Engineering Controllers for Swarm Robotics via Reachability Analysis in Hybrid Systems

Francesco Leofante, Stefan Schupp, Erika Abraham, Armando Tacchella

KEYWORDS

Simulation of Control Systems, Model Checking of Hybrid Systems Swarm Robotics

ABSTRACT

In this paper we propose hybrid systems and reachability analysis to verify properties in swarm robotics systems, i.e., teams of robots performing cooperative tasks without any centralized coordination. We discuss the challenges that are to be faced and we report on the experience gained from applying hybrid formalisms to the verification of swarm robotics systems.

INTRODUCTION

Swarm robotics systems are specific kind of distributed autonomous mobile robotic systems wherein a set of robots cooperatively perform a task, without any centralized coordination [28]. The peculiar feature of robot swarms is that, although individual robots are governed by relatively simple reactive controllers, complex behaviors arise in the swarm, leading the team to achieve goals that would defy each single robot in isolation, or would require more expensive robots to be achieved as effectively as the swarm does. Interest in this research field has been increasing in the last decade, since it allows for robust, flexible, efficient and scalable solutions — see, e.g, [5] for a review.

While understanding individual robot behavior is easy, predicting the overall swarm behavior is difficult, and thus engineering controllers for individual robots that will guarantee a desired swarm behavior is not a straightforward task. Traditionally, the analysis of swarms is carried out either by testing real robot implementations, or by computational simulations – see, e.g., [20], [22]. However these approaches provide little guarantees on the actual swarm behavior as they suffer from intrinsically incomplete coverage. As suggested by many authors — see, e.g., [31], [29], [18], [6] — desired levels of assurance in swarm behavior can be obtained via formal methods. However, considering the work done in this direction, we realized that most approaches abstract away details about the continuous dynamics of the robots, which may indeed be crucial for the emergence of desired behaviors or, on the contrary, might hamper the ability of the swarm to reach the desired goals when specific circumstances arise.

In this paper, we propose to fill this gap by attacking the problem of engineering swarm robotics systems through hybrid formalisms. In particular, hybrid automata [16] can be leveraged to define richer models. Once such models are available, reachability analysis can be employed to overcome the incomplete coverage of testing and simulation. Here we report on the application of reachability analysis of hybrid automata to verify controllers for swarm robotics. Modeling challenges are discussed, together with features and limitations of tools for reachability analysis. Preliminary results are shown to support feasibility. Our contribution can be summarized as follows:

- Introducing hybrid automata and reachability analysis as a method to formally engineer robot swarms; to the extent of our knowledge, this is the first contribution in this direction, with previous works focusing on models that abstract from physical dynamics.
- Modelling and analysing two simple but realistic swarm robotics systems, one entailing synchronization without central coordination, and another considering collision avoidance in exploration tasks.

SWARM ROBOTIC SYSTEMS

Swarm robotics draws from the "simple control rules of various biological societies, particularly ants, bees and birds to the development of similar behaviors in cooperative robot systems" [28]. A swarm can be characterized as in [5]: robots are autonomous; robots are situated in some environment and act to modify it; robot's sensing and communication capabilities are local; robots do not have access to centralized control and/or global knowledge; robots cooperate to accomplish a task. The cooperation among robots happens in different ways, including explicit communication through, e.g., a wireless network, or implicitly through *stigmergy* [3], i.e., robots sense changes made by other robots, and then adjust their behavior accordingly. In any case, the collective behavior is not predefined to any extent at the global level, but it is most likely to be *emergent*, i.e., the result of several local robot-to-robot and robot-to-environment interactions. Here we take as working definition the one given in [24] asserting that emergent behaviors are characterized by two properties: (*i*) they are manifested by global states or time-extended patterns which are not explicitly programmed in, but result from local interactions among a system's components; (*ii*) they are considered interesting based on some observer-established metric.

Francesco Leofante and Armando Tacchella are with "Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi" (DIBRIS), University of Genoa, Viale Causa 13, 16145 Genoa, Italy. E-mail: francesco.leofante@edu.unige.it,armando.tacchella@unige.it. Stefan Schupp and Erika Abraham are with the Computer Science Department at RWTH Aachen University, Germany. E-mail:abraham@informatik.rwth-aachen.de, stefan.schupp@cs.rwth-aachen.de. The corresponding author is Armando Tacchella.

Communications of the ECMS, Volume 33, Issue 1,
Proceedings, ©ECMS Mauro Iacono, Francesco Palmieri,
Marco Gribaudo, Massimo Ficco (Editors)
ISBN: 978-3-937436-65-4/978-3-937436-66-1(CD) ISSN 2522-2414

From an engineering point of view, the main advantage of swarm robotics systems lies in the fact that all robots in the swarm feature the same architecture whereon the same software stack and the same set of reactive control rules are implemented. Furthermore, swarms are inherently robust, because the failure of a single robot rarely affects the success of the whole task. From an abstract point of view, examples of tasks which can be accomplished using swarms include aggregation, pattern formation, object clustering/building, and self-assembling [5]. More concretely, in [32] it is suggested that the following tasks might be best solved by swarms: (*i*) environmental monitoring, e.g., detecting the accidental leakage of a chemical; (*ii*) operating in dangerous/harsh environments, e.g., decommissioning nuclear power plants or clearing corridors in mining fields; (*iii*) containing damage, e.g., amassing oil from a fractured tank of a carrier. In all these cases, the robustness, cheapness, scalability and reconfigurability of swarms are key to their performances in operations. In this work we consider swarms of MarXbots [4], ground-based robots that use differential-drive treels. MarXbots have been conceived and built through several European projects focusing on swarm robotics — see, e.g, [26]; successful applications of such robots can be found, e.g., in [11] and [10]. To support our experimentation, we use ARGoS [30], a simulator designed to efficiently simulate complex experiments involving large swarms of robots, developed within the same European projects mentioned above.

