Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Verifying Emergent Properties of Swarms

Panagiotis Kouvaros and Alessio Lomuscio
Department of Computing, Imperial College London, UK
{p.kouvaros, a.lomuscio}@imperial.ac.uk

Abstract

We investigate the general problem of establishing whether a swarm satisfies an emergent property. We put forward a formal model for swarms that accounts for their nature of unbounded collections of agents following simple local protocols. We formally define the decision problem of determining whether a swarm satisfies an emergent property. We introduce a sound and complete procedure for solving the problem. We illustrate the technique by applying it to the Beta aggregation algorithm.

1 Introduction

Robot swarms have been put forward as an alternative to single-robot systems for a wide variety of applications including search and rescue, de-mining, and surveillance. Robots in a swarm are behaviourally identical agents interacting with each other and the environment by following relatively simple protocols [Bonabeau *et al.*, 1999]. Due to their nature, individual agents in a swarm can interact in subtle ways, thereby displaying global properties for the swarm system that are difficult to predict. Yet, for robot swarms to be deployed in key applications, guarantees about their behaviours need to be provided.

Recent research has begun to address the problem of verifying that robot swarms behave according to their specifications. This is a challenging problem because, in contrast to traditional systems, swarms are typically deployed in varying number of units. It is tolerated, expected even, that some units may develop a fault while the swarm is operating, or that a swarm may be deployed in varying number of components depending on the domain. So for validation purposes it is not sufficient to verify that a swarm satisfies a given property of interest for a determined number of components; instead, we are required to ascertain that the property is satisfied for *any number of agents in the swarm*. This entails verifying an unbounded number of systems and leads to a problem that is undecidable in general [Apt and Kozen, 1986].

In this paper we address a related yet different problem, namely when it is the case that a swarm displays an *emergent behaviour*. While no commonly accepted formal definition exists, by global or emergent behaviour it is often meant a configuration, or a characteristic property of the swarm as a whole, that may be difficult to predict with certainty when analysing the behaviours of the individual agents [Bonabeau *et al.*, 1999]. A notable characteristic of flocking, shoaling and herd behaviour is that the emergent behaviour may be exhibited only when some conditions on the number of agents present is met.

In this paper we aim to develop the foundations for establishing that, given the individual description of the agents, a certain emergent behaviour will be displayed by the system. We are interested in establishing a lower bound on the number of agents to be present for a given emergent property to be displayed. The methodology that we put forward is sound and complete; so if the swarm admits the emergent behaviour, then this fact will be established. In this case the method also guarantees that any swarm composed of *any number of agents greater than the bound* will exhibit the behaviour.

Related Work. We are not aware of previous work addressing the question above. Traditional research in verification of swarm systems involves the application of model checking techniques to particular instances of known swarm algorithms [Dixon *et al.*, 2012; Brambilla *et al.*, 2014; Kloetzer and Belta, 2007]. Systems over a certain size become quickly intractable; hence, while a property can be shown to be satisfied, no consideration on emergent behaviour can be drawn. Much closer to our work are the techniques described in [Kouvaros and Lomuscio, 2013b; 2015] where methods for the parameterised verification of interleaved interpreted systems are put forward. These, however, cannot be used to verify emergence as we do here, as they are tailored to establishing that a system meets its specifications irrespective of the number of components considered. In contrast, here we identify whether population thresholds exist for the swarm to display an emergent behaviour. While the abstract models we introduce below resemble the ones used in [Kouvaros and Lomuscio, 2015], the semantics and the notion of simulation we develop here are radically different, including all the procedures which, crucially, depend on the property under examination.

2 Swarm Systems

Swarm systems are typically composed of an arbitrary number of simple and identical robotic agents interacting with the environment. In line with common theoretical assumptions made in the literature [Sahin, 2005; Brambilla *et al.*, 2013] we here assume that each agent operates on a two-dimensional arena and communicates with its peers and the environment by means of a wireless sensor of limited range. The framework we present can be extended for agents operating in three dimensions, such as flying machines, but the assumed arena will make our presentation more concise. We consider the arena to be finite and we allow it to wrap around, similarly to a sphere, i.e., for an $\alpha \times \alpha$ arena, the cell $(1, 1)$ is one position to the right of the cell $(1, \alpha)$. We also assume that the robots update their state with high frequency; so we can model them by assuming synchronicity even if at any instance some robots may be idle [Dixon *et al.*, 2012]. To model dynamic swarms we introduce a variant of interpreted systems [Fagin *et al.*, 2003], the standard semantics for multi-agent systems, and extend it to agent templates [Kouvaros and Lomuscio, 2015].

Models for swarms. We represent robots in a swarm by means of agents modelled by a generic *template agent* $\mathcal{R}(\alpha, \beta) = (L, I, Act, P, t)$ operating in an $\alpha \times \alpha \in \mathbb{N} \times \mathbb{N}$ arena with localised communication capabilities of a limited range $\beta \in \mathbb{N}$. A template agent $\mathcal{R}(\alpha, \beta)$ comprises a set of local states $L = Loc \times S$ representing the Cartesian product between the set $Loc = \{1, \dots, \alpha\} \times \{1, \dots, \alpha\}$ of the arena square cells and the set S of *private* local states. The private states represent internal configurations of the agent, whereas Loc represents the position of the agent in the arena. Given a local state $l = (x, y, s)$, a local state $l' = (x', y', s')$ with $\min(|x-x'|, \alpha-|x-x'|) \leq \beta, \min(|y-y'|, \alpha-|y-y'|) \leq \beta$ is said to be in the communication range for the agent when in l, or, equivalently, in its *neighbourhood*. We denote this by $l \ltimes l'$. For an agent i in state l, an agent j in state l' is said to be in agent i's neighbourhood, denoted by $i \ltimes j$, if $l \ltimes l'$. Furthermore, $\mathcal{R}(\alpha, \beta)$ is defined by considering a set $I \subseteq L$ of initial local states and a nonempty set Act of (local) actions. When α, β are assumed to be given, we simply write $\mathcal{R}$ for $\mathcal{R}(\alpha, \beta)$. The framework can accommodate a finite number of template agents; for simplicity we do not pursue this here.

