

# Synchronous Gathering Without Multiplicity Detection: A Certified Algorithm

Thibaut Balabonski<sup>3</sup>, Amélie Delga<sup>2,4</sup>, Lionel Rieg<sup>1</sup>, Sébastien Tixeul<sup>4,5</sup>,  
and Xavier Urbain<sup>2,3</sup>(✉)

<sup>1</sup> Collège de France, 75006 Paris, France

<sup>2</sup> École Nat. Sup. d'Informatique Pour l'Industrie et l'Entreprise (ENSIE),  
91025 Évry, France

<sup>3</sup> LRI, CNRS UMR 8623, Université Paris-Sud,  
Université Paris-Saclay, Orsay, France  
Xavier.Urbain@lri.fr

<sup>4</sup> UPMC Sorbonne Universités, LIP6-CNRS 7606, Paris, France

<sup>5</sup> Institut Universitaire de France, Paris, France

**Abstract.** In mobile robotic swarms, the gathering problem consists in coordinating all the robots so that in finite time they occupy the same location, not known beforehand. Multiplicity detection refers to the ability to detect that more than one robot can occupy a given position. When the robotic swarm operates synchronously, a well-known result by Cohen and Peleg permits to achieve gathering, provided robots are capable of multiplicity detection.

We present a new algorithm for synchronous gathering, that does *not* assume that robots are capable of multiplicity detection, nor make any other extra assumption. Unlike previous approaches, our proof correctness is certified in the model where the protocol is defined, using the Coq proof assistant.

## 1 Introduction

Networks of mobile robots have captured the attention of the distributed computing community, as they promise new applications (rescue, exploration, surveillance) in potentially dangerous (and harmful) environments. Since its initial presentation [19], this computing model has grown in popularity<sup>1</sup> and many refinements have been proposed (see [14] for a recent state of the art). From a theoretical point of view, the interest lies in characterising the exact conditions for solving a particular task.

*A computing model for mobile robots.* In the model we consider, robots operate in Look-Compute-Move cycles. In each cycle a robot “Looks” at its surroundings and obtains (in its own coordinate system) a snapshot containing some information about the locations of all robots. Based on this visual information, the

---

<sup>1</sup> The 2016 SIROCCO Prize for Innovation in Distributed Computing was awarded to Masafumi Yamashita for this line of work.

robot “Computes” a destination location (still in its own coordinate system) and then “Moves” towards the computed location. When the robots are oblivious, the computed destination in each cycle depends only on the snapshot obtained in the current cycle (and not on the past history of execution). The snapshots obtained by the robots are not necessarily consistently oriented in any manner.

The execution model significantly impacts the solvability of collaborative tasks. Three different levels of synchronisation have been considered. The strongest model [19] is the fully synchronised (FSYNC) model where each stage of each cycle is performed simultaneously by all robots. On the other hand, the asynchronous model [14] (ASYNC) allows arbitrary delays between the Look, Compute and Move stages and the movement itself may take an arbitrary amount of time, possibly a different amount for each robot. In the semi-synchronous (SSYNC) model [19], which lies somewhere between the two extreme models, time is discretised into rounds and in each round an arbitrary subset of the robots are active. The active robots in a round perform exactly one atomic Look-Compute-Move cycle in that round. It is assumed that the scheduler (seen as an adversary) is fair in the sense that it guarantees that in any configuration, any robot is activated within a finite number of steps.

Furthermore, the scheduler has the ability to stop a robot before it has completed its move, provided the robot has already moved by some positive distance  $\delta$ . Now, if a robot  $r$  wants to move by some distance  $d < \delta$ , once activated by the scheduler, the scheduler then cannot stop  $r$  until it completes its movement. The value of  $\delta$  is unknown to the robots, and is just meant to prevent the scheduler to make them move by infinitely small distances. These stoppable moves are referred to as *flexible* moves in the remainder of the paper.

*The gathering problem.* The gathering problem is one of the benchmarking tasks in mobile robot networks, and has received a considerable amount of attention (see [14] and references herein). The gathering task consists in making all robots (considered as dimensionless points in a two dimensional Euclidean space) reach a single point, not known beforehand, in finite time. A foundational result [19] shows that in the SSYNC model, no oblivious deterministic algorithm can solve gathering for two robots<sup>2</sup>. This result can be extended [11] to the bivalent case, that is, when an even number of robots is initially evenly split in exactly two locations. In general, without extra assumptions in the execution model (*e.g.* a common coordinate system, persistent memory, the ability to detect multiple robots at a given location, use of probabilistic variables, etc.), it is impossible to solve gathering [16] for any set of at least two robots in the SSYNC model. As all possible executions in SSYNC are also possible in ASYNC, those impossibilities also hold in ASYNC. Hence, the only possibility to solve gathering without extra assumptions is to consider the FSYNC model.

