Certified Universal Gathering in $\mathbb{R}^2$ for Oblivious Mobile Robots*

Pierre Courtieu,[1] Lionel Rieg,[2] Sébastien Tixeuil,[5,6] Xavier Urbain[3,4,7]

[1] CÉDRIC – Conservatoire national des arts et métiers, Paris, F-75141
[2] Collège de France, Paris, F-75006
[3] ENSIIE, Évry, F-91025
[4] LRI, CNRS UMR 8623, Université Paris-Sud, Université Paris-Saclay, Orsay, F-91405
[5] UPMC Sorbonne Universités, LIP6-CNRS 7606
[6] Institut Universitaire de France
[7] Contact author: Xavier.Urbain@lri.fr

Abstract. We present a unified formal framework for expressing mobile robots models, protocols, and proofs, and devise a protocol design/proof methodology dedicated to mobile robots that takes advantage of this formal framework.
As a case study, we present the first formally certified protocol for oblivious mobile robots evolving in a two-dimensional Euclidean space. In more details, we provide a new algorithm for the problem of universal gathering mobile oblivious robots (that is, starting from any initial configuration that is not bivalent, using any number of robots, the robots reach in a finite number of steps the same position, not known beforehand) without relying on a common orientation nor chirality. We give very strong guaranties on the correctness of our algorithm by *proving formally* that it is correct, using the COQ proof assistant.
This result demonstrates both the effectiveness of the approach to obtain new algorithms that use as few assumptions as necessary, and its manageability since the amount of developed code remains human readable.

1 Introduction

Networks of mobile robots 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 [20], this computing model has grown in popularity[1] 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.

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

* A preliminary version of this work appears as a 3-page-long Brief Announcement in PODC'16.
[1] The 2016 SIROCCO Prize for Innovation in Distributed Computing was awarded to Masafumi Yamashita for this line of work.

system) a snapshot containing some information about the locations of all robots. Based on this visual information, the 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 [20] 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 this paper, we consider the semi-synchronous (SSYNC) model [20], which lies somewhere between the two extreme models. In the SSYNC model, 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.

Designing and proving mobile robot protocols is notoriously difficult. The diversity of model variants makes it extremely onerous to check whether a particular property of a robot protocol holds in a particular setting. Even worse, checking whether a property that holds in a particular setting also holds in another setting that is not strictly contained in the first one often requires a completely new proof, even if the proof argument is very similar. The lack of proof reusability between model variants is a major problem for investigating the viability of new solutions or implementations of existing protocols (that are likely to execute in a more concrete execution model). Also, oblivious mobile robot protocols are mostly based on observing geometric constructions and deriving invariants from those observations. As the protocols are typically written in an informal high level language, assessing whether they conform to a particular model setting is particularly cumbersome, and may lead to hard to find mismatches. Hence, solely relying on handcrafted protocols, models and proofs is likely to introduce subtle errors that eventually lead to catastrophic failures when the system is actually deployed. 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], [8], [11,12], [18].

Related Work. Model-checking proved useful to find bugs in existing literature [3] and assess formally published algorithms [12],[3], in a simpler setting where robots evolve in a *discrete space* where the number of possible positions 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.* [18] 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 positions and the number of robots are known, they cannot be used in the continuous space where robots positions take values in a set that is not enumerable, and they cannot permit to establish results that are valid for any number of robots.

Developed for the COQ proof assistant,[2] the Pactole[3] framework enabled the use of high-order logic to certify impossibility results [2] for the problem of convergence: for any positive ε, robots are required to reach locations that are at most ε apart. Another classical impossibility result that was certified using the Pactole framework is the impossibility of gathering starting from a bivalent configuration [11]. While the proof assistant approach seems a sensible path for establishing certified results for mobile robots that evolve in a continuous space, until this paper there exists no *positive* certified result in this context. Expressing mobile robot protocols in a formal framework that permits certification poses a double challenge: how to express the protocol (which can make use of complex geometric abstractions that must be properly defined within the framework), and how to write the proof?