THE PROBLEM OF SWARM ENGINEERING

Swarm robotics is the subject of an intense research effort, as witnessed by dedicated conferences, e.g., ANTS, and journals, e.g., Swarm Intelligence[1]. However, in spite of some recent contributions [9], [23], [1], swarm robotics systems are still largely confined to academia [5] where they enjoy the status of research prototypes built to collect empirical evidence. In this setting, engineering efforts can be kept at a bare minimum, i.e., enough to survive a single experiment, so there is no strong push towards introducing rigorous, but also time-consuming, approaches. We posit that engineering swarm robotics systems amounts to (*i*) getting a swarm of robots to perform a given task and (*ii*) making sure that repeatable and reliable behavior is obtained from the swarm with sufficient confidence. While requirement (*i*) has been the subject of extensive research – see [5] for a recent review – requirement (*ii*) is still not considered mainstream. This is not surprising, because for a swarm robotics system the relation between the definition and the implementation of the system "behavior" is somewhat non-standard with respect to other distributed robotics implements, and other engineered systems as well.

Indeed, while swarm-level requirements can be expressed with sufficient precision, the causal links among the conditions stated by the requirements and the actual implementation of robot controllers are not explicit. As an example, consider aggregation behavior. The goal of aggregation is to group all the robots of a swarm in a region of the environment [5]. Notice that this requirement can be stated precisely, e.g., "*after a predefined time lapse the maximum distance between any two robots shall be less than a given threshold*". However, one of the most common methods to implement aggregation is based on probabilistic finite state controllers – see, e.g., [14] – whereby each robot is equipped with a simple reactive controller that explores the environment and, when it detects other robots, it decides stochastically whether to join or leave the aggregate. Empirical evidence shows that a swarm of robots equipped with such simple reactive strategy can indeed aggregate [14], but the correct design and implementation of a single controller is clearly not enough to increase confidence in such overall behavior. As mentioned in [5], testing with simulated and real robots has to be performed in order to observe desired behaviors in swarms, but simulation may not account for all the subtleties of real robots and environments, whereas testing with real robots is extremely time consuming and potentially very expensive.

The fact that traditional system engineering techniques may not be suitable for swarm robotics has opened the path to alternative techniques, most noticeably those based on formal verification [31], [29], [18], [6]. To the best of our knowledge, the first contribution along this line of investigation is [31], wherein the authors investigated the applicability of formal methods to the verification and validation of spacecraft using swarm technology. More specifically, they considered a number of approaches including Communicating Sequential Processes (CSP), process algebras, X-machines and Unity Logic. However, their conclusion at the time of the contribution (2004) was that none of the approaches had all the properties required to assure correct behavior and interactions of swarms in the context of the ANTS (Autonomous Nano Technology Swarm) concept mission. In [29], agent-oriented software engineering (AOSE) are investigated to provide support to developing swarm robotics systems, still in the context of the ANTS mission. However, no mention related to assuring behaviors is to be found in the contribution which is largely confined to model-based design and implementation techniques. A series of papers by Clare Dixon et al. – see, e.g., [18] for the most recent contribution in the series – explores the potential of modeling robot swarms using a composition of (probabilistic) finite state machines and of proving swarm-level requirements using model checking of (probabilistic) temporal logic. Noticeably, in the case of probabilistic models and logic, the model of each single robot controller is very close to the actual implementation, and model checking of relevant properties is reported to be feasible for swarms of relatively small size – less than 4 robots according to the experiments in [18]. The main limitation of these approaches is that the dynamics of the robots and the interactions between robots and the

[1]Publisher's link: http://link.springer.com/journal/11721

environment are neglected, whereas in robot swarms these could be crucial to the performances of the whole system. Finally, in [6] an approach based on probabilistic finite state machines and probabilistic temporal logic is put forth in order to provide property-based design of swarm robotic systems. The case study of aggregation is presented and studied in the context of a swarm of 10 robots. Also in this case, the robot model and the interaction model abstract away the concrete robot dynamics.

HYBRID SYSTEMS REACHABILITY ANALYSIS

In the following, we briefly introduce the syntax and semantics of hybrid automata, and motivate their usage in the context of swarm robotic systems.