Cohen and Peleg [9] proposed the *center of gravity* (*a.k.a.* CoG) algorithm (the robots aim for the location that is the barycenter of all observed robot locations) for the purpose of convergence (a weaker requirement than

<sup>2</sup> <http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR.Impossibility.html>.

gathering, which mandates robots to reach locations that are arbitrarily close to one another) in the SSYNC model. They demonstrate that for the FSYNC model, robots actually solve gathering since they eventually all become closer than  $\delta$  from the barycenter, and hence all reach it in the next round.

However, the CoG algorithm does not prevent more than one robot to occupy the exact same location before gathering, even if they start from distinct locations. For example, consider two robots  $r_1$  and  $r_2$  aligned toward the barycenter at some round, at distances  $d_1$  and  $d_2$  ( $d_1 < d_2$ ) that are both greater than  $\delta$ , respectively. Then, the scheduler stops  $r_1$  after  $\delta$  and  $r_2$  at the same location. Robots  $r_1$  and  $r_2$  now occupy the same location. One immediate consequence of this observation is that in the next round, to compute the barycenter, observing robots must take into account both  $r_1$  and  $r_2$ . That is, using the CoG algorithm, robots must make use of *multiplicity detection*, *i.e.* be able to detect how many robots occupy simultaneously a given location.

Overall, the question of gathering feasibility in FSYNC without multiplicity detection (nor any other additional assumption) remained open.

*Formal methods for mobile robots.* Designing and proving mobile robot protocols is notoriously difficult. Formal methods encompass a long-lasting path of research that is meant to overcome errors of human origin. Not surprisingly, this mechanised approach to protocol correctness was successively used in the context of mobile robots [2, 3, 5, 6, 11, 13, 15, 17].

Model-checking proved useful to find bugs in existing literature [3] and assess formally published algorithms [3, 13, 17], in a simpler setting where robots evolve in a *discrete space* where the number of possible locations is finite. Automatic program synthesis (for the problem of perpetual exclusive exploration in a ring-shaped discrete space) is due to Bonnet *et al.* [5], and can be used to obtain automatically algorithms that are “correct-by-design”. The approach was refined by Millet *et al.* [15] for the problem of gathering in a discrete ring network. As all aforementioned approaches are designed for a discrete setting where both the number of locations and the number of robots are known, they cannot be used in the continuous space where the robots locations take values in a set that is not enumerable, and they cannot permit to establish results that are valid for any number of robots.

The use of a mechanical proof assistant like CoQ<sup>3</sup> allows for more genericity as this approach is not limited to *particular instances* of algorithms. Recent uses of CoQ in Distributed Computing include that of Castéran *et al.* [7], who use CoQ and their library Loco to prove positive and negative results about subclasses of LC systems, and that of Altisen *et al.* [1], who provide a CoQ framework to study self-stabilizing algorithms.

Developed for the CoQ proof assistant,<sup>4</sup> the Pactole<sup>5</sup> framework enabled the use of high-order logic to certify impossibility results [2] for the problem of

<sup>3</sup> <http://coq.inria.fr>.

<sup>4</sup> <http://coq.inria.fr>.

<sup>5</sup> Available at <http://pactole.lri.fr>.

convergence: for any positive  $\varepsilon$ , robots are required to reach locations that are at most  $\varepsilon$  apart. Another classical impossibility result that was certified using the Pactole framework is the impossibility of gathering starting from a bivalent configuration [11]. Recently, positive certified results for SSYNC gathering with multiplicity detection were provided by Courtieu *et al.* [12].

*Our contribution.* We propose a protocol for oblivious mobile robot gathering in FSYNC that does not require multiplicity detection (nor any other extra assumption). Our protocol, called CoGiL (for Center of Gravity of inhabited Locations), is derived from CoG as follows: robots aim to the barycenter of observed *occupied* locations (that is, without considering how many robots occupy a given location). We also present a proof of correctness for our CoGiL protocol.

Unlike previous approaches, our proof is *certified* in the model where the protocol is defined, using the Coq proof assistant. Throughout this paper, links to the COQ development are italicised in the footnotes. The sources package is available at <http://pactole.lri.fr>, as well as its online [html](#) documentation.

*Roadmap.* Section 2 describes our formal framework, while our case study is developed in Sect. 3. Section 4 gives some insights about the benefits of our methodology for mobile robot protocol design.

## 2 A Formal Model to Prove Robot Protocols

To certify results and to guarantee the soundness of theorems, we use COQ, a Curry-Howard-based interactive proof assistant enjoying a trustworthy kernel. The (functional) language of COQ is a very expressive  $\lambda$ -calculus: the *Calculus of Inductive Constructions* (CIC) [10]. In this context, datatypes, objects, algorithms, theorems and proofs can be expressed in a unified way, as terms.