Our contribution. Our first contribution is a unified formal framework for expressing mobile robots models, protocols, and proofs. This framework is motivated by the fact that many of the observed errors in published papers come from a mismatch between the advertised model and the model that is actually used for writing the proofs. For example, some dining philosophers protocols were expressed and proved in a high-level atomicity model, but advertised as working in a lower-level atomicity model, revealed to be incorrect in the lower-level atomicity model (see the work of Adamek *et al.* [1] and references herein). Sometimes, the mismatch between the proof and the advertised model is more subtle: a perpetual exclusive exploration protocol the proof of which did not consider all possible behaviours in the advertised model ASYNC was used to exhibit a counter example in such a setting (See the work of Berard *et al.* [3] and references therein). A unified formalisation whose consistency can be mechanically assessed is a huge asset for designing correct solutions, whose correctness can be certified. As we

[2] http://coq.inria.fr
[3] Available at http://pactole.lri.fr

used a subset of the same framework for certifying impossibility results [2],[11], consistency between negative and positive results is also guaranteed.

Our second contribution is a protocol design/proof methodology dedicated to mobile robots. We advocate the joint development of both the mobile robot protocol and its correctness proof, by taking advantage of the COQ proof assistant features. The proof assistant is typically able to check whether the proof of a particular theorem/lemma/corollary is valid. So replacing particular clauses of those theorems/lemmas/corollaries statements makes the proof assistant check whether the proof still is acceptable for the new statement. We used this feature to lift a preliminary version of this paper (uni-dimensional setting [10]) to a Euclidean bi-dimensional space: the proof assistant checked which arguments were still valid in the new setting. This feature also proved useful when slightly changing parts of the algorithm: the impact of the changes on the proofs were immediate. Also, it becomes easy to remove or weaken hypotheses from the protocol, as the proof assistant makes it obvious if they are not used in the proof arguments. Finally, our methodology includes a formal way to guarantee whether the "global" view of the system (as seen from the protocol prover point of view) is effectively realisable given the hypotheses assumed in the model.

We instantiate our framework and methodology to actually design and prove correct a new protocol for oblivious mobile robot universal gathering problem. The mobile robot gathering problem is a benchmarking problem in this context and can be informally defined as follows: robots have to reach in a finite number of steps the same location, not known beforehand. In more details, we present a new gathering algorithm for robots operating in a continuous space that *(i)* can start from any configuration that is not bivalent (that is, the robots are not initially equally placed in exactly two locations, since gathering is impossible in this case), *(ii)* does not put restriction on the number of robots, *(iii)* does not assume that robots share a common chirality (no common notion of "left" and "right"). To our knowledge, this is the first certified positive (and constructive) result in the context of oblivious mobile robots. It demonstrates both the effectiveness of the approach to obtain new algorithms that are truly generic facilitating the possibility to get rid of unnecessary assumptions, and its manageability since the amount of developed code remains human readable. Our bottom-up approach permits to lay sound theoretical foundations for future developments in this domain.

Throughout this paper, links to the COQ development are denoted by a ⇝ symbol in the margin. 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 Section 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 λ-calculus: the *Calculus of Inductive Constructions* (CIC) [9]. 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 [19] 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 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 $\mathtt{G}$ unless it is necessary to characterise the number of robots. Robots are distributed in space, at places called *locations*. We call a *configuration* 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.

Now if we are under the assumption that robots are anonymous and indistinguishable, we have to make sure that those identifiers are not used by the embedded algorithm.

- *Spectrum.* The computation of any robot's target location is based on the perception they get from their environment, that is, in an SSYNC 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 position. *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 *weak* global multiplicity detection could be the set of inhabited locations, i.e., without any multiplicity information. In a *strong* global multiplicity setting, the multiset 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 (see definition of `round` below) the demon configuration and spectrum are transformed (recentered, 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.* Rounds in this SSYNC setting are characterised with set of oblivious robots receiving their new frame of reference, if activated. We call *demonic action* this operation together with the logical properties ensuring, for example, that new frames of reference make sense. *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].

