Formal Methods in System Design (2020) 56:55–89
https://doi.org/10.1007/s10703-019-00335-y

Parameterized verification of algorithms for oblivious robots on a ring

Arnaud Sangnier[1] · Nathalie Sznajder[2] · Maria Potop-Butucaru[2] · Sébastien Tixeuil[2]

Published online: 30 July 2019

Abstract
We study verification problems for autonomous swarms of mobile robots that self-organize and cooperate to solve global objectives. In particular, we focus in this paper on the model proposed by Suzuki and Yamashita of anonymous robots evolving in a discrete space with a finite number of locations (here, a ring). A large number of algorithms have been proposed working for rings whose size is not a priori fixed and can be hence considered as a parameter. Handmade correctness proofs of these algorithms have been shown to be error-prone, and recent attention had been given to the application of formal methods to automatically prove those. Our work is the first to study the verification problem of such algorithms in the parameterized case. We show that safety and reachability problems are undecidable for robots evolving asynchronously. On the positive side, we show that safety properties are decidable in the synchronous case, as well as in the asynchronous case for a particular class of algorithms. Several other properties of the protocol can be decided as well. Decision procedures rely on an encoding in Presburger arithmetics formulae that can be verified by an SMT-solver. Feasibility of our approach is demonstrated by the encoding of several case studies.

Keywords Formal verification · SMT-solver · Swarms of robots

This work has been partly supported by the ANR research program ANR FREDDA (ANR-17-CE40-0013).

✉ Arnaud Sangnier
sangnier@irif.fr

Nathalie Sznajder
nathalie.sznajder@lip6.fr

Maria Potop-Butucaru
maria.potop-butucaru@lip6.fr

Sébastien Tixeuil
sebastien.tixeuil@lip6.fr

[1] IRIF - Univ Paris Diderot, Paris, France

[2] Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, 75005 Paris, France

1 Introduction

We consider sets of mobile oblivious robots evolving in a discrete space, with a finite number of locations, modeled as a graph. Robots follow the seminal model by Suzuki and Yamashita [23]: they do not remember their past actions, they cannot communicate explicitly, and are disoriented.

However, they can sense their environment and detect the positions of the other robots on the graph. If several robots share the same position—i.e. the same node of the graph—(forming a *tower*, or multiplicity point), other robots may or may not detect the tower. If robots have *weak* multiplicity detection, they are assumed to sense a tower on a position, but are not able to count the actual number of robots in this tower. With *strong* multiplicity detection, they are able to count the exact number of robots on a given position. In case they have no multiplicity detection, robots simply detect occupied positions. In this paper, we assume strong multiplicity detection is available to all robots.

Each robot behaves according to the following cycle: it takes a snapshot of its environment, then it computes its next move (either stay idle or move to an adjacent node in the ring), and at the end of the cycle, it moves according to its computation. Such a cycle is called a look-compute-move cycle. Since robots cannot rely on a common sense of direction, directions that are computed in the compute phase are only *relative* to the robot. Robots are anonymous and execute the same algorithm to achieve together a given objective. When the underlying graph is symmetric (e.g. a ring), a lot of configurations of the robots become also symmetric, making the problem of designing a protocol for the robot especially intricate. To tell apart its two sides, a robot relies on a description of the ring in both clockwise and counter-clockwise direction, which gives two views of the configuration. There are two consequences to this fact. First, if the two views are identical, meaning that the robot is on an axis of symmetry, it cannot distinguish the two directions and thus either decides to stay idle, or to move. In the latter case, the robot move becomes a non-deterministic choice between the two available directions. Second, when two robots have the same two views of the ring, the protocol commands them to move in the same relative direction, but this might result in moves in actual opposite directions for the two robots. Such a symmetrical situation is pictured in Fig. 1.

Different objectives for ring-shaped discrete spaces have been studied in the literature [15]: gathering—starting from any initial configuration, all the robots must gather on the same node, not known beforehand, and then stop [16]; exploration with stop—starting from any initial configuration, the robots reach a configuration where they all are idle and, in the meanwhile, all the positions of the ring have been visited by a robot [14]; exclusive perpetual exploration—starting from any tower-free configuration, each position of the ring is visited infinitely often and no multiplicity point ever appears [5,10].

Existing execution models consider different types of synchronization for the robots: in the fully synchronous model (FSYNC), all robots evolve simultaneously and complete a full look-compute-move cycle. The semi-synchronous model (SSYNC) consider runs that

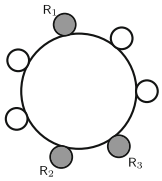


Fig. 1 A disoriented robot R_1—symmetric robots R_2 and R_3 will move in opposite directions

evolve in phases: at each phase, an arbitrary subset of the robots is scheduled for a full look-compute-move cycle, which is executed simultaneously by all robots of the subset. Finally, in the asynchronous model (ASYNC), robots evolve freely at their own pace. In particular, a robot can move according to a computation based on an obsolete observation of its environment, as others robots may have moved in between. Algorithms in the literature are typically parameterized by the number of robots and/or number of positions in the ring. In this work we focus on formally verifying algorithms parameterized by the number of ring positions only, assuming a fixed number of robots.

1.1 Related work

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. Unsurprisingly, this mechanized approach to protocol correctness was successively used in the context of mobile robots [1–4,6,12,18].

When robots are *not* constrained to evolve on a particular topology (but instead are allowed to move freely in a bidimensional Euclidian space), the Pactole (http://pactole.lri.fr) framework has been proven useful. Developed for the Coq proof assistant, Pactole enabled the use of high-order logic to certify impossibility results [1] 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 with Pactole is the impossibility of gathering starting from a bivalent configuration [8]. Recently, positive certified results for SSYNC gathering with multiplicity detection [9], and for FSYNC gathering without multiplicity detection [2] were provided. However, as of now, no Pactole library is dedicated to robots that evolve on discrete spaces.

In the discrete setting that we consider, model checking has shown to be useful both to find bugs in existing literature [4,13] and to assess formally published algorithms [4,12]. Automatic program synthesis (for the problem of perpetual exclusive exploration in a ring-shaped discrete space) is due to Bonnet et al. [6], 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 bounded setting where both the number of locations and the number of robots are known, they cannot permit to establish results that are valid for any number of locations.

Recently, Aminof et al. [20] presented a general framework for verifying properties of mobile robots evolving on graphs, where the graphs are a parameter of the problem. While our model could be encoded in their framework, their undecidability proof relies on persistent memory used by the robots, hence is not applicable to the case of oblivious robots we are interested in. Also, they obtain decidability for a subcase that is not relevant for robot protocols like those we consider.

1.2 Contributions

In this work, we aim at verifying protocols for swarms of robots for any number of locations, hence to remove the limitation encountered with the classical model checking approach taken so far for this problem.

We provide a formal definition of the problem, where the protocol can be described as a quantifier free Presburger formula. This logic, whose satisfiability problem is known to be decidable, happens to be powerful enough to express existing algorithms in the literature.

Objectives of the robots are also described by Presburger formulae and we consider two problems: when the objective of the robots is a safety objective—robots have to avoid the configurations described by the formula (SAFE), and when it is a reachability objective (REACH). We show that REACH is undecidable in any semantics, while SAFE is decidable only in FSYNC and SSYNC. We also show that when the protocol is *uniquely-sequentializable* or *pending-bounded*, safety properties become decidable also in the asynchronous case. Finally, we show practical applicability of this approach by using an SMT-solver to verify safety properties for two algorithms from the literature.

We strongly believe that our formalism should be used when describing and presenting such protocols, because it is formal and non-ambiguous, and thus eliminates any unclarity often found in the literature. Moreover, if totally automated verification in the parameterized setting seems unfeasible, our method provides a way to perform a "sanity check" of the protocol, and to automatically prove intermediate lemmas, that can then be used as formally proved building blocks of a handmade correctness proof.

This work has been first presented in [21]. In this version, we have included some detailed proofs, proposed a new extension to verify safety properties in the asynchronous case, namely when the protocols are pending bounded, and finally we have added more experimental results together with detailed information about our case studies.

2 Model of robots evolving on a ring

2.1 Formal model

In this section we present the formal language to describe mobile robots protocols as well as the way it is interpreted.

2.1.1 Preliminaries

For $a, b \in \mathbb{Z}$ such that $a \leq b$, we denote by $[a, b]$ the set $\{c \in \mathbb{Z} \mid a \leq c \leq b\}$. For $a \in \mathbb{Z}$ and $b \in \mathbb{N}$, we write $a \odot b$ for the natural $d \in [0, (b-1)]$ such that there exists $j \in \mathbb{Z}$ and $a = b \cdot j + d$ (for instance $-1 \odot 3 = 2$). Note that $\odot$ corresponds to the modulo operator, but we recall its definition to avoid ambiguity, especially when a is negative.

We recall the definition of Existential Presburger (EP) formulae. Let Y be a countable set of variables. First we define the grammar for terms

$$\mathtt{t} ::= x \mid \mathtt{t} + \mathtt{t} \mid a \cdot \mathtt{t} \mid \mathtt{t} \mod a,$$

where $a \in \mathbb{N}$ and $x \in Y$, and then the grammar for formulae is given by

$$\phi ::= \mathtt{t} \bowtie b \mid \phi \wedge \phi \mid \phi \vee \phi \mid \exists x.\phi,$$

where $\bowtie \in \{=, \leq, \geq, <, >\}$, $x \in Y$, and $b \in \mathbb{N}$. We sometimes write a formula ϕ as $\phi(x_1, \ldots, x_k)$ to underline that $x_1, \ldots, x_k$ are the free variables of ϕ. Negation of formulae is not allowed to forbid universal quantification over variables. However, observe that inequality between a term $\mathtt{t}$ and an integer b can be obtained with the formula $\mathtt{t} < b \vee \mathtt{t} > b$. The set of Quantifier Free Presburger (QFP) formulae is obtained by the same grammar deleting the elements $\exists x.\phi$. Note that when dealing with QFP formulae, we allow as well negations of formulae.

We say that a vector $V = \langle d_1, \ldots, d_k \rangle$ satisfies an EP formula $\phi(x_1, \ldots, x_k)$, denoted by $V \models \phi$, if the formula obtained by replacing each x_i by d_i holds. Given a formula ϕ with free variables $x_1, \ldots, x_k$, we write $\phi(d_1, \ldots, d_k)$ the formula where each x_i is replaced by d_i. We let $[\![\phi(x_1, \ldots, x_k)]\!] = \{\langle d_1, \ldots, d_k \rangle \in \mathbb{N}^k \mid \langle d_1, \ldots, d_k \rangle \models \phi\}$ be the set of models of the formula. In the sequel, we use Presburger formulae to describe configurations of the robots, as well as protocols.

2.1.2 Configurations and robot views

In this paper, we consider a fixed number $k > 0$ of robots and, except when stated otherwise, we assume the identities of the robots are taken from the set $\mathcal{R} = \{R_1, \ldots, R_k\}$. We may sometimes identify $\mathcal{R}$ with the set of indices $\{1, \ldots, k\}$. On a ring of size $n \geq k$, a *(k,n)-configuration* of the robots (or simply a configuration if n and k are clear from the context) is given by a vector $\mathbf{p} \in [0, n-1]^k$ associating to each robot R_i its position $\mathbf{p}(i)$ on the ring. We assume w.l.o.g. that positions are numbered in the clockwise direction.

A *view* of a robot on this configuration gives the distances between the robots, starting from its neighbor, i.e. the robot positioned on the next occupied node (a distance equal to 0 meaning that two robots are on the same node). A *view* $\mathbf{V} = \langle d_1, \ldots, d_k \rangle \in [0, n]^k$ is a k-tuple such that $\sum_{i=1}^{k} d_i = n$ and $d_1 \neq 0$. We require the first constraint to ensure that the sum of the distances between consecutive robots is equal to the length of the ring, otherwise a vector in $[0, n]^k$ cannot represent a correct view of the entire ring. The second constraint is for normalization questions that we will detail later. Observe that it is possible to obtain a view with $d_1 \neq 0$ by putting 0 at the 'end' of the view instead. We let $\mathcal{V}_{n,k}$ be the set of possible views for k robots on a ring of size n. Notice that all the robots sharing the same position should have the same view. Finally, for a view $\mathbf{V} = \langle d_1, \ldots, d_k \rangle \in [0, n]^k$, we note $\overleftarrow{\mathbf{V}} = \langle d_j, \ldots, d_1, 0, \ldots, 0 \rangle$ the corresponding view when looking at the ring in the opposite direction, where j is the greatest index such that $d_j \neq 0$.

Example 1 Suppose, as it is represented on Fig. 2, that, on a ring of size 10, two robots, R_1 and R_2, are on the same position of the ring (say position 1), R_3 is at position 4, R_4 is at position 8, and R_5 is at position 9. Then, a view of R_1 and R_2 is $\langle 3, 4, 1, 2, 0 \rangle$. It is interpreted by the fact that there is a robot at a distance 3 (R_3), a robot a at distance $3 + 4$ (R_4) and so on. We point out that all the robots at the same position share the same view. As a matter of fact in the example of Fig. 2, there is a robot at distance $3 + 4 + 1 + 2 = 10$ from R_1 (resp. R_2), which is R_2 (resp. R_1). The sum of the d_i always corresponds to the size of the ring. Here, the last element of the view of R_1 is 0, meaning that there is a distance 0 between the last robot (here R_2) and R_1. When looking in the opposite direction, their view becomes: $\langle 2, 1, 4, 3, 0 \rangle$

Given a configuration $\mathbf{p} \in [0, n-1]^k$ and a robot $\mathsf{R}_i \in \mathcal{R}$, the view of robot R_i when looking in the clockwise direction, is given by $\mathbf{V}_\mathbf{p}[i \rightarrow] = \langle d_i(i_1), d_i(i_2) - d_i(i_1), \ldots, n - d_i(i_{k-1}) \rangle$,

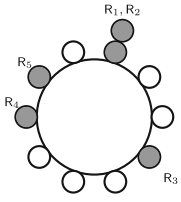


Fig. 2 A configuration with a tower $\mathsf{R}_1, \mathsf{R}_2$

where, for all $j \neq i$, $d_i(j) \in [1, n]$ is such that $(\mathbf{p}(i) + d_i(j)) \odot n = \mathbf{p}(j)$ and $i_1, \ldots, i_k$ are indexes pairwise different such that $0 < d_i(i_1) \leq d_i(i_2) \leq \cdots \leq d_i(i_{k-1})$. When robot R_i looks in the opposite direction, its view of to the configuration $\mathbf{p}$ is $\mathbf{V}_\mathbf{p}[\leftarrow i] = \overleftarrow{\mathbf{V}_\mathbf{p}[i \rightarrow]}$. At each step, when taking a snapshot of the ring, a robot captures the views in both clockwise and anti-clockwise directions.

2.1.3 Protocols

In our context, a protocol for networks of k robots is given by a QFP formula respecting some specific constraints.

Definition 1 (*Protocol*) A protocol is a QFP formula $\phi(x_1, \ldots, x_k)$ such that for all views $\mathbf{V}$ the following holds: if $\mathbf{V} \models \phi$ and $\mathbf{V} \neq \overleftarrow{\mathbf{V}}$ then $\overleftarrow{\mathbf{V}} \not\models \phi$.

A robot uses the protocol to know in which direction it should move according to the following rules. As we have already stressed, all the robots that share the same position have the same view of the ring. Given a configuration $\mathbf{p}$ and a robot $R_i \in \mathcal{R}$, if $\mathbf{V}_\mathbf{p}[i \rightarrow] \models \phi$, then the robot R_i moves in the clockwise direction, if $\mathbf{V}_\mathbf{p}[\leftarrow i] \models \phi$ then it moves in the opposite direction, if none of $\mathbf{V}_\mathbf{p}[i \rightarrow]$ and $\mathbf{V}_\mathbf{p}[\leftarrow i]$ satisfies ϕ then the robot should not move. The conditions expressed in Definition 1 imposes hence a direction when $\mathbf{V}_\mathbf{p}[i \rightarrow] \neq \mathbf{V}_\mathbf{p}[\leftarrow i]$. In case $\mathbf{V}_\mathbf{p}[i \rightarrow] = \mathbf{V}_\mathbf{p}[\leftarrow i]$, the robot is disoriented and it can hence move in one direction or the other. For instance, consider the configuration $\mathbf{p}$ pictured on Fig. 1: here, $\mathbf{V}_\mathbf{p}[1 \rightarrow] = \langle 3, 1, 3\rangle = \mathbf{V}_\mathbf{p}[\leftarrow 1]$. Note that such a semantics enforces that the behavior of a robot is not influenced by its direction. In fact consider two symmetrical configurations $\mathbf{p}$ and $\mathbf{p}'$ such that $\mathbf{V}_\mathbf{p}[i \rightarrow] = \overleftarrow{\mathbf{V}_{\mathbf{p}'}[i \rightarrow]}$ for each robot R_i. If $\mathbf{V}_\mathbf{p}[i \rightarrow] \models \phi$ (resp. $\mathbf{V}_\mathbf{p}[\leftarrow i] \models \phi$), then necessarily $\mathbf{V}_{\mathbf{p}'}[\leftarrow i] \models \phi$ (resp. $\mathbf{V}_{\mathbf{p}'}[i \rightarrow] \models \phi$), and the robot in $\mathbf{p}'$ moves in the opposite direction than in $\mathbf{p}$ (and the symmetry of the two configurations is maintained). Note that checking if a given QFP formula is a protocol is decidable; it is not difficult to write an QFP formula that is satisfiable if and only if the formula ϕ is not a protocol. Details will be given in Sect. 4.

We now formalize the way movement is decided. Given a protocol ϕ and a view $\mathbf{V}$, the moves of any robot whose clockwise direction view is $\mathbf{V}$ are given by:

$$move(\phi, \mathbf{V}) = \begin{cases} \{+1\} & \text{if } \mathbf{V} \models \phi \text{ and } \mathbf{V} \neq \overleftarrow{\mathbf{V}} \\ \{-1\} & \text{if } \overleftarrow{\mathbf{V}} \models \phi \text{ and } \mathbf{V} \neq \overleftarrow{\mathbf{V}} \\ \{-1, +1\} & \text{if } \mathbf{V} \models \phi \text{ and } \mathbf{V} = \overleftarrow{\mathbf{V}} \\ \{0\} & \text{otherwise} \end{cases}$$

Here $+1$ (resp. -1) stands for a movement of the robot in the clockwise (resp. anticlockwise) direction.

Example 2 Consider the configuration with three robots depicted in Fig. 3. In this configuration, each robot has the same view $\mathbf{V}$ equal to $\langle 3, 3, 3\rangle$ and such that $\overleftarrow{\mathbf{V}} = \mathbf{V}$. If now

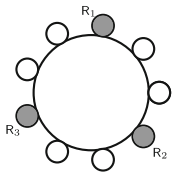


Fig. 3 A configuration with three robots

we consider the protocol $\phi(x_1, x_2, x_3) := (x_1 = x_3) \wedge (x_1 > 1) \wedge (x_3 > 1)$ (which corresponds to a rule from the bigger protocol given in [5]), then for each of the robots, the move according to the protocol will be non-deterministically either in the clockwise or anti-clockwise direction.

2.2 Different possible semantics

We now describe different transition relations between configurations. We assume that robots have a two-phase behavior : (1) look at the ring and (2) according to their view, compute and perform a movement. In this context, we consider three different modes. In the *semi-synchronous mode*, in one step, some of the robots look at the ring and move. In the *synchronous mode*, in one step, all the robots look at the ring and move. In the *asynchronous mode*, in one step a single robot performs a single action: look at the ring, if the last thing it did was a movement, or move, if the last thing it did was to look at the ring. As a consequence, its movement decision is a consequence of the view of the ring it has in its memory. In the remainder of the paper, we fix a protocol ϕ and we consider a set $\mathcal{R}$ of k robots.

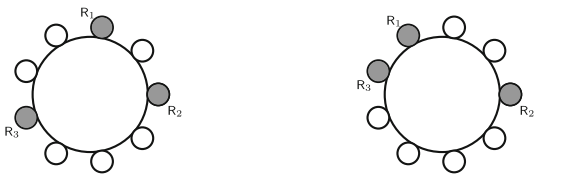
2.2.1 Semi-synchronous mode

We begin by providing the semantics in the semi-synchronous case. For this matter we define the transition relation $\hookrightarrow_\phi \subseteq [0, n-1]^k \times [0, n-1]^k$ (simply noted $\hookrightarrow$ when ϕ is clear from the context) between configurations. We have $\mathbf{p} \hookrightarrow \mathbf{p}'$ if there exists a subset $I \subseteq \mathcal{R}$ of robots such that, for all $i \in I$, $\mathbf{p}'(i) = (\mathbf{p}(i) + m) \odot n$, where $m \in move(\phi, \mathbf{V_p}[i \to])$, and for all $i \in \mathcal{R} \backslash I$, $\mathbf{p}'(i) = \mathbf{p}(i)$.