The template agent's actions are performed in compliance with a protocol $P : L \rightarrow \mathcal{P}(Act)$ that defines which actions may be performed at a given local state. The evolution of the agent is characterised by a (local) transition function $t : L \times L_E \times Act \times \mathcal{P}(Act) \times Act_E \rightarrow L$. The transition function returns the next agent's local state given the current agent's local state, the environment's state (see below), the agent's action, the set of actions performed by the agents in its neighbourhood, and the environment's action.

The template environment $\mathcal{E} = (L_E, I_E, Act_E, P_E, t_E)$ is defined by a nonempty set of states L_E, a nonempty set of initial states $I_E \subseteq L_E$, and a nonempty set of actions Act_E. $\mathcal{E}$ also includes a protocol $P_E : L_E \rightarrow \mathcal{P}(Act_E)$, and a transition function $t_E : L_E \times Act_E \times \mathcal{P}(Act) \rightarrow L_E$ returning the environment's next local state given its current state, its action, and the set of actions performed by all the agents in the swarm.

We define a *swarm system* (SS) as a tuple $\mathcal{S} = \langle \mathcal{R}, \mathcal{E}, \mathcal{V} \rangle$, where $\mathcal{R}$ is a template agent, $\mathcal{E}$ is a template environment and $\mathcal{V} : L \times L_E \rightarrow \mathcal{P}(AP)$ is a labelling function on the Cartesian product of the template agent's states and the environment's states for a set AP of atomic propositions. $\mathcal{S}$ gives an abstract description of an unbounded collection of *concrete swarm systems*, each one obtained by instantiating the SS with the number of actual agents in the system. So, differently from the generic SS $\mathcal{S} = \langle \mathcal{R}, \mathcal{E}, \mathcal{V} \rangle$, given an $n \geq 1$ the concrete swarm $\mathcal{S}(n)$ will encode precisely n agents interacting with the environment.

Let $\{1, \dots, n\}$ denote the set of the concrete agents in $\mathcal{S}(n)$. For any $i \in \{1, \dots, n\}$, the i-th concrete agent $R_i = (L_i, I_i, Act_i, P_i, t_i)$ is defined as follows: $L_i = L \times \{i\}$; $I_i = I \times \{i\}$; $Act_i = Act$; P_i is given by $a \in P_i((l, i))$ iff $a \in P(l)$; and t_i is given by $t_i((l, i), l_E, a, A, a_E) = (l', i)$ iff $t(l, a_E, a, A, a_E) = l'$. So, each local state of a concrete agent is made of the template local states indexed by the name of the agent in question and inherits from its template the actions, the protocols and the transition function.

A *global state* $g = (l_1, \dots, l_n, l_E)$ is a tuple of local states for all the agents in the system and the environment; it represents a snapshot of the system at a particular instant of time. Given a global state g we write $g.i$ to denote the local state of agent i in g. Given an agent $i \in \{1, \dots, n\}$, we let $\mathfrak{N}(g, i) = \{j \in \{1, \dots, n\} \mid i \ltimes j\}$ to denote the set of agents in the neighbourhood of i when at a global state g.

Consider $ACT(n) = Act_1 \times \dots Act_n \times Act_E$ to be the set of all possible joint actions that may be performed in $\mathcal{S}(n)$. Given $\bar{a} \in ACT(n)$, let $\bar{a}.i$ denote the action of agent i in $\bar{a}$. The agents' local protocols and local evolution functions determine the temporal evolution of the system's global states via a sequence of joint actions as defined below.

Definition 1 (Global transition relation). *The global transition relation $\mathcal{T}(n) \subseteq G(n) \times ACT(n) \times G(n)$ on a set $G(n)$ of concrete global states of a concrete system $\mathcal{S}(n)$ is defined as $(g, \bar{a}, g') \in \mathcal{T}(n)$ iff the following hold.*

1. *$\bar{a}.E \in P_E(g.E)$ and $t_E(g.E, \bar{a}.E, A) = g'.E$, where $A = \{\bar{a}.i \mid i \in \{1, \dots, n\}\}$;*
2. *For all $i \in \{1, \dots, n\}$, we have $\bar{a}.i \in P_i(g.i)$ and $t_i(g.i, g.E, \bar{a}.i, A, a.E) = g'.i$, where $A = \{\bar{a}.j \mid j \in \mathfrak{N}(g, i)\}$.*

So, for $\mathcal{S}(n)$ to transition from a global state g to a global state g' via a joint action $\bar{a}$, the following is required: the origin and target states and actions for the environment are in compliance with the environment's protocol and transition function; the origin and target local states and actions of each participant are in compliance with their respective concrete protocols and transition functions; finally, for every agent i, the set of actions performed by the agents in i's neighbourhood is in compliance with i's transition function. Thus, the global transition relation reflects the localised, interactive nature of swarms.

A path is a finite or infinite sequence $\pi = g^1\bar{a}^1g^2\bar{a}^2g^3\dots$ with $(g^i, \bar{a}^i, g^{i+1}) \in \mathcal{T}(n)$, for every $i \geq 1$ whenever applicable. Given a path π, we write $\pi(i)$ for the i-th state in π. The set of all paths originating from a state g is denoted by $\Pi(g)$. A global state g is said to be reachable from a global state g^1 if there is a path $\pi \in \Pi(g^1)$ such that $\pi(i) = g$, for some $i \geq 1$.

We now have all the ingredients to define our concrete semantics, i.e., the notion of concrete swarm system.

Definition 2 (Concrete swarm system). *Given a swarm system $\mathcal{S} = \langle \mathcal{R}, \mathcal{E}, \mathcal{V} \rangle$ and $n \geq 1$, the concrete swarm system*

(CSS) $\mathcal{S}(n)$ is a tuple $\mathcal{S}(n) = \langle \mathcal{G}(n), \mathcal{I}(n), \mathcal{T}(n), \mathcal{V}(n)\rangle$, where $G(n) \subseteq L_1 \times \ldots \times L_n \times L_E$ is the set of global states reachable via $\mathcal{T}(n)$ from the initial global states $I(n) = I_1 \times \ldots \times I_n \times I_E$, and $\mathcal{V}(n) : \mathcal{G}(n) \to \mathcal{P}(AP \times \{1, \ldots, n\})$ is a labelling function for a set $AP \times \{1, \ldots, n\}$ of local atomic propositions such that $(p, i) \in \mathcal{V}(n)(g)$ iff $p \in \mathcal{V}(g.i, g.E)$.

