,d_2)]_{\equiv}$ with $-1 \neq d_1 \neq d_2$ and $d_2 \neq -1$, the proposed strategy depends on whether $rep([(d_1,d_1,d_2)]_{\equiv}) = (d_1,d_1,d_2)$ *or* (d_2,d_1,d_1).
  - If $rep([(d_1,d_1,d_2)]_{\equiv}) = (d_1,d_1,d_2)$ then the two symmetrical robots get closer to the last robot. The edge relation leads to $((d_1,d_1,d_2),(\curvearrowright,\uparrow,\curvearrowleft))$.
  - If $rep([(d_1,d_1,d_2)]_{\equiv}) = (d_1,d_1,d_2)$ then the disoriented robot moves. The edge relation leads to $((d_2,d_1,d_1),(\uparrow,\uparrow,?))$.
- if the configuration is rigid (in$[(d_1,d_2,d_3)]_{\equiv}$ with $-1 < d_1 < d_2 < d_3$)the edge relation leads to three possibilities :
  - The robot with the minimum view gets closer to its nearest neighbor. In this case the edge relation leads to $((d_1,d_2,d_3),(\curvearrowright,\uparrow,\uparrow))$.
  - The robot with the maximum view gets closer to its nearest neighbor.In this case the edge relation leads to $((d_1,d_2,d_3),(\uparrow,\uparrow,\curvearrowleft))$.
  - The robot with the minimum view and the robot with the maximum view get closer to their nearest neighbor. In this case the edge relation leads to $((d_1,d_2,d_3),(\curvearrowright,\uparrow,\curvearrowleft))$. This strategy is the two above strategies made simultaneously.

  Thus the edge relation for rigid configuration leads to: $\{((d_1,d_2,d_3),(a_1,\uparrow,a_2))\}$, with $a_1 \in \{\curvearrowright,\uparrow\}$, $a_2 \in \{\uparrow,\curvearrowleft\}$ and $a_1 \neq a_2$.

From Theorem 8, one can translate the decision functions for each configuration into a distributed algorithm. Among the possible strategies we present below the strategy that moves the robot with the minimum view and the robot with the maximum view closer to their nearest neighbor in the rigid configurations. Thus we obtain the following

distributed algorithm: if the view of the robot r is $view(r) = \{(y,-1,z),(z,-1,y)\}$ with $y < z$, r robot moves in order to increment z and decrement y. If $view(r) = \{(x,x,z),(z,x,x)\}$ with $x < z$ then r moves to increment z and decrement x,if $view(r) = \{(z,x,z),(z,x,z)\}$ with $x < z$ then r moves in any direction,if $view(r) = \{(x,y,z),(z,y,x)\}$ with $x < y < z$ then r moves to increment z and decrement x,if $view(r) = \{(y,x,z),(z,x,y)\}$ with $x < y < z$ then r moves to increment z and decrement y, and when r has a different view than the above, it remains idle.

The above algorithm is correct by construction for various values of n ($3 \leq n \leq 15$, $n = 100$). The following theorem proves that it is also correct for any ring of size n. Due to space limitation the proof by induction of the theorem is omitted.

Theorem 9. *In a ring of any size $n > 3$ starting from any configuration (except periodic ones) the above 3-gathering algorithm eventually reaches a gathering configuration.*

5 Conclusions and discussions

We proposed a formal method based on reachability games that permits to automatically generate distributed algorithms for mobile autonomous robots solving a global task. The task of gathering on a ring-shaped network was used as a case study. We hereby discuss current limitations and future works.

While our construction generates algorithms for a particular number of robots k and ring size n, the game encoding we propose enables to easily tackle the gathering problem for any given k and n, provided as inputs, since k and n are parameters of the arena described in Section 3. Also, we focused on the atomic FSYNC and SSYNC models. Breaking the atomicity of Look-Compute-Move cycles (that is, considering automatic algorithm production for the ASYNC model [13]) implies that robots cannot maintain a current global view of the system (their own view may be outdated), nor be aware of the view of other robots (that may be outdated as well). Then, our two-players game encoding is not feasible anymore. A natural approach would be to use distributed games, but they are generally undecidable as previously stated. So, a completely new approach is required for the automatic generation of non-atomic mobile robot algorithms.