Definition 1 (Hybrid automata: Syntax [16]) A *hybrid automaton* is a tuple $\mathcal{H} = (Loc, Var, Flow, Inv, Edge, Init)$ consisting of:

- A finite set Loc of *locations* or *control modes*.
- A finite ordered set $Var = \{x_1, \ldots, x_n\}$ of real-valued *variables*; we also use the vector notation $\vec{x} = (x_1, \ldots, x_n)$. The number n is called the *dimension* of $\mathcal{H}$. By Var we denote the set $\{\dot{x}_1, \ldots, \dot{x}_n\}$ of dotted variables (which represent first derivatives during continuous change), and by Var' the set $\{x'_1, \ldots, x'_n\}$ of primed variables (which represent values directly after a discrete change). Furthermore, $Pred_X$ is the set of all predicates with free variables from X.
- $Flow : Loc \to Pred_{Var \cup \dot{Var}}$ specifies for each location its *flow* or *dynamics*.
- $Inv : Loc \to Pred_{Var}$ assigns to each location an *invariant*.
- $Edge \subseteq Loc \times Pred_{Var} \times Pred_{Var \cup Var'} \times Loc$ is a finite set of *discrete transitions* or *jumps*. For a jump $(l_1, g, r, l_2) \in Edge$, l_1 is its *source* location, l_2 is its *target* location, g specifies the jump's *guard*, and r its *reset* function, where primed variables represent the state after the step.
- $Init : Loc \to Pred_{Var}$ assigns to each location an *initial* predicate.

The evolution of a hybrid system over time can be described by a run in the respective hybrid automaton, which includes both continuous evolution (flow) and the discrete state changes (jump). The formal semantics of a hybrid automaton is defined as follows.

Definition 2 (Hybrid automata: Semantics) The one-step semantics of a hybrid automaton $\mathcal{H} = (Loc, Var, Flow, Inv, Edge, Init)$ of dimension n is specified by the following operational semantics rules:

$$\frac{\begin{array}{c} l \in Loc \quad \vec{v}, \vec{v}' \in \mathbb{R}^\kappa \\ f : [0,\delta] \to \mathbb{R}^\kappa \quad \upsilon/\approx = \dot{\upsilon} : (\nu, \delta) \to \mathbb{R}^\kappa \quad \upsilon(\nu) = \vec{\tilde{\upsilon}} \quad \upsilon(\delta) = \vec{\tilde{\upsilon}}' \\ \forall \epsilon \in (0,\delta).\ f(\epsilon), \dot{f}(\epsilon) \models Flow(l) \quad \forall \epsilon \in [0,\delta].\ f(\epsilon) \models Inv(l) \end{array}}{(l, \vec{v}) \xrightarrow{\delta} (l, \vec{v}')} \text{ Rule}_{\text{flow}}$$

$$\frac{e = (l, g, r, l') \in Edge \quad \vec{v}, \vec{v}' \in \mathbb{R}^\kappa \quad \vec{\tilde{\upsilon}} \models \eth \quad \vec{\tilde{\upsilon}}, \vec{\tilde{\upsilon}}' \models \diagdown \quad \vec{\tilde{\upsilon}}' \models Inv(l')}{(l, \vec{v}) \xrightarrow{e} (l', \vec{v}')} \text{ Rule}_{\text{jump}}$$

A *path* of $\mathcal{H}$ is a (finite or infinite) sequence $(l_0, \vec{v}_0) \xrightarrow{\delta_0} (l_1, \vec{v}_1) \xrightarrow{e_1} (l_2, \vec{v}_2) \xrightarrow{\delta_2} (l_3, \vec{v}_3) \xrightarrow{e_3} (l_4, \vec{v}_4) \xrightarrow{\delta_4} \ldots$ with $(l_i, \vec{v}_i)$ states of H, $\delta_i \in \mathbb{R}_{\geq \nu}$, $e_i \in Edge$, and $\vec{v}_0 \models Init(l_0) \wedge Inv(l_0)$. A state $(l, \vec{v})$ is *reachable* in $\mathcal{H}$ if there is a path $(l_0, \vec{v}_0) \xrightarrow{\delta_0} (l_1, \vec{v}_1) \xrightarrow{e_1} (l_2, \vec{v}_2) \xrightarrow{\delta_2} \ldots$ of $\mathcal{H}$ with $(l, \vec{v}) = (l_i, \vec{v}_i)$ for some $i \geq 0$.

Once a hybrid automaton of a given hybrid system has been formalized, we might be interested in analyzing its behavior. Given that the hybrid systems we are considering are physical agents that can act and modify the environment, we might want to enforce stringent safety requirements upon their behavior. Such requirements can be formalized as sets of states to be avoided in the state space of a hybrid automaton.

The *reachability problem* for hybrid automata is the problem to decide whether a given state (or any state from a given set) is reachable by a hybrid automaton. As the reachability problem for hybrid automata is in general undecidable, some approaches aim at computing an *over-approximation* of the set of reachable states of a given hybrid automaton. We focus on approaches based on *flowpipe construction*, which iteratively over-approximate the set of reachable states by the union of a set of state sets – see *e.g.* [12] for further details. To *represent* a state set, typically either a *geometric* or a *symbolic* representation is used. Geometric representations specify state sets by geometric objects like boxes [27], (convex) polytopes [34], zonotopes [15], or ellipsoids [19], whereas symbolic representations use, e.g., support functions [21] or Taylor models [7]. These representations might have major differences in the precision of the representation (the size of over-approximation), the memory requirements and the computational effort needed to apply operations like intersection, union, linear transformation, Minkowski sum or test for emptiness.