The reader will find in [4] a very comprehensive overview and good practices with reference to COQ. Developing a proof in a proof assistant may nonetheless be tedious, or require expertise from the user. To make this task easier, we are actively developing (under the name Pactole) a formal model, as well as lemmas and theorems, to specify and certify results about networks of autonomous mobile robots. It is designed to be robust and flexible enough to express most of the variety of assumptions in robots network, for example with reference to the considered space: discrete or continuous, bounded or unbounded.

We do not expect the reader to be an expert in COQ but of course the specification of a model for mobile robots in COQ requires some knowledge of the proof assistant. We want to stress that the framework eases the developer's task. The notations and definitions we give hereafter should be simply read as typed functional expressions.

The Pactole model has been sketched in [2, 11]; we recall here its main characteristics.

We use two important features of COQ: a formalism of *higher-order* logic to quantify over programs, demons, etc., and the possibility to define *inductive*

and *coinductive* types [18] to express inductive and coinductive datatypes and properties. Coinductive types are in particular of invaluable help to express infinite behaviours, infinite datatypes and properties on them, as we shall see with demons.

Robots<sup>6</sup> are anonymous, however we need to identify some of them in the proofs. Thus, we consider given a finite set of *identifiers*, isomorphic to a segment of  $\mathbb{N}$ . We hereafter omit this set  $G$  unless it is necessary to characterise the number of robots. Robots are distributed in space, at places called *locations*. We call a *configuration*<sup>7</sup> a *function* from the set of identifiers to the space of locations.

From that definition, there is information about identifiers contained in configurations, notably, equality between configurations does *not* boil down to the equality of the multisets of inhabited locations.

Under the assumption that robots are anonymous and indistinguishable, we have to make sure that the embedded algorithm does not make use of those identifiers.

*Spectrum*.<sup>8</sup> The computation of any robot's target location is based on the perception they get from their environment, that is, in an FSYNC execution scheme, from a configuration. The result of this observation may be more or less accurate, depending on sensors' capabilities. A robot's perception of a configuration is called a *spectrum*. To allow for different assumptions to be studied, we leave abstract the type *spectrum* (`Spect.t`) and the notion of spectrum of a location. *Robograms*, representing protocols, will then output a location when given a spectrum (instead of a configuration), thus guaranteeing that assumptions over sensors are fulfilled. For instance, the spectrum for anonymous robots with *strong* global multiplicity detection (this capacity refers to the ability to count exactly how many robots occupy any observed location) could be the multiset of inhabited locations. In a setting where robots do not enjoy the detection of multiplicity and just know if a location is inhabited or not, the *set* of inhabited locations is a suitable spectrum.

In the following we will distinguish a *demon* configuration (resp. spectrum), expressed in the global frame of reference, from a *robot* configuration (resp. spectrum), expressed in the robot's own frame of reference. At each step of the distributed protocol the demon configuration and spectrum are transformed (recentered, mirrored, rotated, and scaled) into the considered robots ones before being given as parameters to the robogram. Depending on assumptions, zoom and rotation factors may be constant or chosen by the demon at each step, shared by all robots or not, etc.

*Demon for flexible movements.* As moves under consideration are *flexible*, robots either reach their goal when it is at most at a certain absolute distance  $\delta$ , or travel at least  $\delta$  towards their goal, stopping to an arbitrary location (possibly the computed goal).

<sup>6</sup> <http://pactole.lri.fr/pub/cffg2d/html/Pactole.Robots.html#Robots>.

<sup>7</sup> <http://pactole.lri.fr/pub/cffg2d/html/Pactole.Configurations.html#Configuration>.

<sup>8</sup> <http://pactole.lri.fr/pub/cffg2d/html/Pactole.Configurations.html#Spectrum>.

Rounds<sup>9</sup> in this FSYNC setting are thus characterised by each of the oblivious robots getting both its new frame of reference, and the ratio of its actual movement over its computed destination.

We call *demonic action* this operation together with the logical properties ensuring, for example, that new frames of reference make sense, and that the provided ratio belongs to the  $[0, 1]$  interval. *Demons* are streams of demonic actions. As such, they are naturally defined in COQ as a coinductive construct. Synchrony constraints (e.g. fairness) may be defined as coinductive properties on demons, as detailed in [2, 11].

The Pactole framework provides theorems that state the equivalence between rigid movements models and flexible models when the ratio of actual movement is always 1. Developments in a rigid context may thus be written free of cumbersome irrelevant details.