The problem of synthesis for parameterized systems is a challenging path for future research. Also, the size of the game increases quickly with the number of robots; it is expected that to-be-discovered optimizations and/or heuristics will help bringing algorithm production more practical. Finally, we believe that part of our encoding (typically, configurations and transitions between configurations) can be reused for different problems on ring-shaped networks, such as exploration with stop or perpetual exploration and easily extended to other topologies.

References

1. M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable specifications of reactive systems. In *Proc. of ICALP'89*, volume 372 of *LNCS*, pages 1–17. Springer, 1989.
2. C. Auger, Z. Bouzid, P. Courtieu, S. Tixeuil, and X. Urbain. Certified impossibility results for byzantine-tolerant mobile robots. In *Proc. of SSS*, volume 8255 of *LNCS*, pages 178–190. Springer, 2013.

3. G. Behrmann, A. David, F. E., K. Larsen, and L. D. UPPAAL-Tiga: Time for playing games! In *Proc. of CAV 2007*, volume 4590 of *LNCS*, pages 121–125. Springer, 2007.
4. B. Bérard, L. Millet, M. Potop-Butucaru, S. Tixeuil, and Y. Thierry-Mieg. Vérification formelle et robots mobiles. In *Proc. of Algotel 2013*, 2013.
5. F. Bonnet, X. Défago, F. Petit, M. Potop-Butucaru, and S. Tixeuil. Brief announcement: Discovering and assessing fine-grained metrics in robot networks protocols. In *Proc. of SSS 2012*, volume 7596 of *LNCS*, pages 282–284. Springer, 2012.
6. J. R. Büchi and L. H. Landweber. Solving sequential conditions by finite-state strategies. *Trans. Amer. Math. Soc.*, 138:295–311, 1969.
7. A. Church. Logic, arithmetics, and automata. In *Proc. of Int. Congr. of Mathematicians*, pages 23–35, 1963.
8. E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. of IBM Workshop on Logics of Programs*, 1981.
9. G. D'Angelo, A. Navarra, and N. Nisse. Gathering and exclusive searching on rings under minimal assumptions. In *Proc of ICDCN 2014*, volume 8314 of *LNCS*, pages 149–164. Springer, 2014.
10. G. D'Angelo, G. D. Stefano, and A. Navarra. How to gather asynchronous oblivious robots on anonymous rings. In *Proc. of DISC 2012*, volume 7611 of *LNCS*, pages 326–340. Springer, 2012.
11. G. D'Angelo, G. D. Stefano, A. Navarra, N. Nisse, and K. Suchan. A unified approach for different tasks on rings in robot-based computing systems. In *IPDPS Workshops*, pages 667–676, 2013.
12. S. Devismes, A. Lamani, F. Petit, P. Raymond, and S. Tixeuil. Optimal grid exploration by asynchronous oblivious robots. In *Proc. of SSS*, pages 64–76. Springer, 2012.
13. P. Flocchini, G. Prencipe, and N. Santoro. *Distributed Computing by Oblivious Mobile Robots*. Morgan & Claypool Publishers, 2012.
14. T. Izumi, T. Izumi, S. Kamei, and F. Ooshita. Mobile robots gathering algorithm with local weak multiplicity in rings. In *Proc. of SIROCCO*, volume 6058 of *LNCS*, pages 101–113. Springer, 2010.
15. S. Kamei, A. Lamani, F. Ooshita, and S. Tixeuil. Asynchronous mobile robot gathering from symmetric configurations without global multiplicity detection. In *Proc. of SIROCCO*, volume 6796 of *LNCS*, pages 150–161. Springer, 2011.
16. S. Kamei, A. Lamani, F. Ooshita, and S. Tixeuil. Gathering an even number of robots in an odd ring without global multiplicity detection. In *Proc. of MFCS*, volume 7464 of *LNCS*, pages 542–553. Springer, 2012.
17. R. Klasing, E. Markou, and A. Pelc. Gathering asynchronous oblivious mobile robots in a ring. *Theor. Comput. Sci.*, 390(1):27–39, 2008.
18. Z. Manna and P. Wolper. Synthesis of communicating processes from temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 6(1):68–93, 1984.
19. R. Mazala. Infinite games. In *Automata, Logics, and Infinite Games*, volume 2500 of *LNCS*, pages 23–42. Springer, 2001.
20. F. Ooshita and S. Tixeuil. On the self-stabilization of mobile oblivious robots in uniform rings. In *Proc. of SSS*, volume 7596 of *LNCS*, pages 49–63. Springer, 2012.
21. G. L. Peterson and J. H. Reif. Multiple-person alternation. In *Proc. of FOCS'79*, pages 348–363. IEEE Computer Society Press, 1979.
22. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. of POPL'89*, pages 179–190. ACM, 1989.
23. G. D. Stefano and A. Navarra. Optimal gathering of oblivious robots in anonymous graphs. In *Proc. of SIROCCO*, volume 8179 of *LNCS*, pages 213–224. Springer, 2013.
24. I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, pages 1347–1363, 1999.