2.2.2 Synchronous mode

The transition relation $\Rightarrow_\phi \subseteq [0, n-1]^k \times [0, n-1]^k$ (simply noted $\Rightarrow$ when ϕ is clear from the context) describing synchronous movements is very similar to the semi-synchronous case, except that all the robots have to move. Then $\mathbf{p} \Rightarrow \mathbf{p}'$ if $\mathbf{p}'(i) = (\mathbf{p}(i) + m) \odot n$ with $m \in move(\phi, \mathbf{V_p}[i \to])$ for all $i \in \mathcal{R}$.

Example 3 Take again the configuration with three robots depicted in Fig. 3 together with the protocol $\phi(x_1, x_2, x_3) := (x_1 = x_3) \wedge (x_1 > 1) \wedge (x_3 > 1)$. Then, the configuration represented in Fig. 4a represents a possible successor configuration in the semi-synchronous mode where only the robot R_2 has moved. The configuration represented in Fig. 4b represents a possible successor configuration in the synchronous mode. In this case, we will have many possible successor configurations, even in the synchronous mode, since all the robots are disoriented, hence all their movements are non-deterministically chosen.



(a) Semi-synchronous mode **(b)** Synchronous mode

Fig. 4 Successor configurations of configuration depicted in Fig. 3

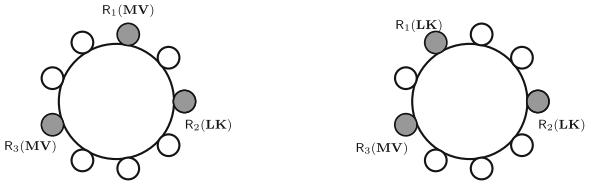
2.2.3 Asynchronous mode

The definition of the transition relation for the asynchronous mode is a bit more involved, for two reasons: first, the move of each robot does not depend on the current configuration, but on the last view of the robot. Second, in one step a robot either looks or moves. As a consequence, an *asynchronous configuration* is a tuple $\langle \mathbf{p}, \mathbf{s}, \mathbf{V} \rangle$ where $\mathbf{p} \in [0, n-1]^k$ gives the current configuration, $\mathbf{s} \in \{\mathbf{LK}, \mathbf{MV}\}^k$ gives, for each robot, its internal state (**LK** stands for ready to look and **MV** stands for compute and move) and $\mathbf{V} \in (\mathcal{V}_{n,k})^k$ stores, for each robot, the view (in the clockwise direction) it had the last time it looked at the ring.

The transition relation for asynchronous mode is hence defined by a binary relation $\rightsquigarrow_\phi$ (or simply $\rightsquigarrow$) working on $[0, n-1]^k \times \{\mathbf{LK}, \mathbf{MV}\}^k \times (\mathcal{V}_{n,k})^k$ and defined as follows: $\langle \mathbf{p}, \mathbf{s}, \mathbf{V} \rangle \rightsquigarrow \langle \mathbf{p}', \mathbf{s}', \mathbf{V}' \rangle$ iff there exists a robot $R_i \in \mathcal{R}$ such that the following conditions are satisfied:

- for all $R_j \in \mathcal{R}$ such that $j \neq i$, $\mathbf{p}'(j) = \mathbf{p}(j)$, $\mathbf{s}'(j) = \mathbf{s}(j)$ and $\mathbf{V}'(j) = \mathbf{V}(j)$,
- if $\mathbf{s}(i) = \mathbf{LK}$ then $\mathbf{s}'(i) = \mathbf{MV}$, $\mathbf{V}'(i) = \mathbf{V}_{\mathbf{p}}[i \rightarrow]$ and $\mathbf{p}'(i) = \mathbf{p}(i)$, i.e. if the robot that has been scheduled was about to look, then the configuration of the robots does not change, and this robot updates its view of the ring according to the current configuration. Finally, it changes its internal state to **MV**.
- If $\mathbf{s}(i) = \mathbf{MV}$ then $\mathbf{s}'(i) = \mathbf{LK}$, $\mathbf{V}'(i) = \mathbf{V}(i)$ and $\mathbf{p}'(i) = (\mathbf{p}(i) + m) \odot n$, with $m \in \mathit{move}(\phi, \mathbf{V}(i))$, i.e. if the robot was about to move, then it changes its internal state and moves according to the protocol applied to *its last view of the ring*.

Example 4 Following Examples 2 and 3, from the configuration $\mathbf{p} = \langle 0, 3, 6 \rangle$ depicted in Fig. 3, we consider the asynchronous configuration $\langle \mathbf{p}, \mathbf{s}, \mathbf{V} \rangle$ where $\mathbf{s} = \langle \mathbf{MV}, \mathbf{MV}, \mathbf{MV} \rangle$ and $\mathbf{V} = \langle \langle 3, 3, 3 \rangle, \langle 3, 3, 3 \rangle, \langle 3, 3, 3 \rangle \rangle$. This configuration captures the situation where each of the robots has just taken a snapshot and is ready to move. We add a rule to the protocol and consider the formula $\phi(x_1, x_2, x_3) := ((x_1 = x_3) \wedge (x_1 > 1) \wedge (x_3 > 1)) \vee ((x_2 = 2) \wedge (x_3 = 3))$. We describe now a possible sequence of asynchronous configurations $\langle \mathbf{p}, \mathbf{s}, \mathbf{V} \rangle \rightsquigarrow_\phi \langle \mathbf{p}_1, \mathbf{s}_1, \mathbf{V}_1 \rangle \rightsquigarrow_\phi \langle \mathbf{p}_2, \mathbf{s}_2, \mathbf{V}_2 \rangle \rightsquigarrow_\phi \langle \mathbf{p}_3, \mathbf{s}_3, \mathbf{V}_3 \rangle$. In this asynchronous run, R_2 has moved first, as shown in Fig. 5a, leading to $\mathbf{p}_1 = \langle 0, 2, 6 \rangle$, $\mathbf{s}_1 = \langle \mathbf{MV}, \mathbf{LK}, \mathbf{MV} \rangle$, and $\mathbf{V}_1 = \mathbf{V}$. Next, R_1 moves, as shown in Fig. 5b, and $\mathbf{p}_2 = \langle 8, 2, 6 \rangle$, $\mathbf{s}_2 = \langle \mathbf{LK}, \mathbf{LK}, \mathbf{MV} \rangle$ and $\mathbf{V}_2 = \mathbf{V}$. The next asynchronous configuration is reached when R_2 has taken a snapshot again, hence $\mathbf{p}_3 = \mathbf{p}_2$, $\mathbf{s}_3 = \langle \mathbf{LK}, \mathbf{MV}, \mathbf{MV} \rangle$ and $\mathbf{V}_3 = \langle \langle 3, 3, 3 \rangle, \langle 4, 2, 3 \rangle, \langle 3, 3, 3 \rangle \rangle$. Now one can see that both R_2 and R_3 are ready to move, but their move will be computed based on views of different configurations. In particular, the move of R_3 will not correspond to the current configuration anymore, and if robots R_1 and R_2 continue to evolve, its view can become more and more outdated, in an unbounded way.



(a) Robot R_2 has moved **(b)** Robot R_1 has moved

Fig. 5 Successor asynchronous configurations of configuration depicted in Fig. 3

2.2.4 Runs

A semi-synchronous (resp. synchronous) ϕ-run (or a run according to a protocol ϕ) is a (finite or infinite) sequence of configurations $\rho = \mathbf{p}_0\mathbf{p}_1\ldots$, where for all $0 \leq i < |\rho|$, $\mathbf{p}_i \hookrightarrow_\phi \mathbf{p}_{i+1}$ (resp. $\mathbf{p}_i \Rightarrow_\phi \mathbf{p}_{i+1}$). Moreover, if $\rho = \mathbf{p}_0 \cdots \mathbf{p}_n$ is finite, then there is no $\mathbf{p}$ such that $\mathbf{p}_n \hookrightarrow_\phi \mathbf{p}$ (respectively $\mathbf{p}_n \Rightarrow_\phi \mathbf{p}$). An asynchronous ϕ-run is a (finite or infinite) sequence of asynchronous configurations $\rho = \langle \mathbf{p}_0, \mathbf{s}_0, \mathbf{V}_0\rangle\langle \mathbf{p}_1, \mathbf{s}_1, \mathbf{V}_1\rangle \cdots$ where, for all $0 \leq i < |\rho|$, $\langle \mathbf{p}_i, \mathbf{s}_i, \mathbf{V}_i\rangle \rightsquigarrow_\phi \langle \mathbf{p}_{i+1}, \mathbf{s}_{i+1}, \mathbf{V}_{i+1}\rangle$ and such that $\mathbf{s}_0(i) = \mathbf{LK}$ for all $i \in [1, k]$. As a consequence, the value of $\mathbf{V}_0$ has no influence on the actual asynchronous run. We let $\mathbf{Runs}_{\mathrm{ss}}(\phi)$ (respectively $\mathbf{Runs}_{\mathrm{s}}(\phi)$, $\mathbf{Runs}_{\mathrm{as}}(\phi)$) be the set of semi-synchronous (respectively synchronous, asynchronous) ϕ-runs.

We let $\mathbf{Post}_{\mathrm{ss}}(\phi, \mathbf{p}) = \{\mathbf{p}' \mid \mathbf{p} \hookrightarrow_\phi \mathbf{p}'\}$, $\mathbf{Post}_{\mathrm{s}}(\phi, \mathbf{p}) = \{\mathbf{p}' \mid \mathbf{p} \Rightarrow_\phi \mathbf{p}'\}$ and $\mathbf{Post}_{\mathrm{as}}(\phi, \mathbf{p}) = \{\mathbf{p}' \mid$ there exist $\mathbf{V}, \mathbf{s}', \mathbf{V}'$ s.t. $\langle \mathbf{p}, \mathbf{s}_0, \mathbf{V}\rangle \rightsquigarrow_\phi \langle \mathbf{p}', \mathbf{s}', \mathbf{V}'\rangle\}$, with $\mathbf{s}_0(i) = \mathbf{LK}$ for all $i \in [1, k]$. Note that in the asynchronous case we impose all the robots to be ready to look. We respectively write $\hookrightarrow_\phi^*$, $\Rightarrow_\phi^*$ and $\rightsquigarrow_\phi^*$ for the reflexive and transitive closure of the relations $\hookrightarrow_\phi$, $\Rightarrow_\phi$ and $\rightsquigarrow_\phi$, and we define $\mathbf{Post}^*_{\mathrm{ss}}(\phi, \mathbf{p})$, $\mathbf{Post}^*_{\mathrm{s}}(\phi, \mathbf{p})$ and $\mathbf{Post}^*_{\mathrm{as}}(\phi, \mathbf{p})$ by replacing in the definition of $\mathbf{Post}_{\mathrm{ss}}(\phi, \mathbf{p})$, $\mathbf{Post}_{\mathrm{s}}(\phi, \mathbf{p})$ and $\mathbf{Post}_{\mathrm{as}}(\phi, \mathbf{p})$ the relations $\hookrightarrow_\phi$, $\Rightarrow_\phi$ and $\rightsquigarrow_\phi$ by their reflexive and transitive closure accordingly.

We now come to our first result that shows that when a protocol has a special shape, the three semantics are identical. For this matter, we introduce the notion of *activatable* robots in a configuration $\mathbf{p}$ which is given by the set $\mathrm{Act}_\phi(\mathbf{p}) = \{i \in \mathcal{R} \mid \mathit{move}(\phi, \mathbf{V}_{\mathbf{p}}[i \to]) \neq \{0\}\}$.

Definition 2 A protocol ϕ is said to be *uniquely-sequentializable* if, for all configurations $\mathbf{p}$, $|\mathrm{Act}_\phi(\mathbf{p})| \leq 1$.

It will be shown in Sect. 4 that this property is decidable. When ϕ is uniquely-sequentializable, at any moment at most one robot moves. Consequently, in that specific case, the three semantics are equivalent as stated by the following theorem.

Theorem 1 *If a protocol ϕ is uniquely-sequentializable, then for every configuration $\boldsymbol{p}$, $\boldsymbol{Post}^*_{\mathrm{s}}(\phi, \boldsymbol{p}) = \boldsymbol{Post}^*_{\mathrm{ss}}(\phi, \boldsymbol{p}) = \boldsymbol{Post}^*_{\mathrm{as}}(\phi, \boldsymbol{p})$.*

Before proving this theorem, we establish that, when ϕ is a uniquely-sequentializable protocol, asynchronous ϕ-runs have the property that when a robot has an obsolete view, it won't move until it has its view updated.

Proposition 1 *Let ϕ be a uniquely-sequentializable protocol, and let $\rho = \langle \boldsymbol{p}_0, \boldsymbol{s}_0, \boldsymbol{V}_0\rangle \langle \boldsymbol{p}_1, \boldsymbol{s}_1, \boldsymbol{V}_1\rangle \cdots \in \boldsymbol{Runs}_{\mathrm{as}}(\phi)$ be an asynchronous ϕ-run. For every robot $i \in \mathcal{R}$, for all $0 \leq \ell < |\rho|$, if $\boldsymbol{V}_\ell(i) \neq \boldsymbol{V}_{\boldsymbol{p}_\ell}[i \to]$ and $\boldsymbol{s}_\ell(i) = \boldsymbol{MV}$ then $\mathit{move}(\phi, \boldsymbol{V}_\ell(i)) = \{0\}$.*

Proof of Proposition 1 We show it by induction on the length ℓ of prefixes of ρ. For $\ell = 0$, it is obvious since $\mathbf{s}_0(i) = \mathbf{LK}$ for all $i \in \mathcal{R}$. Let ℓ such that $0 < \ell + 1 < |\rho|$ and let $i \in \mathcal{R}$ such that $\mathbf{s}_{\ell+1}(i) = \mathbf{MV}$ and $\mathbf{V}_{\ell+1}(i) \neq \mathbf{V}_{\mathbf{p}_{\ell+1}}[i \to]$. If $\mathbf{s}_\ell(i) = \mathbf{LK}$, then $\mathbf{V}_{\ell+1}(i) = \mathbf{V}_{\mathbf{p}_\ell}[i \to]$ and $\mathbf{p}_\ell = \mathbf{p}_{\ell+1}$, which is impossible. Hence, $\mathbf{s}_\ell(i) = \mathbf{MV}$ and $\mathbf{V}_\ell(i) = \mathbf{V}_{\ell+1}(i)$. Moreover, either $\mathbf{V}_\ell(i) \neq \mathbf{V}_{\mathbf{p}_\ell}[i \to]$ and by induction hypothesis, $\mathit{move}(\phi, \mathbf{V}_\ell(i)) = \{0\}$, which yields $\mathit{move}(\phi, \mathbf{V}_{\ell+1}(i)) = \{0\}$. Or $\mathbf{V}_\ell(i) = \mathbf{V}_{\mathbf{p}_\ell}[i \to]$, then $\mathbf{V}_{\mathbf{p}_\ell}[i \to] \neq \mathbf{V}_{\mathbf{p}_{\ell+1}}[i \to]$ and there exists another robot $j \neq i$ such that $\mathbf{s}_\ell(j) = \mathbf{MV}$ and $\mathbf{s}_{\ell+1}(j) = \mathbf{LK}$ and $\mathit{move}(\phi, \mathbf{V}_\ell(j)) \neq \{0\}$. By induction hypothesis, $\mathbf{V}_\ell(j) = \mathbf{V}_{\mathbf{p}_\ell}[j \to]$. Since ϕ is uniquely-sequentializable, $\mathit{move}(\phi, \mathbf{V}_{\mathbf{p}_\ell}[i \to]) = \mathit{move}(\phi, \mathbf{V}_\ell(i)) = \mathit{move}(\phi, \mathbf{V}_{\ell+1}(i)) = \{0\}$. □

This result can now be used to prove Theorem 1.

Proof of Theorem 1 Let ϕ be a protocol and $\mathbf{p}$ a configuration. From the definitions, it is obvious that $\mathbf{Post}^*_{\mathrm{s}}(\phi, \mathbf{p}) \subseteq \mathbf{Post}^*_{\mathrm{ss}}(\phi, \mathbf{p}) \subseteq \mathbf{Post}^*_{\mathrm{as}}(\phi, \mathbf{p})$. We show that $\mathbf{Post}^*_{\mathrm{as}}(\phi, \mathbf{p}) \subseteq \mathbf{Post}^*_{\mathrm{s}}(\phi, \mathbf{p})$ when ϕ is uniquely-sequentializable.

We show by induction on $m \in \mathbb{N}$ that, for any $\mathbf{V} \in (\mathcal{V}_{n,k})^k$, if $\langle \mathbf{p}, \mathbf{s}_0, \mathbf{V}\rangle \rightsquigarrow^m_\phi \langle \mathbf{p}', \mathbf{s}', \mathbf{V}'\rangle$ then $\mathbf{p}' \in \mathbf{Post}^*_{\mathrm{s}}(\phi, \mathbf{p})$, where $\langle \mathbf{p}, \mathbf{s}_0, \mathbf{V}\rangle \rightsquigarrow^m_\phi \langle \mathbf{p}', \mathbf{s}', \mathbf{V}'\rangle$ means that there exist $\langle \mathbf{p}_1, \mathbf{s}_1, \mathbf{V}_1\rangle, \ldots, \langle \mathbf{p}_{m-1}, \mathbf{s}_{m,-1}, \mathbf{V}_{m-1}\rangle$ such that $\langle \mathbf{p}, \mathbf{s}_0, \mathbf{V}\rangle \rightsquigarrow_\phi \langle \mathbf{p}_1, \mathbf{s}_1, \mathbf{V}_1\rangle \rightsquigarrow_\phi \ldots \langle \mathbf{p}_{m-1}, \mathbf{s}_{m-1}, \mathbf{V}_{m-1}\rangle \rightsquigarrow_\phi \langle \mathbf{p}', \mathbf{s}', \mathbf{V}'\rangle$. For $m = 0$, it is obvious. Let now $m \in \mathbb{N}$ and let $\langle \mathbf{p}', \mathbf{s}', \mathbf{V}'\rangle, \langle \mathbf{p}'', \mathbf{s}'', \mathbf{V}''\rangle \in [0, n-1]^k \times \{\mathbf{LK}, \mathbf{MV}\}^k \times (\mathcal{V}_{n,k})^k$, such that $\langle \mathbf{p}, \mathbf{s}_0, \mathbf{V}\rangle \rightsquigarrow^m_\phi \langle \mathbf{p}', \mathbf{s}', \mathbf{V}'\rangle \rightsquigarrow_\phi \langle \mathbf{p}'', \mathbf{s}'', \mathbf{V}''\rangle$. By induction hypothesis, $\mathbf{p}' \in \mathbf{Post}^*_{\mathrm{s}}(\phi, \mathbf{p})$. Let $i \in \mathcal{R}$ such that $\mathbf{s}''(i) \neq \mathbf{s}'(i)$. If $s'(i) = \mathbf{LK}$ then by definition, $\mathbf{p}'' = \mathbf{p}'$ and $\mathbf{p}'' \in \mathbf{Post}^*_{\mathrm{s}}(\phi, \mathbf{p})$. Otherwise, if $\mathbf{V}_{\mathbf{p}'}[i \rightarrow] \neq \mathbf{V}'(i)$, then by Proposition 1, $move(\phi, \mathbf{V}'(i)) = \{0\}$ and then $\mathbf{p}' = \mathbf{p}''$. Finally, if $\mathbf{V}_{\mathbf{p}'}[i \rightarrow] = \mathbf{V}'(i)$, either $move(\phi, \mathbf{V}'(i)) = \{0\}$ and $\mathbf{p}'' = \mathbf{p}'$ or, $move(\phi, \mathbf{V}'(i)) = move(\phi, \mathbf{V}_{\mathbf{p}'}[i \rightarrow]) \neq \{0\}$ and, since ϕ is uniquely-sequentializable, for all $j \neq i$, $move(\phi, \mathbf{V}_{\mathbf{p}'}[j \rightarrow]) = \{0\}$ and $\mathbf{p}' \Rightarrow \mathbf{p}''$. Hence $\mathbf{p}'' \in \mathbf{Post}^*_{\mathrm{s}}(\phi, \mathbf{p})$. □

2.3 Problems under study

In this work, we aim at verifying properties on protocols where we assume that the number of robots is fixed (equal to $k > 0$), but the size of the ring is a parameter and satisfies a given property. Note that when executing a protocol the size of the ring never changes. For our problems, we consider a ring property that is given by a QFP formula $\texttt{Ring}(y)$, a set of bad configurations given by a QFP formula $\texttt{Bad}(x_1, \ldots, x_k)$, and a set of good configurations given by a QFP formula $\texttt{Goal}(x_1, \ldots, x_k)$. We then define two general problems to address the verification of such algorithms, each of which declined according to the desired semantics of execution: the SAFE_m problem, and the REACH_m problem with $m \in \{\mathrm{ss}, \mathrm{s}, \mathrm{as}\}$.