For a given state set p, flowpipe-construction-based approaches compute the successors of the states from p by first over-approximating the set of states reachable via time evolution (*flowpipe*) and afterwards the set of states reachable from the flowpipe as jump successors. Time evolution is usually restricted to a *time horizon* (either per location or for the whole execution), which is divided into smaller time steps. The states reachable from p via one time step are over-approximated by a state set p_1, for which again the time successors p_2 via one time step are computed. This procedure is repeated until the time horizon is reached or the successor set gets empty (due to the violation of the current location's invariant). The union of the resulting state sets $p_1, \ldots, p_k$, which are called *flowpipe segments*, over-approximates the flowpipe. For each outgoing jump and each of the flowpipe segments the jump successors are computed, to which the above procedure is applied iteratively until a given upper bound (*jump depth*) on the number of jumps is reached or until a fixed point is detected. A non-exhaustive list of software implementations of flowpipe-construction-based methods includes CORA [2], FLOW* [8], HYCREATE [17], HYPRO [33] and SPACEEX [13].

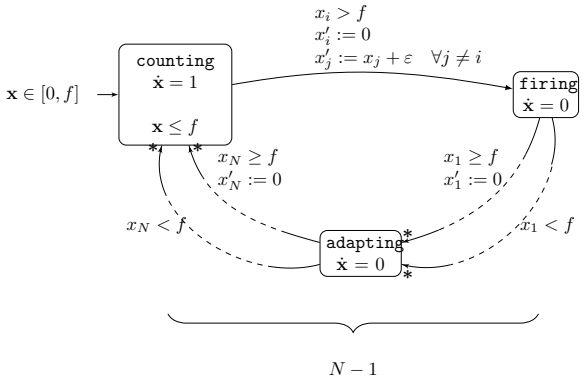


Fig. 1: Hybrid automaton for the synchronization problem.

CASE STUDIES

This section shows experimental results for two problems from the field of swarm robotics, which are analyzed using hybrid systems reachability analysis.

A. Synchronization without central coordination

A.0.a **Problem description.**. This experiment reproduces the behavior of pulse-coupled oscillators as described in [25] and implemented in the ARGoS simulation environment (see Figure 3). In particular, the model involves a population of MarXbots, each of which is equipped with a LED. Each robot is characterized by a state variable x_i representing a counter, subject to the following dynamics:

$$\dot{x_i} = 1 \quad 0 \leq x_i \leq f, \quad i = 1, \ldots, N$$

where f represents the so-called *firing threshold*. When $x_i = f$, robot i flashes its central LED and x_i is reset to zero. Robots are assumed to interact by a simple form of pulse coupling: when robot i flashes, it increases all the other robots' counters by an amount defined by α, or pulls them to firing, according to the following relation:

$$x_i = f \implies x_j' := \begin{cases} \alpha \cdot x_j & \text{if } \alpha \cdot x_j < f \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \neq i \tag{1}$$

This problem represents a good example of how global swarm behaviors can emerge in decentralized distributed systems without being explicitly specified by individual control algorithms.

A.0.b **Modeling.**. The automaton for the synchronization problem is shown in Figure 2 following the notation presented in Definition 1. In addition to what is already defined, the symbol * placed next to a transition denotes *urgency*, which forces the control to take the respective transition as soon as it is enabled.

Despite the simple dynamics involved in this experiment, this problem poses some interesting challenges. First of all, not all the tools for reachability analysis support parallel composition – to our knowledge only SpaceEx supports this feature. To overcome this limitation, we derived a non-compositional model which can subsequently be verified. Furthermore, encoding the behavior described in Equation 1 is non-trivial, as it involves a case distinction. To reproduce such behavior, we adopted the following solution. The case distinction is encoded in guarded discrete transitions, two for each variable. Instead of introducing exponentially many transitions to cover all combinations for all variables, we make use of $N - 1$ intermediate locations where no changes occur in the continuous dynamics – we refer to such locations as *empty*. Note that the introduction of those additional locations should not increase the complexity of the reachability analysis as no flow occurs and no time passes. Our solution employs urgency and *multi-edges* (i.e., two or more transitions connect the same pair of locations), however not all tools for linear reachability support such features. Of course, other modeling choices could have been made, e.g., eliminate multi-edges by introducing additional intermediate empty locations with exclusive transition guards, or enforce urgency with a combination of invariants and guards. However this would have resulted in an unnecessarily complex model.

A.0.c **Reachability analysis results.**. We carried out our experiments using our own implementation based on our `C++` library for reachability analysis HyPro. We modeled a system consisting of three robots [2] starting with different initial sets for the counter variables ($x_1 \in [0, 5], x_2 \in [15, 20], x_3 \in [30, 35]$) and an update factor $\alpha = 1.1$. A plot of the set of reachable states for a predefined time bound of $300s$ can be found in Figure 4a, which shows the set of reachable states for all robots combined in one plot. Results of reachability analysis show that global synchronization emerges over time. In particular, after a short time period the counter values x_2, x_3 already are almost perfectly synchronized. Counter variables are updated by factors proportional to their current values resulting in different convergence behavior as shown in Figure 4b). Here we can see that as soon as x_3 reaches the firing threshold, x_1 and x_2 undertake discrete updates of different quantities. Approximate synchronization is reached after about $250s$.