Hence a swarm system $\mathcal{S}$ is a concise description of an unbounded collection $\{\mathcal{S}(n) \mid n \geq 1\}$ of concrete swarm systems, each made of a different number of identical agents.

Swarm properties. We adopt the temporal-epistemic logic CTLK [Penczek and Lomuscio, 2003] as a specification language for robot swarms. By means of CTLK we can express properties of the agents as well as their states of knowledge over time. Given a set of agents $\{1, \ldots, n\}$ and a set of local atomic propositions $AP \times \{1, \ldots, n\}$, the CTLK formulae are defined by the following BNF grammar:

$$\phi ::= p \mid \neg\phi \mid \phi \vee \phi \mid EX\phi \mid E[\phi U \phi] \mid EG\phi \mid K_i\phi.$$

where $p \in AP \times \{1, \ldots, n\}$ and $i \in \{1, \ldots, n\}$. The epistemic modality $K_i\phi$ represents "agent i knows that ϕ" [Fagin *et al.*, 2003]. The temporal formula $EX\phi$ stands for "there is a path such that ϕ holds at the next step"; $E[\phi U \psi]$ denotes "there is a path such that at some point ψ holds and before then ϕ is true along the path"; and $EG\phi$ means "there is a path such that ϕ is globally true". Dual temporal modalities prefixed by the path quantifier A ("for all paths") can be defined as standard [Huth and Ryan, 2004]. The interpretation of CTLK formulae on a CSS $\mathcal{S}(n) = \langle \mathcal{G}(n), \mathcal{I}(n), \mathcal{T}(n), \mathcal{V}(n)\rangle$ is given as usual [Clarke *et al.*, 1999; Fagin *et al.*, 2003]: the temporal modalities are interpreted by means of the global transition relation, and the epistemic modalities are interpreted by the epistemic possibility relations $\sim_{i \in \{1,\ldots,n\}} = \{(g, g') \in \mathcal{G}(n) \times \mathcal{G}(n) \mid g.i = g'.i\}$. We write $(\mathcal{S}(n), g) \models \phi$ to denote that the formula ϕ is true at state g in $\mathcal{S}(n)$. We refer to [Penczek and Lomuscio, 2003] for the usual definition of the satisfaction relation $\models$. A formula ϕ is said to be true in $\mathcal{S}(n)$, denoted $\mathcal{S}(n) \models \phi$, if $(\mathcal{S}(n), \iota) \models \phi$ for every $\iota \in I(n)$.

As it is typical in the analysis of unbounded systems [Clarke *et al.*, 1989], we consider specifications that reflect the parametric nature of swarms. To do this, we use propositional variables to index the atomic propositions and epistemic modalities appearing in a specification. We write $\phi(\{v_1, \ldots, v_m\})$ to indicate that each of the variables $v_1, \ldots, v_m$ appears in an atomic proposition or epistemic modality in ϕ. We verify SS against properties of the form $\forall v_1 \ldots \forall v_m \left(\bigwedge_{i,j \in \{1,\ldots,m\}} \neg(v_i = v_j) \to \phi(\{v_1, \ldots, v_m\})\right)$ where $\forall$ is a universal quantifier over the variables. We denote such a formula as $\forall_V \phi(V)$, where $V = \{v_1, \ldots, v_m\}$, and we say that $\forall_V \phi(V)$ is an m-indexed formula if $|V| = m$. In other words, when evaluated on a concrete system $\mathcal{S}(n)$, the formula $\forall_V \phi(V)$ denotes a CTLK formula expressing the conjunction of all its ground instantiations under any assignment for V from the domain $\{1, \ldots, n\}$. For example, assume a foraging scenario [Liu *et al.*, 2007], and suppose that we want to express the following property: "whenever a robot starts searching for food, it knows that it will eventually deposit food in the nest". We can express this property with the 1-indexed formula $\forall_{\{i\}} AG(s_i \to K_i(AFd_i))$, where s (search) and d (deposit) are atomic propositions that hold in the appropriate states, and $AF\phi \triangleq A[\top U \phi]$. When evaluated on $\mathcal{S}(2)$, the formula denotes $AG(s_1 \to K_1(AFd_1)) \wedge AG(s_2 \to K_2(AFd_2))$.

Following the symmetric nature of $\forall_V(\phi_V)$, the formula can be evaluated by considering only the ground instantiation obtained by assigning pairwise distinct values to the variables in V from the domain $\{1, \ldots, m\}$ [Kouvaros and Lomuscio, 2013a]. For example, the above formula can be evaluated simply by considering its ground instantiation $AG(s_1 \to K_1(AFd_1))$. For the rest of the paper, an m-indexed formula equivalently refers to its aforementioned ground instantiation.

Identification of emergent behaviour. Swarm robots are typically designed to achieve a *global* behaviour [Bonabeau *et al.*, 1999]. Often, properties of interest displayed during the agents' interactions are referred to as *emergent behaviour*. The formal analysis of emergent behaviour has been mainly studied on predefined number of participants [Brambilla *et al.*, 2014; Dixon *et al.*, 2012]. However, no technique based on fixed numbers of participants can ensure the robustness of a swarm with respect to a global behaviour. When the number of participants varies the overall behaviour may change. For instance a simple local protocol might ensure that 10 flying robots remain in a flocking formation, only for this pattern to break down by adding a further robot to the swarm. This is an issue that does not arise in single robot systems and presents a severe difficulty for the swarm approach [Brambilla *et al.*, 2013]. One approach recently put forward aims to establish that a specification holds on a swarm for any number of agents in the system [Kouvaros and Lomuscio, 2015]. However, the method addresses a related, but different issue; moreover it suffers from constraints in terms of which swarms can be modelled, essentially forcing every pair of concrete systems to be behaviourally identical. In contrast here we aim to obtain the threshold over which an emergent property holds, irrespective of the behaviours displayed by the swarm instances under the boundary condition.

To do this we begin by defining emergent behaviour formally. Intuitively the emergent behaviour is displayed by the swarm only when the swarm is sufficiently populated.

Definition 3 (Emergent behaviour). *Given an SS $\mathcal{S}$, an m-indexed formula ϕ is said to be an* emergent behaviour *exhibited by $\mathcal{S}$ if there is an emergence threshold $t \in \mathbb{N}$ such that for all $t' \geq t$ we have $\mathcal{S}(t') \models \phi$.*

Thus, differently from the standard treatment of unbounded systems concerning the satisfaction of a formula by all concrete systems [Emerson and Kahlon, 2000], we are here interested in whether or not an emergence threshold exists. The definition above immediately gives raise to the following decision problem.