- *Robogram.* Robograms 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}.
```

3 Case study: A Universal Gathering for Mobile Oblivious Robots

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 tasks consists in all robots (considered as dimensionless points in a Euclidean space) reaching a single point, not known beforehand, in finite time. A foundational result [20] shows that in the FSYNC or SSYNC models, no oblivious deterministic algorithm can solve gathering for two robots without additional assumptions [17]. 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. On the other hand, it is possible to solve gathering if $n > 2$ robots start from initially distinct positions, provided robots are endowed with multiplicity detection: that is, a robot is able to determine the number of robots that occupy a given position.

While probabilistic solutions [20],[16] can cope with arbitrary initial configuration (including bivalent ones), most of the deterministic ones in the literature [14] assume robots always start from distinct locations (that is, the initial configuration contains no multiplicity points). Some recent work was devoted to relaxing this hypothesis in the deterministic case. Dieudonné and Petit [13] investigated the problem of gathering from *any* configuration (that is, the initial configuration can contain arbitrary multiplicity points): assuming that the number of robots is odd (so, no initial bivalent configuration can exist), they provide a deterministic algorithm for gathering starting from any configuration. Bouzid *et al.* [6] improved the result by also allowing an even number of robots to start from configurations that contain multiplicity points (albeit the initial bivalent configuration is still forbidden due to impossibility results in this case). In that sense, the algorithm of Bouzid *et al.* [6] is *universal* in the sense that it works for all gatherable configurations, including those with multiplicity points. The assumption that robots have a common chirality was removed in a context where robots may fail-stop in an unexpected manner [7].

A general description on how to characterise a solution to the problem of gathering has been given in [11]. We specialise this definition here to take into account that an initial configuration is not bivalent. This is straightforward: any robogram r is a solution w.r.t. a demon d if for every configuration *cf* that is not bivalent (that is $\neg$ `forbidden`), there is a point pt to which all robots will eventually gather (and stay) in the execution defined by r and d, and starting from *cf*.

We present a new gathering algorithm for robots operating in a continuous space that *(i)* can start from any configuration that is not bivalent, *(ii)* does not put restriction on the number of robots, *(iii)* does not assume that robots share

a common chirality. We give very strong guarantees on the correctness of our algorithm by *proving formally* that it is correct, using the Coq proof assistant.