*Robogram.* Robograms<sup>10</sup> may be naturally defined in a *completely abstract manner*, without any concrete code, in our COQ model. They consist of an actual algorithm `pgm` that represents the considered protocol and that takes a spectrum as input and returns a location, and a compatibility property `pgm_compat` stating that target locations are the same if equivalent spectra are given (for some equivalence on spectra).

```
Record robogram :=
  {pgm :> Spect.t → Location.t;
   pgm_compat : Proper (Spect.eq ⇒ Location.eq) pgm}.
```

*Execution of a round.* The actual location of arrival for a robot is determined by the protocol, which computes a local target from the perceived spectrum, and the demon-provided ratio which is applied to the local target to obtain a chosen target. If the distance between the robot's original location and its chosen target is more than  $\delta$  then the robot stops at the chosen target, otherwise it reaches its protocol-computed destination (local target). This concise way of proceeding ensures that either the protocol-computed destination is reached or at least  $\delta$  is travelled.

### 3 Center of Gravity Algorithms

*Notations.* In the sequel, we denote by:  $C$  a configuration,  $C(r)$  the location of Robot  $r$  in Configuration  $C$ , and  $S_C$  the global spectrum associated to  $C$ .

<sup>9</sup> <http://pactole.lri.fr/pub/cffg2d/html/Pactole.FlexibleFormalism.html>.

<sup>10</sup> <http://pactole.lri.fr/pub/cffg2d/html/Pactole.CommonFormalism.html#Sig.robogram>.

### 3.1 Center of Gravity Algorithms Variants

Cohen and Peleg [8,9] define the CoG algorithm as depicted in Algorithm 1. A robot simply moves toward the center of gravity of all robots locations. Since robots may occupy the same location in space, the proper calculation of the center of gravity implies that the robots are capable of strong global multiplicity detection: for each inhabited location, the robots can count the number of robots on that location.

---

**Algorithm 1.** Protocol CoG (for Robot  $r$  in Configuration  $C$ )

---

Move toward the centre of gravity of robot locations  $c_{pos} = \frac{1}{|C|} \times \sum_{r \in C} C(r)$

---

We define the CoGiL algorithm in Algorithm 2. Here, we do not assume that robots are capable of multiplicity detection, so robots simply move toward the center of gravity of inhabited locations. Note that the number of those inhabited locations is not necessarily monotonically decreasing.

---

**Algorithm 2.** Protocol CoGiL (for Robot  $r$  in Configuration  $C$ )

---

Move toward the centre of gravity of inhabited locations  $c_{pos} = \frac{1}{|S_C|} \times \sum_{p \in S_C} p$

---

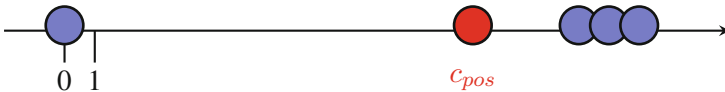
Although CoGiL is extremely similar to CoG, proving its correctness is not. For example, Cohen and Peleg [8] first used in the conference version of their paper moments of inertia as a monotonically decreasing measure to prove the convergence of CoG:

$$I(q) = \frac{1}{|C|} \times \sum_{r \in C} \|C(r) - q\|^2$$

Expressing this measure with the observed spectrum gives:

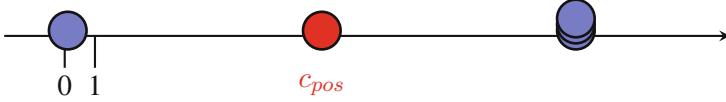
$$I(q) = \frac{1}{|S_C|} \times \sum_{p \in S_C} \|p - q\|^2$$

Now, without strong global multiplicity detection, it is possible that this measure is *not* monotonically decreasing for  $c_{pos}$ . For example, consider four robots in a one-dimension metric space, localised at locations 0; 17; 18; 19.



The center of gravity of the inhabited locations  $c_{pos}$  is at 13.5 and  $I(c_{pos}) = 61.25$ . Now, consider that  $\delta = 0.1$ . A possible following configuration is that the

robot on 0 has moved by  $\delta$  toward  $c_{pos}$  and the others have stopped at location 16.9.



The center of gravity of the inhabited locations  $c_{pos}$  is now at 8.5, and  $I(c_{pos}) = 70.56$ , which is strictly greater than its previous value. So, the proof argument appearing in Cohen and Peleg's conference paper [8] does not extend to the case without global strong multiplicity detection.

Fortunately, the underlying idea of the proof appearing in the journal version of Cohen and Peleg [9] can be extended to the case without multiplicity detection. We thus construct our certified proof along the main arguments of theirs.

### 3.2 Formalisation, and Key Points to Prove Correctness

*Gathering in the context of flexible movements.* A way to state Gathering and Convergence has been already described in [2, 11]. Those definitions take place in a context where movements are rigid, and thus the specification of what a solution to Gathering is has to be generalised for the case of flexible movements. We name `gathered_at pt` the property of a configuration the robots of which are all gathered at<sup>11</sup> the same location  $pt$ . We say that a location  $pt$  and an execution enjoy the property `Gather` if all robots are gathered at  $pt$  for all rounds of the (infinite) execution.

