A Counter Abstraction Technique for the Verification of Robot Swarms

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

Abstract

We study parameterised verification of robot swarms against temporal-epistemic specifications. We relax some of the significant restrictions assumed in the literature and present a counter abstraction approach that enable us to verify a potentially much smaller abstract model when checking a formula on a swarm of any size. We present an implementation and discuss experimental results obtained for the alpha algorithm for robot swarms.

1 Introduction

A robot swarm is a collection of behaviourally identical robotic agents (Şahin and Winfield 2008). Typically, each agent interacts with its peers and the environment by following simple, local, nature-inspired protocols that enable the swarm to achieve sophisticated, global behaviour (Engelbrecht 2006). Swarm robots are attractive as they may provide an efficient and robust mechanism to achieve complex tasks (Beni 2005; Tan and Zheng 2013), such as remote surveillance (Spears et al. 2005), search and rescue (Pettinaro et al. 2002), and de-mining (Zafar, Qazi, and Baig 2006). This is in contrast with single-robot systems that are often extremely complex, expensive and offer little or no fault-tolerance.

While robot swarms are appealing, predicting their collective behaviour is known to be problematic as this may depend on the number of robots present and their interaction among themselves and with their environment. Yet, without assurances on their resulting behaviour, robot swarms cannot be deployed in a large range of scenarios. Model checking (Clarke, Grumberg, and Peled 1999) is a widely used verification technique in reactive systems and Multi-Agent Systems (MAS). It involves encoding a system S by means of a mathematical model M_S and checking whether the model M_S satisfies a property P, or $M_S \models \phi_S$, where ϕ_S is a logical formula representing the property P under analysis. A fundamental assumption in traditional model checking is that all the components present in the system are defined and modelled at design time. This cannot be assumed in robotic swarms where we often do not know how many robots will be active in the swarm initially and while the swarm evolves. Therefore we need to be able to validate properties of the swarm *irrespective of the number of robots present*. The aim of the paper is to provide a general methodology for the verification of temporal and epistemic properties of a variety of swarms with an unbounded number of participants.

The rest of the paper is organised as follows. In Section 2 we formalise swarms via a form of interpreted systems and give the syntax of our specification language. In Section 3 we present a novel approach for solving the verification problem for swarm systems by giving sufficient conditions for the identification of a cutoff. In Section 4 we present an implementation and discuss our results in the context of the alpha algorithm for swarms (Winfield et al. 2008). We conclude in Section 5 by discussing related work.

2 Semantics for Robot Swarms

Interpreted systems are a standard semantics for describing multi-agent systems (Fagin et al. 2003). Given the asynchronous nature of swarm-based systems, we here develop a variant of interleaved interpreted systems (Lomuscio, Penczek, and Qu 2010), a class of interpreted systems constraining the interleaved evolution of the agents' actions, to give a model for the evolution of swarms. Specifically, we extend and generalise parameterised interleaved interpreted systems (Kouvaros and Lomuscio 2013b), a semantics used to model multi-agent systems with an unbounded number of agents. In this setting we make the following assumptions on a swarm system: (i) the robotic agents composing the swarm are behaviourally identical and relatively simple; (ii) each agent interacts with its peers and the environment; (iii) each agent may evolve asynchronously from the rest of the swarm; (iv) each agent may become inactive while the rest of the swarm continues its evolution; (v) the number of agents present in the swarm is unknown at design time. For the common technical details we follow the presentation in (Kouvaros and Lomuscio 2013b).

Swarm Systems

We assume that a swarm is composed of an arbitrary number of identical robotic agents interacting with an environment. The behaviour of the agents and the environment is specified by a *template agent* $\mathcal{T} = (L, \iota, Act, P, t)$ and a *template environment* $\mathcal{E} = (L_E, \iota_E, Act_E, P_E, t_E)$. A template agent

$\mathcal{T}$ defines a nonempty set of local states L, a unique initial state $\iota \in L$, and a nonempty set of actions Act. Actions are performed in compliance with a protocol $P : L \to \mathcal{P}(Act)$ that selects which actions may be performed at a given state. The evolution of the template local states is characterised by a transition function $t : L \times Act \to L$ returning the next local state given the template's local state and action. Similarly, $\mathcal{E}$ is associated with a nonempty set of local states L_E, an initial state $\iota_E \in L_E$, a nonempty set of actions Act_E, a protocol P_E, and a transition function t_E. We include the "null" actions ϵ and ϵ_E to the sets of actions Act and Act_E respectively. It is assumed that the protocol P is such that for every $l \in L$ we have that $\epsilon \in P(l)$ (i.e., the null action is enabled at every template state); the transition function t is such that $t(l, \epsilon) = l$ (i.e, the local state does not change whenever the null action is performed). The environment's null action ϵ_E is similarly described.