```
Definition solGathering (r : robogram) (d : demon) :=
  ∀ cf, ¬ forbidden cf → ∃ pt : R2, WillGather pt (execute r d cf).
```

3.1 Setting and Protocol

We consider a set of nG anonymous robots that are oblivious and equipped with global strong multiplicity detection (i.e., they are able to count the number of robots that occupy any given position). The demon is supposed to be fair, and the execution model is SSYNC. The space in which robots move (the set of locations) is the real plane $\mathbb{R}^2$; they do not share any common direction, nor any chirality. Any initial configuration is accepted as long as it is not bivalent (including those with multiplicity points).

Protocol. The protocol we propose uses multiplicity to build the set of towers of maximal height. If there is a unique tower of maximal height, i.e., a unique location of highest multiplicity, this location is the destination of each activated robot. Otherwise, the inhabited locations on the smallest enclosing circle (SEC) are taken into account to define a *target*.

In our case robots enjoy strong global multiplicity detection: as noticed in Section 2 the spectrum of a configuration is the multiset of all its robots' locations. It is said to be *clean* if inhabited locations are either on the SEC or at target. When it is not clean (*dirty*), robots on SEC (or at target) stay where they are, the others move to the target, thus cleaning the spectrum. In a clean spectrum, any activated robot moves to the target. A configuration is said to be clean if and only if its spectrum is clean.

The important operation is thus to define a convenient target. Our target depends on how many inhabited locations are on the SEC. If there is only one, then the whole spectrum is reduced to a single location and all robots are already gathered. When the number of towers on the SEC is not equal to 3, the target is the center of the SEC. Critical situations occur when towers on the SEC define a triangle. If this triangle is *equilateral*, we cannot break the symmetry between its vertices and the target is the center of the SEC (which is also the triangle's barycenter). On the contrary, in all other cases we can break the symmetry and select a particular vertex as the target. If the triangle is *isosceles* and not equilateral, the target is the vertex opposite to its base. Finally if the triangle is *scalene*, the target is the vertex opposite to its longest side. Let us rephrase that description in informal pseudo-code. See Section 2 for its formal version, that is the Coq definition of our algorithm. For a spectrum s, let support(s) be the set of locations in s, let max(s) be the set of locations of maximal multiplicity in s,

and let $\text{SEC}(s)$ be the smallest enclosing circle of s. Let *dest* be the destination to be computed. Remember that $(0,0)$ is always the location of a robot in its own frame of reference.

if $\max(s) = \emptyset$ **then** *dest* $:= (0,0)$ (* absurd case *)
else if $\max(s) = \{p\}$ **then** *dest* $:= p$
else begin (* first compute target then dest depending on cleanliness *)
  if $\text{support}(s) \cap \text{SEC}(s) = \emptyset$ **then** *dest* $:= (0,0)$ (* absurd case *)
  else if $\text{support}(s) \cap \text{SEC}(s) = \{p\}$ **then** *target* $:= p$ (* already gathered *)
  else if $\text{support}(s) \cap \text{SEC}(s) = \{p_1, p_2, p_3\}$ **then** (* triangle cases *)
    if equilateral(p_1, p_2, p_3) **then** *target* $:=$ barycenter(p_1, p_2, p_3)
    else if isosceles(p_1, p_2, p_3) **then** *target* $:=$ opposite of base(p_1, p_2, p_3)
    else *target* $:=$ opposite of longest(p_1, p_2, p_3)
  else *target* $:=$ center$(\text{SEC}(s))$;
  if $\forall p \in s, p \in \text{SEC}(s)$ **or** $p =$ *target* **then** *dest* $:=$ *target* (* clean $\Rightarrow$ go to target *)
  else if $(0,0) \in \text{SEC}(s)$ **or** $(0,0) =$ *target* **then** *dest* $:= (0,0)$ (* dirty $\Rightarrow$ clean config *)
    else *dest* $:=$ *target*
  end

Phases of the algorithm. We characterise several cases of the protocol, called *phases*, which depend on what is perceived from the configuration, and which are mutually exclusive in an execution: Gathered robots, the Majority case where there is a unique tower of maximal height, the three triangle cases (Equilateral, Isosceles, Scalene), and finally the General case. To ease the proof of termination, we chose to consider differently an instance of the general case, namely the Diameter case where $\text{support}(s) \cap \text{SEC}(s)$ contains exactly two points (in which case they are a diameter of the SEC).

For all cases that need the computation of a target, we moreover distinguish between clean and dirty situations. Note that from any dirty version of a case, the only two other reachable cases are its clean version and Majority. This leaves us with twelve phases: Gathered (the success situation), Majority (Maj), Diameter clean (Dc) and dirty (Dd), Equilateral, Isosceles, Scalene clean (Ec, Ic, Sc) and dirty (Ed, Id, Sd), and General clean (Gc) and dirty (Gd). Figure 1 summarises the reachability relation between cases.

3.2 Key points to prove correctness

Some properties are fundamental in our proof that the algorithm actually solves Gathering. Namely, that robots move towards the same location, that a legal configuration cannot evolve into a forbidden (that is: bivalent) one, and finally that the configuration is eventually reduced to a single inhabited location.

Expressing the robogram in the global frame of reference. The first step towards reasoning about a robogram is to leave the robots local frames of reference and rephrase the robogram in the demon global frame of reference. This step is