Appendix A: proof of Theorem 8.

We describe first how to transform a tuple of movements for the robots created for one configuration, so they perform the same movements on an another equivalent configuration. Let $C = (d_1, \ldots, d_k) \in \mathcal{C}$ and $C' \equiv C$. For a tuple of moves $m = (m_i)_{i \in \{1,\ldots,k\}} \in \mathbb{M}^k$ associated with C, we define the moves $m^{C'}$ that will represent the same decisions for the robots, on the configuration C'. Thanks to Lemma 2, we know that either $C = (d_i, \ldots, d_k, d_1, \ldots, d_{i-1})$ or $C = (d_{i-1}, \ldots, d_1, d_k, \ldots, d_i)$, for some $1 \leq i \leq k$. In the first case, we let

$$m^C_j = m_{(i+j-1)}$$

for $1 \leq j \leq k$, with the sum in the indices is modulo k, as usual. In the second case,

$$m^C_j = \begin{cases} \curvearrowright & \text{if } m_{i-j} = \curvearrowleft \\ \curvearrowleft & \text{if } m_{i-j} = \curvearrowright \\ \uparrow & \text{if } m_{i-j} = \uparrow \end{cases}$$

for all $1 \leq j \leq k$, with the addition modulo k.

The next result states that the equivalence classes are compatible with equivalent movements of the robots.

Theorem 10. *For all $C \equiv C'$ two equivalent configurations, for all $(m_1, \ldots, m_k) \in \mathbb{M}^k$ tuple of moves associated with C, $C \oplus (m_i)_{i \in \{1,\ldots,k\}} \equiv C' \oplus (m^{C'}_i)_{i \in \{1,\ldots,k\}}$.*

Theorem 8 The winning region for the game $(\mathcal{A}_{\text{gather}}, W)$ corresponds exactly to the set of configurations from which the robots can achieve the gathering.

Proof. Let $\mathcal{F}$ be the algorithm described as follows: for each observation of a robot $(d_1, \ldots, d_k) \in \mathcal{C}$, $\mathcal{F}(d_1, \ldots, d_k) = (d_1 + 1, \ldots, d_k - 1)$ or $\mathcal{F}(d_1, \ldots, d_k) = (d_1 - 1, \ldots, d_k + 1)$, meaning that, if a robot sees the sequence $(d_1, \ldots, d_k)$ it decides either to increase the distance d_1 (and hence decrease the distance d_k) or to decrease d_1. Since robots have no chirality, the algorithm is such that $\mathcal{F}(d_1, \ldots, d_k) = (d_1 + 1, \ldots, d_k - 1)$ if and only if $\mathcal{F}(d_k, \ldots, d_1) = (d_k - 1, \ldots, d_1 + 1)$, and if $(d_1, \ldots, d_k) = (d_k, \ldots, d_1)$, $\mathcal{F}(d_1, \ldots, d_k) = (d_1, \ldots, d_k)$ or the algorithm does not precise wether $\mathcal{F}(d_1, \ldots, d_k) = (d_1 + 1, \ldots, d_k - 1)$ or $\mathcal{F}(d_1, \ldots, d_k) = (d_1 - 1, \ldots, d_k + 1)$. In the vein of v'-compatible tuples defined in Definition 7, we define for all configuration $C = (d_1, \ldots, d_k) \in \mathcal{C}$, a $(\mathcal{F}, C)$-compatible tuple $m(C, \mathcal{F}) \in \mathbb{M}^k$ as follows: for all $1 \leq i \leq k$,