Definition 4 (Emergence identification problem). *The* emergence identification problem *(EIP) takes as input an m-indexed formula ϕ and an SS $\mathcal{S}$ and returns as output the threshold $t \in \{0, m, m+1, \ldots\}$ for ϕ and $\mathcal{S}$. If $t = 0$, then $\mathcal{S}$ admits no threshold for ϕ.*

We do not present the problem above as outputting true and false, but this can be done without loss of generality.

The definition of threshold above is related to the previously adopted, but stronger definition of cutoffs in [Kouvaros and Lomuscio, 2015], which refer to the number of agents that is sufficient to consider when evaluating *any formula on the system*. We stress the two notions are different and require a separate technical treatment. While procedures for solving the parameterised verification problem are typically incomplete, as we show in the next section, by giving a specification-dependent procedure we can obtain a sound and *complete* procedure to solve the EIP.

3 Emergence Identification Procedure

We now put a formal technique to solve the EIP defined above. Clearly we cannot solve the EIP by generating all the swarm concrete instances, as the number of systems to consider is unbounded. Instead, we identify an emergence threshold for either the property in question or its negation. Note that if the system does not admit any emergent behaviour, we can still use the notion of threshold above to establish the negation of the property under consideration. Also observe that we do not assume the threshold to be minimal; so if an SS admits an emergence threshold for a given emergent property ϕ, then it admits infinitely many.

Cycle-Stuttering Simulation. We begin by defining the degree of temporal depth for a CTLK formula by counting the temporal operators nested in a formula. We then define a simulation relation that preserves CTLK formulas up to a level of depth. In the following, we fix an m-indexed formula ϕ and two CSS $\mathcal{S}(n) = \langle \mathcal{G}(n), \mathcal{I}(n), \mathcal{T}(n), \mathcal{V}(n)\rangle$ and $\mathcal{S}(n') = \langle \mathcal{G}(n'), \mathcal{I}(n'), \mathcal{T}(n'), \mathcal{V}(n')\rangle$, where $n, n' \geq m$.

Definition 5. *The* temporal depth $td(\phi)$ *of* ϕ *is inductively defined as follows.*

$$\begin{aligned}
td(p) &\triangleq 0, & td(\phi \wedge \psi) &\triangleq \max(td(\phi), td(\psi)),\\
td(\neg\phi) &\triangleq td(\phi), & td(EX\phi) &\triangleq td(\phi)+1,\\
td(K_i\phi) &\triangleq td(\phi), & td(EG\phi) &\triangleq td(\phi)+1,\\
& & td(E[\phi U \psi]) &\triangleq \max(td(\phi), td(\psi))+1.
\end{aligned}$$

where ϕ, ψ are CTLK formulae and $i \in \{1, \dots, m\}$.

The notion of temporal depth finds a correspondence in the *cycle-stuttering simulation equivalence* that we now introduce. Intuitively, $\mathcal{S}(n)$ and $\mathcal{S}(n')$ are cycle-stuttering simulation equivalent if they are behaviourally identical modulo stuttering of cyclic behaviours. To formally define this equivalence, we first introduce some preliminary definitions.

A substring $\delta = \pi(i), \dots, \pi(j)$ of a concrete path π in $\mathcal{S}(n)$ is said to be a cycle if $\mathcal{V}(n)(\pi(i)) = \mathcal{V}(n)(\pi(j))$. A cycle δ' in π corresponds to the x-th repetition of δ if $\delta' = \delta^x$, e.g., $\delta' = abababa = (aba)^3$. A cyclic-decomposition of π is an inductive partition of π into alternating *non-cyclic* and *cyclic* blocks. Each non-cyclic block is a finite sequence of states in π without repetitions, whereas each cyclic block corresponds to a (possibly infinite) repetition of a cycle in π and it can be further partitioned into cyclic and non-cyclic blocks. For example, consider the path $\pi = abcbcbdabcbcbda$. The partition $[[a][bcb]^2[da]]^2$ is a cyclic decomposition of π, where $[[a][bcb]^2[da]]^2$ is a cyclic block, $[a], [da]$ are non-cyclic blocks, and $[bcb]^2$ is a cyclic block; the partition $[a][bcb]^2[da][bcb]^2[da]$ is also a cyclic decomposition of π. Given a non-cyclic block C, we write $C(i)$ for the i-th state in C, and $|C|$ for the sequence's length. For a cyclic block $\mathfrak{C}^x$, we write $\mathfrak{C}^x[i]$ for the sequence of states corresponding to the i-th repetition of the cycle associated with $\mathfrak{C}^x$. If $\mathfrak{C}^x$ cannot be decomposed further, then we treat each $\mathfrak{C}^x[i]$ as a non-cyclic block, otherwise we treat it as a cyclic decomposition.

Definition 6. *The relations $\approx_d \subseteq \mathcal{G}(n) \times \mathcal{G}(n')$ and $\cong_d \subseteq \mathcal{S}(n) \times \mathcal{S}(n')$ are defined as follows.*

We start by defining two states g and g' to be $\approx_0$-related if $\mathcal{V}(n)(g) = \mathcal{V}(n')(g')$. Then, for all $d \geq 0$, we define $\cong_d$ as follows. Two paths π and π' (either both finite or both infinite) are $\cong_d$-related if there is a cyclic decomposition $C_1\mathfrak{C}_1^{x_1}C_2\mathfrak{C}_2^{x_2}\dots$ of π and a cyclic decomposition $C'_1\mathfrak{C}'^{x'_1}_1C'_2\mathfrak{C}'^{x'_2}_2\dots$ of π' such that for all $i \geq 1$, $C_i \propto_d C'_i$ and $\mathfrak{C}_i^{x_i} \propto_d \mathfrak{C}'^{x'_i}_i$, where

- $C_i \propto_d C'_i$ if: (i) $|C_i| = |C'_i|$; (ii) for each $1 \leq j \leq |C_i|$, we have $C_i(j) \approx_d C'_i(j)$.
- $\mathfrak{C}_i^{x_i} \propto_d \mathfrak{C}'^{x'_i}_i$ if: (i) either $x_i = x'_i$ or $x_i > d+1, x'_i > d+1$; (ii) for every pair $\mathfrak{C}_i^{x_i}[z], \mathfrak{C}'^{x'_i}_i[z']$ of repetitions of $\mathfrak{C}_i^{x_i}$ and $\mathfrak{C}'^{x'_i}_i$, we have $\mathfrak{C}_i^{x_i}[z] \propto_{d'} \mathfrak{C}'^{x'_i}_i[z']$, where $d' = \min(d+1, x_i - z, x'_i - z') - 1$.