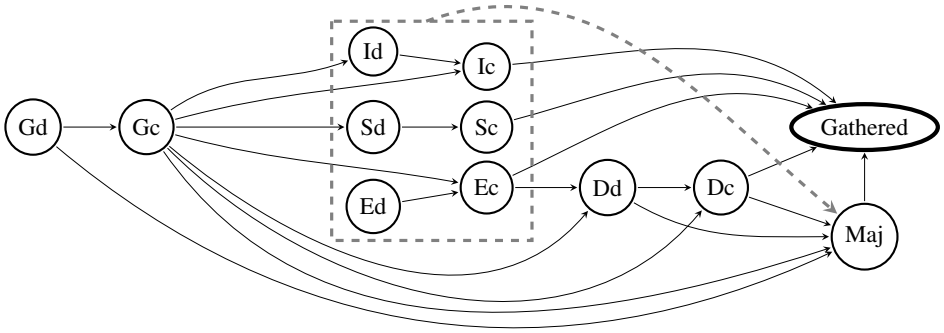

Fig. 1. Reachability graph for the distinguished categories of spectra. For clarity's sake, self loops are omitted. The boxed area contains the triangle cases; they are all linked to Maj.

always left implicit in pen-and-paper proofs but it is actually not trivial: it relies on the fact that the protocol uses only geometrical concepts that are invariant under the allowed changes of frame, here scaling, rotation, and translation. Using a formal framework ensures that this overlooked proof is indeed done and correct. This in turn gives a global version of the `round` function and creates a global view of the configuration after one round (lemma `round_simplify`).

Robots that move go to the same location. Note that by robots "that move" we explicitly mean robots that change location during the round, *not* robots that are activated (some of which may not move). Robots enjoy global strong multiplicity detection, hence they all detect the number of highest towers, they share the same notion of SEC, and they all compute the same number of towers on the SEC. Moreover, in both non-equilateral triangle cases, pointing out the longest side or the base side is not ambiguous as relative distances compare the same way for all robots. Hence, cleanliness and targets are the same for all activated robots, which means that computed destinations are the same.

Further note that we actually just showed that all moving robots are in the same phase of the protocol, and that the resulting destination does not depend on the frame of reference of the robot.

Bivalent positions are unreachable. We require that the initial configuration does not consist of exactly two towers with the same multiplicity. One of the key points ensuring this algorithm's correctness is that there is no way to reach a position that is bivalent from a position that is *not* bivalent. Consider two configurations C_0 and C_1, C_1 being bivalent and resulting from C_0 by some round. Let us denote by $|x|_0$ (resp. $|x|_1$) the multiplicity of location x in C_0 (resp. in C_1). By definition, C_1 consists of two locations l_1 and l_2 such that

$|l_1|_1 = |l_2|_1 = \frac{nG}{2}$. As all moving robots go to *the same location*, we can assume without loss of generality that robots moved to, say, l_1, adding to its original multiplicity $|l_1|_0$ (which might have been 0). Since the configuration is now bivalent, this means that l_2 was inhabited in C_0 and such that $|l_2|_0 \geq \frac{nG}{2}$ (some robot in l_2 might have moved to l_1). There cannot have been only one inhabited location l distinct from l_2 before the round because either $|l|_0 = |l_2|_0 = \frac{nG}{2}$ but we supposed the configuration was not bivalent, or $|l|_0 < \frac{nG}{2} < |l_2|_0$ but then by phase Majority robots would have moved to l_2 and not l_1. Hence C_0 consisted of l_2 and several inhabited l_i ($i \neq 2$) amongst which the robots not located in l_2 were distributed, but then none of the l_i could have held more than $\frac{nG}{2} - 1$ robots, hence phase Majority should have applied and robots should have moved to l_2, a contradiction. Interestingly, this argument makes no reference to the dimension of the space. It applies as is to both [10] and this work.

Eventually no-one moves. The termination of the algorithm is ensured by the existence of a measure decreasing at each round involving a moving robot for a well-founded ordering. We then conclude using the assumption that the demon is fair.

To define the measure, we associate a weight to each of the protocol's phases (see Section 3.1) as follows: Maj $\mapsto 0$, Dc $\mapsto 1$, Dd $\mapsto 2$, Ec, Ic, Sc $\mapsto 3$, Ed, Id, Sd $\mapsto 4$, Gc $\mapsto 5$, Gd $\mapsto 6$. Note that these weights decrease along the arcs of Figure 1. We may now map any configuration C_i to a $(p_i, m_i^{p_i}) \in \mathbb{N} \times \mathbb{N}$ such that p_i is the weight of the phase for the moving robots, and:

- m_i^0 is the number of robots that are *not at* the unique location of maximal multiplicity, and
- $m_i^{p_i>0}$ is $\begin{cases} \text{- } \sharp\text{robots that are } \textit{not at} \text{ target if } C_i \text{ is clean, or} \\ \text{- } \sharp\text{robots that are } \textit{neither at} \text{ target } \textit{nor} \text{ on SEC if } C_i \text{ is dirty.} \end{cases}$

Let $>_{\mathbb{N}}$ be the usual ordering on natural numbers, the relevant ordering $\succ$ is defined as the lexicographic extension of $>_{\mathbb{N}}$ on pairs: $(p, m) \succ (p', m')$ iff either $p >_{\mathbb{N}} p'$, or $p =_{\mathbb{N}} p'$ and $m >_{\mathbb{N}} m'$.
It is well-founded since $>_{\mathbb{N}}$ is well-founded. We show that for any round on a configuration C_k resulting in a *different* configuration C_{k+1} (that is, some robots have moved), $(p_k, m_k^{p_k}) \succ (p_{k+1}, m_{k+1}^{p_{k+1}})$, hence proving that eventually there is no more change in successive configurations.

As convincing that they may seem, the arguments above do not constitute a formal proof at all, and should not be ultimately relied upon. At best, they may give an intuition that the protocol is correct. To obtain formal guarantees, we define the protocol and do the proof in our Coq framework.

3.3 Formalising the Protocol, Key Points, and the Main Theorem

Formal description of the protocol. The type of locations is $\mathbb{R}^2$ (noted `R2.t` and defined as `R*R` from the type `R` of the Coq library on axiomatic reals). The robogram as described in Section 3.1 is:

```
Definition gatherR2_pgm (s : Spect.t) : R2.t :=
 match Spect.support (Spect.max s) with  (* max height towers?*)
 | nil ⇒ (0, 0)    (* None? only happens when no robot          *)
 | pt :: nil ⇒ pt    (* Unique highest tower? go there          *)
 | _ :: _ :: _ ⇒    (* Otherwise                                 *)
   if is_clean s then target s else  (* All on SEC/target ?    *)
   if (0, 0) ∈ (SECT s) then (0, 0) else target s
 end.
```

Target is defined as follows, in critical situations where exactly three inhabited positions are on the SEC target depends on the shape of the triangle (here isosceles *excludes* equilateral):

```
Function target_triangle (pt1 pt2 pt3 : R2.t) : R2.t :=
 match classify_triangle pt1 pt2 pt3 with (* Kind of triangle? *)
 | Equilateral ⇒ barycenter_3_pts pt1 pt2 pt3 (* To barycenter *)
 | Isosceles p ⇒ p
 | Scalene ⇒ opposite_of_max_side pt1 pt2 pt3
 end.
Function target (s : Spect.t) : R2.t :=
 match on_SEC (Spect.support s ) with (*♯inhabited locations on SEC?*)
 | nil ⇒ (0, 0)                                        (* None? *)
 | pt :: nil ⇒ pt        (* Unique loc. on SEC? ⇒ gathered!    *)
 | pt1 :: pt2 :: pt3 :: nil ⇒ target_triangle pt1 pt2 pt3
 | _ ⇒ center (SEC l)           (* Gen. case: center of SEC    *)
 end.
```

Note that this is almost exactly an actual robot code. The instantiated robogram (in the sense of Section 2) binding together this code and its compatibility property is defined under the name `gatherR2`.

Formal proofs of key points and of the main theorem. The key steps of our proof can be written as relatively straightforward statements. Theorem `round_simplify` expresses the configuration after one round in the global fame of reference, without making reference to local frames of each robot. Its proof uses several lemmas expressing the invariance of the geometric properties used by the robogram.

Theorem `same_destination` states that two moving robots id_1 and id_2 (i.e., that change locations during the round) compute the same destination location (in the demon's frame of reference). By case on the phases of the robogram, and on the structure of the provided code. The formal proof is about 20 lines long and uses Theorem `round_simplify`.

```
Theorem same_destination : ∀ da cf id1 id2,
  In id1 (moving gatherR2 da cf) → In id2 (moving gatherR2 da cf)
 → round gatherR2 da cf id1 = round gatherR2 da cf id2.
```

Theorem `never_forbidden` says that for all demonic action *da* and configuration *cf*, if *cf* is not bivalent (*i.e.* not `forbidden`), then the configuration after the round is not bivalent.

```
Theorem never_forbidden: ∀ da cf, ¬forbidden cf → ¬forbidden (round gatherR2 da cf).
```

The proof is done by a case analysis on the set of towers of maximum height at the beginning. If there is none, this is absurd; if there is exactly one, the resulting configuration will have the same highest tower, a legal configuration. Now if there are at least two highest towers, then if the resulting configuration is bivalent, at least one robot has moved (otherwise the original configuration would be bivalent, to the contrary of what is assumed), and all robots that move go to the same of the resulting two towers. The rest is arithmetics, as described on page 10. The proof of this key point is around 100 lines of Coq script. Note that as remarked on page 10 the argument is the same in $\mathbb{R}$ or $\mathbb{R}^2$, hence we were able to reuse the Coq script developed earlier for [10] (and thus in our libraries) to prove this statement.

It remains to state that for all demonic action *da* and configuration *conf*, if *conf* is not bivalent, and if there is at least one robot moving this round, then the configuration resulting from the round defined by *da* and our robogram on *conf* is smaller than *conf*. The ordering relation on configurations, called `lt_config`, is the one described in Section 3.2. The theorem stating the correctness of our robogram is then simply: for all demon d that is fair, `gatherR2` is a solution with reference to d.