$$m(C, \mathcal{F})_i = \begin{cases} \curvearrowright & \text{if } \mathcal{F}(d_i, \ldots, d_k, d_1, \ldots, d_{i-1}) = (d_i + 1, \ldots, d_k, d_1, \ldots, d_{i-1} - 1) \\ \curvearrowleft & \text{if } \mathcal{F}(d_i, \ldots, d_k, d_1, \ldots, d_{i-1}) = (d_i - 1, \ldots, d_k, d_1, \ldots, d_{i-1} + 1) \\ \uparrow & \text{if } \mathcal{F}(d_i, \ldots, d_k, d_1, \ldots, d_{i-1}) = (d_i, \ldots, d_k, d_1, \ldots, d_{i-1}) \end{cases}$$

Moreover, $m(C, \mathcal{F})_i \in \{\curvearrowright, \curvearrowleft\}$ if $\mathcal{F}(d_i, \ldots, d_k, d_1, \ldots, d_{i-1})$ does not precise any direction (when the robot is disoriented). Then, applying the algorithm on a configuration $\mathcal{F}$, we obtain a configuration C' such that $C \xrightarrow{\mathcal{F}} C'$ if and only if $C' = C \oplus (m(C, \mathcal{F}))_{1 \leq i \leq k}$, where $(m(C, \mathcal{F})_{1 \leq i \leq k}$ is $(\mathcal{F}, C)$-compatible. Thanks to the properties

of the algorithm recalled at the beginning of the proof, we see that, for $C \equiv C'$, if $(m(\mathcal{F},C))_{1\leq i\leq k}$ and $(m(\mathcal{F},C'))_{1\leq i\leq k}$ are respectively $(\mathcal{F},C)$-compatible and $(\mathcal{F},C')$-compatible, $(m(\mathcal{F},C'))_i = m_i^C$, for all $1 \leq i \leq k$. Then, by Theorem 10, we obtain that if $C \xrightarrow{\mathcal{F}} C_1$ and $C' \xrightarrow{\mathcal{F}} C_2$, then $C_1 \equiv C_2$. We can then abuse notations and write $\mathbb{C} \xrightarrow{\mathcal{F}} \mathbb{C}'$, with $\mathbb{C}, \mathbb{C}' \in \mathcal{C}/\equiv$.

We can turn this algorithm into a decision function $f : \mathcal{V} \to \Delta$ as follows: let $\{view_1, view_2\} \in \mathcal{V}$, and assume that $view_1 < view_2$ with $<$ being the lexicographic order. Let $o \in p^{-1}(view_1)$ be an observation compatible with the view $view_1$ (we recall that p is the projection of an observation for a robot in a tower to its view that removes the elements equal to -1). Then $f(\{view_1, view_2\}) = \mathcal{F}(o)$. Since the algorithm $\mathcal{F}$ takes the same decision for all the robots in a tower, hence for all $o \in p^{-1}(view_1)$, this definition indeed translates the algorithm into a decision function. Once the decision function is defined, the strategy is straightforward. Let $\sigma : V_p \to V_o$ be a memoryless strategy defined, for all $\mathbb{C} \in V_p$ by $\sigma(\mathbb{C}) = (rep(\mathbb{C}), (a_1, \ldots, a_k))$ where the a_i are defined according to the decision function f we have just defined. Let $\mathbb{C} \in V_p$ and $v' \in V_o$ such that $(C, (a_1, \ldots, a_k)) = v' = \sigma(\mathbb{C})$. Thanks to the properties of $\mathcal{F}$ related to the anonymity and absence of chirality of the robots, one can be convinced that any v'-compatible tuple is also a $(\mathcal{F},C)$-compatible tuple. Hence, if $(v', \mathbb{C}') \in E$, then there exists a v'-compatible tuple $(m_1, \ldots, m_k)$ such that $C \oplus (m_i)_{1\leq i\leq k} \in \mathbb{C}'$. Since $(m_1, \ldots, m_k)$ is also $(\mathcal{F},C)$-compatible, $\mathbb{C} \xrightarrow{\mathcal{F}} \mathbb{C}'$. Then, any configuration from which the algorithm ensures the gathering is winning for $(\mathcal{A}_{\text{gather}}, W)$.