So, $C_i \propto_d C'_i$ if: (i) the blocks have the same length; and (ii) each pair $(C_i(j), C'_i(j))$ of states is $\approx_d$-related. Two blocks $\mathfrak{C}_i^{x_i}$ and $\mathfrak{C}'^{x'_i}_i$ are $\propto_d$-related if: (i) the blocks agree up to $d+1$ on the number of repetitions of their associated cycles; and (ii) each pair of repetitions $(\mathfrak{C}_i^{x_i}[z], \mathfrak{C}'^{x'_i}_i[z'])$ is $\propto_{d'}$-related, where $d'+1$ is the greatest number of repetitions in $\{1, \dots, d+1\}$ such that $\mathfrak{C}_i^{x_i}[z]$ and $\mathfrak{C}'^{x'_i}_i[z']$ can both "see" $d'+1$ repetitions ahead in their blocks.

Finally, we define $\approx_{d+1}$ as follows. Two states g and g' are $\approx_{d+1}$-related if: i) for every path $\pi \in \Pi(g)$ there is a path $\pi' \in \Pi(g')$ such that $\pi \cong_d \pi'$; and ii) for every path $\pi' \in \Pi(g')$ there is a path $\pi \in \Pi(g)$ such that $\pi' \cong_d \pi$.

Definition 7. *We say that $\mathcal{S}(n)$ and $\mathcal{S}(n')$ are cycle-stuttering simulation equivalent, denoted by $\mathcal{S}(n) \equiv_d \mathcal{S}(n')$, if $\iota \approx_d \iota'$ for every pair (ι, ι') of initial states in $\mathcal{I}(n) \times \mathcal{I}(n')$.*

The following proposition can easily be established.

Proposition 1. *Let $\pi \cong_d \pi'$. Then for each state $\pi(i)$, there is a state $\pi'(i')$ such that $\pi(i) \approx_d \pi'(i')$. Further, for each state $\pi(j)$ in $\pi(1), \dots, \pi(i)$, there is a state $\pi'(j')$ in $\pi'(1), \dots, \pi'(i')$ such that $\pi(j) \approx_d \pi'(j')$.*

Formulas with temporal nesting up to d are preserved by cycle-stuttering simulation of degree d.

Theorem 1. *If $\mathcal{S}(n) \equiv_d \mathcal{S}(n')$ and $td(\phi) \leq d$, then $\mathcal{S}(n) \models \phi$ iff $\mathcal{S}(n') \models \phi$.*

Proof sketch. The proof is by induction on d in which each case is shown by structural induction on ϕ. The $\phi = EX\psi$ case follows by observing that the successor states of two $\approx_d$-related states are $\approx_{d-1}$-related. The cases of $EG[\psi_1 U \psi_2]$

and $EG\psi$ are shown using Proposition 1. The case for $K_i\psi$ ($i \in \{1, \ldots, m\}$) is shown as follows. Let $g \approx_d g'$. If $g \models K_i\psi$, then $g^1 \models \psi$ for all g^1 with $g \sim_i g^1$. Assume g'^1 such that $g' \sim_i g'^1$. By Proposition 1, there is a state g^2 in $\mathcal{G}(n)$ with $g'^1 \approx_d g^2$; therefore $g \sim_i g^2$, hence $g^2 \models \psi$. So, the inductive hypothesis gives $g'^1 \models \psi$, hence $g' \models K_i\psi$. $\square$

Emergence Threshold Identification Procedure. We can now give an emergence threshold identification procedure (ETIP) for solving the emergence identification problem. ETIP takes as input an SS $\mathcal{S}$ and an m-indexed formula ϕ, and returns an integer $t \geq m$ representing an emergence threshold for $\mathcal{S}$ and ϕ. If ETIP returns $t = 0$, then ϕ is not an emergent property for $\mathcal{S}$.

ETIP first calls an auxilliary potential threshold identification procedure (PTIP). PTIP takes as input $\mathcal{S}$ and ϕ and returns an integer t' corresponding to either an emergence threshold for ϕ on $\mathcal{S}$, or an emergence threshold for $\neg\phi$ on $\mathcal{S}$. Then, following the potential threshold computation, ETIP employs standard model checking algorithms for CTLK, e.g., those implemented by MCMAS [Lomuscio *et al.*, 2009], to check whether $\mathcal{S}(t') \models \phi$. If the latter is the case, then t' is an emergence threshold and thus ϕ is an emergent behaviour exhibited by $\mathcal{S}$. In this case ETIP returns $t = t'$; if $\mathcal{S}(t') \not\models \phi$, ETIP returns returns $t = 0$.

We now describe the PTIP, which returns an integer $t' > m$ such that $\mathcal{S}(n) \equiv_{td(\phi)} \mathcal{S}(n')$, for every $n, n' \geq t'$. By Theorem 1, the latter implies that t' is an emergence threshold for either ϕ or $\neg\phi$. PTIP operates by means of three steps.

Step 1: Construction of the Abstract Model. In the first step, an abstract model $\hat{\mathcal{S}}(m)$ is built. $\hat{\mathcal{S}}(m)$ is an abstraction of any concrete system from $\mathcal{S}$ of arbitrary size. Its definition is inspired by models used in [Emerson and Namjoshi, 1996], but modified to represent the current semantics. $\hat{\mathcal{S}}(m)$ is defined as the tuple $\hat{\mathcal{S}}(m) = \langle \hat{\mathcal{G}(m)}, \hat{\mathcal{I}(m)}, \hat{\mathcal{T}}(m) \rangle$, where $\hat{\mathcal{G}}(m) \subseteq L_1 \times \ldots \times L_m \times (\mathcal{P}(L) \setminus \{\emptyset\}) \times L_E$ is the set of abstract states; $\hat{\mathcal{I}}(m) = I_1 \times \ldots \times I_m \times (\mathcal{P}(I) \setminus \{\emptyset\}) \times I_E$ is the set of initial abstract states; and $\hat{\mathcal{T}}(m) \subseteq \hat{\mathcal{G}}(m) \times \Lambda \times \hat{\mathcal{G}}(m)$ is the abstract transition relation for a set of labels $\Lambda = \mathcal{P}(L \times Act \times \mathcal{P}(Act) \times L) \setminus \{\emptyset\}$.