We extend parameterised interleaved interpreted systems (Kouvaros and Lomuscio 2013b) to define a *swarm system* (SS) as a tuple $\mathcal{S} = \langle \mathcal{T}, \mathcal{E}, \mathcal{V} \rangle$, where $\mathcal{V} : L \to \mathcal{P}(AP)$ is a labelling function on the template agent's states for a set AP of template atomic propositions. $\mathcal{S}$ gives a generic description of an unbounded collection of *concrete swarm systems*, each one obtained by setting the parameter prescribing to the number of agents in the system. The concrete agents can interact either by full synchronisation among all of them with the environment, or by pairwise synchronisation between one agent and the environment. They can also evolve asynchronously at any time. Each communication pattern is the result of performing, respectively, *global-synchronous*, *agent-environment*, and *asynchronous* actions. Formally, the sets of actions $Act = A \cup AE \cup GS \cup \{\epsilon\}$ and $Act_E = A_E \cup AE \cup GS \cup \{\epsilon_E\}$ are decomposed into disjoint sets of asynchronous (A, A_E), agent-environment (AE), and global-synchronous (GS) actions. These types of actions encode our assumptions on how the agents may interact. Specifically, A actions enable a robot to evolve asynchronously; AE actions enable a robot to interact with its environment; GS actions enable the agents to interact among themselves; finally, ϵ actions enable a robot to dynamically become inactive while the swarm evolves.

Consider an SS $\mathcal{S}$. If we are given an integer $n \geq 1$ representing how many agents are present in a swarm, we can construct the concrete swarm system $\mathcal{S}(n)$ obtained by taking n instantiations of $\mathcal{T}$ and one instantiation of $\mathcal{E}$. Let $\mathcal{A}^{1,n}$ denote the set of concrete agents $\{1, \ldots, n\}$ and let $\dot{\mathcal{A}}^{1,n} = \mathcal{A}^{1,n} \cup \{\mathcal{E}(n)\}$ denote, additionally, the concrete environment $\mathcal{E}(n)$. For any $i \in \mathcal{A}^{1,n}$, the i-th concrete agent is defined as follows: $L_i = L \times \{i\}$; $\iota_i = \iota \times \{i\}$; $Act_i = A_i \cup AE_i \cup GS \cup \{\epsilon_i\}$, where $A_i = A \times \{i\}$ and $AE_i = AE \times \{i\}$. Given $a \in Act_i$, let $a_\mathcal{T}$ to denote the corresponding template action, i.e, if $a = (b, i) \in A_i \cup AE_i$, then $a_\mathcal{T} = b$, otherwise, if $a \in GS$, then $a_\mathcal{T} = a$. Agent i's protocol P_i is defined as $a \in P_i((l, i))$ iff $a_\mathcal{T} \in P(l)$; its evolution function t_i is defined as $t_i((l, i), a) = (l', i)$ iff $t(l, a_\mathcal{T}) = l'$. $\mathcal{E}(n)$ is defined as follows: $L_E(n) = L_E$, $\iota_E(n) = \iota_E$, $Act_E(n) = A_E \cup \{AE_i\}_{i \in \mathcal{A}^{1,n}} \cup GS \cup \{\epsilon_E\}$; $P_E(n)$ is given by $a \in P_E(n)(l)$ iff $a_\mathcal{T} \in P_E(l)$ and $t_E(n)$ is given by $t_E(n)(l, a) = l'$ iff $t_E(l, a_\mathcal{T}) = l'$.

A *global state* $g = (l_1, \ldots, l_n, l_E)$ is a tuple of local states for all the agents in the system and it corresponds to a description of the system at a particular instant of time. For a global state g we write $g.i$ to denote the template local state of agent i in g. The system's global states evolve over time in compliance with the agents' local protocols and local evolution functions, thereby inducing a global transition relation. For $Y \in \{Act, A, AE, GS\}$ and $X \subseteq \dot{\mathcal{A}}^{1,n}$, let $Y_X = \bigcup_{i \in X} Y_i$ denote the union of the set of actions Y for each agent in X, e.g., $A_{\mathcal{A}^{1,n}} = \bigcup_{i \in \mathcal{A}^{1,n}} A_i$. For $a \in Act_{\dot{\mathcal{A}}^{1,n}}$, let $Agent(a) = \left\{ i \in \dot{\mathcal{A}}^{1,n} : a \in Act_i \right\}$ denote the set of agents admitting the concrete action a in their repertoire.