**Definition** `Gather (pt : Loc.t) (e : execution) : Prop := Streams.forever (Streams.instant (gathered_at pt)) e.`

`WillGather pt e` means that the (infinite) execution  $e$  is *eventually* `Gathered` for  $pt$ . That is: there is a (finitely) reachable instant in  $e$  for which  $pt$  and what remains of  $e$  fulfils `Gather`.

**Definition** `WillGather (pt : Loc.t) (e : execution) : Prop := Streams.eventually (Gather pt) e.`

We may now characterise that a robogram  $r$  is a solution to the Gathering problem for a demon  $d$ , in the context of  $\delta$ -flexible movements.<sup>12</sup> It takes into account the minimal distance of travel  $\delta$  that is necessary to define the execution.

**Definition** `FullSolGathering (r : robogram) (d : demon)  $\delta$  :=  $\forall$  config,  $\exists$  pt : Loc.t, WillGather pt (execute  $\delta$  r d config).`

<sup>11</sup> [http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.FlexDefinitions.html#FlexGatheringDefs.gathered\\_at](http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.FlexDefinitions.html#FlexGatheringDefs.gathered_at).

<sup>12</sup> <http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.FlexDefinitions.html#FlexGatheringDefs.FullSolGathering>.



*Expressing the protocol in Pactole.* The space of locations is  $\mathbb{R}^2$  and its type is `R2.t` in the following. Writing the algorithm is straightforward in our framework, and the CoQ implementation is almost exactly an actual robot code. Let `ffgatherR2_pgm` denote the code of the algorithm<sup>13</sup>, which takes a spectrum as an input and returns a location, and let `ffgatherR2` denote the robogram, that is the code and its properties (invariance through equivalent spectra).

```
Definition ffgatherR2_pgm (s : Spect.t) : R2.t :=
  let spect := Spect.M.elements s in
    match spect with
      | nil  $\Rightarrow$  (0, 0)                                (* no robot *)
      | pt :: nil  $\Rightarrow$  pt                             (* gathered *)
      | _ :: _ :: _  $\Rightarrow$  barycenter spect
    end.
```

The function computing the barycenter<sup>14</sup> is simply:

```
Definition barycenter (E: list R2.t) : R2.t :=
  1 / (INR (List.length E)) * (List.fold_left R2.add E R2.origin).
```

where `INR` injects a natural number into reals.

*The robogram can be expressed in the demon's frame of reference.* The input spectrum given to the code above is expressed in the robot's frame of reference (it is a local code). As noticed in [12], we establish explicitly and formally that it is sufficient to reason about the protocol in the frame of reference of the demon. The geometrical concepts in use in the protocol are invariant under the changes of frame that are allowed: scaling, rotation, symmetry and translation, hence we can express the global configuration after one round without making reference to the frames of each robot<sup>15</sup> (Lemma `round_simplify`).

*Eventually no-one moves.* The main difficulty is to establish that after a finite number of steps, no robot will change its location. This amounts to finding a measure that decreases for a well founded ordering along with the execution.

To this goal, we consider the maximal distance<sup>16</sup>  $dm(C)$  between any two robots in a configuration  $C$ .

```
Definition measure (conf: Config.t) :  $\mathbb{R}$  :=
  max_dist_spect (spectrum_of conf).
```

<sup>13</sup> [http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.FSyncFlexNoMultAlgorithm.html#GatheringinR2.ffgatherR2\\_pgm](http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.FSyncFlexNoMultAlgorithm.html#GatheringinR2.ffgatherR2_pgm).

<sup>14</sup> <http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.R2geometry.html#barycenter>.

<sup>15</sup> [http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.FSyncFlexNoMultAlgorithm.html#GatheringinR2.round\\_simplify](http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.FSyncFlexNoMultAlgorithm.html#GatheringinR2.round_simplify).

<sup>16</sup> [http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.FSyncFlexNoMultAlgorithm.html#GatheringinR2.max\\_dist\\_spect\\_ex](http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.FSyncFlexNoMultAlgorithm.html#GatheringinR2.max_dist_spect_ex).

If this distance is less than  $\delta$  then after one step all robots are gathered and we are done (Theorem [round\\_last\\_step](#)<sup>17</sup>). If not, we prove that if a configuration  $C_1$  is obtained after one round from a configuration  $C_0$  such that  $dm(C_0) > \delta$ , then  $dm(C_1) \leq dm(C_0) - \delta$ . This part is established through Theorem [round\\_lt\\_config](#):<sup>18</sup>

**Theorem** [round\\_lt\\_config](#):  $\forall d \text{ conf } \delta, \delta > 0$   
 $\rightarrow \text{FullySynchronous } d$   
 $\rightarrow \delta \leq \text{measure conf}$   
 $\rightarrow \text{measure (round } \delta \text{ ffgatherR2 (head } d) \text{ conf)} \leq \text{measure conf} - \delta$ .