```
Theorem round_lt_config: ∀ da conf, ¬ forbidden conf
   → moving gatherR2 da conf ≠ nil → lt_config (round gatherR2 da conf) conf.
Theorem Gathering_in_R2 : ∀ d, Fair d → solGathering gatherR2 d.
```

The proof is led by well-founded induction on the `lt_config` relation. If all robots are gathered, then it is done. If not, by fairness some robots will have to move, thus a robot will be amongst the first to move. (Formally, this is an induction using fairness.) We conclude by using the induction hypothesis (of our well-founded induction) as this round decreases the measure on configurations (theorem `round_lt_config`). This proof of the main theorem is interestingly small as it is only 20 lines long. The whole file dedicated to specification and certification of our algorithm (`Algorithm.v`) consists of 478 lines of definitions, specification and intermediate lemmas, and 2836 lines of actual proof.

4 Discussion and Perspectives

The Distributed Computing community is known to have fundamental algorithms tightly coupled with their proof of correctness. The mobile robot setting is no exception, as the minimal hypotheses a protocol must make to solve a given

problem are extremely difficult to identify without actually writing the corresponding correctness proofs (that is, an intuitive approach is often detrimental to the correctness of the result to be established, as recent errors found in the literature proved [1]). In a formal proof approach to obtain mechanically certified protocols, our framework and methodology clearly contributes to two main phases in a verified development.

Firstly the *specification* phase, where all objects, definitions, algorithms, statements and expected properties are expressed without any ambiguity, in a higher order type theoretic functional environment. The lack of ambiguity is a key feature to enable the early detection of inconsistencies between the problem specification, the algorithmic proposal, and the execution model. We emphasise the fact that there is no need to be an expert with the Coq proof assistant to use our framework in this phase. Clear and unequivocal specifications are indeed a fundamental step towards correct algorithms.

Secondly the *proof* phase, where properties are proved to hold for the relevant executions. This phase is of course more demanding on the expertise side, so our goal when constructing the framework was to provide useful libraries and proof techniques that can be reused in other contexts, enabling more automation to the protocol designer. Considering reusability, useful assets brought by the current work are the notions of gathering, SSYNC demons, etc., developments on geometry in $\mathbb{R}^2$ and smallest enclosing circles, and the proof of `never_forbidden` [10]. Those will most likely prove useful in future developments. When developing the protocol for our case study, we decided to modify the protocol code several times, either to fix a newly discovered bug, or to ease the writeup of the proofs. This classical design stage was streamlined by the use of a formal language based on the Curry-Howard isomorphism [15] where both activities can be done in a uniform way. In such a setting, correcting the algorithm amounts to modifying the algorithm definition, and replaying the proofs certification process after adapting the proof scripts written previously. The mechanised verification of the proofs makes this process fast and trustworthy, compared to a purely handcrafted approach.

Perspectives A next step would be to add more dimensions to the considered Euclidean space. As the framework is highly parametric, specifying another space in which robots move is not a dramatic change: the type of locations is a parameter, it is left abstract throughout the majority of the formalism, in which a concrete instance is not needed. Another interesting evolution would be to take into account the more general ASYNC model, that is when Look-Compute-Move cycles and stages are not atomic anymore. Describing behaviours that are ASYNC in Coq may nonetheless add to the intricacy of formal proofs,

and relevant libraries to ease the task of the developer will have to be provided accordingly.

References

1. Jordan Adamek, Mikhail Nesterenko, and Sébastien Tixeuil. Evaluating and optimizing stabilizing dining philosophers. In *11th European Dependable Computing Conference, EDCC 2015, Paris, France, September 7-11, 2015*, pages 233–244. IEEE, 2015.