Definition 1 (Global transition relation)**.** *The global transition relation $\mathcal{R}(n) \subseteq G(n) \times G(n)$ on a set $G(n)$ of global states is defined as $(g, g') \in \mathcal{R}(n)$ iff there is $\bar{a} = (a_1, \ldots, a_n, a_E) \in Act_1 \times \cdots \times Act_n \times Act_E(n)$ and $a \in Act_{\dot{\mathcal{A}}^{1,n}}$ such that for all $i \in Agent(a)$, we have that $a_i = a$ and $t_i(g.i, a_i) = g'.i$; and for all $i \in \dot{\mathcal{A}}^{1,n} \setminus Agent(a)$, we have that $a_i = \epsilon_i$ and $t_i(g.i, a_i) = g'.i = g.i$. In brief we write $g \to_a g'$ if $\mathcal{R}(g, g')$ by means of action a.*

Given the above, we observe the following: (i) only one local action is performed in the system at any time; (ii) for an action to be performed, every agent admitting said action in its repertoire has to perform it at the round. So, communication in SS is by means of shared actions. It follows from the agents' concrete definitions that a global transition from a global state g can only happen in either one of the following cases: (i) an A (A_E respectively) action is enabled for an agent (the environment respectively) at g; (ii) an AE action is enabled for an agent and the environment at g; (iii) a GS action is enabled for all the agents and the environment at g. We assume that the joint null action is always enabled thereby inducing a serial global transition relation.

A path π is a sequence $\pi = g^1 a^1 g^2 a^2 g^3 \ldots$ such that $g^i \to_{a^i} g^{i+1}$, for every $i \geq 1$. 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 define the concrete semantics.

Definition 2 (Concrete swarm system)**.** *Given $\mathcal{S} = \langle \mathcal{T}, \mathcal{E}, \mathcal{V} \rangle$ and $n \geq 1$, the concrete swarm system (CSS) $\mathcal{S}(n)$ is a tuple $\mathcal{S}(n) = \langle \mathcal{G}(n), \iota(n), \mathcal{R}(n), \mathcal{V}(n) \rangle$, where $G(n) \subseteq L_1 \times \ldots \times L_n \times L_E(n)$ is the set of global states reachable via $\mathcal{R}(n)$ from the initial global state $\iota(n) = ((\iota, 1), \ldots, (\iota, n), \iota_E(n))$, and $\mathcal{V}(n) : \mathcal{G}(n) \to \mathcal{P}(AP \times \mathcal{A}^{1,n})$ is a labelling function for the set $AP \times \mathcal{A}^{1,n}$ of concrete atomic propositions such that $(p, i) \in \mathcal{V}(n)(g)$ iff $p \in \mathcal{V}(g.i)$.*

In line with the parametric nature of robot swarms, an SS $\mathcal{S}$ gives a concise description of an unbounded collection $\{\mathcal{S}(n) : n \geq 1\}$ of CSS, each composed of a different number of identical agents.

$$\begin{tikzcd}
{[\mathbf{11},-,\mathbf{F},\mathbf{C},\mathbf{p1}]} \arrow[r, "north"] \arrow[d, dotted, "{south, east \atop west}"'] & {[\mathbf{11},\mathbf{N},\mathbf{F},\mathbf{C},\mathbf{s1}]} \arrow[r, "syn"] & {[\mathbf{11},\mathbf{N},\mathbf{F},\mathbf{C},\mathbf{p2}]} \arrow[r, "move"] & {[\mathbf{12},\mathbf{N},\mathbf{F},\mathbf{C},\mathbf{s2}]} \arrow[r, "syn"] & {[\mathbf{12},\mathbf{N},\mathbf{F},\mathbf{C},\mathbf{p3a}]} \arrow[d, "12"] \\
\cdots\cdots & {[\mathbf{12},\mathbf{S},\mathbf{C},\neg\mathbf{C},\mathbf{p2}]} \arrow[l, dotted] & {[\mathbf{12},\mathbf{S},\mathbf{C},\neg\mathbf{C},\mathbf{s3b}]} \arrow[l, "syn"'] & {[\mathbf{12},\mathbf{N},\mathbf{F},\mathbf{C},\mathbf{p3b}]} \arrow[l, "\neg con\_12"'] \arrow[u, dotted, "con\_12"'] & {[\mathbf{12},\mathbf{N},\mathbf{F},\mathbf{C},\mathbf{s3a}]} \arrow[l, "syn"']
\end{tikzcd}$$

Figure 1: Fragment of the template agent of the AA for a 2×2 grid and wireless range of 1.

$$\begin{tikzcd}
{[\mathbf{110},\mathbf{120},\mathbf{210},\mathbf{220},\mathbf{p1}]} \arrow[r, "syn"] & {[\mathbf{110},\mathbf{120},\mathbf{210},\mathbf{220},\mathbf{p2}]} \arrow[r, "syn"] & {[\mathbf{110},\mathbf{120},\mathbf{210},\mathbf{220},\mathbf{p3a}]} \arrow[r, "12"] \arrow[u, dotted, "{11,21,22,syn}"'] \arrow[loop below, "21"] & {[\mathbf{110},\mathbf{121},\mathbf{210},\mathbf{220},\mathbf{p3a}]} \arrow[d, "21"] \arrow[u, dotted, "{11,12,22,syn}"'] \\
& {[\mathbf{110},\mathbf{121},\mathbf{212},\mathbf{220},\mathbf{p3b}]} \arrow[u, "syn"] \arrow[loop below, "{\neg con\_11, \neg con\_12, con\_21, \neg con\_22}"] & {[\mathbf{110},\mathbf{121},\mathbf{212},\mathbf{220},\mathbf{p3a}]} \arrow[l, "syn"'] \arrow[d, dotted, "{11,12,22}"] & {[\mathbf{110},\mathbf{121},\mathbf{211},\mathbf{220},\mathbf{p3a}]} \arrow[l, "21"'] \arrow[d, dotted, "{11,12,22,syn}"] \\
& & {} & {}
\end{tikzcd}$$

Figure 2: Fragment of the template environment of the AA for a 2×2 grid and wireless range of 1.