We may then take as a relevant indication for a configuration  $C$  the natural number  $m(C) = \lceil \frac{dm(C)}{\delta} \rceil$  and define accordingly the ordering we use:

**Definition** [lt\\_config](#)  $\delta \times y :=$   
 $(\mathbb{Z}.\text{to\_nat } (\text{up}(\text{measure } x / \delta))) < (\mathbb{Z}.\text{to\_nat } (\text{up}(\text{measure } y / \delta)))$ .

which is well-founded over the naturals.

The crucial step is to prove that for any two inhabited locations  $p_1$  and  $q_1$  in  $C_1$ ,  $\|p_1 - q_1\| \leq dm(C_0) - \delta$ . Let us denote by  $\mathbf{b}$  the location of the barycenter of inhabited locations in  $C_0$ . As locations  $p_1$  and  $q_1$  are inhabited in  $C_1$ , we can assume that some robots  $P$  and  $Q$  occupying them in  $C_1$  were previously in  $C_0$  at respectively  $p_0$  and  $q_0$ . Now let us perform a case analysis on whether  $\|p_0 - \mathbf{b}\|$  and  $\|q_0 - \mathbf{b}\|$  are greater or equal to  $\delta$ ; the only interesting case is the non-degenerate one where both are greater. In this case,  $P$  and  $Q$  move towards  $\mathbf{b}$ , and in particular  $p_1 = p_0 + \kappa \times (\mathbf{b} - p_0)$  and  $q_1 = q_0 + \mu \times (\mathbf{b} - q_0)$  for  $\kappa, \mu \in [0, 1]$ . Let us suppose  $\kappa \leq \mu$  (the other case is symmetrical), then  $\|p_1 - q_1\| = \|(p_0 + \kappa \times (\mathbf{b} - p_0)) - (q_0 + \mu \times (\mathbf{b} - q_0))\| \leq (1 - \kappa) \times dm(C_0) \leq dm(C_0) - \delta$  by Thales's Basic Proportionality Theorem and since the distance from any robot to  $\mathbf{b}$ , the barycenter of locations, is less than or equal to  $dm(C_0)$ .

This argument can be trusted it as it is formally certified in our mechanical framework.

*Robots stay gathered forever.* As there is only one phase in the algorithm, the computed target is always the barycenter of the inhabited locations, which is the same for all robots. We need however technical lemmas to complete the final proof. Firstly that when robots are gathered, they will stay forever at the same location, namely:<sup>19</sup>

**Lemma** [gathered\\_at\\_OK](#) :  $\forall \delta d \text{ conf pt, gathered\_at pt conf}$   
 $\rightarrow \text{Gather pt (execute } \delta \text{ ffgatherR2 } d \text{ conf)}$ .

<sup>17</sup> [http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.FSyncFlexNoMultAlgorithm.html#GatheringinR2.round\\_last\\_step](http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.FSyncFlexNoMultAlgorithm.html#GatheringinR2.round_last_step).

<sup>18</sup> [http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.FSyncFlexNoMultAlgorithm.html#GatheringinR2.round\\_lt\\_config](http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.FSyncFlexNoMultAlgorithm.html#GatheringinR2.round_lt_config).

<sup>19</sup> [http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.FSyncFlexNoMultAlgorithm.html#GatheringinR2.gathered\\_at\\_OK](http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.FSyncFlexNoMultAlgorithm.html#GatheringinR2.gathered_at_OK).

The counterpart is that a robot that is not at the barycenter of inhabited locations will actually move (that is, it will change its location).<sup>20</sup>

**Lemma** `not_barycenter_moves`:  $\forall \delta \, d \, \text{conf} \, \text{gid}, \delta > 0$   
 $\rightarrow \text{FullySynchronous } d$   
 $\rightarrow \neg \text{R2.eq } (\text{conf } \text{gid}) \, (\text{barycenter } (\text{Spect.M.elements } (!! \, \text{conf})))$   
 $\rightarrow \neg \text{R2.eq } (\text{round } \delta \, \text{ffgatherR2 } (\text{Streams.hd } d) \, \text{conf } \text{gid})$   
 $(\text{conf } \text{gid})$ .

We are now ready to tackle the final proof.

The final theorem<sup>21</sup> states that for all positive  $\delta$ , the robogram `ffgatherR2` is a solution to the gathering problem in FSYNC.

**Theorem** `FSGathering_in_R2` :  $\forall \delta \, d, \delta > 0$   
 $\rightarrow \text{FullySynchronous } d$   
 $\rightarrow \text{FullSolGathering } \text{ffgatherR2 } d \, \delta$ .