An abstract state $\gamma = (l_1, \ldots, l_m, X, l_E)$ consists of a concrete component $\gamma.c = (l_1, \ldots, l_m, l_E)$ and an abstract component $\gamma.\hat{a} = X$. γ represents any concrete state g in $\mathcal{S}(n)$, $n > m$, in which: (i) the environment is at local state l_E; (ii) the agents $1, \ldots, m$ are at local states $l_1, \ldots, l_m$, respectively; (iii) $X = \{g.i \mid i \in \{m+1, \ldots, n\}\}$ is the set of local states of the agents $m+1, \ldots, n$. Thus $\gamma.c$ encodes the atomic propositions on which ϕ is built, and $\gamma.\hat{a}$ encodes how an arbitrary number of agents may interfere with the state of $\gamma.c$. To see this, consider a transition $(\gamma, \Xi, \gamma') \in \hat{\mathcal{T}}(m)$, representing a set of transitions between the concrete states represented by γ and γ'. The label Ξ indicates the template transitions enabling the agents to participate in a global transition. In other words, a tuple $(l, a, A, l') \in \Xi$ indicates that in a global transition at least one agent is in state l; this agent performs action a, its neighbours perform the set of actions A, and the agent moves to state l'.

For a set $\Xi \subseteq \Lambda$, let $Act(\Xi) = \{a \mid \exists l, l', A. (l, a, A, l') \in \Xi\}$ to be the set of actions performed by Ξ. Given a template state $l \in L$, consider $\hat{\mathfrak{N}}(\Xi, l) = \{a \mid \exists l', l'', A. (l', a, A, l'') \in \Xi$ and $l \ltimes l'\}$ to be the set of actions performed by Ξ in the neighbourhood of l. The construction of $\hat{\mathcal{S}}(m)$ is completed by defining $\hat{\mathcal{T}}(m)$ as follows: (γ, Ξ, γ') iff there is a joint action $\bar{a} \in Act_1 \times \ldots \times Act_m \times Act_E$ such that:

1. The transition is valid for the environment: $\bar{a}.E \in P_E(\gamma.c.E)$ and $t_E(\gamma.c.E, \bar{a}.E, A) = \gamma'.c.E$, where $A = Act(\Xi) \cup \{\bar{a}.i \mid i \in \{1, \ldots, m\}\}$;
2. The transition is valid for a concrete agent: for all $i \in \{1, \ldots, m\}$, we have $\bar{a}.i \in P_i(\gamma.c.i)$ and $t_i(\gamma.c.i, \gamma.c.E, \bar{a}.i, A, \bar{a}.E) = \gamma'.c.i$, where $A = \hat{\mathfrak{N}}(\Xi, \gamma.c.i) \cup \{\bar{a}.j \mid j \in \mathfrak{N}(\gamma.c, i)\}$;
3. The transition is valid for the agents represented by $\gamma.\hat{a}$: For all $(l, a, A, l') \in \Xi$, we have $l \in \gamma.\hat{a}$, $l' \in \gamma'.\hat{a}$, $a \in P(l)$, $t(l, \gamma.c.E, a, A, \bar{a}.E) = l'$, and $A = \hat{\mathfrak{N}}(\Xi, l) \cup \{\bar{a}.i \mid i \in \{1, \ldots, m\}$ and $l \ltimes \gamma.c.i\}$;
4. For every $l \in \gamma.\hat{a}$, there is a transition label in Ξ to a state $l' \in \gamma'.\hat{a}$, and for every $l' \in \gamma'.\hat{a}$, there is a transition label in Ξ from a state $l \in \gamma.\hat{a}$.

A path in $\hat{\mathcal{S}}(m)$ is either a finite or infinite sequence $\gamma^1 \Xi^1 \gamma^2, \ldots$ with $(\gamma^i, \Xi^i, \gamma^{i+1}) \in \hat{\mathcal{T}}(m)$, for every $i \geq 1$ whenever applicable. We now establish a correspondence between any CSS $\mathcal{S}(n)$, $n > m$, and $\hat{\mathcal{S}}(m)$ as follows. Define δ_n to map concrete states from $\mathcal{S}(n)$ to abstract states in $\hat{\mathcal{S}}(m)$ as $\delta_n(g) = (g.1, \ldots, g.m, \{g.j \mid j \in \{m+1, \ldots, n\}\}, g.E)$. Assume ζ_n to map tuples in $\mathcal{T}(n)$ to tuples in Λ by $\zeta_n(g, \bar{a}, g') = \{(l, a, A, l') \mid \exists i \in \{m+1, \ldots, n\}$ s.t. $g.i = l, \bar{a}.i = a, A = \{\bar{a}.j \mid j \in \mathfrak{N}(g, i)\}$, and $g'.i = l'\}$. Finally, define θ_n to map paths in $\mathcal{S}(n)$ to paths in $\hat{\mathcal{S}}(m)$ by $\theta_n(g^1 \bar{a}^1 g^2 \ldots) = \delta_n(g^1) \zeta_n(g^1, \bar{a}^1, g^2) \delta_n(g^2) \ldots$.

Lemma 1. *For a path π in $\mathcal{S}(n)$, $\theta_n(\pi)$ is a path in $\hat{\mathcal{S}}(m)$.*

The above gives the correspondence between $\hat{\mathcal{S}}(m)$ and a CSS $\mathcal{S}(n)$. Following this, the covering lemma from [Emerson and Namjoshi, 1996] can be adapted here to show that for every path π in $\mathcal{S}(n)$ and for every $n' > n$, there is a path π' in $\mathcal{S}(n')$ such that $\theta_n(\pi) = \theta_{n'}(\pi')$. Thus $\mathcal{S}$ can only exhibit additional behaviour as the number of agents grows. This gives rise to the completeness of our procedure as it implies an emergence threshold for either ϕ or its negation. The threshold corresponds to a $t' \geq m$ with $\mathcal{S}(n) \equiv_{td(\phi)} \mathcal{S}(n')$, for every $n, n' \geq t'$. We now proceed to compute it by counting the sufficient number of agents to enable every behaviour (modulo cyclic-stuttering) in $\hat{\mathcal{S}}(m)$. To do this, we do not consider infinite paths as they may require an infinite number of concrete agents. In other words, we first prune the computation tree of $\hat{\mathcal{S}}(m)$.