B. Collision avoidance in exploration tasks

B.0.a **Problem description.**. This experiment models a scenario where MarXbots perform collision avoidance while exploring a square environment. Given arbitrary initial poses (i.e., position and heading), robots start moving following straight line trajectories. Each robot is equipped with a range sensor used to detect static and dynamic obstacles – *e.g.* walls and robots – within a fixed sensing distance. If the sensed distance between two robots is below a security threshold, a collision avoidance behavior is triggered. Our experiment implements a simple collision avoidance scheme where robots always prevent collisions by turning by a fixed angle. Angles are defined using the classical notation

[2]Note that this number has been chosen for the sake of readability of the plots. Higher swarm sizes have successfully been analyzed.

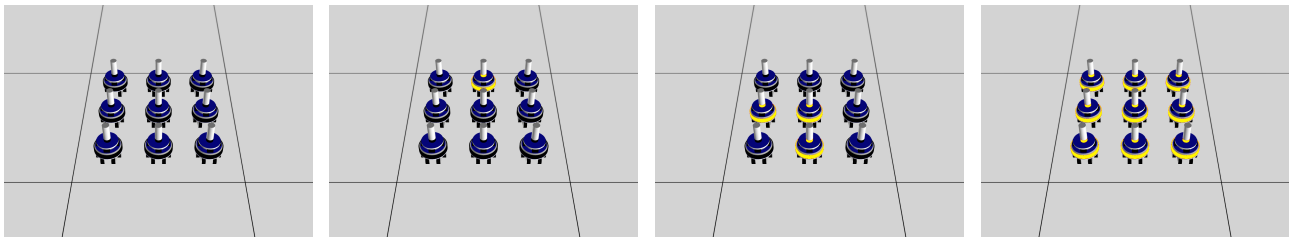
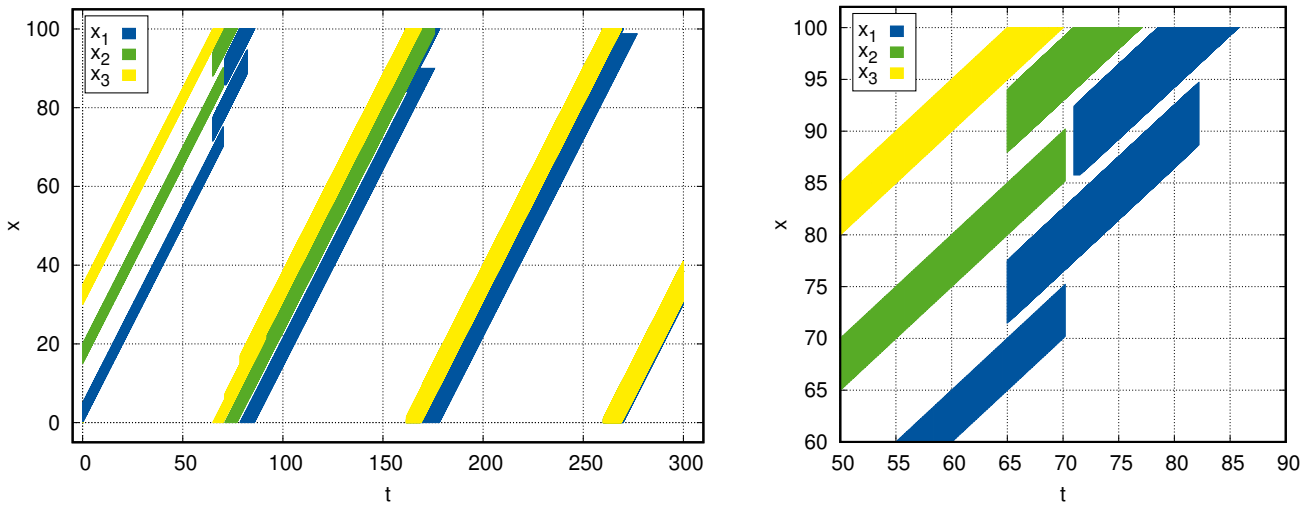


Fig. 2: Synchronization of flashing behavior within a swarm.



(a) Sets of reachable states.

(b) Detail of Fig. 4a.

Fig. 3: Reachable sets for the synchronization of three robots. Figure 4a shows the evolution of reachable sets over a time horizon of $300s$, while Figure 4b shows the effect of the first discrete jumps in more detail.