2. Cédric Auger, Zohir Bouzid, Pierre Courtieu, Sébastien Tixeuil, and Xavier Urbain. Certified Impossibility Results for Byzantine-Tolerant Mobile Robots. In Teruo Higashino, Yoshiaki Katayama, Toshimitsu Masuzawa, Maria Potop-Butucaru, and Masafumi Yamashita, editors, *Stabilization, Safety, and Security of Distributed Systems - 15th International Symposium (SSS 2013)*, volume 8255 of *Lecture Notes in Computer Science*, pages 178–186, Osaka, Japan, November 2013. Springer-Verlag.
3. Béatrice Berard, Laure Millet, Maria Potop-Butucaru, Yann Thierry-Mieg, and Sébastien Tixeuil. Formal verification of Mobile Robot Protocols. Technical report, LIP6 , LINCS , IUF, May 2013.
4. Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. Springer-Verlag, 2004.
5. François Bonnet, Xavier Défago, Franck Petit, Maria Potop-Butucaru, and Sébastien Tixeuil. Discovering and assessing fine-grained metrics in robot networks protocols. In *33rd IEEE International Symposium on Reliable Distributed Systems Workshops, SRDS Workshops 2014, Nara, Japan, October 6-9, 2014*, pages 50–59. IEEE, 2014.
6. Zohir Bouzid, Shantanu Das, and Sébastien Tixeuil. Gathering of mobile robots tolerating multiple crash faults. In *ICDCS*, pages 337–346, Philadelphia, Pennsylvania, USA, July 2013. IEEE Computer Society.
7. Quentin Bramas and Sébastien Tixeuil. Wait-free gathering without chirality. In Christian Scheideler, editor, *Structural Information and Communication Complexity - 22nd International Colloquium, SIROCCO 2015, Montserrat, Spain, July 14-16, 2015, Post-Proceedings*, volume 9439 of *Lecture Notes in Computer Science*, pages 313–327. Springer, 2015.
8. Béatrice Bérard, Pierre Courtieu, Laure Millet, Maria Potop-Butucaru, Lionel Rieg, Nathalie Sznajder, Sébastien Tixeuil, and Xavier Urbain. Formal Methods for Mobile Robots: Current Results and Open Problems. *International Journal of Informatics Society*, 7(3):101–114, 2015. Invited Paper.
9. Thierry Coquand and Christine Paulin-Mohring. Inductively Defined Types. In Per Martin-Löf and Grigori Mints, editors, *International Conference on Computer Logic (Colog'88)*, volume 417 of *Lecture Notes in Computer Science*, pages 50–66. Springer-Verlag, 1990.
10. Pierre Courtieu, Lionel Rieg, Sébastien Tixeuil, and Xavier Urbain. A Certified Universal Gathering Algorithm for Oblivious Mobile Robots. *CoRR*, abs/1506.01603, 2015.
11. Pierre Courtieu, Lionel Rieg, Sébastien Tixeuil, and Xavier Urbain. Impossibility of Gathering, a Certification. *Information Processing Letters*, 115:447–452, 2015.
12. Stéphane Devismes, Anissa Lamani, Franck Petit, Pascal Raymond, and Sébastien Tixeuil. Optimal Grid Exploration by Asynchronous Oblivious Robots. In Andréa W. Richa and Christian Scheideler, editors, *Stabilization, Safety, and Security of Distributed Systems - 14th International Symposium (SSS 2012)*, volume 7596 of *Lecture Notes in Computer Science*, pages 64–76, Toronto, Canada, October 2012. Springer-Verlag.
13. Yoann Dieudonné and Franck Petit. Self-stabilizing gathering with strong multiplicity detection. *Theoretical Computer Science*, 428:47–57, 2012.

14. Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. *Distributed Computing by Oblivious Mobile Robots*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2012.
15. William A. Howard. The formulae-as-types notion of construction. In J. Roger Hindley Jonathan P. Seldin, editor, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, London, 1980.
16. Taisuke Izumi, Tomoko Izumi, Sayaka Kamei, and Fukuhito Ooshita. Feasibility of polynomial-time randomized gathering for oblivious mobile robots. *IEEE Transactions on Parallel and Distributed Systems*, 24(4):716–723, 2013.
17. Taisuke Izumi, Samia Souissi, Yoshiaki Katayama, Nobuhiro Inuzuka, Xavier Défago, Koichi Wada, and Masafumi Yamashita. The gathering problem for two oblivious robots with unreliable compasses. *SIAM Journal of Computing*, 41(1):26–46, 2012.
18. Laure Millet, Maria Potop-Butucaru, Nathalie Sznajder, and Sébastien Tixeuil. On the synthesis of mobile robots algorithms: The case of ring gathering. In Pascal Felber and Vijay K. Garg, editors, *Stabilization, Safety, and Security of Distributed Systems - 16th International Symposium, (SSS 2014)*, volume 8756 of *Lecture Notes in Computer Science*, pages 237–251, Paderborn, Germany, sep 2014. Springer-Verlag.
19. Davide Sangiorgi. *Introduction to Bisimulation and Coinduction*. Cambridge University Press, 2012.
20. Ichiro Suzuki and Masafumi Yamashita. Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. *SIAM Journal of Computing*, 28(4):1347–1363, 1999.