It is proven via well-founded induction over `lt_config` and by case analysis: if the robots are already gathered or will be gathered at the next step then we are done, else we use `round_lt_config`. That last proof is about 30 lines of COQ.

## 4 Discussion and Perspectives

We presented the first FSYNC gathering protocol, CoGiL, that does not require robots to be capable of multiplicity detection (nor any other extra assumptions), closing the only remaining open case in Prencipe’s set of impossibility results [16]. We advocate that proofs for even small variants of oblivious mobile robot protocols (such a CoGiL, which is a minor variant of Cohen and Peleg’s CoG protocol) should be thoroughly checked from the beginning, using mechanised support such as a proof assistant. This methodology enabled the possibility to present a proof for our protocol, whose correctness can be certified.

We want to stress that, even if the actual development of a formal proof remains a difficult task, the *specifications* of properties and protocols in our framework do not require a strong expertise with the COQ proof assistant. As an illustration, many of the specifications appearing in this paper, most notably the specification of the actual protocol, were developed by one of the authors while a M1-level trainee (first year master, Bologna process).

We believe a thorough revision of other results in the context of oblivious mobile robots will lay a solid foundation for further research advances. Thanks to the collaborative effort of the Pactole framework, reuse of previous achievements is facilitated and encouraged.

**Acknowledgements.** The authors are grateful to the reviewers who provided constructive comments and helped to improve the presentation of this work.

<sup>20</sup> [http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.FSyncFlexNoMultAlgorithm.html#GatheringinR2.not\\_barycenter\\_moves](http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.FSyncFlexNoMultAlgorithm.html#GatheringinR2.not_barycenter_moves).

<sup>21</sup> [http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.FSyncFlexNoMultAlgorithm.html#GatheringinR2.FSGathering\\_in\\_R2](http://pactole.lri.fr/pub/cffg2d/html/Pactole.Gathering.InR2.FSyncFlexNoMultAlgorithm.html#GatheringinR2.FSGathering_in_R2).

## A Axioms of the Formalisation

In the main file `FSyncFlexNoMultAlgorithm.v`, the last command: `Print Assumptions Gathering_in_R2` shows all the axioms upon which the proof of correctness of our algorithm for gathering in  $\mathbb{R}^2$  relies, in total 31 axioms. Here, we break them down. They can be classified in three categories:

- The first category is the axiomatisation of reals numbers from the COQ standard library. It is by far the biggest number of axioms, and they are not listed here.
- The second category is the description of the problem.

```
nG : nat
Hyp_nG : 2 ≤ nG
```

As one can see, it simply means that our proof is valid for any number `nG` of robots greater than or equal to 2. Notice that with one robot or less, the problem is not interesting (trivially solved).

- The third category contains usual geometric properties that are not part of our library: firstly some properties about barycenters that we think could be provable from its axiomatisation but are currently left as axioms, that the barycenter is unique and the result of the function computing the barycenter is indeed a barycenter:

```
barycenter_n_unique : ∀ (E : list R2.t) (a b : R2.t),
  is_barycenter_n E a → is_barycenter_n E b → R2.eq a b

barycenter_n_spec : ∀ E : list R2.t,
  is_barycenter_n E (barycenter E)
```

Finally that similarities can be expressed with an orthogonal matrix, a zoom factor and a translation. The orthogonal matrix and the scaling factor are combined into two column vectors `u` and `v`.

```
similarity_in_R2 : ∀ sim : Sim.t, ∃ u v t : R2.t,
  R2norm u = Sim.zoom sim
  ∧ R2norm v = Sim.zoom sim ∧ perpendicular u v ∧ (∀ pt :
  R2.t,
    sim pt = (product u pt * u + product v pt * v + t)  $\mathbb{R}^2$ )
```

## References

1. Altisen, K., Corbineau, P., Devismes, S.: A framework for certified self-stabilization. In: Albert, E., Lanese, I. (eds.) FORTE 2016. LNCS, vol. 9688, pp. 36–51. Springer, Heidelberg (2016). doi:[10.1007/978-3-319-39570-8\\_3](https://doi.org/10.1007/978-3-319-39570-8_3)
2. Auger, C., Bouzid, Z., Courtieu, P., Tixeuil, S., Urbain, X.: Certified impossibility results for byzantine-tolerant mobile robots. In: Higashino, T., Katayama, Y., Masuzawa, T., Potop-Butucaru, M., Yamashita, M. (eds.) SSS 2013. LNCS, vol. 8255, pp. 178–190. Springer, Heidelberg (2013). doi:[10.1007/978-3-319-03089-0\\_13](https://doi.org/10.1007/978-3-319-03089-0_13)