The SAFE_m problem is to decide, given a protocol ϕ and two formulae $\texttt{Ring}$ and $\texttt{Bad}$, whether for every size $n \in \mathbb{N}$ and every (k, n)-configuration $\mathbf{p}$, if $n \in [\![\texttt{Ring}]\!]$ and $\mathbf{p} \notin [\![\texttt{Bad}]\!]$, then $\mathbf{Post}^*_m(\phi, \mathbf{p}) \cap [\![\texttt{Bad}]\!] = \emptyset$.

The REACH_m problem is to decide given a protocol ϕ and two formulae $\texttt{Ring}$ and $\texttt{Goal}$ whether for every size $n \in \mathbb{N}$ and every (k, n)-configuration $\mathbf{p}$, if $n \in [\![\texttt{Ring}]\!]$ then $\mathbf{Post}^*_m(\phi, \mathbf{p}) \cap [\![\texttt{Goal}]\!] \neq \emptyset$.

Example 5 We can state in our context the problem that consists in checking that a protocol ϕ working with three robots never leads to a collision (i.e. to a configuration where two robots are on the same position on the ring) for rings of size strictly bigger than 6. In that case we use the SAFE_m problem with $\texttt{Ring} := y > 6$ and $\texttt{Bad} := x_1 = x_2 \vee x_2 = x_3 \vee x_1 = x_3$.

Another property that can be checked in our context is the fact that no matter where all the robots are, they can all gather in the same point. Assume we want to verify this for a protocol with 4 robots working on rings of size strictly greater than 10, then we can use the REACH_m problem with $\texttt{Ring} := y > 10$ and $\texttt{Goal} := x_1 = x_2 = x_3 = x_4$. If the answer to this problem is positive, we know that the property is verified, in other words we know that for all rings of size strictly greater than 10, from all configurations in such rings, the robots can eventually gather on some position. On the other hand, if the answer is negative, we have found a size of the ring and an initial position from which $\texttt{Goal}$ is never attained.

Remark 1 Note that the two problems are not dual due to the quantifiers. As we will see later, in some cases we are able solve the former problem but not the latter.

3 Undecidability results

In this section, we present undecidability results for the two aforementioned problems. The proofs rely on the encoding of the run of a deterministic k-counter machine. A deterministic k-counter machine consists of k integer-valued registers (or counters) called c_1, ..., c_k, and a finite list of labelled instructions L. Each instruction is either of the form $\ell : c_i = c_i + 1$; `goto` ℓ', or ℓ : `if` $c_i > 0$ `then` $c_i = c_i - 1$; `goto` ℓ'; `else goto` ℓ'', where $i \in [1, k]$. We also assume the existence of a special instruction ℓ_h : `halt`. Configurations of a k-counter machine are elements of $L \times \mathbb{N}^k$, giving the current instruction and the current values of the registers. The initial configuration is $(\ell_0, 0, \ldots, 0)$, and the set of halting configurations is $\mathsf{HALT} = \{\ell_h\} \times \mathbb{N}^k$. Given a configuration $(\ell, n_1, \ldots, n_k)$, the successor configuration $(\ell', n'_1, \ldots, n'_k)$ is defined in the usual way and we write $(\ell, n_1, \ldots, n_k) \vdash (\ell', n'_1, \ldots, n'_k)$. For the transitive closure of $\vdash$, we write $\vdash^+$. A run of a k-counter machine is a (finite or infinite) sequence of configurations $(\ell_0, n_1^0, \ldots, n_k^0), (\ell_1, n_1^1, \ldots, n_k^1) \cdots$, where $(\ell_0, n_1^0, \ldots, n_k^0)$ is the initial configuration, and, for all $i \geq 0$, $(\ell_i, n_1^i, \ldots, n_k^i) \vdash (\ell_{i+1}, n_1^{i+1}, \ldots, n_k^{i+1})$. The run is finite if and only if it ends in a halting configuration, i.e. in a configuration in HALT. It is well known that the halting problem for deterministic 2-counter machines, which consists in determining whether a given machine halts, is undecidable [19]. We will now see how we can encode such a problem to prove that the safety problem is undecidable in the asynchronous mode.

Theorem 2 *$SAFE_{\text{as}}$ is undecidable.*

The proof relies on a reduction from the halting problem of a deterministic two-counter machine M to $\mathsf{SAFE}_{\text{as}}$ with $k = 42$ robots. It is likely that an encoding using less robots might be used for the proof, but for the sake of clarity, we do not seek the smallest possible amount of robots. The idea is to simulate the run of M in a way that ensures that a collision occurs if and only if M halts. Positions of robots on the ring are used to encode values of counters and the current instruction of the machine. The protocol for k robots makes sure that movements of the robots simulate correctly the run of M. Moreover, one special robot moves only when the initial configuration is encoded, and another only when the final configuration is encoded. The collision is ensured in the following sequence of actions of the robots: when the initial configuration is encoded, the first robot computes its action but does not move immediately. When the halting configuration is reached, the second robot computes its action and moves, then the first robot finally completes its move, entailing the collision. Note that if the ring is not big enough to simulate the counter values then the halting configuration is never reached and there is no collision.

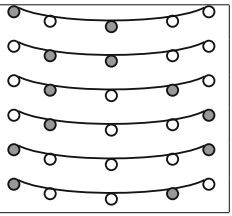
Instead of describing configurations of the robots by providing their respective positions on the ring, we use a sequence of letters F or R, representing respectively a free node and a node occupied by a robot. When a letter $\mathsf{A} \in \{\mathsf{F}, \mathsf{R}\}$ is repeated i times, we use the notation A^i, when it is repeated an arbitrary number of times (including 0), we use A^*. We also use the notation B_i as a shorthand for $\mathsf{FR}^i\mathsf{F}$, i.e. a block of i consecutive robots between two empty positions. These fixed blocks of robots will be used as immutable marks on the ring between which the other robots will move to encode configurations of the 2-counter machine. A *machine-like* configuration is a configuration of the form

$$\mathsf{B}_3\mathsf{F}^*\mathsf{R}\mathsf{F}^*\mathsf{B}_4\mathsf{F}^*\mathsf{R}\mathsf{F}^*\mathsf{B}_5\mathsf{F}^*\mathsf{R}\mathsf{F}^*\mathsf{B}_6\mathsf{F}^*\mathsf{R}\mathsf{F}^*\mathsf{B}_7\mathsf{F}^*\mathsf{R}\mathsf{F}^*\mathsf{B}_8\, P_1P_2P_3P_4P_5\mathsf{R}\mathsf{F}\mathsf{R},$$

where $P_1P_2 \in \{\mathsf{RF}, \mathsf{FR}\}$ and exactly one $P_i = \mathsf{R}$ for $i \in \{3, 4, 5\}$, and $P_j = \mathsf{F}$ for all $j \in \{3, 4, 5\} \backslash \{i\}$ (see Table 1 for a graphic representation of the section $P_1P_2P_3P_4P_5$ of the

Table 1 Different types of machine-like configurations

stable configuration	$\mathsf{R}_{tt}\mathsf{FR}_t\mathsf{FF}$
moving1 configuration	$\mathsf{FR}_{tt}\mathsf{R}_t\mathsf{FF}$
moving2 configuration	$\mathsf{FR}_{tt}\mathsf{FR}_t\mathsf{F}$
moving3 configuration	$\mathsf{FR}_{tt}\mathsf{FFR}_t$
stabilizing1 configuration	$\mathsf{R}_{tt}\mathsf{FFFR}_t$
stabilizing2 configuration	$\mathsf{R}_{tt}\mathsf{FFR}_t\mathsf{F}$



ring). In fact, thanks to our encoding, each robot sees a different sequence when it looks at the ring. For instance the first robot in the block B_3 sees on its 'right' the sequence

$$\mathsf{RRFF}^*\mathsf{RF}^*\mathsf{B}_4\mathsf{F}^*\mathsf{RF}^*\mathsf{B}_5\mathsf{F}^*\mathsf{RF}^*\mathsf{B}_6\mathsf{F}^*\mathsf{RF}^*\mathsf{B}_7\mathsf{F}^*\mathsf{RF}^*\mathsf{B}_8 P_1 P_2 P_3 P_4 P_5\mathsf{RFRF}$$

and the second robot of the same block sees

$$\mathsf{RFF}^*\mathsf{RF}^*\mathsf{B}_4\mathsf{F}^*\mathsf{RF}^*\mathsf{B}_5\mathsf{F}^*\mathsf{RF}^*\mathsf{B}_6\mathsf{F}^*\mathsf{RF}^*\mathsf{B}_7\mathsf{F}^*\mathsf{RF}^*\mathsf{B}_8 P_1 P_2 P_3 P_4 P_5\mathsf{RFRFR}$$

which is distinguishable from the view of the first one. The same reasoning can be applied to each pair of robots in our configuration. The blocks B_i are used to ensure that each robot can determine where is its place in the configuration. Hence, in the rest of the proof we abuse notations and describe the protocol using different names for the different robots, according to their position in the ring, even if they are formally anonymous.

We let $\mathcal{R}$ be the set of robots involved. A machine-like configuration

$$\mathsf{B}_3\mathsf{F}^{n_1}\mathsf{R}_{c_1}\mathsf{F}^*\mathsf{B}_4\mathsf{F}^{n_2}\mathsf{R}_{c_2}\mathsf{F}^*\mathsf{B}_5\mathsf{F}^m\mathsf{R}_c\mathsf{F}^n\mathsf{B}_6\mathsf{F}^i\mathsf{R}_\ell\mathsf{F}^{i'}\mathsf{B}_7\mathsf{F}^p\mathsf{R}_{\ell'}\mathsf{F}^r\mathsf{B}_8\mathsf{R}_{tt}\mathsf{FR}_t\mathsf{FFR}_g\mathsf{FR}_d$$

is said to be *stable* because of the positions of robots R_t and R_{tt} (see Table 1). Moreover, it encodes the configuration (ℓ_i, n_1, n_2) of M (due to the relative positions of robots R_{c_1}, R_{c_2} and R_ℓ respectively to B_3, B_4 and B_6). In the following, we distinguish between configurations of the 2-counter machine and configurations of the robots by calling them respectively M-configurations and ϕ-configurations. For a stable and machine-like ϕ-configuration $\mathbf{p}$, we let $M(\mathbf{p})$ be the M-configuration encoded by $\mathbf{p}$. We first present the part of the algorithm simulating the behavior of M. We call this algorithm ϕ'. Since the machine is deterministic, only one instruction is labelled by ℓ_i, known by every robot. The simulation follows different steps, according to the positions of the robots R_t and R_{tt}, as pictured in Table 1.

We explain the algorithm ϕ' on the configuration (ℓ_i, n_1, n_2) with the transition ℓ_i : `if` $c_1 > 0$ `then` $c_1 = c_1 - 1$; `goto` ℓ_j; `else goto` $\ell_{j'}$.

- When in a stable configuration, robot R_{tt} first moves to obtain a moving1 configuration.
- In a moving1 configuration, robot R_c moves until it memorizes the current value of c_1. More precisely, in a moving1 configuration where $n_1 \neq m$, robot R_c moves : if $n_1 > m$, and $n \neq 0$, R_c moves towards B_6, if $n_1 < m$, it moves towards B_5, if $n_1 > m$ and $n = 0$, it does not move.
- In a moving1 configuration where $n_1 = m$, R_t moves to obtain a moving2 configuration.
- In a moving2 configuration, if $n_1 = m \neq 0$, then R_{c_1} moves towards B_3, hence encoding the decrementation of c_1.
- In a moving2 configuration, if $n_1 = m = 0$ or if $n_1 \neq m$, (then the modification of c_1 is either impossible, or already done), robot $\mathsf{R}_{\ell'}$ moves until it memorizes the position of robot R_ℓ: if $p < i$, and $r \neq 0$, $\mathsf{R}_{\ell'}$ moves towards B_8; if $p > i$, $\mathsf{R}_{\ell'}$ moves towards B_7.

- In a moving2 configuration, if $p = i$ and ($n_1 = m = 0$ or $n_1 \neq m$), then R_t moves to obtain a moving3 configuration.
- In a moving3 configuration, if $n_1 = m = 0$, and robot $\mathsf{R}_{\ell'}$ encodes ℓ_i (i.e. $p = i$), then $c_1 = 0$ and robot R_ℓ has to move until it encodes $\ell_{j'}$. If on the other hand $n_1 < m$, then robot R_ℓ moves until it encodes ℓ_j. More precisely, if $n_1 = m = 0$, and the position encoded by R_ℓ is smaller than j' ($i < j'$), and if $i' \neq 0$, then R_ℓ moves towards B_7. If $n_1 = m = 0$, and the position encoded by R_ℓ is greater than j', R_ℓ moves towards B_6. If $n_1 \neq m$, then robot R_ℓ moves in order to reach a position where it encodes ℓ_j (towards B_6 if $i > j$, towards B_7 if $i < j$ and $i' \neq 0$).
- In a moving3 configuration, if the position encoded by $\mathsf{R}_{\ell'}$ is ℓ_i, if $n_1 = m = 0$ and the position encoded by R_ℓ is $\ell_{j'}$, or if $n_1 \neq m$, and the position encoded by R_ℓ is ℓ_j, then the transition has been completely simulated : the counters have been updated and the next transition is stored. The robots then return to a stable configuration: robot R_{tt} moves to obtain a stabilizing1 configuration.
- In a stabilizing1 configuration, robot R_t moves to obtain a stabilizing2 configuration.
- In a stabilizing2 configuration, robot R_t moves to obtain a stable configuration.

For other types of transitions, the robots move similarly. When in a stable configuration encoding a configuration in HALT, no robot moves. We describe now the algorithm ϕ that simply adds to ϕ' the two following rules. Robot R_g (respectively R_d) moves in the direction of R_d (respectively in the direction of R_g) if and only if the robots are in a stable machine-like configuration, and the encoded configuration of the machine is $(\ell_0, 0, 0)$ (respectively is in HALT). Recall that since the configuration is machine-like, the distance between R_g and R_d is 2.

On all configurations that are not machine-like, the algorithm makes sure that no robot moves. This implies that once R_g or R_d has moved, no robot with an up-to-date view ever moves. One can easily be convinced that the algorithm can be expressed by a QFP formula ϕ.

Let the formulae

$$\texttt{Bad}(p_1, \ldots, p_{42}) = \bigvee_{\substack{i,\, j \in [1,42] \\ i \neq j}} (p_i = p_j)$$

that is satisfied by all the configurations where two robots share the same position, and

$$\texttt{Ring}(y) = y \geq 0$$

We show that M halts if and only if there exists a size $n \in [\![\texttt{Ring}]\!]$, a $(42, n)$-configuration $\mathbf{p}$ with $\mathbf{p} \notin [\![\texttt{Bad}]\!]$, such that $\mathbf{Post}^*_{\text{as}}(\phi, \mathbf{p}) \cap [\![\texttt{Bad}]\!] \neq \emptyset$.

For that matter we use several claims about ϕ'. First observe that according to the different rules stable, moving1, etc., ϕ' is uniquely-sequentializable. This means that, in each configuration, at most one robot is programmed to move.

Then, thanks to Theorem 1, it is enough to study the synchronous successors of a configuration. It is then not difficult to establish the following claim.

Claim 1 *Let $\boldsymbol{p}$ be a machine-like ϕ-configuration. Then for every configuration $\boldsymbol{p}' \in \boldsymbol{Post}^*_{\text{as}}(\phi', \boldsymbol{p}) = \boldsymbol{Post}^*_{\text{s}}(\phi', \boldsymbol{p})$, $\boldsymbol{p}'$ is machine-like.*

The next claim states that, from a machine-like stable configuration encoding some M-configuration (ℓ, n_1, n_2), an asynchronous ϕ'-run will lead to the encoding of the successor M-configuration of (ℓ, n_1, n_2), provided that the ring on which robots evolve is big enough.

Claim 2 *Let $\langle \boldsymbol{p}, \boldsymbol{s}, \boldsymbol{V} \rangle$ be a stable and machine-like asynchronous ϕ-configuration with $\boldsymbol{p}$ of the following form:*

$$B_3 F^{n_1} R_{c_1} F^{n_1'} B_4 F^{n_2} R_{c_2} F^{n_2'} B_5 F^m R_c F^n B_6 F^i R_\ell F^{i'} B_7 F^p R_{\ell'} F^r B_8 R_{tt} F R_t F F R_g F R_d$$

such that $M(\boldsymbol{p}) = (\ell_i, n_1, n_2)$ is a non-halting M-configuration. Assume that $\boldsymbol{p}$ has the following properties:

(i) *if $n_1 > m$ then $n \geq n_1 - m$ (if ℓ_i modifies c_1) or if $n_2 > m$ then $n \geq n_2 - m$ (if ℓ_i modifies c_2),*
(ii) *if ℓ_i increments c_1 (resp. c_2) then $n_1' > 0$ (resp. $n_2' > 0$), and*
(iii) *$i + i' = p + r = |L|$, where L is the set of instructions of the 2-counter machine.*

Then there exists $\langle \boldsymbol{p}', \boldsymbol{s}', \boldsymbol{V}' \rangle$ such that $\langle \boldsymbol{p}, \boldsymbol{s}, \boldsymbol{V} \rangle \rightsquigarrow_{\phi'}^* \langle \boldsymbol{p}', \boldsymbol{s}', \boldsymbol{V}' \rangle$ with $\boldsymbol{p}'$ stable and machine-like, $\boldsymbol{s}'(R_g) = \boldsymbol{s}(R_g)$, $\boldsymbol{s}'(R_d) = \boldsymbol{s}(R_d)$, $\boldsymbol{V}'(R_g) = \boldsymbol{V}(R_g)$ and $\boldsymbol{V}'(R_d) = \boldsymbol{V}(R_d)$, such that $M(\boldsymbol{p}) \vdash M(\boldsymbol{p}')$.

Finally, we state the fact that, from an asynchronous configuration encoding (ℓ, n_1, n_2), the first stable asynchronous configuration reached in an asynchronous ϕ'-run indeed encodes the successor of (ℓ, n_1, n_2).

Claim 3 *Let $\langle \boldsymbol{p}, \boldsymbol{s}, \boldsymbol{V} \rangle$ and $\langle \boldsymbol{p}', \boldsymbol{s}', \boldsymbol{V}' \rangle$ be two asynchronous configurations, with $\boldsymbol{p}$ and $\boldsymbol{p}'$ stable. If there exists $k > 0$ such that $\langle \boldsymbol{p}, \boldsymbol{s}, \boldsymbol{V} \rangle \rightsquigarrow_{\phi'}^k \langle \boldsymbol{p}', \boldsymbol{s}', \boldsymbol{V}' \rangle$, then $M(\boldsymbol{p}) \vdash^+ M(\boldsymbol{p}')$.*

All these claims follow directly from the definition of ϕ'. We can now give the Proof of Theorem 2.