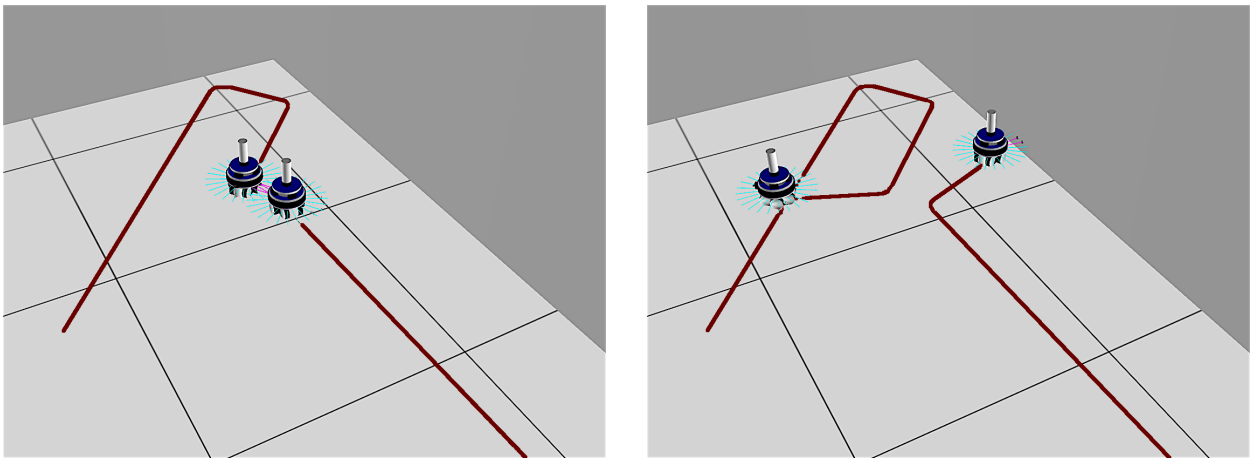
where zero reflects a heading along the x-axis and angles increase counterclockwise. This behavior will be active until both robots are sufficiently far away from each other, that is, their distance is above the security threshold. A slightly different behavior is triggered whenever a static obstacle, is detected. In this case, the heading of the robot is changed following simple reflection-like behavior. This experiment, albeit in a simplified form, shows how reachability analysis can be employed to analyze robotics systems involving differential drive actuation, field of view of sensors, heading of robots as well as collision avoidance. A simulation in ARGoS of the behaviors described above is shown in Figure 5.

B.0.b **Modeling.**. The automaton for the collision avoidance problem is shown in Figure 6. The notation is analogous to the one used in Figure 2. We use $\mathcal{B}$ to represent the workspace of the robot and by $d_{i,j}$ we denote the distance between two robots, which has to be above a security threshold ε. A clock c is used to model timing constraints on the execution of the controller, *e.g.* the frequency at which the controller executes.

Modeling this scenario is non-trivial, due to intricacies of the problem as well as limitations of the tools. First of all, our scenario includes complex behaviors leading to large models. Given the presence of nonlinear terms in the kinematic model of the robots, only Flow* could be used for the analysis of the model. The tool does not provide support for urgency, forcing us to use clock invariants to obtain a similar behavior. Moreover, if two robots get too close we impose a turn of a fixed angle – e.g., $\frac{\pi}{3}$ to the right in the example shown here. Since $\theta_i \in [0, 2\pi]$ for each robot, additional locations (labeled as `adapting` θ_i) have to be introduced to check, and adjust if needed, the value of each of the robots' headings. Modeling the collision avoidance for walls requires to take into account the robot's pose upon their detection. To do this we enumerate all possible configurations in which the robot is about to leave the workspace and for each we introduce transitions and locations to update the robots' heading accordingly. For the sake of simplicity this is represented in Figure 6 by means of the predicates `atB`(x, y, θ) and `update`(θ) and the location `wall avoidance`.

To do this, we introduce additional locations (labeled as `wall avoidance`) to update the heading of each robot, as Flow* does not support multi-edges.

B.0.c **Reachability Analysis results.**. As the dynamics of this model is nonlinear due to the involved transcendental functions, we could not make use of our own library, which is only suitable for linear hybrid



(a) Sensors detect a violation of the safety area.

(b) Collision avoidance is triggered, robots turn.

Fig. 4: Collision avoidance behavior with two robots and static obstacles. Trajectories followed by robots are also depicted.

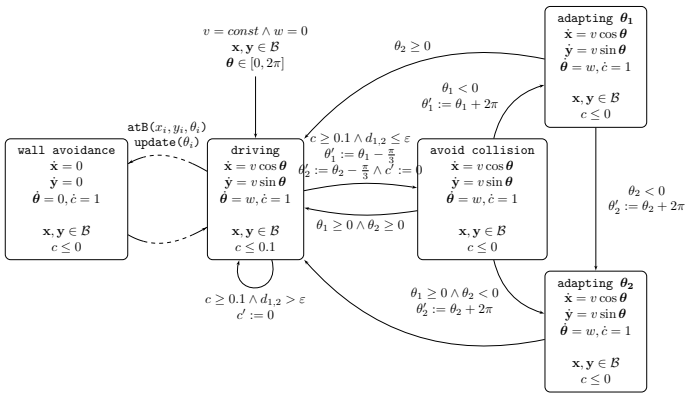


Fig. 5: Hybrid automaton for the collision avoidance with two robots. To ease model understanding not all locations are depicted here.

automata and used FLOW* instead. During analysis we tracked the position of two robots behaving according to the automaton presented before. Our first results (see Figure 7a) show the trajectories of two robots obtained by using an exact initial pose p_i – here $p_1 = (4, 1, \frac{3\pi}{4})$ and $p_2 = (2, 1, \frac{\pi}{4})$. Initially both robots drive straight until the distance between them is below the safety threshold ε. The control switches to collision avoidance behavior instantly adjusting the heading by a fixed amount. Note that more than one loop between `driving` and `avoid collision` might be needed before a safe distance is established again. Furthermore we can observe, that after bouncing off the walls, both robots encounter each other again, resulting in a second evasive maneuver. The presented plots show a projection of the trajectory abstracting away time. Therefore an intersection of trajectories does not necessarily imply a collision of the robots.