3. Bérard, B., Lafourcade, P., Millet, L., Potop-Butucaru, M., Thierry-Mieg, Y., Tixeuil, S.: Formal verification of mobile robot protocols. *Distributed Computing* (2016)
4. Bertot, Y., Castéran, P.: *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. Springer (2004)
5. Bonnet, F., Défago, X., Petit, F., Potop-Butucaru, M., Tixeuil, S.: Discovering and assessing fine-grained metrics in robot networks protocols. In *33rd IEEE International Symposium on Reliable Distributed Systems Workshops, SRDS Workshopps*, Nara, Japan, 6–9 October, pp. 50–59. IEEE (2014)
6. Bérard, B., Courtieu, P., Millet, L., Potop-Butucaru, M., Rieg, L., Sznajder, N., Tixeuil, S., Urbain, X.: Formal methods for mobile robots: current results and open problems. *Int. J. Inf. Soc.* **7**(3), 101–114 (2015). Invited Paper
7. Castéran, P., Filou, V.: Tasks, types and tactics for local computation systems. *Stud. Inform. Univ.* **9**(1), 39–86 (2011)
8. Cohen, R., Peleg, D.: Robot convergence via center-of-gravity algorithms. In: Kráľovič, R., Sýkora, O. (eds.) *SIROCCO 2004*. LNCS, vol. 3104, pp. 79–88. Springer, Heidelberg (2004). doi:[10.1007/978-3-540-27796-5\\_8](https://doi.org/10.1007/978-3-540-27796-5_8)
9. Cohen, R., Peleg, D.: Convergence properties of the gravitational algorithm in asynchronous robot systems. *siam j. comput.* **34**(6), 1516–1528 (2005)
10. Coquand, T., Paulin, C.: Inductively defined types. In: Martin-Löf, P., Mints, G. (eds.) *COLOG 1988*. LNCS, vol. 417, pp. 50–66. Springer, Heidelberg (1990). doi:[10.1007/3-540-52335-9\\_47](https://doi.org/10.1007/3-540-52335-9_47)
11. Courtieu, P., Rieg, L., Tixeuil, S., Urbain, X.: Impossibility of gathering, a certification. *Inf. Process. Lett.* **115**, 447–452 (2015)
12. Courtieu, P., Rieg, L., Tixeuil, S., Urbain, X.: Certified universal gathering in  $\mathbb{R}^2$  for oblivious mobile robots. In: Gavoille, C., Ilcinkas, D. (eds.) *DISC 2016*. LNCS, vol. 9888, pp. 187–200. Springer, Heidelberg (2016). doi:[10.1007/978-3-662-53426-7\\_14](https://doi.org/10.1007/978-3-662-53426-7_14)
13. Devismes, S., Lamani, A., Petit, F., Raymond, P., Tixeuil, S.: Optimal grid exploration by asynchronous oblivious robots. In: Richa, A.W., Scheideler, C. (eds.) *SSS 2012*. LNCS, vol. 7596, pp. 64–76. Springer, Heidelberg (2012). doi:[10.1007/978-3-642-33536-5\\_7](https://doi.org/10.1007/978-3-642-33536-5_7)
14. Flocchini, P., Prencipe, G., Santoro, N.: *Distributed Computing by Oblivious Mobile Robots. Synthesis Lectures on Distributed Computing Theory*. Morgan & Claypool Publishers (2012)
15. Millet, L., Potop-Butucaru, M., Sznajder, N., Tixeuil, S.: On the synthesis of mobile robots algorithms: the case of ring gathering. In: Felber, P., Garg, V. (eds.) *SSS 2014*. LNCS, vol. 8756, pp. 237–251. Springer, Heidelberg (2014). doi:[10.1007/978-3-319-11764-5\\_17](https://doi.org/10.1007/978-3-319-11764-5_17)
16. Prencipe, G.: Impossibility of gathering by a set of autonomous mobile robots. *Theoret. Comput. Sci.* **384**(2–3), 222–231 (2007)
17. Aminof, B., Murano, A., Rubin, S., Zuleger, F.: Verification of asynchronous mobile-robots in partially-known environments. In: Chen, Q., Torroni, P., Villata, S., Hsu, J., Omicini, A. (eds.) *PRIMA 2015*. LNCS (LNAI), vol. 9387, pp. 185–200. Springer, Heidelberg (2015). doi:[10.1007/978-3-319-25524-8\\_12](https://doi.org/10.1007/978-3-319-25524-8_12)
18. Sangiorgi, D.: *Introduction to Bisimulation and Coinduction*. Cambridge University Press (2012)
19. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: formation of geometric patterns. *SIAM J. Comput.* **28**(4), 1347–1363 (1999)

Stabilization, Safety, and Security of Distributed  
Systems

18th International Symposium, SSS 2016, Lyon, France,  
November 7-10, 2016, Proceedings  
Bonakdarpour, B.; Petit, F. (Eds.)  
2016, XIII, 432 p. 72 illus., Softcover  
ISBN: 978-3-319-49258-2