Proof of Theorem 2 Assume that M halts. There is then a finite bound $K \in \mathbb{N}$ on the values of the counters during the run. We show hereafter an asynchronous ϕ-run leading to a collision. Let $\langle \mathbf{p}_0, \mathbf{s}_0, \mathbf{V}_0 \rangle$ be the initial configuration of the run, with $\mathbf{p}_0$ encoding $(\ell_0, 0, 0)$, i.e. of the form:

$$\mathsf{B}_3 \mathsf{R}_{c_1} \mathsf{F}^K \mathsf{B}_4 \mathsf{R}_{c_2} \mathsf{F}^K \mathsf{B}_5 \mathsf{R}_c \mathsf{F}^K \mathsf{B}_6 \mathsf{R}_\ell \mathsf{F}^{|L|} \mathsf{B}_7 \mathsf{R}_{\ell'} \mathsf{F}^{|L|} \mathsf{B}_8 \mathsf{R}_{tt} \mathsf{F} \mathsf{R}_t \mathsf{F} \mathsf{F} \mathsf{R}_g \mathsf{F} \mathsf{R}_d,$$

(hence such that $M(\mathbf{p}_0) = (\ell_0, 0, 0)$), and $\mathbf{s}_0(i) = \mathbf{LK}$ for all $i \in \mathcal{R}$.
First, R_g takes a snapshot of the ring, i.e. $\langle \mathbf{p}_0, \mathbf{s}_0, \mathbf{V}_0 \rangle \rightsquigarrow \langle \mathbf{p}_1, \mathbf{s}_1, \mathbf{V}_1 \rangle$ where

$$\mathbf{p}_1 = \mathbf{p}_0,$$

$$\mathbf{s}_1(i) = \begin{cases} \mathbf{MV} & \text{if } i = \mathsf{R}_g \\ \mathbf{s}_0(i) & \text{otherwise,} \end{cases}$$

$$\mathbf{V}_1(i) = \begin{cases} \mathbf{V}_{\mathbf{p}_0}[g \rightarrow] & \text{if } i = \mathsf{R}_g \\ \mathbf{V}_0(i) & \text{otherwise.} \end{cases}$$

From now on, robots will simulate M. It is easy to check that $\mathbf{p}_1$ satisfies conditions (i), (ii) and (iii) of Claim 2. Then, $\langle \mathbf{p}_1, \mathbf{s}_1, \mathbf{V}_1 \rangle \rightsquigarrow_{\phi'}^* \langle \mathbf{p}_2, \mathbf{s}_2, \mathbf{V}_2 \rangle$, with $\mathbf{p}_2$ stable and machine-like, and such that $(\ell_0, 0, 0) \vdash M(\mathbf{p}_2)$. Again, one can check that $\mathbf{p}_2$ satisfies also conditions (i), (ii) and (iii) of Claim 2. Then, since M halts, by applying iteratively Claim 2, we have $\langle \mathbf{p}_1, \mathbf{s}_1, \mathbf{V}_1 \rangle \rightsquigarrow_{\phi'}^* \langle \mathbf{p}_n, \mathbf{s}_n, \mathbf{V}_n \rangle$ with $M(\mathbf{p}_n)$ a halting configuration. Moreover, by Claim 2, $\mathbf{s}_n(\mathsf{R}_d) = \mathbf{s}_0(\mathsf{R}_d) = \mathbf{LK}$, and $\mathbf{s}_n(\mathsf{R}_g) = \mathbf{s}_0(\mathsf{R}_g) = \mathbf{MV}$.

Now, the ϕ-run continues with robot R_d being scheduled to look, and then robots R_g and R_d moving, leading to a collision. Formally: $\langle \mathbf{p}_n, \mathbf{s}_n, \mathbf{V}_n \rangle \rightsquigarrow \langle \mathbf{p}, \mathbf{s}, \mathbf{V} \rangle \rightsquigarrow \langle \mathbf{p}', \mathbf{s}', \mathbf{V}' \rangle \rightsquigarrow \langle \mathbf{p}'', \mathbf{s}'', \mathbf{V}'' \rangle$, with

$$\mathbf{p} = \mathbf{p}_n$$
$$\mathbf{s}(i) = \begin{cases} \mathbf{MV} & \text{if } i = \mathsf{R}_d \\ \mathbf{s}_n(i) & \text{otherwise.} \end{cases}$$
$$\mathbf{V}(i) = \begin{cases} \mathbf{V}_{\mathbf{p}_n}[d \rightarrow] & \text{if } i = \mathsf{R}_d \\ \mathbf{V}_n(i) & \text{otherwise.} \end{cases}$$

Since $\mathbf{p}_n$ encodes a halting configuration, with distance between R_g and R_d equal to 2, applying the protocol makes R_d moves. Then,

$$\mathbf{p}'(i) = \begin{cases} \mathbf{p}(i) - 1 & \text{if } i = \mathsf{R}_d \\ \mathbf{p}(i) & \text{otherwise} \end{cases}$$
$$\mathbf{s}'(i) = \begin{cases} \mathbf{LK} & \text{if } i = \mathsf{R}_d \\ \mathbf{s}(i) & \text{otherwise.} \end{cases}$$
$$\mathbf{V}' = \mathbf{V}$$

Last, R_g completes the move computed from its view at the beginning of the run. Formally:

$$\mathbf{p}''(i) = \begin{cases} \mathbf{p}'(i) + 1 & \text{if } i = \mathsf{R}_g \\ \mathbf{p}'(i) & \text{otherwise} \end{cases}$$
$$\mathbf{s}''(i) = \begin{cases} \mathbf{LK} & \text{if } i = \mathsf{R}_g \\ \mathbf{s}'(i) & \text{otherwise.} \end{cases}$$
$$\mathbf{V}'' = \mathbf{V}'$$

since $\mathbf{V}''(\mathsf{R}_g) = \mathbf{V}_n(\mathsf{R}_g) = \mathbf{V}_1(\mathsf{R}_g) = \mathbf{V}_{\mathbf{p}_0}[\mathsf{R}_g \rightarrow]$, and $\mathbf{p}_0$ is a stable machine-like configuration such that $M(\mathbf{p}_0) = (\ell_0, 0, 0)$. Hence, $\mathbf{p}''(\mathsf{R}_g) = \mathbf{p}''(\mathsf{R}_d)$ and $\mathbf{p}'' \in \mathbf{Post}^*_{as}(\phi, \mathbf{p}_0) \cap [\![\mathtt{Bad}]\!]$.

Conversely assume there is an asynchronous ϕ-run ρ leading to a collision. We show first that ρ contains moves from R_g *and* R_d. By Claim 1, if ρ is an asynchronous ϕ'-run, there is no collision (either ρ starts in a non machine-like configuration and no robot moves, or it starts in a machine-like configuration and it never reaches a collision, since machine-like configurations are collision-free by construction). Hence ρ contains moves from R_g and/or R_d. Suppose by contradiction that R_g moves in ρ and not R_d. Before the first move of R_g, the run is a ϕ'-run. The prefix ending in the first move of R_g can be written $\rho' \cdot \langle \mathbf{p}_1, \mathbf{s}_1, \mathbf{V}_1 \rangle \langle \mathbf{p}_2, \mathbf{s}_2, \mathbf{V}_2 \rangle$ with $\mathbf{p}_2(\mathsf{R}_g) = \mathbf{p}_1(\mathsf{R}_g) - 1$ and $\mathbf{p}_2(i) = \mathbf{p}_1(i)$ for all $i \neq \mathsf{R}_g$, and $\rho' \cdot \langle \mathbf{p}_1, \mathbf{s}_1, \mathbf{V}_1 \rangle$ a prefix of a ϕ'-run. Then $\mathbf{p}_1$ is a machine-like configuration and there is at most one robot $i \notin \{\mathsf{R}_g, \mathsf{R}_d\}$ such that $move(\phi', \mathbf{V}_1(i)) = move(\phi', \mathbf{V}_2(i)) \neq \{0\}$. Once R_g has moved, no robot can take the decision to move anymore. Then $\rho = \rho' \cdot \langle \mathbf{p}_1, \mathbf{s}_1, \mathbf{V}_1 \rangle \langle \mathbf{p}_2, \mathbf{s}_2, \mathbf{V}_2 \rangle \langle \mathbf{p}_3, \mathbf{s}_3, \mathbf{V}_3 \rangle \cdot \rho''$ with $\mathbf{p}_3(i) = \mathbf{p}_2(i)$ for all the robots $i \in \mathcal{R}$ but possibly one, and then $\mathbf{p}_3$ being the only configuration appearing in ρ''. According to ϕ, $\mathbf{p}_3$ is collision free, so such a run could not lead to a collision.

We know now that R_d moves in this run. Hence, there is a configuration $\langle \mathbf{p}, \mathbf{s}, \mathbf{V} \rangle$ in ρ where R_d has just been scheduled to look and is hence such that $\mathbf{s}(\mathsf{R}_d) = \mathbf{MV}$, $\mathbf{V}(\mathsf{R}_d) = \mathbf{V}_{\mathbf{p}}[\mathsf{R}_d \rightarrow]$, with $move(\phi, \mathbf{V}(\mathsf{R}_d)) \neq 0$. By definition of ϕ, $\mathbf{p}$ is machine-like, stable and $M(\mathbf{p})$ is a halting M-configuration. Moreover, $\rho = \rho' \cdot \langle \mathbf{p}, \mathbf{s}, \mathbf{V} \rangle \cdot \rho''$, with ρ' a ϕ'-run. After $\langle \mathbf{p}, \mathbf{s}, \mathbf{V} \rangle$, the only robots that can move are R_g and R_d. Indeed, assume there is some robot $j \notin \{\mathsf{R}_g, \mathsf{R}_d\}$ such that $\mathbf{s}(j) = \mathbf{MV}$. If $move(\phi, V(j)) \neq \{0\}$, by Proposition 1, and since $move(\phi, V(j)) = move(\phi', V(j))$, $V(j) = \mathbf{V}_{\mathbf{p}}[j \rightarrow]$, and it is impossible since $M(\mathbf{p})$ is a

halting configuration. If R_g does not move, then all the following configurations in the run are equal to $\mathbf{p}'$ with $\mathbf{p}'(i) = \mathbf{p}(i)$ for all $i \neq \mathsf{R}_d$ and $\mathbf{p}'(\mathsf{R}_d) = \mathbf{p}(\mathsf{R}_d) + 1$. In that case, ρ does not lead to a collision, which is again a contradiction with the hypothesis.

Hence, we know that in ρ both R_g and R_d move. Moreover, we can order these actions in the run. From the above reasoning we deduce that in ρ, at some point R_g has been scheduled to look, then later on, R_d has been scheduled to look, and finally R_g and R_d have moved provoking a collision. Formally, ρ is in the following form: $\rho = \rho' \cdot \langle \mathbf{p}_0, \mathbf{s}_0, \mathbf{V}_0 \rangle \langle \mathbf{p}_1, \mathbf{s}_1, \mathbf{V}_1 \rangle \cdot \rho'' \cdot \langle \mathbf{p}_2, \mathbf{s}_2, \mathbf{V}_2 \rangle \langle \mathbf{p}_3, \mathbf{s}_3, \mathbf{V}_3 \rangle \langle \mathbf{p}_4, \mathbf{s}_4, \mathbf{V}_4 \rangle \langle \mathbf{p}_5, \mathbf{s}_5, \mathbf{V}_5 \rangle$ with ρ', ρ'' asynchronous ϕ'-runs, $\mathbf{s}_0(\mathsf{R}_g) = \mathbf{LK}$, $\mathbf{s}_1(\mathsf{R}_g) = \mathbf{MV}$, and for all other robots i, $\mathbf{s}_0(i) = \mathbf{s}_1(i)$, and $\mathbf{p}_0 = \mathbf{p}_1$, and $move(\phi, \mathbf{V}_1(\mathsf{R}_g)) \neq \{0\}$. After the sub-run ρ'', robot R_d looks, i.e. $\mathbf{p}_2 = \mathbf{p}_3$, $\mathbf{s}_2(\mathsf{R}_d) = \mathbf{LK}$, $\mathbf{s}_3(\mathsf{R}_d) = \mathbf{MV}$, $move(\phi, \mathbf{V}_3(\mathsf{R}_d)) \neq \{0\}$. Finally, configurations $\mathbf{p}_4$ and $\mathbf{p}_5$ correspond to movements of R_g and R_d in any order. For instance, and without loss of generality, $\mathbf{p}_4(\mathsf{R}_d) = \mathbf{p}_3(\mathsf{R}_d) + 1$, and for all $i \neq \mathsf{R}_d$, $\mathbf{p}_3(i) = \mathbf{p}_4(i)$ and $\mathbf{p}_5(\mathsf{R}_g) = \mathbf{p}_4(\mathsf{R}_g) - 1$ and for all $i \neq \mathsf{R}_g$, $\mathbf{p}_4(i) = \mathbf{p}_5(i)$. In both cases, since $move(\phi, \mathbf{V}_1(\mathsf{R}_g)) \neq \{0\}$, we deduce that $\mathbf{p}_0$ is a machine-like stable configuration such that $M(\mathbf{p}_0) = C_0$, and since $move(\phi, \mathbf{V}_3(\mathsf{R}_d)) \neq \{0\}$, then $\mathbf{p}_2$ is a machine-like stable configuration encoding a halting M-configuration C_h. Hence, from Claim 3, since $\mathbf{p}_0 \rightsquigarrow^+ \mathbf{p}_2$ then $C_0 \vdash^+ C_h$ and M halts. □

Observe that the previous result only holds for the asynchronous mode: the proof relies on the fact that a robot can have a look and perform the movement later on. As soon as robot R_g or robot R_d has moved, all the robots stop moving. Hence it is only because robot R_g can look at the beginning of the simulation, and then *stay idle for the whole simulation of the 2-counter machine*, until robot R_d looks at the moment where the halting configuration is encoded, that the collision can occur. We see now that for the reachability problem, the issue is different. In fact, the following theorem states that for all the three presented modes, this latter problem is undecidable.

Theorem 3 *$REACH_m$ is undecidable, for $m \in \{\text{ss}, \text{s}, \text{as}\}$.*

In that case, the proof relies on a reduction from the repeated reachability problem of a deterministic three-counter zero-initializing bounded-strongly-cyclic machine M, which is undecidable [17]. A (deterministic) counter machine is *zero-initializing* if from the initial location ℓ_0, it first sets all the counters to 0. Moreover, an infinite run is said to be *space-bounded* if there is a value $K \in \mathbb{N}$ such that all the values of all the counters stay below K during the run. A counter machine M is *bounded-strongly-cyclic* if, starting from any configuration, any space-bounded infinite run visits ℓ_0 infinitely often. The repeated reachability problem we consider is expressed as follows: given a 3-counter zero-initializing bounded-strongly-cyclic machine M without any halting configuration, does there exist an infinite space-bounded run of M starting from the initial configuration? A configuration of M is encoded in the same fashion as in the Proof of Theorem 2, with 3 robots encoding the values of the counters. A machine-like configuration in that case is of the form $\mathsf{B}_3\mathsf{F}^{n_1}\mathsf{R}_{c_1}\mathsf{F}^*\mathsf{B}_4\mathsf{F}^{n_2}\mathsf{R}_{c_2}\mathsf{F}^*\mathsf{B}_5\mathsf{F}^{n_3}\mathsf{R}_{c_3}\mathsf{F}^*\mathsf{B}_6\mathsf{F}^m\mathsf{R}_c\mathsf{F}^n\mathsf{B}_7\mathsf{F}^i\mathsf{R}_\ell\mathsf{F}^{i'}\mathsf{B}_8\mathsf{F}^p\mathsf{R}_{\ell'}\mathsf{F}^r\mathsf{B}_9\mathsf{R}_{tt}\mathsf{F}\mathsf{R}_t\mathsf{F}\mathsf{F}$. A transition of M is simulated by the algorithm in a similar way than in the Proof of Theorem 2, with the difference that *if a counter is supposed to be increased, the corresponding robot moves accordingly even if there is no room to do it, yielding a collision*. Hence, in any machine-like non stable configuration, exactly one robot moves and the only finite runs are ending in a collision, which is not machine-like. The algorithm that governs the robots in that case is called $\overline{\phi}$ and is also uniquely-sequentializable.

Let `Machine_like` be a three QFP formulae (with 50 free variables) such that $\mathbf{p} \in [\![\texttt{Machine\_like}]\!]$ if and only if $\mathbf{p}$ is machine-like. We then let `Goal` =

$\neg$`Machine_like` and `Ring(y)` $= y \geq 0$. We will also use a QFP formula `Collision` such that $\mathbf{p} \in [\![\texttt{Collision}]\!]$ if and only if $\mathbf{p}(i) = \mathbf{p}(j)$ for some $i, j \in \mathcal{R}$.

We now show that there is a size $n \in [\![\texttt{Ring}]\!]$, and a $(50, n)$-configuration $\mathbf{p}$ such that $\mathbf{Post}^*_{\mathrm{s}}(\overline{\phi}, \mathbf{p}) \cap [\![\texttt{Goal}]\!] = \emptyset$ if and only if there exists an infinite space-bounded run of M. From Theorem 1, since $\overline{\phi}$ is uniquely-sequentializable, this also provides an undecidability proof for $\mathsf{REACH}_{\mathrm{ss}}$ and $\mathsf{REACH}_{\mathrm{as}}$. We use the following claims, reminiscent of the claims used in the Proof of Theorem 2.

Claim 4 *Let $\boldsymbol{p}$ be a machine-like configuration. Then, for all configurations $\boldsymbol{p}' \in \boldsymbol{Post}^*_{\mathrm{s}}(\overline{\phi}, \boldsymbol{p})$, we have $\boldsymbol{p}' \in [\![\texttt{Machine\_like} \vee \texttt{Collision}]\!]$.*

The two following claims establish the correspondence between successive configurations of the counter machine and successive configurations of the robots.

Claim 5 *Let $\boldsymbol{p}$ be a stable and machine-like synchronous $\overline{\phi}$-configuration of the following form:*

$$B_3F^{n_1}R_{c_1}F^{n'_1}B_4F^{n_2}R_{c_2}F^{n'_2}B_5F^{n_3}R_{c_3}F^{n'_3}B_6F^{m}R_{c}F^{n}B_7F^{i}R_{\ell}F^{i'}B_8F^{p}R_{\ell'}F^{r}B_9R_{tt}FR_tFF.$$

Let $M(\boldsymbol{p}) = (\ell_i, n_1, n_2, n_3)$ be the encoded M-configuration. Assume that $\boldsymbol{p}$ has the following properties.

(i) *if $n_1 > m$ (respectively $n_2 > m$, $n_3 > m$), then $n \geq n_1 - m$ (respectively $n \geq n_2 - m$, $n \geq n_3 - m$)(if ℓ_i modifies c_1—respectively c_2 or c_3),*
(ii) *if ℓ_i increments c_1 (resp. c_2 or c_3) then $n'_1 > 0$ (resp. $n'_2 > 0$, or $n'_3 > 0$), and*
(iii) *$i + i' = p + r = |L|$.*

Then there exists a configuration $\boldsymbol{p}'$ stable and machine-like such that $\boldsymbol{p}' \in \boldsymbol{Post}^*_{\mathrm{s}}(\overline{\phi}, \boldsymbol{p})$ and $M(\boldsymbol{p}) \vdash M(\boldsymbol{p}')$.

Claim 6 *Let $\boldsymbol{p}, \boldsymbol{p}'$ be two stable, machine-like configurations. If there exists some $k > 0$ such that $\boldsymbol{p} \Rightarrow^k_{\overline{\phi}} \boldsymbol{p}'$ and for all $0 < j < k$, if $\boldsymbol{p} \Rightarrow^j_{\overline{\phi}} \boldsymbol{p}''$ then $\boldsymbol{p}''$ is not stable, then $M(\boldsymbol{p}) \vdash M(\boldsymbol{p}')$.*

The last claim formalizes the fact that the protocol makes the robots evolve infinitely from stable configuration to stable configuration, unless a collision occurs, which stops all the robots.

Claim 7 *Let $\boldsymbol{p}$ be a machine-like configuration, which is not stable. Then, either $|\boldsymbol{Post}^*_{\mathrm{s}}(\overline{\phi}, \boldsymbol{p})|$ is finite, or there exists $\boldsymbol{p}' \in \boldsymbol{Post}^*_{\mathrm{s}}(\overline{\phi}, \boldsymbol{p})$ with $\boldsymbol{p}'$ stable.*

We can now give the Proof of Theorem 3.

Proof of Theorem 3 Assume there is an infinite space-bounded run of M, and let $K \in \mathbb{N}$ be the maximal value of all the counters during this run. Let $\mathbf{p}_0$ be the ϕ-configuration having the following word-representation:

$$\mathsf{B}_3\mathsf{R}_{c_1}\mathsf{F}^K\mathsf{B}_4\mathsf{R}_{c_2}\mathsf{F}^K\mathsf{B}_5\mathsf{R}_{c_3}\mathsf{F}^K\mathsf{B}_6\mathsf{R}_{c}\mathsf{F}^K\mathsf{B}_7\mathsf{R}_{\ell}\mathsf{F}^{i'}\mathsf{B}_8\mathsf{F}^{p}\mathsf{R}_{\ell'}\mathsf{F}^{r}\mathsf{B}_9\mathsf{R}_{tt}\mathsf{FR}_t\mathsf{FF}.$$

Hence $M(\mathbf{p}_0)$ is the initial configuration of M. It is easy to show that, for all $\mathbf{p} \in \mathbf{Post}^*_{\mathrm{s}}(\overline{\phi}, \mathbf{p}_0)$, $\mathbf{p}$ satisfies conditions (i), (ii), and (iii) of Claim 5. Hence, by applying iteratively Claim 5, we can build an infinite $\overline{\phi}$-run $\rho = \mathbf{p}_0\mathbf{p}_1 \cdots$ such that $\mathbf{p}_i \notin [\![\texttt{Collision}]\!]$ for all $i \geq 0$. By Claim 4, $\mathbf{p}_i \in [\![\texttt{Machine\_like}]\!]$ for all $i \geq 0$, then $\mathbf{p}_i \notin [\![\texttt{Goal}]\!]$ for all $i \geq 0$. Moreover, for all machine-like configurations $\mathbf{p}$, for all $i \in \mathcal{R}$, $\mathbf{V_p}[i \rightarrow] \neq \mathbf{V_p}[\leftarrow i]$,

then $\mathbf{p}$ can have at most one successor by the relation $\Rightarrow_{\overline{\phi}}$, and ρ is the only ϕ-run starting from $\mathbf{p}_0$. Hence, $\mathbf{Post}^*_{\mathrm{s}}(\overline{\phi}, \mathbf{p}_0) \cap [\![\texttt{Goal}]\!] = \emptyset$.