The Swarm System of the Alpha Algorithm

As a case study we consider the alpha algorithm (AA) for robot swarms (Winfield et al. 2008). This is a *swarm aggregation* algorithm, where the robots have to form a cluster in an area of the environment (Brambilla et al. 2012; Berman et al. 2009; Martinoli, Ijspeert, and Mondada 1999).

Each robot i is equipped with a wireless sensor of limited range. Through local communication the i-th robot can observe the number $N(i)$ of robots in the range of its sensor, i.e., the number of its neighbours. Robot i is said to be connected if its neighbourhood is composed of at least α robots, for a threshold α; i.e., $N(i) \geq \alpha$. The behaviour of each robot is characterised by its connectivity status and by whether it is in *forward (motion) mode* or in *coherence (motion) mode*. More specifically, if a robot is in forward mode and connected, then it moves forward; if it is in forward mode, but not connected, then it performs a 180° turn and changes its motion mode to 'coherence'; if it is in coherence mode, but not connected, then it moves forward; finally, if it is in coherence mode and connected, then it performs a random 90° turn and changes its motion mode to 'forward'.

We can encode the alpha algorithm as an SS, where the robots are assumed to be moving on a finite square grid which wraps round, i.e., for an $N \times N$ grid, a robot moving east from position $(1, N)$ gets to position $(1, 1)$. The states of the template agent are given as tuples of four components: the position on the grid, the direction of movement ($North, East, South, West$), the motion mode ($Forward, Coherence$), the connectivity status ($Connected, \neg Connected$), and the phase ($p1, p2, p3a, p3b$) of the encoding. The encoding is in terms of three phases. For each phase $p1, p2, p3a, p3b$ an action is performed by each of the robots followed by a change of the phase component to $s1, s2, s3a, s3b$ respectively. This is followed by the global synchronous action syn thereby simulating the synchronous semantics[1] (see Figures 1 and 2). The robots are initially in square $(1, 1)$. In phase 1 each robot performs one of the asynchronous actions $north, east, south, west$, thereby choosing an initial random direction. In phase 2 each robot moves forward one step by performing the asynchronous action $move$. Phase 3 is responsible for updating the connectivity status of each of the robots. This is done in terms of 2 steps. In phase 3a each robot performs the agent-environment action (x, y), where (x, y) is the location of the robot. A template environment's state has a component $((x, y), z)$, where $z \in \{1, \ldots, \alpha + 1\}$, for each square (x, y) in the grid. The environment increases z (up to $\alpha + 1$) each time an (x, y) action is performed. Following this, the environment can deduce whether or not a robot in (x, y) is connected. In phase 3b, for each location (x, y) the environment's protocol enables the agent-environment action $con\_(x, y)$ if the sum of z's in the range of (x, y) is at least $\alpha + 1$, otherwise it enables the agent-environment action $\neg con\_(x, y)$. Thus each robot can update its connectivity status by synchronising with the environment through the con actions. This happens for all robots in a sequence of steps before the system may move to the next phase; upon each synchronisation, the corresponding agent updates its direction and motion mode as described in the previous paragraph. Then, the system goes back to phase 2 where the counters in the environment are reset to zero in repeating the cycle.

[1]Note that as stated in (Dixon et al. 2012), since the robots move at the same speed and their connectivity status is updated with high frequency, synchronicity is an adequate assumption.

Specifications for Robot Swarms

We express properties of robot swarms in the temporal-epistemic logic ACTL*K$\backslash X$ (Lomuscio, Penczek, and Qu 2010). This logic combines the epistemic logic S5 with the temporal logic ACTL$^* \setminus X$(the universal fragment of CTL* without the ne$X$t operator X). To express properties that are independent of the number of robots in the swarm, we use 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

$$\forall v_1 \ldots \forall v_m \left(\bigwedge_{i,j \in \{1,\ldots,m\}} \neg(v_i = v_j) \rightarrow \phi(\{v_1, \ldots, v_m\}) \right)$$

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 $\mathcal{S}(n)$, $\forall_V \phi(V)$ denotes a specific ACTL*K$\backslash X$ formula expressing the conjunction of all its ground instantiations under any assignment for V from the domain $\mathcal{A}^{1,n}$. For example, consider the *connectedness property* (Dixon et al. 2012) for the AA "each robot knows that it will be connected infinitely often". We can express this property with the specification $\phi_{AA} = \forall_{\{v\}} K_v GF con_v$, where *con* is a template atomic proposition holding in the template states indicating connectedness. When evaluated on $\mathcal{S}(2)$, ϕ_{AA} is a shortcut for $K_1 GF con_1 \wedge K_2 GF con_2$. We refer to (Kouvaros and Lomuscio 2013b) for more details on indexed ACTL*K$\backslash X$ including the definition of the satisfaction relation $\models$.