In a second experiment, we specified the starting positions within sets while the heading was fixed as before. The projections of the set of reachable states for both robots after the first maneuver are shown in Figure 7b. From the results we can see the impact of the larger initial sets, as the flowpipes containing the possible trajectories show branching behavior. Branching is due to the non-determinism introduced by working with sets. In general, sets contain more trajectories, which can satisfy certain predicates at different points in time. In this case, while parts of the flowpipe already allow to end the evasive maneuver (*i.e.* the guard predicate to `avoid collision` is not satisfied), others still contain trajectories whose distance is below the threshold ε forcing the control to continue collision avoidance. The observed effect for even small initial sets already indicates that classical approaches based on simulation are prone to miss certain cases. Reachability analysis instead tracks every single trajectory starting from the initial set and can provide conclusive results. However this comes at the price of increased computational effort as this approach accounts for all possible systems behaviors due to the introduced non-determinism on single maneuvers.

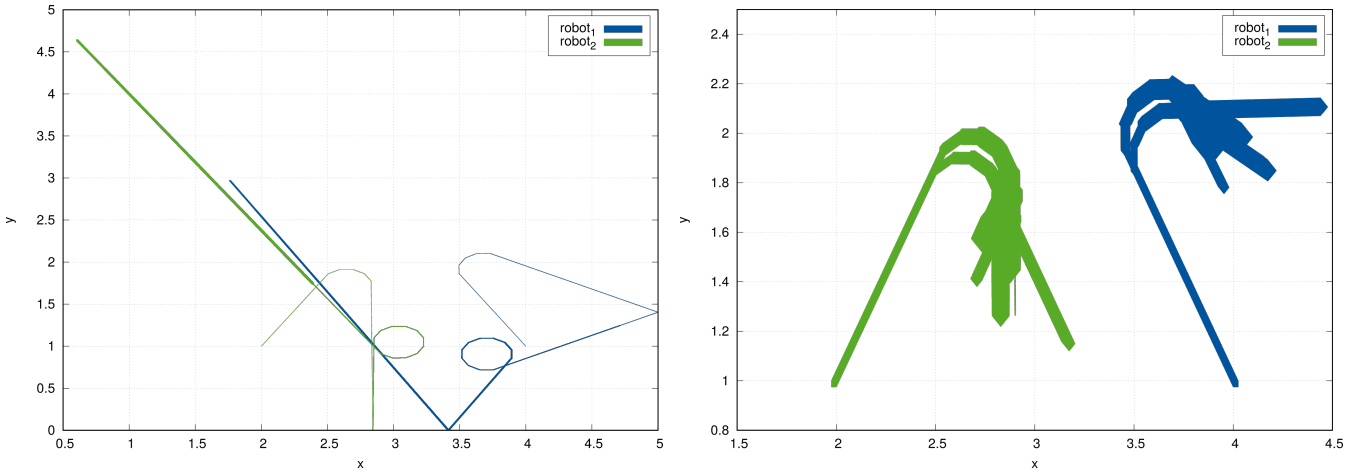
CONCLUSIONS

In this paper we have shown how algorithms and tools for verification of hybrid systems could be employed to analyze controllers for swarm robotics. Reachability analysis via flowpipe construction was used as the basis for the verification of global swarm behavior. Experimental results show the feasibility and potential of our approach, nevertheless several challenges are yet to be addressed in order to increase the applicability of such technique to robotics.

REFERENCES

[1] Alonso-Mora, J., Lohaus, S.H., Leemann, P., Siegwart, R., Beardsley, P.A.: Gesture based human - multi-robot swarm interaction and its application to an interactive display. In: Proc. of ICRA'15 (2015)

[2] Althoff, M., Dolan, J.M.: Online verification of automated road vehicles using reachability analysis. IEEE Transaction on Robotics 30(4), 903–918 (2014)



(a) Simulation. (b) Reachable sets.

Fig. 6: Collision avoidance with two robots. A sample simulation run of the model is shown in Figure 7a. The reachable sets for two robots performing collision avoidance are depicted in Figure 7b.

[3] Beckers, R., Holland, O.E., Deneubourg, J.L.: From local actions to global tasks: Stigmergy and collective robotics. In: Prerational Intelligence: Interdisciplinary Perspectives on the Behavior of Natural and Artificial Systems, vol. 2, pp. 1008–1022. Springer (2000)

[4] Bonani, M., Longchamp, V., Magnenat, S., Rétornaz, P., Burnier, D., Roulet, G., Vaussard, F., Bleuler, H., Mondada, F.: The MarXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In: Proc. of IROS'10. pp. 4187–4193 (2010)

[5] Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M.: Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence 7(1), 1–41 (2013)

[6] Brambilla, M., Pinciroli, C., Birattari, M., Dorigo, M.: Property-driven design for swarm robotics. In: Proc. of AAMAS'12. pp. 139–146 (2012)

[7] Chen, X., Ábrahám, E., Sankaranarayanan, S.: Taylor model flowpipe construction for non-linear hybrid systems. In: Proc. of RTSS'12. pp. 183–192 (2012)