Conversely, assume that there is $n \in \mathbb{N}$ and a $(50, n)$-configuration $\mathbf{p}_0$ such that $\mathbf{Post}^*_{\mathrm{s}}(\overline{\phi}, \mathbf{p}_0) \cap [\![\texttt{Goal}]\!] = \emptyset$. Hence, for all $\mathbf{p} \in \mathbf{Post}^*_{\mathrm{s}}(\overline{\phi}, \mathbf{p}_0)$, $\mathbf{p} \in [\![\texttt{Machine\_like}]\!]$. According to the definition of the protocol, there is a unique synchronous $\overline{\phi}$-run starting from $\mathbf{p}_0$. By definition of the protocol, all the machine-like configurations have a successor configuration. Hence, the run ρ starting from $\mathbf{p}_0$ is infinite. Let $\rho = \mathbf{p}_0\mathbf{p}_1 \cdots$ be such a run and assume that $\mathbf{p}_0$ is not stable. Then, by Claim 7, there exists $i \geq 0$, such that $\mathbf{p}_i$ is stable. By definition of $\overline{\phi}$, $\mathbf{p}_{i+1}$ is not stable, but by Claims 7 and 6, there exists $j > i + 1$ such that $\mathbf{p}_j$ is stable and $M(\mathbf{p}_i) \vdash M(\mathbf{p}_j)$. By iterating this reasoning, we can build an infinite run of M starting in $M(\mathbf{p}_i)$. Let K be the maximal number of positions between respectively B_3 and B_4, B_4 and B_5 and B_5 and B_6 in $\mathbf{p}_0$. It is easy to see that this distance is an invariant of any configuration in ρ. Hence, for any $k \geq 0$ such that $\mathbf{p}_k$ is stable, $M(\mathbf{p}_k) = (\ell, n_1, n_2, n_3)$ with $n_i \leq K$ for $i \in \{1, 2, 3\}$, and the infinite run of M built from ρ is indeed space-bounded. Let $C_0 \vdash C_1 \vdash \ldots$ be such a run. Since M is bounded-strongly-cyclic, there exists $i \geq 0$ such that $C_i = (\ell_0, n_1, n_2, n_3)$ with $n_i \in \mathbb{N}$ for $i = \{1, 2, 3\}$, and since M is zero-initializing, then there exists $j \geq i$ such that $C_j = (\ell_0, 0, 0, 0)$. Hence, M has an infinite space-bounded run from $(\ell_0, 0, 0, 0)$. □

4 Decidability results

In this section, we show that even though $\mathsf{SAFE}_{\mathrm{as}}$, $\mathsf{REACH}_{\mathrm{as}}$, $\mathsf{REACH}_{\mathrm{ss}}$ and $\mathsf{REACH}_{\mathrm{s}}$ are undecidable, the other cases $\mathsf{SAFE}_{\mathrm{s}}$ and $\mathsf{SAFE}_{\mathrm{ss}}$ can be reduced to the complement of the satisfiability problem for EP formulae, which is decidable and NP-complete [7]. This result may seem strange at a first sight but it can easily be explained by the fact that robots protocols are most of the time designed to work without any assumption on the initial configuration, except that it is not a bad configuration, hence we can restrict the analysis to one-step successor configurations.

4.1 Reducing safety to successor checking

The first step towards decidability is to remark that to solve $\mathsf{SAFE}_{\mathrm{s}}$ and $\mathsf{SAFE}_{\mathrm{ss}}$, it is enough to look at the one-step successor. Let ϕ be a protocol over k robots and `Ring` and `Bad` be respectively a ring property and a set of bad configurations. We have then the following lemma.

Lemma 1 *Let $n \in \mathbb{N}$ such that $n \in [\![\texttt{Ring}]\!]$ and $m \in \{\mathrm{s}, \mathrm{ss}\}$. There exists a (k, n)-configuration $\boldsymbol{p}$ with $\boldsymbol{p} \notin [\![\texttt{Bad}]\!]$, such that $\boldsymbol{Post}^*_m(\phi, \boldsymbol{p}) \cap [\![\texttt{Bad}]\!] \neq \emptyset$ iff there exists a (k, n)-configuration $\boldsymbol{p}'$ with $\boldsymbol{p}' \notin [\![\texttt{Bad}]\!]$, such that $\boldsymbol{Post}_m(\phi, \boldsymbol{p}') \cap [\![\texttt{Bad}]\!] \neq \emptyset$.*

Proof The $\Leftarrow$ direction is obvious. For the $\Rightarrow$ direction, if there exists a synchronous or semi-synchronous run $\rho = \mathbf{p}_0\mathbf{p}_1 \ldots \mathbf{p}_n$ with $\mathbf{p}_0 = \mathbf{p}$ and $\mathbf{p}_n$ being the first configuration in ρ belonging to $[\![\texttt{Bad}]\!]$, then by taking $\mathbf{p}' = \mathbf{p}_{n-1}$ we have $\mathbf{p}' \notin [\![\texttt{Bad}]\!]$ and $\mathbf{Post}_m(\phi, \mathbf{p}') \cap [\![\texttt{Bad}]\!] \neq \emptyset$. □

4.2 Encoding successor computation in Presburger

We now describe various EP formulae to be used to express the successor configuration in synchronous and semi-synchronous mode.

First we show how to express the view of some robot R_i in a configuration $\mathbf{p}$, with the following formula:

$$\begin{aligned}
&\texttt{ConfigView}_i(y, p_1, \ldots, p_k, d_1, \ldots, d_k)\\
&:= \exists d'_1, \ldots, d'_{k-1}, i_1, \ldots, i_{k-1}. \bigwedge_{j=1}^{k-2} d'_j \leq d'_{j+1}\\
&\wedge \bigwedge_{\ell=1}^{k-1} \left(\bigvee_{j=1, j\neq i}^{k} (p_j = p_i + d'_\ell \vee p_j + y = p_i + d'_\ell) \wedge i_\ell = j \right)\\
&\wedge 0 < d'_1 \wedge \bigwedge_{j=1}^{k-1} d'_j \leq y \wedge \bigwedge_{\ell\neq j} i_\ell \neq i_j\\
&\wedge d_1 = d'_1 \wedge \bigwedge_{j=2}^{k-1} d_j = d'_j - d'_{j-1} \wedge d_k = y - d'_{k-1}.
\end{aligned}$$

Note that this formula only expresses in the syntax of Presburger arithmetic the definition of $\mathbf{V_p}[i \rightarrow]$ where the variable y is used to store the length of the ring, $p_1, \ldots, p_k$ represent $\mathbf{p}$ and the variables $d_1, \ldots, d_k$ represent the view. In fact, we have the following statement.

Lemma 2 *For all $i \in [1, k]$, we have $n, \boldsymbol{p}, \boldsymbol{V} \models \texttt{ConfigView}_i$ if and only if $\boldsymbol{V} = \boldsymbol{V_p}[i \rightarrow]$ on a ring of size n.*

Proof Assume $n, \mathbf{p}, \mathbf{V} \models \texttt{ConfigView}_i$. Then, there exist $k-1$ variables, $d'_1, \ldots, d'_{k-1} \in [1, n]$ such that $0 < d'_1 \leq d'_2 \leq \cdots \leq d'_{k-1}$. Moreover, thanks to the hypothesis on the variables $i_1, \ldots, i_{k-1}$, we deduce that there exists a bijection $f : [1, k-1] \rightarrow [1, k-1]$ such that for all $j \neq i$ we have $i_{f(j)} = j$ and $p_j = (p_i + d'_{f(j)}) \mod n$. Finally, $d_1 = d'_1$ and for all $j \in [2, k-1], d_j = d'_j - d'_{j-1}$ and $d_k = n - d'_{k-1}$. Hence, if we consider the configuration $\mathbf{p}$ defined by $\mathbf{p}(j) = p_j$ for all $j \in [1, n]$, then $\langle d_1, \ldots, d_k\rangle = \mathbf{V_p}[i \rightarrow]$. Conversely, let $\mathbf{p}$ be a (k, n)-configuration and $\mathbf{V_p}[i \rightarrow] = \langle d_1, \ldots, d_k\rangle$. Then, $n, \mathbf{p}, \mathbf{V} \models \texttt{ConfigView}_i$. Indeed, by definition of the view, we let $d_i(j) \in [1, n]$ be such that $(\mathbf{p}(i) + d_i(j)) \odot n = \mathbf{p}(j)$ for all $j \neq i$ and we let $i_1, \ldots, i_{k-1}$ be a permutation of positions such that $0 < d_i(i_1) \leq d_i(i_2) \leq \cdots \leq d_i(i_{k-1})$. Then, for all $j \in [2, k-1]$, $d_j = d_i(i_j) - d_i(i_{j-1})$, $d_1 = d_i(i_1)$ and $d_k = n - d_i(i_{k-1})$. By interpreting the variables $d'_1, \ldots, d'_{k-1}$ by respectively $d_i(i_1), \ldots, d_i(i_{k-1})$, it is easy to see that the formula is satisfied. □

We also use the formula $\texttt{ViewSym}(d_1, \ldots, d_k, d'_1, \ldots, d'_k)$ to compute the symmetry of a view.

$$\begin{aligned}
&\texttt{ViewSym}(d_1, \ldots, d_k, d'_1, \ldots, d'_k)\\
&:= \bigvee_{j=1}^{k} \left(\bigwedge_{\ell=j+1}^{k} (d_\ell = 0 \wedge d'_\ell = 0) \wedge \bigwedge_{\ell=1}^{j} d'_\ell = d_{j-\ell+1} \right)
\end{aligned}$$

Lemma 3 *For all $n \in \mathbb{N}$, for all views $\boldsymbol{V}, \boldsymbol{V'} \in [0, n]^k$, we have $\boldsymbol{V}, \boldsymbol{V'} \models \texttt{ViewSym}$ if and only if $\boldsymbol{V'} = \overleftarrow{\boldsymbol{V}}$.*

Proof Given $n \in \mathbb{N}, d_1, \ldots, d_k, d'_1, \ldots, d'_k \in [0, n]$ such that $d_1 \neq 0$ we have $\langle d'_1, \ldots, d'_k \rangle = \overleftarrow{\langle d_1, \ldots, d_k \rangle}$ if and only if there exists $1 \leq j \leq k$ such that $d_\ell = 0$ for all $j+1 \leq \ell \leq k$ and $d'_1 = d_j$, ..., $d'_j = d_1$ and $d'_\ell = 0$ for all $j+1 \leq \ell \leq k$ (by definition), if and only if $d_1, \ldots, d_k, d'_1, \ldots, d'_k \models \texttt{ViewSym}$. □

We are now ready to introduce the formula $\texttt{Move}_i^\phi(y, p_1, \ldots, p_k, p')$, which is true if and only if, on a ring of size n (represented by the variable y), the move of robot R_i according to the protocol ϕ from the configuration $\mathbf{p}$ leads R_i to the new position p'. Here the variables $p_1, \ldots, p_k$ characterize $\mathbf{p}$.

$$
\begin{aligned}
&\texttt{Move}_i^\phi(y, p_1, \ldots, p_k, p') \\
&\quad := \exists d_1, \ldots, d_k, d'_1, \ldots, d'_k. \\
&\texttt{ConfigView}_i(y, p_1, \ldots, p_k, d_1, \ldots, d_k) \\
&\qquad \wedge \texttt{ViewSym}(d_1, \ldots, d_k, d'_1, \ldots, d'_k) \\
&\qquad \wedge \Big[\Big(\phi(d_1, \ldots, d_k) \wedge \big((p_i < y - 1 \wedge p' = p_i + 1) \\
&\qquad \vee (p_i = y - 1 \wedge p' = 0) \big) \Big) \\
&\qquad \vee \Big(\phi(d'_1, \ldots, d'_k) \wedge \big((p_i > 0 \wedge p' = p_i - 1) \\
&\qquad \vee (p_i = 0 \wedge p' = y - 1) \big) \Big) \\
&\qquad \vee \Big(\neg\phi(d_1, \ldots, d_k) \wedge \neg\phi(d'_1, \ldots, d'_k) \wedge (p' = p_i) \Big) \Big]
\end{aligned}
$$

Lemma 4 *For all $n \in \mathbb{N}$, for all (k, n)-configurations $\boldsymbol{p}$ and $p' \in [0, n-1]$, for all $i \in [1, k]$, we have $n, \boldsymbol{p}, p' \models \texttt{Move}_i^\phi$ if and only if $p' = (\boldsymbol{p}(i) + m) \odot n$ with $m \in move(\phi, \mathbf{V_p}[i \rightarrow])$.*

Proof We have $n, \mathbf{p}, \mathbf{p}' \models \texttt{Move}_i^\phi$ if and only if there exist $d_1, \ldots, d_k, d'_1, \ldots, d'_k \in [0, n]$ such that $\langle d_1, \ldots, d_k \rangle = \mathbf{V_p}[i \rightarrow]$ (by Lemma 2) and $\langle d'_1, \ldots, d'_k \rangle = \overleftarrow{\langle d_1, \ldots, d_k \rangle} = \mathbf{V_p}[\leftarrow i]$ (by Lemma 3) and either (*a*) $\mathbf{V_p}[i \rightarrow] \models \phi$ and $p' = (p_i + 1) \odot n$, or (*b*) $\mathbf{V_p}[\leftarrow i] \models \phi$ and $p' = (p_i - 1) \odot n$, or (*c*) $\mathbf{V_p}[i \rightarrow] \not\models \phi$, $\mathbf{V_p}[\leftarrow i] \not\models \phi$ and $p' = p_i$, if and only if either (*a*) $1 \in move(\phi, \mathbf{V_p}[i \rightarrow])$ and $p' = (p_i + 1) \odot n$ or (*b*) $-1 \in move(\phi, \mathbf{V_p}[i \rightarrow])$ and $p' = (p_i - 1) \odot n$ or (*c*) $move(\phi, \mathbf{V_p}[i \rightarrow]) = \{0\}$ and $p' = p_i$ if and only if $p' = (p_i + m) \odot n$ with $m \in move(\phi, \mathbf{V_p}[i \rightarrow])$. □

Now, given two (k, n)-configurations $\mathbf{p}$ and $\mathbf{p}'$, and a k-protocol ϕ, it is easy to express the fact that $\mathbf{p}'$ is a successor configuration of $\mathbf{p}$ according to ϕ in a semi-synchronous run (resp. synchronous run); for this we define the two formulae $\texttt{SemiSyncPost}_\phi(y, p_1, \ldots, p_k, p'_1, \ldots, p'_k)$ and $\texttt{SyncPost}_\phi(y, p_1 \ldots, p_k, p'_1, \ldots, p'_k))$ as follows:

$$
\begin{aligned}
&\texttt{SemiSyncPost}_\phi(y, p_1, \ldots, p_k, p'_1, \ldots, p'_k) \\
&\quad := \bigvee_{i=1}^{k} \Big(\texttt{Move}_i^\phi(y, p_1, \ldots, p_k, p'_i) \\
&\quad \wedge \bigwedge_{j=1, j \neq i}^{k} ((p'_j = p_j) \vee \texttt{Move}_j^\phi(y, p_1, \ldots, p_k, p'_j)) \Big)
\end{aligned}
$$

$$\begin{aligned}&\texttt{SyncPost}_\phi(y, p_1, \ldots, p_k, p'_1, \ldots, p'_k)\\ &:= \bigwedge_{i=1}^{k} \texttt{Move}_i^\phi(y, p_1, \ldots, p_k, p'_i)\end{aligned}$$

Lemma 5 *For all $n \in \mathbb{N}$ and all (k, n)-configurations $\boldsymbol{p}$ and $\boldsymbol{p}'$, we have:*

1. *$\boldsymbol{p} \hookrightarrow \boldsymbol{p}'$ if and only if $n, \boldsymbol{p}, \boldsymbol{p}' \models \texttt{SemiSyncPost}_\phi$,*
2. *$\boldsymbol{p} \Rightarrow \boldsymbol{p}'$ if and only if $n, \boldsymbol{p}, \boldsymbol{p}' \models \texttt{SyncPost}_\phi$.*

4.3 Results

Now, gathering the results above, in particular Lemmas 1 and 5, allows to conclude that $\mathsf{SAFE}_{\mathrm{ss}}$ and $\mathsf{SAFE}_{\mathrm{s}}$ can be expressed in Presburger arithmetic, hence the following theorem.

Theorem 4 *$\mathsf{SAFE}_{\mathrm{s}}$ and $\mathsf{SAFE}_{\mathrm{ss}}$ are decidable and in co-*NP.

Proof We consider a ring property $\texttt{Ring}(y)$, a protocol ϕ for k robots (which is a QFP formula) and a set of bad configurations given by a QFP formula $\texttt{Bad}(x_1, \ldots, x_k)$. We know by Lemma 1 that there exists a size $n \in \mathbb{N}$ with $n \in [\![\texttt{Ring}]\!]$, and a (k, n)-configuration $\mathbf{p}$ with $\mathbf{p} \notin [\![\texttt{Bad}]\!]$, such that $\mathbf{Post}_{\mathrm{s}}^*(\phi, \mathbf{p}) \cap [\![\texttt{Bad}]\!] \neq \emptyset$ if and only if there exists a (k, n)-configuration $\mathbf{p}'$ with $\mathbf{p}' \notin [\![\texttt{Bad}]\!]$, such that $\mathbf{Post}_m(\phi, \mathbf{p}') \cap [\![\texttt{Bad}]\!] \neq \emptyset$. By Lemma 5, this latter property is true if and only if the following formula is satisfiable:

$$\begin{aligned}&\texttt{SyncPost}_\phi(y, p_1, \ldots, p_k, p'_1, \ldots, p'_k)\\ &\quad \wedge \texttt{Ring}(y) \wedge \neg\texttt{Bad}(p_1, \ldots, p_k)\\ &\quad \wedge \texttt{Bad}(p'_1, \ldots, p'_k)\end{aligned}$$

For the semi-synchronous case, we replace $\texttt{SyncPost}_\phi$ by $\texttt{SemiSyncPost}_\phi$. The co- NP upper bound is obtained by the fact that the built formula is an EP formula of polynomial size if the size of the formulae ϕ, $\texttt{Ring}$ and $\texttt{Goal}$ and that the satisfiability problem for EP formulae is NP-complete [7]. □

4.4 Restrictions to deal with asynchrony

We have seen that when we consider the asynchronous mode, then $\mathsf{SAFE}_{\mathrm{as}}$ is undecidable. However, decidability can be regained by considering some restrictions on the protocol. First, when it is uniquely-sequentializable, i.e. when in each configuration at most one robot can decide to move then Theorems 1 and 4 lead us to the following result.

Corollary 1 *When the protocol ϕ is uniquely-sequentializable, $\mathsf{SAFE}_{\mathrm{as}}$ is decidable.*

We will see that it is possible to relax a bit this restriction by allowing more robots to be able to move in a given configuration, but only if the following points are respected: the movement of a strict subset of these robots does not allow new robots to move and does not make the robots that could move originally change their move direction. This restriction can be formalized as follows.

Definition 3 A protocol ϕ is *pending-bounded* if, for every configuration $\mathbf{p}$, for all configuration $\mathbf{p}' \in \mathbf{Post}_{\mathrm{ss}}(\mathbf{p}) \backslash \mathbf{Post}_{\mathrm{s}}(\mathbf{p})$, for every $i \in \mathcal{R}$, $move(\phi, \mathbf{V}_{\mathbf{p}'}[i \rightarrow]) \subseteq move(\phi, \mathbf{V}_{\mathbf{p}}[i \rightarrow])$ and if $\mathbf{p}(i) \neq \mathbf{p}'(i)$ then $i \notin \mathrm{Act}_\phi(\mathbf{p}')$.

We will now show that this new restriction, which is more general than the property of being uniquely-sequentializable, allows us to regain decidability for the safety problem in asynchronous mode.