3 Model Checking Swarm Systems

In contrast with the existing literature on swarm systems, we here introduce a counter abstraction procedure (Pnueli, Xu, and Zuck 2002) for the verification of an SS independent from the number of agents in the system. In the verification community this is referred to as the parameterised verification problem (Clarke, Grumberg, and Browne 1989).

Definition 3 (Parameterised model checking problem (PMCP))**.** *Given an SS $\mathcal{S}$ and an m-indexed formula $\forall_V \phi(V)$, determine whether or not the following holds:*

$$\forall n \geq m : \mathcal{S}(n) \models \forall_V \phi(V)$$

Clearly, the PMCP involves checking an unbounded number of systems; so the problem cannot by solved by traditional model checking techniques. To solve this, we define a counter abstraction procedure and show how an SS may still be verified.

Abstract Swarm System

We define the abstract SS and show that any m-indexed formula $\forall_V \phi(V)$ satisfied by the abstract system is satisfied by every concrete system composed of at least m agents. To do this, we first observe that an m-indexed formula $\forall_V \phi_V$ can be evaluated by considering only its ground instantiation $\phi[m]$, obtained by assigning to the variables in V pairwise distinct values in $\mathcal{A}^{1,m}$ (Kouvaros and Lomuscio 2013b). Intuitively, this results from the symmetric nature of a CSS since each ground instantiation of $\forall_V \phi(V)$ refers to an m-tuple of identical (up to re-indexing) agents. In other words, $\phi[m]$ is equivalent to any ground instantiation of $\forall_V \phi(V)$; thus $\phi[m]$ is equivalent to $\forall_V \phi(V)$. For example the formula $\phi_{AA} = \forall_{\{v\}} K_v GF con_v$ can be evaluated simply by considering its single conjunct $\phi_{AA}[1] = K_1 GF con_1$. For the rest of the paper, we denote an m-indexed formula $\forall_V \phi_V$ by its instantiation $\phi[m]$. To reason in the abstract system about the agents that the atomic propositions and epistemic modalities in $\phi[m]$ refer to, we abstract all concrete agents but the agents $1, \ldots, m$. To do this, we define an abstract agent and an abstract environment adhering to the counter abstraction of the agents $m+1, \ldots, n$, for any concrete system $\mathcal{S}(n)$ ($n > m$), and then we define the abstract SS as the system built from the abstract agent, the abstract environment, and m concrete agents.

Definition 4 (Abstract agent)**.** *Given an SS $\mathcal{S} = \langle \mathcal{T}, \mathcal{E}, \mathcal{V} \rangle$, the* abstract agent *$\hat{a} = \langle \hat{L}, \hat{\iota}, \hat{Act}, \hat{P}, \hat{t} \rangle$ is defined as follows.*

- *$\hat{L} = \mathcal{P}(L) \backslash \emptyset$ is the set of states; $\hat{\iota} = \{\iota\} \in \hat{L}$ is the initial state;*
- *$\hat{Act} = \hat{A} \cup \hat{AE} \cup GS \cup \{\hat{\epsilon}\}$ is the set of actions, where $\hat{A} = (A \times L \times \{\uparrow, \downarrow\})$ and $\hat{AE} = (AE \times L \times \{\uparrow, \downarrow\})$;*
- *$\hat{P} : \hat{L} \rightarrow \mathcal{P}(\hat{Act})$ is the protocol defined as $\hat{P}(\hat{l}) = \{(a, l, \downarrow), (a, l, \uparrow) \in \hat{A} \cup \hat{AE} : l \in \hat{l}$ and $a \in P(l)\} \cup \{a \in GS :$ for all $l \in \hat{l}$ we have that $l \in P(l)\}$;*
- *$\hat{t} : \hat{L} \times \hat{Act} \rightarrow \hat{L}$ is the evolution function defined as $\hat{t}(\hat{l}, x) = \hat{l}'$ iff $x \in \hat{P}(\hat{l})$ and one of the following holds: (i) if $x \in GS$, then $\hat{l}' = \{l' \in L : \exists l \in \hat{l}. t(l, x) = l'\}$; (ii) if $x = (a, l, \uparrow)$, then $\hat{l}' = \hat{l} \cup \{t(l, a)\}$; (iii) if $x = (a, l, \downarrow)$, then $\hat{l}' = (\hat{l} \setminus \{l\}) \cup \{t(l, a)\}$.*