Reciprocally, let U be the winning region, and $\sigma : V_p \to V_o$ a (memoryless, from Theorem 1) winning strategy. From the arena $\mathcal{A}_{\text{gather}}$, this amounts to having, for each equivalence class $\mathbb{C} \in V_p$ a decision function $f_{\mathbb{C}}$. From these decision functions, one can define a unique memoryless algorithms for the robot, as follows: let $o = \{(d_1, \ldots, d_k), (d_k, \ldots, d_1)\}$ be an observation for a robot. To this observation corresponds a unique equivalence class $\mathbb{C}$. Then we let $\mathcal{F}(d_1, \ldots, d_k) = f_{\mathbb{C}}((d_1, \ldots, d_k), 1)$, and, by definition of $f_{\mathbb{C}}(C, i)$, it follows that $\mathcal{F}(d_1, \ldots, d_k) = (d_1 + 1, \ldots, d_k - 1)$ if and only if $\mathcal{F}(d_k, \ldots, d_1)$ $= (d_k - 1, \ldots, d_1 + 1)$, and if $(d_1, \ldots, d_k) = (d_k, \ldots, d_1)$, $\mathcal{F}(d_1, \ldots, d_k) = (d_1, \ldots, d_k)$ or the algorithm does not precise wether $\mathcal{F}(d_1, \ldots, d_k) = (d_1 + 1, \ldots, d_k - 1)$ or $\mathcal{F}(d_1, \ldots, d_k)$ $= (d_1 - 1, \ldots, d_k + 1)$. We now show that this algorithm is winning from any configuration C such that $[C]_{\equiv} \in U$. Let C be such a configuration, and $(m(C, \mathcal{F}))_{1\leq i\leq k}$ be a $(\mathcal{F},C)$-compatible tuple. Let $\mathbb{C} \in (\mathcal{C}/\equiv)$ be the equivalence class of C and $v' = (C_r, (a_1, \ldots, a_k)) = \sigma(\mathbb{C})$. Consider $(m(C, \mathcal{F}))^{C_r}_{1\leq i\leq k}$ that is obtained by transformation of the movements on the configuration C into equivalent movements in configuration C_r, as explained in Subsection 3.2. It is not difficult to see that $(m(C, \mathcal{F}))^{C_r}_{1\leq i\leq k}$ is v'-compatible. By Theorem 10, we get that $C \oplus (m(C, \mathcal{F}))_{1\leq i\leq k} \equiv C_r \oplus (m(C, \mathcal{F})^{C_r})_{1\leq i\leq k} = C'$. By definition of $\mathcal{A}_{\text{gather}}$, $\mathbb{C}' = [C']_{\equiv}$ is a successor state of v'. Hence, for all succession of configurations $C \xrightarrow{\mathcal{F}} C' \xrightarrow{\mathcal{F}} \ldots$ obtained by successive applications of the algorithm, we have a corresponding play $[C]_{\equiv}, v', [C']_{\equiv}, \ldots$ that is σ-compatible, in the game. Since $[C]_{\equiv}$ is a winning position, the play is winning and will eventually reach the tower position and so will the algorithm $\mathcal{F}$ we have designed. □

important

Appendix B: Generalization of the 3-gathering strategy for rings of any size n.