Step 2: Pruning. We now build the computation tree of $\hat{\mathcal{S}}(m)$ by including all and only the paths in $\hat{\mathcal{S}}(m)$ that do not contain any cyclic repetition for $td(\phi) + 1$ times. More precisely, for a state $\gamma \in \hat{\mathcal{G}}(m)$, we associate a labelled computation tree $\mathfrak{T}(\gamma)$ rooted at γ that is inductively defined as follows.

- $\mathfrak{T}^0(\gamma)$ consists of exactly one node with label γ.
- $\mathfrak{T}^{n+1}(\gamma)$ consists of root node $\mathfrak{r}$ with label γ, and for every transition $(\gamma, \Xi, \gamma') \in \hat{\mathcal{T}}(m)$, $\mathfrak{r}$ has a subtree $\mathfrak{T}^n(\gamma')$ with edge label Ξ iff there is no path in $\mathfrak{T}^{n+1}(\gamma)$ containing a cycle with $td(\phi) + 1$ repetitions.

It is easy to see that there is a $k \in \mathbb{N}$ such that $\mathfrak{T}^k(\gamma)$ is isomorphic to every $\mathfrak{T}^{k'}(\gamma)$ with $k' \geq k$. We write $\mathfrak{T}(\gamma)$ to denote this tree. Note that $\mathfrak{T}(\gamma)$ is finite. A path $\mathfrak{B}$ in $\mathfrak{T}(\gamma)$ is a sequence $\mathfrak{r}^1\Xi^1 \ldots \Xi^{x-1}\mathfrak{r}^x$ such that $\mathfrak{r}^1$ is the root, $\mathfrak{r}^x$ is a leaf, and for every $1 \leq i \leq x-1$, Ξ^i is the label for the edge between $\mathfrak{r}^i$ and $\mathfrak{r}^{i+1}$. We write $\mathfrak{B}(i)$ for the i-th node in $\mathfrak{B}$, $\mathfrak{B}(i, \Lambda)$ for the i-th label in $\mathfrak{B}$, and $|\mathfrak{B}|$ for the number of nodes in $\mathfrak{B}$.

Step 3: Counting. Consider a path $\mathfrak{B}$ in $\mathfrak{T}(\gamma)$. Define $\mathfrak{B}(0, \Lambda) = \emptyset$, and $\mathfrak{B}(i, \Lambda) = \emptyset$ whenever $\mathfrak{B}(i)$ is a leaf node. Then, given a template state l, define $in(\mathfrak{B}(i), l) = |\{l' \mid (l', a, A, l) \in \mathfrak{B}(i-1, \Lambda)\}|$, and $out(\mathfrak{B}(i), l) = |\{l' \mid (l, a, A, l') \in \mathfrak{B}(i, \Lambda)\}|$. So, in any concrete transition $(\gamma, \bar{a}, \gamma')$ that corresponds to $(\mathfrak{B}(i-1), \Xi, \mathfrak{B}(i))$, $out(\mathfrak{B}(i-1), l)$ is the least number of agents that need to transition from l in γ, and $in(\mathfrak{B}(i), l)$ is the least number of agents that need to transition to l in γ'. Thus, let $et(\mathfrak{B}(i), l) = \max(0, out(\mathfrak{B}(i)) - in(\mathfrak{B}(i)))$. Then, define $et(\mathfrak{B}(i)) = \sum_{l \in L} et(\mathfrak{B}(i), l)$ and $et(\mathfrak{B}) = \sum_{i \in |\mathfrak{B}|} et(\mathfrak{B}(i))$. Finally, let $et(\mathfrak{T}(\gamma)) = \max(et(\mathfrak{B}) \mid \mathfrak{B} \text{ is a path in } \mathfrak{T}(\gamma))$ and $et(\hat{\mathcal{S}}(m)) = \max(et(\mathfrak{T}(\iota)) \mid \iota \in \hat{I}(m)) + m$. We obtain the following.

Theorem 2. *$et(\hat{\mathcal{S}}(m))$ corresponds to exactly one of the following: (i) an emergence threshold for $\mathcal{S}$ and ϕ; or (ii) an emergence threshold for $\mathcal{S}$ and $\neg\phi$.*

Proof sketch. The thesis follows by showing that $\mathcal{S}(n) \equiv_{td(\phi)} \mathcal{S}(n')$, for any $n, n' \geq et(\hat{\mathcal{S}}(m))$. □

The above concludes the analysis of the ETIP. The procedure first constructs the abstract model $\hat{\mathcal{S}}(m)$ for the SS $\mathcal{S}$ in question. Then, for a formula ϕ under analysis, it prunes the computation tree of $\hat{\mathcal{S}}(m)$ to include only $td(\phi)$ cyclic repetitions. Finally, the procedure counts the number $t' = et(\hat{\mathcal{S}}(m))$ of agents required for a CSS to include every behaviour of said tree. By Theorem 2, t' corresponds to an emergence threshold either for ϕ or for $\neg\phi$. Since t' is computable, the procedure is complete. This is because an emergence threshold for $\neg\phi$ implies that there is no emergence threshold for ϕ. In other words, if $\mathcal{S}(t') \models \phi$, then t' is an emergence threshold for ϕ, and therefore ϕ is an emergent behaviour. Otherwise, t' is an emergence threshold for $\neg\phi$, and thus ϕ is not an emergent behaviour.

4 Verifying the Beta Aggregation Algorithm

We exemplify the methodology described so far on the Beta aggregation algorithm [Nembrini, 2005] (here described in a simplified manner), a protocol used to aggregate robots somewhere on a grid. To begin, given a robot i, define a *lost robot* j to be a robot satisfying $i \ltimes j$ in the previous time step but not in the current step. Each robot i obeys two simple rules. The first rule concerns the number of i's neighbours that can observe a lost robot in their neighbourhoods. If this number is less than a predefined threshold β for at least one lost robot, then robot i performs a $180°$ turn. The second rule concerns the number of i's neighbours. If the first rule does not apply and the number of i's neighbours increases during a time step, then the robot performs a random $90°$ turn. If neither rule can be applied, the robot moves forward one cell.