Following the definition of the PMCP, we would like to check whether an emergent behaviour encoded as an m-indexed formula holds in the swarm system irrespective of the number of robots. In line with this, the abstract agent represents in a finitary manner the ways that the agents in $\mathcal{A}^{m,n}$ may interfere with the agents in $\mathcal{A}^{1,m}$ in any CSS with at least m agents. In particular, a state in $\hat{L}$ represents the set of template states that the agents in $\mathcal{A}^{m,n}$ may be in in a global state. A template action in $\hat{Act}$ represents all of its concrete instantiations that the agents in $\mathcal{A}^{m,n}$ may perform in a global state. To respect the encoding of the abstract states, two cases have to be considered when a concrete asynchronous action or an agent-environment action a is performed at a global state: (i) there is exactly one concrete agent in template state l at which a is performed; (ii) there are at least two agents. The former case is captured in $\hat{a}$ with the action $(a_\tau, l, \downarrow)$, whereas the latter is captured with the action $(a_\tau, l, \uparrow)$. The former action may cause the abstract state to "shrink" ($\downarrow$) and not include l, whereas the latter action may cause the abstract state to "grow" ($\uparrow$) and include the template state l' as specified by the template transition $t(l, a_\tau) = l'$. Under this light, $\hat{P}$ and $\hat{t}$ are defined accordingly.

The abstract environment is defined as the concrete environment $\mathcal{E}(m)$, but with additional agent-environment actions added for synchronising with the abstract agent. Formally, $\hat{\mathcal{E}}(m) = \langle \hat{L_E}(m), \hat{\iota_E}(m), \hat{Act}_E(m), \hat{P}_E(m), \hat{t_E}(m) \rangle$, where $\hat{L_E}(m) = L_E(m), \hat{\iota_E}(m) = \iota_E(m)$ and $\hat{Act}_E(m) = Act_E(m) \cup \hat{AE}$. The abstract protocol and transition function are defined for $\hat{AE}$ as follows: $(a, l, x) \in \hat{P}_E(m)(l)$ iff

$a \in P_E(l)$ and $\hat{t_E}(l_E, (a, l, x)) = l'_E$ iff $t_E(l_E, a) = l'_E$, for $x \in \{\uparrow, \downarrow\}$.

The abstract SS $\hat{\mathcal{S}}(m) = \langle \hat{\mathcal{G}(m)}, \hat{\iota}(m), \hat{\mathcal{R}}(m), \hat{\mathcal{V}}(m) \rangle$ is constructed by composing m concrete agents, the abstract agent and the abstract environment. The abstract valuation function $\hat{\mathcal{V}}(m) : \hat{\mathcal{G}}(m) \rightarrow \mathcal{P}(AP \times \mathcal{A}^{1,m})$ is given by $p_i \in \hat{\mathcal{V}}(m)(\gamma)$ iff $p \in \mathcal{V}(\gamma.i)$.

A state $\gamma \in \hat{\mathcal{G}}(m)$ represents any concrete state g in which: (i) $\gamma.i = g.i$ for each $i \in \dot{\mathcal{A}}^{1,m}$; (ii) $g.i \in \gamma.\hat{a}$ for each $i \in \mathcal{A}^{m+1,n}$; (iii) for each $l \in \gamma.\hat{a}$, there is an $i \in \mathcal{A}^{m+1,n}$ such that $g.i = l$. Given a concrete state g, let $[g]$ denote the abstract state representing a set of states including g. The abstraction is sound in the sense that every concrete path can be mapped to an abstract path. To see this, suppose that $g \rightarrow_a g'$. If $a \in GS$, then $[g] \rightarrow_a [g']$. If $a \notin GS$, then only one agent, say i, participates in the global transition giving three cases. In the first case, we have $i \in \dot{\mathcal{A}}^{1,m}$; then $[g] \rightarrow_a [g']$. In the second case, $i \in \mathcal{A}^{m+1,n}$ and there is only one agent in $\mathcal{A}^{m+1,n}$ at template state $g.i$ in g; hence $[g] \rightarrow_{(a_\tau, g.i, \downarrow)} [g']$. In the third case, we have $i \in \mathcal{A}^{m+1,n}$ and there are at least two agents in $\mathcal{A}^{m+1,n}$ at template state $g.i$ in g; so $[g] \rightarrow_{(a_\tau, g.i, \uparrow)} [g']$. Thus we get the following.

Theorem 1. $\hat{\mathcal{S}}(m) \models \phi[m]$ *implies* $\mathcal{S}(n) \models \phi[m]$*, for any* $n > m$ *and any* m*-indexed formula* $\phi[m]$.

Following the above, if an m-indexed formula $\phi[m]$ is satisfied by $\hat{\mathcal{S}}(m)$, then we can conclude that the formula is satisfied by every concrete system. However, if $\hat{\mathcal{S}}(m) \not\models \phi[m]$, then we cannot conclude that there is an $n > m$ such that $\mathcal{S}(n) \not\models \phi[m]$.