In this section we prove that the 3-gathering synthesized strategy is correct in the FSYNC model for any size n ($n \in \mathbb{N}$) except from periodic configurations.Let $C = [(x,y,z)]_{\equiv}$ a class of configurations such that $rep(C) = (x,y,z)$, and $d = x+y$ be the *separating-distance* or the class of configuration.

Definition 11. *A gathering configuration is a configuration $c \in \mathcal{C}$ such that $rep([c]_{\equiv}) = (-1,-1,n-1)$(all robots are positioned on the same node).*

Theorem 12. *Starting from any configuration (except periodic ones) the 3-gathering strategy eventually reaches a gathering configuration.*

Proof. The theorem is correct if any of the movements produce by the above strategy is decreasing the separating-distance. The proof directly follows from the Lemmas 13, 14, 15, 16 below. Each one of these lemmas addresses a possible initial class of configurations.

Lemma 13. *Starting from a class of configuration with one tower of size 2, the system executing the 3-gathering strategy eventually reaches a gathering configuration.*

Proof. **Base-case:** Consider the minimal separating distance. For any $z \in \mathbb{N}$, the execution starting from from $[(-1,0,z)]_{\equiv}$ leads to $[(-1,-1,z+1)]_{\equiv}$ which is a gathering class of configuration.

Induction: Assume that starting in a configuration where the separating-distance is i the system executing the strategy eventually reaches a gathering class of configuration. We prove in the following that the above also holds when the separating distance is $i+1$. From $[(-1,i,z)]_{\equiv}$ the execution of the strategy leads to $[(-1,i-1,z+1)]_{\equiv}$. The proof follows from the induction hypothesis.

Lemma 14. *Starting from a symmetrical class of configurations without tower, where $x = y$, the system executing the 3-gathering strategy eventually reaches a gathering configuration.*

Proof. **Base-case:** Consider the class of configurations where the separating distance is minimal : $[(0,0,z)]_{\equiv}$. For any $z \in \mathbb{N}$, the system executing the 3-gathering strategy starting in $[(0,0,z)]_{\equiv}$ reaches $[(-1,-1,z+2)]_{\equiv}$ (the gathering class of configurations).

Induction: Assume that when the separating-distance equals $2i$ a gathering configuration is eventually reached, and prove that it is also true when the separating distance is $2i+2$. From $[(i+1,i+1,z)]_{\equiv}$ the system executing the strategy eventually reaches $[(i,i,z+2)]_{\equiv}$. The lemma follows from the induction hypothesis.

Lemma 15. *Starting from a rigid configuration, the system executing the 3-gathering strategy eventually reaches a gathering configuration.*

In order to prove this lemma for the last edge we fix $x = 0$ and prove that for any y and any z the strategy is correct, and then we use this proof as a base case to prove the lemma for any x, y and z.

Proof. **Base-case2:** When $x = 0$, for any y and z, from $[(0,y,z)]_{\equiv}$ the algorithm leads to $[(-1,y-1,z+2)]_{\equiv}$ where the interdistance is decreased by two. Hence the lemma is true, thanks to lemma 13.

Induction2: We assume that the gathering is made for any y and any z for an $x = i$, and we show that it is also made when $x = i+1$. From $[(x+1,y,z)]_{\equiv}$ the algorithm leads to $[(x-1,y-1,z+2)]_{\equiv}$. Thanks to our induction hypothesis the lemma is true.

Lemma 16. *Starting from a symmetrical class of configurations without tower, where $y = z$, the system executing the 3-gathering strategy eventually reaches a gathering configuration.*

Proof. This lemma is correct since from any of these configurations we have $[(x,y,y)]_{\equiv}$ which leads to $[(x,y-1,y+1)]_{\equiv}$ where the separating distance is decreased. The obtained configuration is a symmetrical configuration if $x = y-1$ or a rigid configuration otherwise, thanks to Lemma 14 and Lemma 5 we know that from these configurations, a gathering configuration is eventually reached.