Theorem 5 *When a protocol ϕ is pending-bounded, then $\boldsymbol{Post}^*_{\mathrm{as}}(\phi, \boldsymbol{p}) = \boldsymbol{Post}^*_{\mathrm{ss}}(\phi, \boldsymbol{p})$ for all configurations $\boldsymbol{p}$.*

In order to prove this theorem, we will need an intermediate proposition. When all the robots are in an internal state where they are ready to look, the corresponding configuration can be an initial configuration of the run. The following proposition establishes that, when a protocol is pending-bounded, in a finite asynchronous run in which no configuration (apart from the first one) could be an initial configuration, any reachable configuration could be reached in one semi-synchronous step from the initial configuration.

Proposition 2 *Let ϕ be a pending-bounded protocol, and consider an asynchronous ϕ-run $\rho = \langle \boldsymbol{p}_0, \boldsymbol{s}_0, \boldsymbol{V}_0\rangle\langle \boldsymbol{p}_1, \boldsymbol{s}_1, \boldsymbol{V}_1\rangle \cdots \langle \boldsymbol{p}_L, \boldsymbol{s}_L, \boldsymbol{V}_L\rangle \in \boldsymbol{Runs}_{\mathrm{as}}(\phi)$ satisfying the two following conditions:*

1. *for all $0 < \ell < L$, there exists $i \in \mathcal{R}$ such that $\boldsymbol{s}_\ell(i) = \boldsymbol{MV}$.*
2. *there is no $0 < \ell \leq L$ and no $i \in \mathcal{R}$ such that $\boldsymbol{s}_{\ell-1}(i) = \boldsymbol{LK}$, $\boldsymbol{s}_\ell(i) = \boldsymbol{MV}$ and $\mathit{move}(\phi, \boldsymbol{V}_\ell(i)) = \{0\}$.*

Then $\boldsymbol{p}_\ell \in \boldsymbol{Post}_{\mathrm{ss}}(\phi, \boldsymbol{p}_0)$ for all $0 < \ell \leq L$.

Proof We consider a ϕ-run $\rho = \langle \mathbf{p}_0, \mathbf{s}_0, \mathbf{V}_0\rangle\langle \mathbf{p}_1, \mathbf{s}_1, \mathbf{V}_1\rangle \cdots \langle \mathbf{p}_L, \mathbf{s}_L, \mathbf{V}_L\rangle$ respecting the stated hypothesis. Observe that any asynchronous run is in fact a succession of sequences of robots that look and sequences of robots that move. We will now consider ρ as a sequence $\langle \mathbf{p}_0, \mathbf{s}_0, \mathbf{V}_0\rangle \overset{\mathbf{LK}_1}{\rightsquigarrow_\phi} \langle \mathbf{p}'_1, \mathbf{s}'_1, \mathbf{V}'_1\rangle \overset{\mathbf{MV}_1}{\rightsquigarrow_\phi} \langle \mathbf{p}'_2, \mathbf{s}'_2, \mathbf{V}'_2\rangle \overset{\mathbf{LK}_2}{\rightsquigarrow_\phi} \cdots \overset{\mathbf{MV}_m}{\rightsquigarrow_\phi} \langle \mathbf{p}'_{2m}, \mathbf{s}'_{2m}, \mathbf{V}'_{2m}\rangle$ where $\mathbf{LK}_j$ is a set of robots that perform a look action, and $\langle \mathbf{p}'_{2j-1}, \mathbf{s}'_{2j-1}, \mathbf{V}'_{2j-1}\rangle$ is the configuration reached from $\langle \mathbf{p}'_{2j-2}, \mathbf{s}'_{2j-2}, \mathbf{V}'_{2j-2}\rangle$ when all the robots from $\mathbf{LK}_j$ have looked (the order does not matter). In fact $\overset{\mathbf{LK}_j}{\rightsquigarrow_\phi}$ is a macro-transition gathering several consecutive transitions from the original run. Similarly, $\mathbf{MV}_j$ is a set of robots that perform a move action, and $\langle \mathbf{p}'_{2j}, \mathbf{s}'_{2j}, \mathbf{V}'_{2j}\rangle$ is the configuration reached from $\langle \mathbf{p}'_{2j-1}, \mathbf{s}'_{2j-1}, \mathbf{V}'_{2j-1}\rangle$ when all the robots from $\mathbf{MV}_j$ have moved. The intermediate configurations successively reached while executing the different look or move actions gathered in this macro transition are of no interest for the proof.

We now show by induction, for all $0 < \ell < m$:

$P_1(\ell)$ for every robot $i \in \bigcup_{j=1}^{\ell} \mathbf{LK}_j$, $\mathit{move}(\phi, \mathbf{V}'_{2\ell-1}(i)) \subseteq \mathit{move}(\phi, \mathbf{V}_{\mathbf{p}_0}[i \to])$,

$P_2(\ell)$ $\mathbf{p}'_{2\ell} \in \mathbf{Post}_{\mathrm{ss}}(\phi, \mathbf{p}_0) \backslash \mathbf{Post}_{\mathrm{s}}(\phi, \mathbf{p}_0)$, with $\bigcup_{s=1}^{\ell} \mathbf{MV}_s$ the set of robots that moved in the semi-synchronous transition.

If $\ell = 1$, for all $i \in \mathbf{LK}_\ell$, $\mathbf{V}'_1(i) = \mathbf{V}_{\mathbf{p}_0}[i \to]$, hence $P_1(1)$ is true. Moreover, $\mathbf{p}'_2(i) = \mathbf{p}_0(i)$ if $i \notin \mathbf{MV}_1$ and $\mathbf{p}'_2(i) = (\mathbf{p}'_1(i) + m) \odot n$, with $m \in \mathit{move}(\phi, \mathbf{V}'_1(i))$, if $i \in \mathbf{MV}_1$. Since $\mathbf{p}'_1 = \mathbf{p}_0$, $\mathbf{MV}_1 \subseteq \mathbf{LK}_1$ by definition, and $\mathbf{V}'_1(i) = \mathbf{V}_{\mathbf{p}_0}[i \to]$ for all $i \in \mathbf{MV}_1$, then $\mathbf{p}'_2 \in \mathbf{Post}_{\mathrm{ss}}(\phi, \mathbf{p}_0)$. Finally, since there exists $i \in \mathcal{R}$ such that $\mathbf{s}'_2(i) = \mathbf{MV}$, and since we have assumed that if $i \in \mathbf{LK}_1$, then $\mathit{move}(\phi, \mathbf{V}'_1(i)) \neq \{0\}$, we also have $\mathbf{p}'_2 \notin \mathbf{Post}_{\mathrm{s}}(\phi, \mathbf{p}_0)$ and $P_2(1)$ is also true.

Let now $1 \leq \ell < m$ and assume that $P_1(j)$ and $P_2(j)$ are true for all $1 \leq j \leq \ell$. We first establish the following facts, that will be useful for the induction step.

– For all $j \leq \ell+1$, $\mathbf{LK}_j \subseteq \mathrm{Act}_\phi(\mathbf{p}_0)$. Indeed, let $j \leq \ell+1$, then by the second condition on the run considered in the proposition, we know that $\mathbf{LK}_j \subseteq \mathrm{Act}_\phi(\mathbf{p}'_{2j-2})$. Moreover, by induction hypothesis, $\mathbf{p}'_{2j-2} \in \mathbf{Post}_{\mathrm{ss}}(\phi, \mathbf{p}_0) \backslash \mathbf{Post}_{\mathrm{s}}(\phi, \mathbf{p}_0)$. Let $i \in \mathrm{Act}_\phi(\mathbf{p}'_{2j-2})$. Then $move(\phi, \mathbf{V}_{\mathbf{p}'_{2j-2}}[i \to]) \neq \{0\}$ and since ϕ is pending-bounded, $move(\phi, \mathbf{V}_{\mathbf{p}'_{2j-2}}[i \to]) \subseteq move(\phi, \mathbf{V}_{\mathbf{p}_0}[i \to])$. Hence, $i \in \mathrm{Act}_\phi(\mathbf{p}_0)$ and $\mathbf{LK}_j \subseteq \mathrm{Act}_\phi(\mathbf{p}_0)$. This fact also yields $\bigcup_{s=1}^{\ell+1} \mathbf{MV}_s \subseteq \bigcup_{s=1}^{\ell+1} \mathbf{LK}_s \subseteq \mathrm{Act}_\phi(\mathbf{p}_0)$.
– For all $j, j' \leq \ell+1$, $\mathbf{LK}_j \cap \mathbf{LK}_{j'} = \emptyset$ and $\mathbf{MV}_j \cap \mathbf{MV}_{j'} = \emptyset$. For the sake of contradiction, let $i \in \mathbf{LK}_j \cap \mathbf{LK}_{j'}$, and assume that $j < j'$. Then there exists some $j \leq r < j'$ such that $i \in \mathbf{MV}_r$, because the robot has to execute its move before looking again. By induction hypothesis, $\mathbf{p}'_{2r} \in \mathbf{Post}_{\mathrm{ss}}(\phi, \mathbf{p}_0) \backslash \mathbf{Post}_{\mathrm{s}}(\phi, \mathbf{p}_0)$. Since ϕ is pending-bounded and $\mathbf{p}'_{2r}(i) \neq \mathbf{p}_0(i)$ because robot i moved in that step, then $i \notin \mathrm{Act}_\phi(\mathbf{p}'_{2r}(i))$. By iterating this reasoning for every position between r and $j'-1$, we obtain that $i \notin \mathrm{Act}_\phi(\mathbf{p}'_{2j'-2})$, and hence $i \notin \mathbf{LK}_{j'}$, which contradicts the hypothesis. Hence $\mathbf{LK}_j \cap \mathbf{LK}_{j'} = \emptyset$. Similarly, if $i \in \mathbf{MV}_j \cap \mathbf{MV}_{j'}$, then robot i has necessarily looked before the move in the macro-step $\mathbf{MV}_j$ and between the two moves, and there exist some $r \leq j \leq r' \leq j'$ with $i \in \mathbf{LK}_r \cap \mathbf{LK}'_r$. Since $\mathbf{LK}_r \cap \mathbf{LK}_{r'} = \emptyset$, this is impossible too.
– Finally, we show that $\bigcup_{s=1}^{\ell+1} \mathbf{MV}_s \subsetneq \mathrm{Act}_\phi(\mathbf{p}_0)$. We already know from the first item that $\bigcup_{s=1}^{\ell+1} \mathbf{MV}_s \subseteq \mathrm{Act}_\phi(\mathbf{p}_0)$. Assume for the sake of contradiction that $\bigcup_{s=1}^{\ell+1} \mathbf{MV}_s = \mathrm{Act}_\phi(\mathbf{p}_0)$. For $i \in \mathrm{Act}_\phi(\mathbf{p}_0)$, let $s_i \leq \ell+1$ be the unique index such that $i \in \mathbf{MV}_{s_i}$. Then $\mathbf{s}'_{2s_i} = \mathbf{LK}$, and since the elements of $\{\mathbf{LK}_s \mid 1 \leq s \leq \ell+1\}$ are pairwise distinct, $\mathbf{s}'_r = \mathbf{LK}$ for all $2s_i \leq r \leq 2\ell+2$. Finally, $\mathbf{s}'_{2\ell+2}(i) = \mathbf{LK}$ for all $i \in \mathrm{Act}_\phi(\mathbf{p}_0)$, and since $\bigcup_{s=1}^{\ell+1} \mathbf{LK}_s \subseteq \mathrm{Act}_\phi(\mathbf{p}_0)$, $\mathbf{s}'_{2\ell+2}(i) = \mathbf{LK}$ for all $i \in \mathcal{R}$. This contradicts the hypothesis on the considered run. Hence $\bigcup_{s=1}^{\ell+1} \mathbf{MV}_s \subsetneq \mathrm{Act}_\phi(\mathbf{p}_0)$.

We have now the elements to show the induction step. Let $i \in \bigcup_{j=1}^{\ell+1} \mathbf{LK}_j$. If $i \in \mathbf{LK}_{\ell+1}$, $\mathbf{V}'_{2\ell+1}(i) = \mathbf{V}_{\mathbf{p}'_{2\ell}}[i \to]$ by definition. By $P_2(\ell)$, we know that $\mathbf{p}'_{2\ell} \in \mathbf{Post}_{\mathrm{ss}}(\phi, \mathbf{p}_0) \backslash \mathbf{Post}_{\mathrm{s}}(\phi, \mathbf{p}_0)$ and, since the protocol is pending-bounded, $move(\phi, \mathbf{V}_{\mathbf{p}'_{2\ell}}[i \to]) \subseteq move(\phi, \mathbf{V}_{\mathbf{p}_0}[i \to])$. Otherwise, there exists a unique j such that $i \in \mathbf{LK}_j$ and $\mathbf{V}'_{2\ell+1}(i) = \mathbf{V}'_{2j-1}(i) = \mathbf{V}_{\mathbf{p}'_{2j-2}}[i \to]$. By $P_1(2j-2)$, $move(\phi, \mathbf{V}_{\mathbf{p}'_{2j-2}}[i \to]) \subseteq move(\phi, \mathbf{V}_{\mathbf{p}_0}[i \to])$. Hence $move(\phi, \mathbf{V}'_{2\ell+1}(i)) \subseteq move(\phi, \mathbf{V}_{\mathbf{p}_0}[i \to])$ and $P_1(\ell+1)$ is true.

To show $P_2(\ell+1)$, we examine the value of $\mathbf{p}'_{2\ell+2}$. Let $i \in \mathcal{R}$. Then,

$$\mathbf{p}'_{2\ell+2}(i) = \begin{cases} \mathbf{p}'_{2\ell}(i) & \text{if } i \notin \mathbf{MV}_{\ell+1} \\ (\mathbf{p}'_{2\ell}(i) + m) \odot n, \text{ with } m \in move(\phi, \mathbf{V}'_{2\ell+1}(i)) & \text{otherwise.} \end{cases}$$

Let $i \in \mathbf{MV}_{\ell+1} \subseteq \bigcup_{j=1}^{\ell+1} \mathbf{LK}_j$. Since the $\mathbf{MV}_j$ are pairwise distinct, $i \notin \bigcup_{j=1}^{\ell} \mathbf{MV}_j$, and by $P_1(\ell+1)$, $move(\phi, \mathbf{V}'_{2\ell+1}(i)) \subseteq move(\phi, \mathbf{V}_{\mathbf{p}_0}[i \to])$. By $P_2(\ell)$,

$$\mathbf{p}'_{2\ell}(i) = \begin{cases} (\mathbf{p}_0(i) + m) \odot n, m \in move(\phi, \mathbf{V}_{\mathbf{p}_0}[i \to]) & \text{if } i \in \bigcup_{j=1}^{\ell} \mathbf{MV}_j \\ \mathbf{p}_0(i) & \text{otherwise.} \end{cases}$$

Gathering these observations, we obtain

$$\mathbf{p}'_{2\ell+2}(i) = \begin{cases} (\mathbf{p}_0(i) + m) \odot n, m \in move(\phi, \mathbf{V}_{\mathbf{p}_0}[i \to]) & \text{if } i \in \bigcup_{j=1}^{\ell+1} \mathbf{MV}_j \\ \mathbf{p}_0(i) & \text{otherwise.} \end{cases}$$

We can conclude then that $\mathbf{p}'_{2\ell+2} \in \mathbf{Post}_{\mathrm{ss}}(\phi, \mathbf{p}_0)$. Since $\bigcup_{j=1}^{\ell+1} \mathbf{MV}_j \subsetneq \mathrm{Act}_\phi(\mathbf{p}_0)$, not all the activatable robots have moved in this portion of run, hence $\mathbf{p}'_{2\ell+2} \notin \mathbf{Post}_{\mathrm{s}}(\phi, \mathbf{p}_0)$ and $P_2(\ell+1)$ is true.

Il $\ell = m$, it is easy to show that $\mathbf{p}'_{2\ell} \in \mathbf{Post}_{\mathrm{ss}}(\phi, \mathbf{p}_0)$ (and maybe also in $\mathbf{Post}_{\mathrm{s}}(\phi, \mathbf{p}_0)$). Indeed, we can show $P_1(m)$ thanks to $P_2(m-1)$ and all the $P_1(j)$, $0 < j < m$, like in the induction step. Then, with the same reasoning than in the induction step for P_2, we can show that $\mathbf{p}'_{2m} \in \mathbf{Post}_{\mathrm{ss}}(\phi, \mathbf{p}_0)$ (only the fact that $\bigcup_{j=1}^{m} \mathbf{MV}_j \subsetneq \mathrm{Act}_\phi(\mathbf{p}_0)$ is not ensured). So we have the property

$P'(\ell)$ $\mathbf{p}'_{2\ell} \in \mathbf{Post}_{\mathrm{ss}}(\phi, \mathbf{p}_0)$ for all $0 < j \leq m$.

Thanks to this, we can show the proposition. Let $0 < \ell \leq L$.

- If $\mathbf{p}_\ell = \mathbf{p}'_{2k}$ for some $0 < k \leq m$, then by $P'(k)$, $\mathbf{p}_\ell \in \mathbf{Post}_{\mathrm{ss}}(\phi, \mathbf{p}_0)$.
- If $\mathbf{p}_\ell = \mathbf{p}'_{2k+1}$ for some $0 \leq k < m$, then by construction, $\mathbf{p}'_{2k+1} = \mathbf{p}'_{2k}$, and again by $P'(k)$, $\mathbf{p}_\ell \in \mathbf{Post}_{\mathrm{ss}}(\phi, \mathbf{p}_0)$.
- If $\mathbf{p}_\ell$ corresponds to a configuration reached in the middle of a macro-transition $\mathbf{MV}_j$, it means that the run looks like $\langle \mathbf{p}_0, \mathbf{s}_0, \mathbf{V}_0 \rangle \overset{\mathbf{LK}_1}{\rightsquigarrow_\phi} \langle \mathbf{p}'_1, \mathbf{s}'_1, \mathbf{V}'_1 \rangle \overset{\mathbf{MV}_1}{\rightsquigarrow_\phi} \langle \mathbf{p}'_2, \mathbf{s}'_2, \mathbf{V}'_2 \rangle \overset{\mathbf{LK}_2}{\rightsquigarrow_\phi} \cdots \overset{\mathbf{LK}_j}{\rightsquigarrow_\phi} \langle \mathbf{p}'_{2j-1}, \mathbf{s}'_{2j-1}, \mathbf{V}'_{2j-1} \rangle \overset{\mathbf{MV}'_j}{\rightsquigarrow_\phi} \langle \mathbf{p}_\ell, \mathbf{s}_\ell, \mathbf{V}_\ell \rangle \cdots$ with $\mathbf{MV}'_j \subsetneq \mathbf{MV}_j$. Then, $\mathbf{MV}'_j \subseteq \bigcup_{s=1}^{j} \mathbf{LK}_s$. Let $i \in \mathbf{MV}'_j$ the last robot that moved and led to $\mathbf{p}_\ell$. Then, by $P_1(j)$, $move(\phi, \mathbf{V}'_{2j-1}) \subseteq move(\phi, \mathbf{V}_{\mathbf{p}_0}[i \rightarrow])$. By $P_2(j-1)$, $\mathbf{p}'_{2j-2} \in \mathbf{Post}_{\mathrm{ss}}(\phi, \mathbf{p}_0) \backslash \mathbf{Post}_{\mathrm{s}}(\phi, \mathbf{p}_0)$. As above, these two facts allow with a simple computation to prove that $\mathbf{p}_\ell \in \mathbf{Post}_{\mathrm{ss}}(\phi, \mathbf{p}_0)$. □

Equipped with this result, we can now show easily Theorem 5.