Global-Synchronous Simulation and Cutoff Identification

We now define the notion of gs-simulation between an abstract system $\hat{\mathcal{S}}(c-1)$ and a concrete system $\mathcal{S}(c)$. We show that if $\mathcal{S}(c)$ gs-simulates $\hat{\mathcal{S}}(c-1)$, then c is a *swarm cutoff*. A cutoff for a swarm is the number of agents that is sufficient to consider when evaluating a given specification.

Definition 5 (Swarm cutoff)**.** *Given an SS* $\mathcal{S}$ *and* $m \in \mathbb{N}$, $c \in \mathbb{N}$ *is said to be a* swarm cutoff *if the following holds:*

$$\mathcal{S}(c) \models \phi[m] \textit{ iff } \forall n \geq c.\, \mathcal{S}(n) \models \phi[m]$$

for any m*-indexed formula* $\phi[m]$.

Therefore the identification of a cutoff c implies that the PMCP can be solved by model checking all the concrete systems up to $\mathcal{S}(c)$ (Emerson and Kahlon 2000; Clarke et al. 2004; Hanna, Basu, and Rajan 2009; Kaiser, Kroening, and Wahl 2010; Kouvaros and Lomuscio 2013b). Hence our verification procedure is as follows: (i) check whether or not $\hat{\mathcal{S}}(m) \models \phi[m]$; (ii) if so, then terminate; (iii) otherwise, starting from $(\mathcal{S}(m+1), \hat{\mathcal{S}}(m))$, iteratively check if there is a pair of systems in $\left\{ \Big(\mathcal{S}(m+x+1), \hat{\mathcal{S}}(m+x)\Big) : x \geq 0 \right\}$ such that $\mathcal{S}(m+x+1)$ gs-simulates, as defined below, $\hat{\mathcal{S}}(m+x)$; (iv) if such a pair $(\mathcal{S}(c), \hat{\mathcal{S}}(c-1))$ is found, for some $c > m$, then check whether $\mathcal{S}(m), \mathcal{S}(m+1), \ldots, \mathcal{S}(c) \models \phi[m]$ and terminate.

Note that the procedure above may never terminate. Indeed, in the context of SS cutoffs do not always exist (Kouvaros and Lomuscio 2013b). Note, also, that sound and incomplete techniques are typical in parameterised verification (Clarke, Talupur, and Veith 2006; 2008; German and Sistla 1992; Baldan, Corradini, and König 2008) given the problem's general undecidability (Apt and Kozen 1986). Nonetheless incomplete procedures are still of interest as they solve specific verification problems. As we show below the alpha algorithm is one of them.

In the following, for a set of actions X, let $g \rightarrow_X g'$ denote that $g \rightarrow_a g'$ for some $a \in X$, and let $\rightarrow_{X*}$ denote the reflexive and transitive closure of $\rightarrow_X$. We now define the global-synchronous simulation. A concrete system $\mathcal{S}(c)$ is said to gs-simulate the abstract system $\hat{\mathcal{S}}(c-1)$ if $\mathcal{S}(c)$ can simulate $\hat{\mathcal{S}}(c-1)$ be means of the abstract states in which an action in $Act_{\mathcal{A}^{1,m}}$ is enabled.

Definition 6 (gs-simulation)**.** $\mathcal{S}(c)$ *is said to* gs*-simulate* $\hat{\mathcal{S}}(c-1)$ $(c > m)$*, denoted* $\hat{\mathcal{S}}(c-1) \leq_{gs} \mathcal{S}(c)$*, if there is a relation* $\sim_{gs} \subseteq \hat{\mathcal{G}}(c-1) \times L \times \mathcal{G}(c)$ *such that* $(\hat{\iota}(c-1), \iota, \iota(c)) \in \sim_{gs}$ *and whenever* $(\gamma, l, g) \in \sim_{gs}$ *we have the following:*

1. $\gamma.i = g.i$*, for* $i \in \mathcal{A}^{1,m}$*;*
2. *If* $\gamma \rightarrow_{X*} \gamma^1 \rightarrow_a \gamma^2$*, for* $X = (\hat{Act} \cup Act_{\mathcal{A}^{m+1,c-1}}) \setminus GS$ *and some* $a \in Act_{\mathcal{A}^{1,m}}$*, then* $g \rightarrow_{Y*} g^1 \rightarrow_a g^2$*, where* $Y = Act_{\mathcal{A}^{m+1,c}} \setminus GS$*, and the following hold.*
   (a) *If* $a \notin GS$*, then* $(\gamma^2, l, g^2) \in \sim_{gs}$*;*
   (b) *If* $a \in GS$*, then* $h \rightarrow_{(A_i \cup AE_i \cup A_E)*} g^1$*, where* h *is as* g^1 *but* $h.i = l$ *for some* $i \in \mathcal{A}^{1,c}$*, and* $(\gamma^2, g^2.i, g^2) \in \sim_{gs}$*.*

