Information Processing Letters 115 (2015) 447–452

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Impossibility of gathering, a certification

Pierre Courtieu[b], Lionel Rieg[a,b], Sébastien Tixeuil[d,e,f,1], Xavier Urbain[a,b,c,*]

[a] *École Nat. Sup. d'Informatique pour l'Industrie et l'Entreprise (ENSIIE), Évry, F-91025, France*
[b] *Cédric – Conservatoire National des Arts et Métiers, Paris, F-75141, France*
[c] *LRI, CNRS UMR 8623, Université Paris-Sud, Orsay, F-91405, France*
[d] *Sorbonne Universités, UPMC Univ. Paris 06, UMR 7606, LIP6, F-75005, Paris, France*
[e] *CNRS, UMR 7606, LIP6, F-75005, Paris, France*
[f] *Institut Universitaire de France, France*

ARTICLE INFO

Article history:
Received 22 May 2014
Received in revised form 5 October 2014
Accepted 4 November 2014
Available online 11 November 2014
Communicated by M. Yamashita

Keywords:
Analysis of algorithms
Automatic theorem proving
Computational geometry
Distributed computing
Theory of computation

ABSTRACT

Recent advances in Distributed Computing highlight models and algorithms for autonomous swarms of mobile robots that self-organise and cooperate to solve global objectives. The overwhelming majority of works so far considers handmade algorithms and proofs of correctness.

This paper builds upon a previously proposed formal framework to certify the correctness of impossibility results regarding distributed algorithms that are dedicated to autonomous mobile robots evolving in a continuous space. As a case study, we consider the problem of gathering all robots at a particular location, not known beforehand. A fundamental (but not yet formally certified) result, due to Suzuki and Yamashita, states that this simple task is impossible for two robots executing deterministic code and initially located at distinct positions. Not only do we obtain a certified proof of the original impossibility result, we also get the more general impossibility of gathering with an even number of robots, when any two robots are possibly initially at the same exact location.

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 and Yamashita [1] 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 any explicit way. They have however the ability to sense the environment and see the position of the other robots. 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 location computed in the previous phase. The two execution models are denoted by (using recent taxonomy [2]) FSYNC, for fully synchronous, and SSYNC, for semi-synchronous. In the SSYNC model, an arbitrary non-empty subset of robots

[*] Corresponding author at: École Nat. Sup. d'Informatique pour l'Industrie et l'Entreprise (ENSIIE), Évry, F-91025, France.
E-mail addresses: Pierre.Courtieu@cnam.fr (P. Courtieu), Lionel.Rieg@ensiie.fr (L. Rieg), Sebastien.Tixeuil@lip6.fr (S. Tixeuil), Xavier.Urbain@ensiie.fr (X. Urbain).

[1] This author is supported in part by LINCS.

http://dx.doi.org/10.1016/j.ipl.2014.11.001
0020-0190/© 2014 Elsevier B.V. All rights reserved.

execute the three phases synchronously and atomically. In the FSYNC model, all robots execute the three phases synchronously.

One of the benchmarking [2] problems for mobile robots is that of *Gathering*. Regardless of their initial positions, robots have to move in such a way that they eventually stand on the same location, not known beforehand, and remain there thereafter. A key impossibility result for gathering is due to Suzuki and Yamashita [1]: two robots initially located at distinct positions may never gather if they execute a deterministic algorithm. This result is fundamental because any weakening of the initial system hypotheses (*e.g.* anonymity, obliviousness, common sense of direction) makes the problem solvable [3].

Related works Most related to our concern are recent approaches to mechanising the algorithm design or the proof of correctness in the context of autonomous mobile robots [4–8]. Model-checking proved useful to find bugs in existing literature [6] and assess formally published algorithms [5,6], in a simpler setting where robots evolve in a *discrete space* where the number of possible positions is finite. However, no method exists to derive impossibility results using model checking. Automatic program synthesis (for the problem of perpetual exclusive exploration in a ring-shaped discrete space) is due to Bonnet et al. [4], and could be used to prove impossibility in a particular setting (by a side effect, if no algorithm can be generated), yet it exhibits important limitations for studying the gathering problem we focus on here. First, the authors consider only the discrete space setting (with a ring shape). Second, their approach is brute force (it generates every possible algorithm in a particular setting, regardless of the problem to solve). Third, the generator is limited to configurations where *(i)* a location can only host one robot (so, gathering cannot be expressed), and *(ii)* no symmetry appears (which eludes all interesting cases for studying gathering). The approach was recently refined by Millet et al. [8] for the problem of gathering in a discrete ring network. Yet, the tools used prevent algorithm synthesis for more than three robots in a (small) fixed size ring. So, none of those approaches is suitable for positions requiring real numbers, or for establishing results that are valid for any number of robots and any network size.

Developed for the Coq proof assistant,[2] the Pactole framework enabled the use of high-order logic to certify impossibility results [7] for the problem of convergence: for any positive ε, robots are required to reach locations that are at most ε apart. Of course, an algorithm that solves gathering also solves convergence, but the converse is not true. As convergence is solvable in the usual setting, the impossibility results that can be obtained involve Byzantine robots (that is, robots that may exhibit arbitrary, and possibly malicious, behaviours). The impossibility results obtained in previous work using Coq [7] show that convergence is impossible if more than half of the robots are Byzantine in the FSYNC model (or more that one third of the robots are Byzantine in the SSYNC model). These results cannot be directly reused for the case of "Gathering Impossibility" for several reasons. First, they involve the active participation of Byzantine robots to destabilise the correct ones, while the gathering problem involves only correct robots. Second, the possible positions robots may occupy are encoded using rational numbers, while positions in the original model actually use real numbers.

Our contribution In this paper, we consider the construction of a formal proof for the fundamental impossibility result of Suzuki and Yamashita [1], for two robots executing deterministic code and initially located at distinct positions. Our proof builds upon the previously initiated Pactole framework [7] to use actual real numbers as locations instead of rational numbers, and refines the definitions of executions (including scheduling assumptions) to enable the study of executions involving only correct processes. Not only do we obtain a certified proof of the original impossibility result of Suzuki and Yamashita, we also get the more general impossibility result with an even number of robots, when any two robots are possibly initially at the same exact location.

2. Preliminaries

2.1. Certification and the Coq *proof assistant*

To certify results and to guarantee the soundness of theorems, we use the Coq proof assistant, a Curry-Howard based interactive prover enjoying a trustworthy kernel. The Pactole formal model is thus developed in Coq's formal language, a very expressive λ-calculus: the *Calculus of Inductive Constructions* (CIC) [9]. In this (functional) language, datatypes, objects, algorithms, theorems and proofs can be expressed in a unified way, as terms. λ-abstraction is denoted by `fun x:T ⇒ t`, and application is denoted by `t u`. Curry-Howard isomorphism associates proofs and programs, types and logical propositions. Writing a proof of a theorem in this setting amounts to building (interactively in most cases but with the help of tactics) a term the type of which corresponds to the theorem statement. As a term is indeed a *proof* of its type, ensuring the soundness of a proof thus simply consists in type-checking a λ-term.

Coq has already been successfully employed for various tasks such as the formalisation of programming language semantics [10,11] or mathematical developments as involved as the 4-colours [12] or Feit–Thompson [13] theorems. Regarding distributed algorithms, local calculi enjoy a formal model with the Coq library Loco [14].

The reader will find in [15] 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, Pactole proposes a formal model, as well as lemmas and theorem, 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. . . .

[2] http://coq.inria.fr.