Proof of Theorem 5 We show that for all configurations $\mathbf{p}$, $\mathbf{Post}^*_{\mathrm{as}}(\phi, \mathbf{p}) \subseteq \mathbf{Post}^*_{\mathrm{ss}}(\phi, \mathbf{p})$, the inverse inclusion being straightforward. Consider an asynchronous ϕ-run $\rho = \langle \mathbf{p}_0, \mathbf{s}_0, \mathbf{V}_0 \rangle \langle \mathbf{p}_1, \mathbf{s}_1, \mathbf{V}_1 \rangle \cdots \langle \mathbf{p}_m, \mathbf{s}_m, \mathbf{V}_m \rangle \in \mathbf{Runs}_{\mathrm{as}}(\phi)$. First we assume that this run respects the condition (2) of Proposition 2, which is that no robot is scheduled to look unless it induces an actual move, i.e. for all $0 \leq k \leq m$, if $\mathbf{s}_k(i) = \mathbf{MV}$, then $move(\phi, \mathbf{V}_k(i)) \neq \{0\}$. If this is not the case, we can modify the run by deleting the look action and the subsequent move (if it exists) without modifying the reached configurations. We show the property by induction of the length of ρ. If $m = 0$, it is trivial. Assume now by induction hypothesis that $\mathbf{p}_j \in \mathbf{Post}^*_{\mathrm{ss}}(\phi, \mathbf{p}_0)$, for all $j < m$. Let $0 < \ell < m$ be the largest index such that $\mathbf{s}_\ell(i) = \mathbf{LK}$ for all $i \in \mathcal{R}$. If no such index exists, then, by Proposition 2, $\mathbf{p}_m \in \mathbf{Post}_{\mathrm{ss}}(\mathbf{p}_0)$. Otherwise, by induction hypothesis, $\mathbf{p}_\ell \in \mathbf{Post}^*_{\mathrm{ss}}(\phi, \mathbf{p}_0)$. Moreover, $\langle \mathbf{p}_\ell, \mathbf{s}_\ell, \mathbf{V}_\ell \rangle \ldots \langle \mathbf{p}_m, \mathbf{s}_m, \mathbf{V}_m \rangle$ is an asynchronous ϕ-run such that, for all $\ell < j < m$, there exists $i \in \mathcal{R}$ such that $\mathbf{s}_j(i) = \mathbf{MV}$. Note that this run is well initialized because we have $\mathbf{s}_\ell(i) = \mathbf{LK}$ for all $i \in \mathcal{R}$. Then from Proposition 2, we deduce that $\mathbf{p}_m \in \mathbf{Post}_{\mathrm{ss}}(\phi, \mathbf{p}_\ell)$, hence $\mathbf{p}_m \in \mathbf{Post}^*_{\mathrm{ss}}(\phi, \mathbf{p}_0)$. □

Again, from Theorems 4 and 5 we immediately obtain the following result.

Corollary 2 *When the protocol ϕ is pending-bounded, $\mathsf{SAFE}_{\mathrm{as}}$ is decidable.*

4.5 Using logic to verify other interesting properties

Not only the method consisting in expressing the successor computation in Presburger arithmetic allows us to obtain the decidability for $\mathsf{SAFE}_{\mathrm{s}}$ and $\mathsf{SAFE}_{\mathrm{ss}}$, as well as for $\mathsf{SAFE}_{\mathrm{as}}$ in some restricted cases, but it also allows us to express other interesting properties. For instance, we

can compute the successor configuration in asynchronous mode for a protocol ϕ working over k robots thanks to the following formula:

$$\begin{aligned}&\texttt{AsyncPost}_\phi(y, p_1, \ldots, p_k, s_1, \ldots, s_k, v_1 \ldots, v_k, p'_1, \ldots, p'_k, s'_1, \ldots, s'_k, v'_1, \ldots, v'_k)\\ &:= \exists d_1, \ldots, d_k \cdot \bigvee_{i=1}^{k} \Big(\bigwedge_{j \neq i} (p'_j = p_j \wedge s'_j = s_j \wedge v'_j = v_j)\\ &\quad \wedge s'_i = 1 - s_i \wedge \big((s_i = 0 \wedge v'_i = \langle d_1, \ldots, d_k \rangle\\ &\quad \wedge \texttt{ConfigView}_i(n, p_1, \ldots, p_k, d_1, \ldots, d_k) \wedge p'_i = p_i)\\ &\quad \vee (s_i = 1 \wedge v'_i = v_i \wedge \texttt{Move}_i^\phi(n, p_1, \ldots, p_k, p'_i))\big)\Big)\end{aligned}$$

To prove the correctness of this formula for an asynchronous configuration $(\mathbf{p}, \mathbf{s}, \mathbf{V})$ with k robots we make the analogy between the flags **LK** and **MV** and the naturals 0 and 1, which means that in the definition of the vector $\mathbf{s} \in \{\mathbf{LK}, \mathbf{MV}\}^k$, we encode **LK** by 0 and **MV** by 1 and we then apply the definition of $\rightarrow_{as}$.

Lemma 6 *For all $n \in \mathbb{N}$ and all (k, n) asynchronous configurations $\langle \boldsymbol{p}, \boldsymbol{s}, \boldsymbol{V} \rangle$ and $\langle \boldsymbol{p}', \boldsymbol{s}', \boldsymbol{V}' \rangle$, we have $\langle \boldsymbol{p}, \boldsymbol{s}, \boldsymbol{V} \rangle \rightsquigarrow \langle \boldsymbol{p}', \boldsymbol{s}', \boldsymbol{V}' \rangle$ if and only if $n, \boldsymbol{p}, \boldsymbol{s}, \boldsymbol{V}, \boldsymbol{p}', \boldsymbol{s}', \boldsymbol{V}' \models \texttt{AsyncPost}_\phi$.*

Note that one can also express the fact that one configuration is a predecessor of the other in a straightforward way.

As stated earlier, one can also automatically check whether a given formula ϕ is indeed a protocol, thanks to the formula

$$\begin{aligned}&\texttt{Protocol}_\phi() := \neg \exists d_1, \ldots, d_k, d'_1, \ldots, d'_k.\\ &\bigvee_{i=1}^{k} d_i \neq d'_i \wedge \texttt{ViewSym}(d_1, \ldots, d_k, d'_1, \ldots, d'_k) \wedge \phi(d_1, \ldots, d_k) \wedge \phi(d'_1, \ldots, d'_k)\end{aligned}$$

We immediately obtain that

Lemma 7 *A QFP formula ϕ for k variables is a protocol for k robots if and only if $\texttt{Protocol}_\phi$ is satisfiable.*

It is also possible to check whether a protocol ϕ over k robots fits into the hypothesis of Corollary 1, i.e. whether it is uniquely-sequentializable. We define for this matter the formula:

$$\begin{aligned}&\texttt{UniqSeq}_\phi() := \neg \exists y, p_1, \ldots, p_k, p'_1, \ldots, p'_k.\\ &\bigvee_{i \neq j, 1 \leq i, j \leq k} (\texttt{Move}_i^\phi(y, p_1, \ldots, p_k, p'_i) \wedge \texttt{Move}_j^\phi(y, p_1, \ldots, p_k, p'_j)\\ &\wedge p'_i \neq p_i \wedge p'_j \neq p_j.\end{aligned}$$

By applying the definition of uniquely-sequentializable protocol and the result of Lemma 4, we obtain directly the next result.

Lemma 8 *The protocol ϕ is uniquely-sequentializable if and only if the formula $\texttt{UniqSeq}_\phi$ is satisfiable.*

Hence we deduce the following statement.

Theorem 6 *Checking whether a protocol ϕ is uniquely-sequentializable is decidable.*

Moreover we can as well verify whether a protocol ϕ over k robots is pending bounded with a formula that encodes the conditions presented in Definition 3. For this matter we define the formula $\texttt{PendingBounded}_\phi$ that is satisfiable if and only if ϕ is pending-bounded

$$\begin{aligned}
&\texttt{PendingBounded}_\phi() := \neg\exists y, p_1, \ldots, p_k, p'_1, \ldots, p'_k.\\
&\quad \texttt{SemiSyncPost}_\phi(y, p_1, \ldots, p_k, p'_1, \ldots, p'_k)\\
&\quad \wedge \neg\texttt{SyncPost}_\phi(y, p_1, \ldots, p_k, p'_1, \ldots, p'_k)\\
&\quad \wedge \bigvee_{1\le i\le k} \left(p_i \neq p'_i \wedge (\exists q \cdot q \neq p'_i \wedge \texttt{Move}_i^\phi(y, p'_1, \ldots, p'_k, q))\right)\\
&\quad \vee \left(p_i = p'_i \wedge (\exists q . q \neq p'_i \wedge \neg\texttt{Move}_i^\phi(y, p_1, \ldots, p_k, q) \wedge \texttt{Move}_i^\phi(y, p'_1, \ldots, p'_k, q))\right)
\end{aligned}$$

By the definition of pending-bounded protocols and applying the results of Lemmas 4 and 5, we get directly the following property concerning this formula.

Lemma 9 *The protocol ϕ is pending-bounded if and only if the formula $\texttt{PendingBounded}_\phi$ is satisfiable.*

This allows us to deduce this last decidability result.

Theorem 7 *Checking whether a protocol ϕ is pending-bounded is decidable.*

5 Case studies

As explained in the previous section, given a protocol it is possible to reduce the verification of certain properties to the satisfiability problem of some Presburger formulae. In order to see if this translation was useful in practice, we have implemented a small prototype which takes as input a protocol in the form of a formula, an initial set of configurations and a bad set of configurations and generates the Presburger formulae corresponding to the property we want to solve.

For our experiments, we have considered the *exclusive perpetual exploration* algorithms proposed by Blin et al. [5]. The *exclusive perpetual exploration* consists in having robots exploring a ring forever, that is, each node is visited infinitely often by at least one robot. Moreover, for the protocol to be exclusive, no two robots should collide at the same node. For small instances of k (number of robots) and n (number of nodes), it was possible to model-check protocol proposals to ensure their correctness, but going to arbitrary n required a manual proof [4] that was not mechanically verified. For the protocols proposed to solve this problem we have generated the formulae corresponding to the following properties:

1. The proposed formula for the protocol is well defined, i.e. respects the conditions stated in Definition 1;
2. No configuration where two robots collide at the same node is ever reached in the synchronous mode (applying Theorem 4);
3. No configuration where two robots collide at the same node is ever reached in the semi-synchronous mode (applying Theorem 4);

4. The protocol is uniquely-sequentializable; (applying Theorem 6)
5. The protocol is pending-bounded (applying Theorem 7).

We have translated these problems into Presburger formulae in the SMT- LIB format [22] and we have then used the SMT solver Z3 [11] to verify whether the generated formulae were satisfiable or not.

The first algorithm we have studied from [5] is the one using a minimum of 3 robots. For this algorithm, we have considered a correct version and a new one where we have introduced a bug on purpose. Our two models for this algorithm are provided in "Appendix". Note that in this case, when we have studied the absence of collision, we have been able to verify that this property holds (for the correct version of the algorithm) for any ring of size greater than 10, providing hence an automatic correctness criteria. We then studied another algorithm proposed in [5] dedicated as well to the exclusive perpetual exploration of a ring. The goal of this second algorithm was to maximize the number of robots. In that case the verification process is not parametric anymore because the size of the ring is fixed and depends on the number of robots (it is exactly 5 plus the number of robots). As a matter of fact, each time we fix a number of robots, we obtain a different protocol to check and we have studied this algorithm for 5 up to 12 robots. Models of this algorithm for 6 and 7 robots are provided in "Appendix". Note that in this case, the main reason we performed experiments was to see whether the solver could handle relatively big formulae generated by our method.

Table 2 summarizes the results we obtained by running the solver Z3 on the generated files. In this table, we use the symbol ✓ to state that a property holds and in the other case we use ✗. In some cases, the ones indicated by ?, we stopped the solver because the computation was taking too much time (more than 15 min).

We have hence proved automatically that the algorithm using a minimum of 3 robots was safe for any rings of size greater than 10 in the synchronous and semi-synchronous modes. We have also checked that our method was able to detect a bug introduced on purpose (as it is shown in the second column of the table). We have furthermore verified that this algorithm is not uniquely-sequentializable neither pending bounded and as a matter of fact, we cannot deduce any correctness result for the asynchronous mode. An example of the SMT- LIB file used to prove the absence of collision is given in "Appendix".

For the algorithm designed to put a maximum robot on a ring in order to perform the perpetual exploration, we have shown automatically that for any number of robots going from 5 to 12, the protocol is not well specified, in the sense that there exists a configuration in which a robot's view is not symmetric and yet the robot can move in both directions. We have checked manually that this was indeed the case and found that the problem comes from the rules $\mathtt{RL2}_M$ and $\mathtt{RL3}_M$ presented in [5]. However, even if the considered protocol is not well specified, our translation into Presburger formula to check the absence of collision allows to deal with such non deterministic moves. We have shown that the algorithm for 6 robots was safe, but we found some bugs for 5, 7, 8, 9, 10, 11 and 12 robots. It was stated in [5] that the algorithm was not working for 5 robots however the other cases are new bugs. We point out that the bugs are on the version of the algorithm we have manually translated from [5], but on the other hand our way to present the protocol is unambiguous oppositely to the protocol presented in [5] which is sometimes unclear. On the other hand, on these examples, we have as well seen the limit of our methods: with many robots and long protocols, the SMT solver may not be able to finish the computation. In our case, it was not able to handle the files to check that the protocol is uniquely-sequentializable for more than 10 robots and that it is pending bounded for more than 8 robots.

Table 2 Experimental results

Properties	Robot Min 3	Robot Min 3 Bug	Robot Max 5	Robot Max 6	Robot Max 7	Robot Max 8	Robot Max 9	Robot Max 10	Robot Max 11	Robot Max 12
Well-defined	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
Uniquely-sequentializable	✗	✗	✗	✗	✗	✗	✗	?	?	?
Pending-bounded	✗	✗	✗	✗	✗	?	?	?	?	?
No collision in synchronous mode	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗
No collision in semi-synchronous mode	✓	✗	✗	?	✗	✗	✗	✗	✗	✗

6 Conclusion

We have addressed two main problems concerning formal verification of protocols of mobile robots, and answered the open questions regarding decidability of the verification of such protocols, when the size of the ring is given as a parameter of the problem. Note that in such algorithms, robots can start in any position on the ring. Simple modifications of the proofs in this paper allow to obtain undecidability of both the reachability and the safety problem in each of the three presented modes, when the starting configuration of the robots is given. Hence we give a precise view of what can be achieved in the automated verification of protocols for robots in the parameterized setting, and provide a means of partially verifying them. Of course, to fully demonstrate the correctness of a tentative protocol, more properties are required (like, all nodes are visited infinitely often) that are not handled with our approach. Nevertheless, as intermediate lemmas (for arbitrary n) are verified, the whole process of proof writing is both eased and strengthened.

An application of Corollary 1 and Theorem 6 deals with robot program synthesis as depicted in the approach of Bonnet et al. [6]. To simplify computations and save memory when synthesizing mobile robot protocols, their algorithm only generates uniquely-sequentializable protocols (for a given k and n). Now, given a protocol description for a given n, it becomes possible to check whether this protocol remains uniquely-sequentializable for any n. Afterwards, regular safety properties can be devised for this tentative protocol, for all models of computation (that is, FSYNC, SSYNC, and ASYNC). Protocol design is then driven by the availability of a uniquely-sequentializable solution, a serious asset for writing handwritten proofs (for the properties that cannot be automated).

Last, we would like to mention possible applications of our approach for problems whose core properties seem related to reachability only. One such problem is exploration with stop [4]: robots have to explore and visit every node in a network, then stop moving forever, assuming that all robots initial positions are distinct. All of the approaches published for this problem make use of *towers*, that is, locations that are occupied by at least two robots, in order to distinguish the various phases of the exploration process (initially, as all occupied nodes are distinct, there are no towers). Our approach still makes it possible to check if the number of created towers remains acceptable (that is below some constant, typically 2 per block of robots that are equally spaced) from any given configuration in the algorithm, for any ring size n. As before, such automatically obtained lemmas are very useful when writing the full correctness proof.

Appendix: Models for the algorithm with three robots

See Figs. 6, 7, 8, 9, 10 and 11.

```
(define-fun phi ((d1 Int) (d2 Int) (d3 Int)) Bool
 (or
   (and (> d1 4) (= d2 3) (= d3 1))
   (and (= d1 2) (> d2 4) (= d3 3))
   (and (= d1 4) (= d2 2) (> d3 4))
   (and (> d1 4) (> d3 d1) (= d2 1))
   (and (= d1 d3) (> d1 1) (> d2 1))
   (and (> d2 d3) (> d3 d1) (> d1 1))
   (and (= d2 1) (= d3 1))
   (and (= d2 1) (= d3 2))
 )
)
```

Fig. 6 Correct model for the algorithm with 3 robots

```
(define-fun phi ((d1 Int) (d2 Int) (d3 Int)) Bool
 (or
   (and (> d1 4) (= d2 3) (= d3 1))
   (and (= d1 2) (= d2 3) (> d3 4))
   (and (= d1 4) (= d2 2) (> d3 4))
   (and (> d1 4) (> d3 d1) (= d2 1))
   (and (= d1 d3) (> d1 0) (> d2 0))
   (and (> d2 d3) (> d3 d1) (> d1 1))
   (and (= d2 1) (= d3 1))
   (and (= d2 1) (= d3 2))
 )
)
```

Fig. 7 Buggy model for the algorithm with 3 robots

```
(define-fun phi ((d1 Int) (d2 Int) (d3 Int)
(d4 Int) (d5 Int) (d6 Int) ) Bool
(or
;RL1M
(and (= d1 4) (= d2 1) (= d3 3) (= d4 1) (= d5 1) (= d6 1))
;RL2M
(and (= d1 3) (= d2 1) (= d3 3) (= d4 1) (= d5 1) (= d6 2))
;RL3M
(and (= d1 2) (= d2 1) (= d3 3) (= d4 1) (= d5 1) (= d6 3))
;RL4M
(and (= d1 3) (= d2 1) (= d3 1) (= d4 4) (= d5 1) (= d6 1))
;RL5M
(and (= d1 2) (= d2 1) (= d3 1) (= d4 4) (= d5 1) (= d6 2))
;RC2-2M
(and (= d1 6) (= d2 1) (= d3 1) (= d4 1) (= d5 1) (= d6 1))
;RC4-1M
;RC4-2M
;RC4-3M
(and (= d1 5) (= d2 2) (= d3 1) (= d4 1) (= d5 1) (= d6 1))
;RC4-4M
(and (= d1 3) (= d2 4) (= d3 1) (= d4 1) (= d5 1) (= d6 1))
;RC4-5M-x=2
(and (= d1 5)  (= d3 1) (= d4 1) (= d5 2) (= d6 1))
;RC4-5M-x=3
(and (= d1 5)  (= d3 1) (= d4 2) (= d5 1) (= d6 1))
;RC6-1M
(and (= d1 2)  (= d2 4) (= d3 2) (= d4 1) (= d5 1) (= d6 1))
(and (= d1 2)  (= d2 1) (= d3 4) (= d4 2) (= d5 1) (= d6 1))
(and (= d1 2)  (= d2 1) (= d3 4) (= d4 1) (= d5 2) (= d6 1))
(and (= d1 2)  (= d2 1) (= d3 1) (= d4 4) (= d5 2) (= d6 1))
;RC6-2M
(and (= d1 2)  (= d2 4) (= d3 1) (= d4 2) (= d5 1) (= d6 1))
(and (= d1 2)  (= d2 4) (= d3 1) (= d4 1) (= d5 2) (= d6 1))
;RC6-3M
;RC6-5M
(and (= d1 3)  (= d2 2) (= d3 3) (= d4 1) (= d5 1) (= d6 1))
(and (= d1 3)  (= d2 1) (= d3 2) (= d4 1) (= d5 3) (= d6 1))
;RC6-6M
;RC6-7M
;RC6-8M
;RC6-9M
;RC6-10M
(and (= d1 3)  (= d2 1) (= d3 2) (= d4 1) (= d5 1) (= d6 3))
;RC8-1M
(and (= d1 2)  (= d2 3) (= d3 2) (= d4 1) (= d5 1) (= d6 2))
;RC8-2M
;RC8-3M
(and (= d1 2)  (= d2 1) (= d3 3) (= d4 2) (= d5 2) (= d6 1))
(and (= d1 2)  (= d2 1) (= d3 3) (= d4 2) (= d5 1) (= d6 2))
(and (= d1 2)  (= d2 1) (= d3 1) (= d4 3) (= d5 2) (= d6 2))
;RC8-4M
(and (= d1 3)  (= d2 2) (= d3 2) (= d4 2) (= d5 1) (= d6 1))
;RC8-5M
(and (= d1 2)  (= d2 1) (= d3 2) (= d4 1) (= d5 3) (= d6 2))
;RC8-6M
;RC8-7M
;RC8-8M
(and (= d1 2)  (= d2 1) (= d3 2) (= d4 2) (= d5 1) (= d6 3))
;RC10-2M
(and (= d1 2)  (= d2 2) (= d3 2) (= d4 2) (= d5 1) (= d6 2))
;RC10-3M
;RC10-4M
;RC10-5M
(and (= d1 2)  (= d2 1) (= d3 2) (= d4 2) (= d5 2) (= d6 2))
;RC10-7M
;RC10-9M
(and (= d1 2)  (= d2 1) (= d3 2) (= d4 2) (= d5 2) (= d6 2))
;RC10-10M
;RC10-12M
;RC10-14M
;RC10-15M
(and (= d1 2)  (= d2 1) (= d3 2) (= d4 2) (= d5 2) (= d6 2))
)
)
```