Fix a 5×5 arena, assume a communication range of 1, and let $\beta = 0$. We can easily encode the Beta algorithm as an SS $\mathcal{S}$ consisting of a template agent $\mathcal{R}(5, 1)$ that reflects the rules above for the action selection and state evolution. The state encodes (among other concepts) the set Z of occupied cells of the agent's neighbours' neighbourhoods (See [Beta, 2015] for the full encoding).

We say that a robot i is connected to a robot j if j is in the transitive closure of i's neighbours. Note that, given the size of the arena and the communication range, i and j are connected in a global state iff they agree on the sets Z of cells encoded in their local state. Thus, associate a fresh atomic proposition q for each set Q of cells and assign q to each template state with $Z = Q$. Consider the formula $\phi = \forall_{\{v,u\}} K_v AFAG \bigvee_Q (q_v \wedge q_u)$ expressing "every robot knows that eventually it will be forever connected to every other robot". This means that not only the whole swarm will be connected in a single cluster, but everyone will know to be connected. We are interested to check whether ϕ emerges in S as the number of robots increases in the system. Following the procedure of the previous section, we can compute an emergence threshold of 16 (the full calculation is reported in [Beta, 2015]). Thus, we conclude that ϕ is an emergent behaviour iff $\mathcal{S}(16) \models \phi$. The latter query can be solved by any epistemic model checker, which would return false, thereby establishing that $\neg\phi$ is satisfied by every $\mathcal{S}(n)$, for $n \geq 16$. So the robots will not eventually congregate into a stable cluster.

We conclude that the property above is not an emergent property of the Beta protocol. We emphasise that the above result cannot be obtained with traditional model checking nor simulation techniques since an unbounded number of systems would have to be considered.

5 Conclusions and Further Work

We have introduced a technique for determining automatically whether a swarm will display an emergent behaviour irrespective of the number of agents present. The technique is provably sound and complete. While we have put forward a fully automatic methodology, our methodology can also be used to complement simulations for fast prototyping. Specifically, if early simulations on small populations show a particular behaviour to emerge, we can then use the technique here described to check whether this behaviour will still be displayed by larger systems. In future work we intend to implement the technique here presented into a toolkit that can be paired to well-established simulation toolkits to achieve this.

Acknowledgements. The research described in this paper was supported by the EPSRC Research Project Trusted Autonomous Systems (grant No. EP/I00520X/1).

References

[Apt and Kozen, 1986] K.R. Apt and D.C. Kozen. Limits for automatic verification of finite-state concurrent systems. *Information Processing Letters*, 22(6):307–309, 1986.

[Beta, 2015] Beta. The model for the Beta algorithm. http://vas.doc.ic.ac.uk/software/extensions/, 2015.

[Bonabeau *et al.*, 1999] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford university press, 1999.

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

[Brambilla *et al.*, 2014] M. Brambilla, A. Brutschy, M. Dorigo, and M. Birattari. Property-driven design for robot swarms: A design method based on prescriptive modeling and model checking. *ACM Transactions on Autonomous and Adaptive Systems*, 9(4):17, 2014.

[Clarke *et al.*, 1989] E.M. Clarke, O. Grumberg, and M.C. Browne. Reasoning about networks with many identical finite state processes. *Information and Computation*, 81(1):13–31, 1989.

[Clarke *et al.*, 1999] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. The MIT Press, 1999.

[Dixon *et al.*, 2012] C. Dixon, A. Winfield, M. Fisher, and C. Zeng. Towards temporal verification of swarm robotic systems. *Robotics and Autonomous Systems*, 60(11):1429–1441, 2012.

[Emerson and Kahlon, 2000] E. Emerson and V. Kahlon. Reducing model checking of the many to the few. In *Proceedings of CADE'00*, volume 2421 of *LNCS*, pages 236–254. Springer, 2000.

[Emerson and Namjoshi, 1996] E. Emerson and K. Namjoshi. Automatic verification of parameterized synchronous systems. In *Proceedings of CAV'96*, volume 1102 of *LNCS*, pages 87–98. Springer, 1996.

[Fagin *et al.*, 2003] R. Fagin, Y. Moses, J. Y. Halpern, and M. Y. Vardi. *Reasoning about knowledge*. The MIT Press, 2003.

[Huth and Ryan, 2004] M. R. A. Huth and M. D. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems (2nd edition)*. Cambridge University Press, 2004.

[Kloetzer and Belta, 2007] M. Kloetzer and C. Belta. Temporal logic planning and control of robotic swarms by hierarchical abstractions. *Robotics, IEEE Transactions on*, 23(2):320–330, 2007.

[Kouvaros and Lomuscio, 2013a] P. Kouvaros and A. Lomuscio. Automatic verification of parameterised interleaved multi-agent systems. In *Proceedings of AAMAS'13*, pages 861–868. IFAAMAS, 2013.

[Kouvaros and Lomuscio, 2013b] P. Kouvaros and A. Lomuscio. A cutoff technique for the verification of parameterised interpreted systems with parameterised environments. In *Proceedings of IJCAI'13*, pages 2013–2019. AAAI Press, 2013.

[Kouvaros and Lomuscio, 2015] P. Kouvaros and A. Lomuscio. A counter abstraction technique for the verification of robot swarms. In *Proceedings of AAAI'15*. AAAI Press, 2015.

[Liu *et al.*, 2007] W. Liu, A. Winfield, J. S, J.Chen, and L. Dou. Strategies for energy optimisation in a swarm of foraging robots. In *Swarm robotics*, LNCS, pages 14–26. Springer, 2007.

[Lomuscio *et al.*, 2009] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A model checker for the verification of multi-agent systems. In *Proceedings of CAV'09*, volume 5643 of *LNCS*, pages 682–688. Springer, 2009.

[Nembrini, 2005] J. Nembrini. *Minimalist Coherent Swarming of Wireless Networked Autonomous Mobile Robots*. PhD thesis, University of the West of England, 2005.

[Penczek and Lomuscio, 2003] W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundamenta Informaticae*, 55(2):167–185, 2003.

[Sahin, 2005] E. Sahin. Swarm robotics: From sources of inspiration to domains of application. In *Swarm Robotics*, volume 3342 of *LNCS*, pages 10–20. Springer, 2005.