[8] Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: An analyzer for non-linear hybrid systems. In: Proc. of CAV'13. LNCS, vol. 8044, pp. 258–263. Springer (2013)

[9] Delight, M., Ramakrishnan, S., Zambrano, T., MacCready, T.: Developing robotic swarms for ocean surface mapping. In: Proc. of ICRA'16. pp. 5309–5315 (2016)

[10] Ducatelle, F., Caro, G.A.D., Pinciroli, C., Gambardella, L.M.: Self-organized cooperation between robotic swarms. Swarm Intelligence 5(2), 73–96 (2011)

[11] Ferrante, E., Brambilla, M., Birattari, M., Dorigo, M.: Socially-mediated negotiation for obstacle avoidance in collective transport. In: Proc. of DARS'10. pp. 571–583 (2010)

[12] Frehse, G.: An introduction to hybrid automata, numerical simulation and reachability analysis. In: Formal Modeling and Verification of Cyber-Physical Systems. pp. 50–81 (2015)

[13] Frehse, G., Guernic, C.L., Donzé, A., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: SpaceEx: Scalable verification of hybrid systems. In: Proc. of CAV'11. LNCS, vol. 6806, pp. 379–395. Springer (2011)

[14] Garnier, S., Jost, C., Jeanson, R., Gautrais, J., Asadpour, M., Caprari, G., Theraulaz, G.: Aggregation behaviour as a source of collective decision in a group of cockroach-like-robots. In: Proc. of ECAL'05. pp. 169–178 (2005)

[15] Girard, A.: Reachability of uncertain linear systems using zonotopes. In: Proc. of HSCC'05. pp. 291–305 (2005)

[16] Henzinger, T.A.: The theory of hybrid automata. In: Proc. of LICS'96. pp. 278–292 (1996)

[17] HyCreate, `http://stanleybak.com/projects/hycreate/hycreate.html`

[18] Konur, S., Dixon, C., Fisher, M.: Analysing robot swarm behaviour via probabilistic model checking. Robotics and Autonomous Systems 60(2), 199–213 (2012)

[19] Kurzhanskiy, A.A., Varaiya, P.: Ellipsoidal techniques for reachability analysis of discrete-time linear systems. IEEE Transactions on Automatic Control 52(1), 26–38 (2007)

[20] Labella, T.H., Dorigo, M., Deneubourg, J.: Efficiency and task allocation in prey retrieval. In: Proc. of BioADIT'04. pp. 274–289 (2004)

[21] Le Guernic, C., Girard, A.: Reachability analysis of linear systems using support functions. Nonlinear Analysis: Hybrid Systems 4(2), 250–262 (2010)

[22] Liu, W., Winfield, A.F.T., Sa, J., Chen, J., Dou, L.: Strategies for energy optimisation in a swarm of foraging robots. In: Proc. of SAB'06. pp. 14–26 (2006)

[23] Maftuleac, D., Lee, S., Fekete, S.P., Akash, A.K., López-Ortiz, A., McLurkin, J.: Local policies for efficiently patrolling a triangulated region by a robot swarm. In: Proc. of ICRA'15 (2015)

[24] Mataric, M.J.: Designing emergent behaviors: From local interactions to collective intelligence. In: Proc. of SAB'93. pp. 432–441 (1993)

[25] Mirollo, R.E., Strogatz, S.H.: Synchronization of pulse-coupled biological oscillators. SIAM Journal on Applied Mathematics 50(6), 1645–1662 (1990)

[26] Mondada, F., Pettinaro, G.C., Guignard, A., Kwee, I.W., Floreano, D., Deneubourg, J., Nolfi, S., Gambardella, L.M., Dorigo, M.: Swarm-bot: A new distributed robotic concept. Autonomous Robots 17(2-3), 193–221 (2004)

[27] Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to interval analysis. SIAM (2009)

[28] Parker, L.E.: Current state of the art in distributed autnomous mobile robotics. In: Proc. of DARS'00. pp. 3–14 (2000)

[29] Peña, J., Rouff, C.A., Hinchey, M., Cortés, A.R.: Modeling NASA swarm-based systems: using agent-oriented software engineering and formal methods. Software and System Modeling 10(1), 55–62 (2011)

[30] Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Caro, G.D., Ducatelle, F., Birattari, M., Gambardella, L.M., Dorigo, M.: Argos: a modular, parallel, multi-engine simulator for multi-robot systems. Swarm Intelligence 6(4), 271–295 (2012)

[31] Rouff, C., Vanderbilt, A., Hinchey, M.G., Truszkowski, W., Rash, J.L.: Properties of a formal method for prediction of emergent behaviors in swarm-based systems. In: Proc. of SEFM'04. pp. 24–33 (2004)

[32] Sahin, E.: Swarm robotics: From sources of inspiration to domains of application. In: Proc. of SAB'04. pp. 10–20 (2004)

[33] Schupp, S., Abraham, E., Ben Makhlouf, I., Kowalewski, S.: Hypro: A C++ library for state set representations for hybrid systems reachability analysis. In: Proc. of NFM'17. LNCS, Springer International Publishing ((to appear))

[34] Ziegler, G.M.: Lectures on polytopes, vol. 152. Springer Science & Business Media (1995)