Fig. 8 Model for the algorithm with 6 robots on ring of size 11

```
(define-fun phi ((d1 Int) (d2 Int) (d3 Int) (d4 Int)
(d5 Int) (d6 Int) (d7 Int) ) Bool
(or
;RL1M
(and (= d1 4) (= d2 1) (= d3 3) (= d4 1) (= d5 1) (= d6 1) (= d7 1))
;RL2M
(and (= d1 3) (= d2 1) (= d3 3) (= d4 1) (= d5 1) (= d6 1) (= d7 2))
;RL3M
(and (= d1 2) (= d2 1) (= d3 3) (= d4 1) (= d5 1) (= d6 1) (= d7 3))
;RL4M
(and (= d1 3) (= d2 1) (= d3 1) (= d4 1) (= d5 4) (= d6 1) (= d7 1))
;RL5M
(and (= d1 2) (= d2 1) (= d3 1) (= d4 1) (= d5 4) (= d6 1) (= d7 2))
;RC2-2M
(and (= d1 6) (= d2 1) (= d3 1) (= d4 1) (= d5 1) (= d6 1) (= d7 1))
;RC4-1M
;RC4-2M
;RC4-3M
(and (= d1 5) (= d2 2) (= d3 1) (= d4 1) (= d5 1) (= d6 1) (= d7 1))
;RC4-4M
(and (= d1 3) (= d2 4) (= d3 1) (= d4 1) (= d5 1) (= d6 1) (= d7 1))
;RC4-5M-x=2
(and (= d1 5)  (= d3 1) (= d4 1) (= d5 1) (= d6 2) (= d7 1))
;RC4-5M-x=3
(and (= d1 5)  (= d3 1) (= d4 1) (= d5 2) (= d6 1) (= d7 1))
;RC6-1M
(and (= d1 2)  (= d2 4) (= d3 2) (= d4 1) (= d5 1) (= d6 1) (= d7 1))
(and (= d1 2)  (= d2 1) (= d3 4) (= d4 2) (= d5 1) (= d6 1) (= d7 1))
(and (= d1 2)  (= d2 1) (= d3 4) (= d4 1) (= d5 2) (= d6 1) (= d7 1))
(and (= d1 2)  (= d2 1) (= d3 1) (= d4 4) (= d5 2) (= d6 1) (= d7 1))
(and (= d1 2)  (= d2 1) (= d3 1) (= d4 4) (= d5 1) (= d6 2) (= d7 1))
(and (= d1 2)  (= d2 1) (= d3 1) (= d4 1) (= d5 4) (= d6 2) (= d7 1))
;RC6-2M
(and (= d1 2)  (= d2 4) (= d3 1) (= d4 2) (= d5 1) (= d6 1) (= d7 1))
(and (= d1 2)  (= d2 4) (= d3 1) (= d4 1) (= d5 2) (= d6 1) (= d7 1))
(and (= d1 2)  (= d2 4) (= d3 1) (= d4 1) (= d5 1) (= d6 2) (= d7 1))
;RC6-3M
(and (= d1 2)  (= d2 1) (= d3 1) (= d4 4) (= d5 1) (= d6 1) (= d7 2))
;RC6-5M
(and (= d1 3)  (= d2 2) (= d3 3) (= d4 1) (= d5 1) (= d6 1) (= d7 1))
(and (= d1 3)  (= d2 1) (= d3 2) (= d4 1) (= d5 3) (= d6 1) (= d7 1))
;RC6-6M
(and (= d1 2)  (= d2 1) (= d3 1) (= d4 1) (= d5 3) (= d6 3) (= d7 1))
;RC6-7M
;RC6-8M
;RC6-9M
;RC6-10M
(and (= d1 3)  (= d2 1) (= d3 2) (= d4 1) (= d5 1) (= d6 1) (= d7 3))
(and (= d1 3)  (= d2 1) (= d3 1) (= d4 2) (= d5 1) (= d6 1) (= d7 3))
;RC8-1M
(and (= d1 2)  (= d2 3) (= d3 2) (= d4 1) (= d5 1) (= d6 2) (= d7 1))
(and (= d1 2)  (= d2 3) (= d3 2) (= d4 1) (= d5 1) (= d6 1) (= d7 2))
(and (= d1 2)  (= d2 1) (= d3 3) (= d4 1) (= d5 2) (= d6 1) (= d7 2))
```

Fig. 9 Model for the algorithm with 7 robots on ring of size 12 (part I)

```
;RC8-2M
;RC8-3M
(and (= d1 2)  (= d2 1) (= d3 3) (= d4 2) (= d5 2) (= d6 1) (= d7 1))
(and (= d1 2)  (= d2 1) (= d3 3) (= d4 2) (= d5 1) (= d6 2) (= d7 1))
(and (= d1 2)  (= d2 1) (= d3 3) (= d4 2) (= d5 1) (= d6 1) (= d7 2))
(and (= d1 2)  (= d2 1) (= d3 1) (= d4 3) (= d5 2) (= d6 2) (= d7 1))
(and (= d1 2)  (= d2 1) (= d3 1) (= d4 3) (= d5 2) (= d6 1) (= d7 2))
(and (= d1 2)  (= d2 1) (= d3 1) (= d4 3) (= d5 1) (= d6 2) (= d7 2))
(and (= d1 2)  (= d2 1) (= d3 1) (= d4 1) (= d5 3) (= d6 2) (= d7 2))
;RC8-4M
(and (= d1 3)  (= d2 1) (= d3 2) (= d4 1) (= d5 2) (= d6 2) (= d7 1))
(and (= d1 3)  (= d2 2) (= d3 2) (= d4 2) (= d5 1) (= d6 1) (= d7 1))
;RC8-5M
(and (= d1 2)  (= d2 1) (= d3 2) (= d4 1) (= d5 1) (= d6 3) (= d7 2))
;RC8-6M
;RC8-7M
;RC8-8M
(and (= d1 2)  (= d2 1) (= d3 2) (= d4 2) (= d5 1) (= d6 1) (= d7 3))
;RC10-2M
(and (= d1 2)  (= d2 1) (= d3 2) (= d4 1) (= d5 2) (= d6 2) (= d7 2))
(and (= d1 2)  (= d2 2) (= d3 2) (= d4 1) (= d5 2) (= d6 2) (= d7 1))
(and (= d1 2)  (= d2 2) (= d3 2) (= d4 2) (= d5 1) (= d6 1) (= d7 2))
;RC10-3M
(and (= d1 2)  (= d2 2) (= d3 1) (= d4 2) (= d5 2) (= d6 2) (= d7 1))
;RC10-4M
;RC10-5M
(and (= d1 2)  (= d2 1) (= d3 1) (= d4 2) (= d5 2) (= d6 2) (= d7 2))
;RC10-7M
;RC10-9M
;RC10-10M
;RC10-12M
(and (= d1 2)  (= d2 1) (= d3 2) (= d4 2) (= d5 2) (= d6 1) (= d7 2))
;RC10-14M
(and (= d1 2)  (= d2 1) (= d3 2) (= d4 1) (= d5 2) (= d6 2) (= d7 2))
;RC10-15M
(and (= d1 2)  (= d2 1) (= d3 2) (= d4 2) (= d5 2) (= d6 1) (= d7 2))
(and (= d1 2)  (= d2 1) (= d3 2) (= d4 2) (= d5 1) (= d6 2) (= d7 2))
(and (= d1 2)  (= d2 1) (= d3 2) (= d4 1) (= d5 2) (= d6 2) (= d7 2))
(and (= d1 2)  (= d2 1) (= d3 1) (= d4 2) (= d5 2) (= d6 2) (= d7 2))
(and (= d1 2)  (= d2 1) (= d3 2) (= d4 2) (= d5 2) (= d6 2) (= d7 1))
)
)
```

Fig. 10 Model for the algorithm with 7 robots on ring of size 12 (part II)

```
(define-fun Cond ((n Int)) Bool
 (< 10 n)
)

(define-fun IsView ( (n Int) (d1 Int) (d2 Int) (d3 Int)) Bool
(and (<= 0 d1) (<= 0 d2) (<= 0 d3) (= n (+ d1 d2 d3))))

(define-fun ViewSym ((d1 Int) (d2 Int) (d3 Int) (d!1 Int) (d!2 Int) (d!3 Int))
     Bool
(or (and (= d2 0) (= d!2 0) (= d3 0) (= d!3 0) (= d!1 d1)) (and (= d3 0) (= d!3 0)
    (= d!1 d2)  (= d!2 d1))
(and (= d!1 d3) (= d!2 d2) (= d!3 d1))))

(define-fun ConfigView1 ((n Int) (p1 Int) (p2 Int) (p3 Int) (d1 Int) (d2 Int) (d3
     Int)) Bool (exists ((d!1 Int) (d!2 Int)) (and (and (<= 0 d1) (<= 0 d2)(<= 0
     d3)) (and (<= d!1 d!2)) (and (or (or (= p2 (+ p1 d!1)) (= (+ p2 n) (+ p1 d!1)
     )) (or (= p2 (+ p1 d!2)) (= (+ p2 n) (+ p1 d!2)))) (or (or (= p3 (+ p1 d!1))
     (= (+ p3 n) (+ p1 d!1))) (or (= p3 (+ p1 d!2)) (= (+ p3 n) (+ p1 d!2))))) (
     and (or (or (= p2 (+ p1 d!1))(= (+ p2 n) (+ p1 d!1))) (or (= p3 (+ p1 d!1))(=
      (+ p3 n) (+ p1 d!1)))) (or (or (= p2 (+ p1 d!2))(= (+ p2 n) (+ p1 d!2))) (or
      (= p3 (+ p1 d!2))(= (+ p3 n) (+ p1 d!2))))) (and (< 0 d!1) (<= d!1 n) (<= d
     !2 n) (= d1 d!1) (= d2 (- d!2 d!1)) (= d3 (- n (+ d1 d2)))))))

(define-fun ConfigView2 ((n Int) (p1 Int) (p2 Int) (p3 Int) (d1 Int) (d2 Int) (d3
     Int)) Bool (exists ((d!1 Int) (d!2 Int)) (and (and (<= 0 d1) (<= 0 d2)(<= 0
     d3)) (and (<= d!1 d!2)) (and (or (or (= p1 (+ p2 d!1)) (= (+ p1 n) (+ p2 d!1)
     )) (or (= p1 (+ p2 d!2)) (= (+ p1 n) (+ p2 d!2)))) (or (or (= p3 (+ p2 d!1))
     (= (+ p3 n) (+ p2 d!1))) (or (= p3 (+ p2 d!2)) (= (+ p3 n) (+ p2 d!2))))) (
     and (or (or (= p1 (+ p2 d!1))(= (+ p1 n) (+ p2 d!1))) (or (= p3 (+ p2 d!1))(=
      (+ p3 n) (+ p2 d!1)))) (or (or (= p1 (+ p2 d!2))(= (+ p1 n) (+ p2 d!2))) (or
      (= p3 (+ p2 d!2))(= (+ p3 n) (+ p2 d!2))))) (and (< 0 d!1) (<= d!1 n) (<= d
     !2 n) (= d1 d!1) (= d2 (- d!2 d!1)) (= d3 (- n (+ d1 d2)))))))

(define-fun ConfigView3 ((n Int) (p1 Int) (p2 Int) (p3 Int) (d1 Int) (d2 Int) (d3
     Int)) Bool (exists ((d!1 Int) (d!2 Int)) (and (and (<= 0 d1) (<= 0 d2)(<= 0
     d3)) (and (<= d!1 d!2)) (and (or (or (= p1 (+ p3 d!1)) (= (+ p1 n) (+ p3 d!1)
     )) (or (= p1 (+ p3 d!2)) (= (+ p1 n) (+ p3 d!2)))) (or (or (= p2 (+ p3 d!1))
     (= (+ p2 n) (+ p3 d!1))) (or (= p2 (+ p3 d!2)) (= (+ p2 n) (+ p3 d!2))))) (
     and (or (or (= p1 (+ p3 d!1))(= (+ p1 n) (+ p3 d!1))) (or (= p2 (+ p3 d!1))(=
      (+ p2 n) (+ p3 d!1)))) ) (or (or (= p1 (+ p3 d!2))(= (+ p1 n) (+ p3 d!2))) (
     or (= p2 (+ p3 d!2))(= (+ p2 n) (+ p3 d!2))) )) (and (< 0 d!1) (<= d!1 n) (<=
      d!2 n) (= d1 d!1) (= d2 (- d!2 d!1)) (= d3 (- n (+ d1 d2)))))))

(define-fun Move1phi ((n Int) (p1 Int) (p2 Int) (p3 Int) (p!1 Int)) Bool (exists
     ((d1 Int) (d2 Int) (d3 Int) (d!1 Int) (d!2 Int) (d!3 Int)) (and (ConfigView1
     n p1 p2 p3 d1 d2 d3) (ViewSym d1 d2 d3 d!1 d!2 d!3) (or (and (phi d1 d2 d3) (
     or (and (< p1 (- n 1)) (= p!1 (+ p1 1))) (and (= p1 (- n 1)) (= p!1 0)))) (
     and (phi d!1 d!2 d!3) (or (and (> p1 0) (= p!1 (- p1 1))) (and (= p1 0) (= p
     !1 (- n 1))))) (and (not (phi d1 d2 d3)) (not (phi d!1 d!2 d!3)) (= p!1 p1)))
     )))

(define-fun Move2phi ((n Int) (p1 Int) (p2 Int) (p3 Int) (p!2 Int)) Bool (exists
     ((d1 Int) (d2 Int) (d3 Int) (d!1 Int) (d!2 Int) (d!3 Int)) (and (ConfigView2
     n p1 p2 p3 d1 d2 d3) (ViewSym d1 d2 d3 d!1 d!2 d!3) (or (and (phi d1 d2 d3) (
     or (and (< p2 (- n 1)) (= p!2 (+ p2 1))) (and (= p2 (- n 1)) (= p!2 0)))) (
     and (phi d!1 d!2 d!3) (or (and (> p2 0) (= p!2 (- p2 1))) (and (= p2 0) (= p
     !2 (- n 1))))) (and (not (phi d1 d2 d3)) (not (phi d!1 d!2 d!3)) (= p!2 p2)))
     )))

(define-fun Move3phi ((n Int) (p1 Int) (p2 Int) (p3 Int) (p!3 Int)) Bool (exists
     ((d1 Int) (d2 Int) (d3 Int) (d!1 Int) (d!2 Int) (d!3 Int)) (and (ConfigView3
     n p1 p2 p3 d1 d2 d3) (ViewSym d1 d2 d3 d!1 d!2 d!3) (or (and (phi d1 d2 d3) (
     or (and (< p3 (- n 1)) (= p!3 (+ p3 1))) (and (= p3 (- n 1)) (= p!3 0)))) (
     and (phi d!1 d!2 d!3) (or (and (> p3 0) (= p!3 (- p3 1))) (and (= p3 0) (= p
     !3 (- n 1))))) (and (not (phi d1 d2 d3)) (not (phi d!1 d!2 d!3)) (= p!3 p3)))
     )))

(define-fun SyncPost ((n Int) (p1 Int) (p2 Int) (p3 Int) (p!1 Int) (p!2 Int) (p!3
     Int)) Bool (and (Move1phi n p1 p2 p3 p!1) (Move2phi n p1 p2 p3 p!2) (Move3phi
      n p1 p2 p3 p!3)))

(define-fun Bad ((q1 Int) (q2 Int) (q3 Int)) Bool (or (= q1 q2) (= q1 q3) (= q2 q1
     ) (= q2 q3) (= q3 q1) (= q3 q2)))

(define-fun InitNoBad ((n Int) (p1 Int) (p2 Int) (p3 Int)) Bool (and (< p1 n) (<=
     0 p1) (< p2 n) (<= 0 p2) (< p3 n) (<= 0 p3) (not (Bad p1 p2 p3))))

(assert (and (Cond n) (InitNoBad n p1 p2 p3) (Bad p!1 p!2 p!3) (SyncPost n p1 p2
     p3 p!1 p!2 p!3)))
```

Fig. 11 Extract of the SMT- LIB code to check the absence of collision in the algorithm with 3 robots

References

1. Auger C, Bouzid Z, Courtieu P, Tixeuil S, Urbain X (2013) Certified impossibility results for byzantine-tolerant mobile robots. In: Proceedings of SSS'13, volume 8255 of LNCS. Springer, Berlin, pp 178–186
2. Balabonski T, Delga A, Rieg L, Tixeuil S, Urbain X (2016) Synchronous gathering without multiplicity detection: a certified algorithm. In: Proceedings of SSS'16, volume 10083 of LNCS. Springer, Berlin, pp 7–19
3. Bérard B, Courtieu P, Millet L, Potop-Butucaru M, Rieg L, Sznajder N, Tixeuil S, Urbain X (2015) Formal methods for mobile robots: current results and open problems. Int J Inf Soc 7(3):101–114
4. Bérard B, Lafourcade P, Millet L, Potop-Butucaru M, Thierry-Mieg Y, Tixeuil S (2016) Formal verification of mobile robot protocols. Distrib Comput 29:459–587
5. Blin L, Milani A, Potop-Butucaru M, Tixeuil S (2010) Exclusive perpetual ring exploration without chirality. In: Proceedings of DISC'10, volume 6343 of LNCS. Springer, Berlin, pp 312–327
6. Bonnet F, Défago X, Petit F, Potop-Butucaru M, Tixeuil S (2014) Discovering and assessing fine-grained metrics in robot networks protocols. In: Proceedings of SRDS'14. IEEE Press, pp 50–59
7. Borosh I, Treybig L (1976) Bounds on positive integral solutions of linear Diophantine equations. Am Math Soc 55:299–304
8. Courtieu P, Rieg L, Tixeuil S, Urbain X (2015) Impossibility of gathering, a certification. Inf Process Lett 115:447–452
9. Courtieu P, Rieg L, Tixeuil S, Urbain X (2016) Certified universal gathering in $\mathbb{R}^2$ for oblivious mobile robots. In: Proceedings of DISC'16, volume 9888 of LNCS. Springer, Berlin, pp 187–200
10. D'Angelo G, Stefano GD, Navarra A, Nisse N, Suchan K (2013) A unified approach for different tasks on rings in robot-based computing systems. In Proceedings of IPDPSW'13. IEEE Press, pp 667–676
11. de Moura LM, Bjørner N (2008) Z3: an efficient SMT solver. In: TACAS'08, volume 4963 of LNCS. Springer, Berlin, pp 337–340
12. Devismes S, Lamani A, Petit F, Raymond P, Tixeuil S (2012) Optimal grid exploration by asynchronous oblivious robots. In: Proceedings of SSS'12, volume 7596 of LNCS. Springer, Berlin, pp 64–76
13. Doan HTT, Bonnet F, Ogata K (2016) Model checking of a mobile robots perpetual exploration algorithm. In Proceedings of SOFL+MSVL, Revised Selected Papers, volume 10189 of LNCS, pp 201–219
14. Flocchini P, Ilcinkas D, Pelc A, Santoro N (2013) Computing without communicating: ring exploration by asynchronous oblivious robots. Algorithmica 65(3):562–583
15. Flocchini P, Prencipe G, Santoro N (2012) Distributed computing by oblivious mobile robots. Synt. Lect. Distr. Comp. Th. Morgan & Claypool Publishers, San Rafael
16. Kranakis E, Krizanc D, Markou E (2010) The mobile agent rendezvous problem in the ring. Synt. Lect. Distr. Comp. Th. Morgan & Claypool Publishers, San Rafael
17. Mayr R (2003) Undecidable problems in unreliable computations. Theor Comput Sci 297(1–3):337–354
18. Millet L, Potop-Butucaru M, Sznajder N, Tixeuil S (2014) On the synthesis of mobile robots algorithms: the case of ring gathering. In: Proceedings of SSS'14, volume 8756 of LNCS. Springer, Berlin, pp 237–251
19. Minsky ML (1967) Computation: finite and infinite machines. Prentice-Hall Inc, Upper Saddle River
20. Rubin S, Zuleger F, Murano A, Aminof B (2015) Verification of asynchronous mobile-robots in partially-known environments. In: Proceedings of PRIMA'15, volume 9387 of LNCS. Springer, Berlin, pp 185–200
21. Sangnier A, Sznajder N, Potop-Butucaru M, Tixeuil S (2017) Parameterized verification of algorithms for oblivious robots on a ring. In: FMCAD'17. IEEE, pp 212–219
22. SMT-LIB: The satisfiability modulo theory library. http://smtlib.cs.uiowa.edu/
23. Suzuki I, Yamashita M (1999) Distributed anonymous mobile robots: formation of geometric patterns. SIAM J Comput 28(4):1347–1363