As previously discussed, the abstract system may over approximate the concrete systems, i.e., it may be the case that not every abstract path can be mapped to a concrete path. In other words, if a specification does not hold in the abstract system, then no conclusions can be drawn on whether the specification holds in the arbitrary case. However, we can still verify an SS by identifying the systems for which the abstract system captures precisely the concrete behaviours. These systems have the following properties: given enough agents, a concrete instantiation of the system exhibits the "maximal" behaviour encoded in its abstract system; additionally, every bigger system admits the abstract behaviour as well. That is, we are interested in systems for which an emergent behaviour holds whenever a certain number (the cutoff) of agents are present, and continues to hold irrespective of how many additional agents are present in the system. Note that not all swarm systems have the aforementioned properties. Still, some systems of practical importance do. For example, in the case of the AA we expect that a robot will be infinitely often connected irrespective of the number of its peers as long as this number is greater than or equal to α.

Following the above observations, if $\hat{\mathcal{S}}(c-1) \leq_{gs} \mathcal{S}(c)$, for a CSS $\mathcal{S}(c)$, then conditions 1 and 2(a) of the gs-simulation entail that every abstract path can be mapped to

$$\begin{tikzcd}[column sep=large]
2(a) & \gamma \arrow[r, dash, dotted, "\sim_{gs}"] \arrow[d, "\text{Sequence of actions by agents not in } \mathcal{A}^{1,m}"] & l \arrow[r, dash, dotted, "\sim_{gs}"] & g \arrow[d] \\
& \gamma^1 \arrow[d, "A \text{ or } AE \text{ action by an agent in } \mathcal{A}^{1,m}"] & & g^1 \arrow[d] \\
& \gamma^2 \arrow[r, dash, dotted, "\sim_{gs}"] & l \arrow[r, dash, dotted, "\sim_{gs}"] & g^2
\end{tikzcd}$$

$$\begin{tikzcd}[column sep=large]
2(b) & \gamma \arrow[r, dash, dotted, "\sim_{gs}"] \arrow[d, "\text{Sequence of actions by agents not in } \mathcal{A}^{1,m}"] & l \arrow[r, dash, dotted, "\sim_{gs}"] & g \arrow[d, "\text{Every agent that is at template state } l \text{ in } g \text{ can move to a template state } l^1 \text{ at which } a \text{ is enabled}"'] \\
& \gamma^1 \arrow[d, "a \in GS"] & & g^1 \arrow[d, "a \in GS"'] \\
& \gamma^2 \arrow[r, dash, dotted, "\sim_{gs}"] & l^1 \arrow[r, dash, dotted, "\sim_{gs}"] & g^2
\end{tikzcd}$$

Figure 3: Conditions 2(a) (left) and 2(b) (right) of the gs-simulation. Thick arrows indicate the LHS of each condition whereas regular arrows indicate the RHS.

a concrete path in $\mathcal{S}(c)$. Condition 2(b) entails that an abstract path can always be mapped to a concrete path in any CSS $\mathcal{S}(c')$, for $c' > c$. Consequently, these guarantees entail that $\mathcal{S}(c)$ is a cutoff; therefore any behaviour exhibited by $\mathcal{S}(c)$ is not invalidated in a bigger CSS. We now describe the conditions of the gs-simulation in detail (see Figure 3).

Condition 1 insists on the equality of the template local states of the agents that $\phi[m]$ refers to in related global states. Condition 2 insists on the simulation of $Act_{\mathcal{A}^{1,m}}$ actions: whenever the abstract system can reach a state enabling an $Act_{\mathcal{A}^{1,m}}$ action, the concrete system can reach a state enabling the same action. Note that a relation satisfying only requirements 1 and 2a (where 2a is also defined for GS actions) can be used to show that $\mathcal{S}(c)$ stuttering simulates (Lomuscio, Penczek, and Qu 2010) $\hat{\mathcal{S}}(c-1)$, denoted $\hat{\mathcal{S}}(c-1) \leq_{ss} \mathcal{S}(c)$. The latter can be used to show that $\mathcal{S}(n) \leq_{ss} \mathcal{S}(c)$ for any $n \geq c$. That is, if $\phi[m]$ is satisfied by $\mathcal{S}(c)$, then $\phi[m]$ is satisfied by every $\mathcal{S}(n)$ for $n \geq c$. This is because if $\mathcal{S}(n) \leq_{ss} \mathcal{S}(c)$, then $\mathcal{S}(n) \models \phi[m]$ implies $\mathcal{S}(c) \models \phi[m]$ (Lomuscio, Penczek, and Qu 2010). However, to show that c is a cutoff we also need to show that $\mathcal{S}(c) \leq_{ss} \mathcal{S}(n)$ for every $n \geq c$. The difficulty in showing this is the potential blockage of global-synchronous transitions in a bigger system. As the enabling of a global synchronous action depends on the local state of every agent in the system, it is not necessarily the case that the extra agents in a bigger system can reach a local state enabling the GS action. This means that $\mathcal{S}(n)$ may not be able to simulate the $Act_{\mathcal{A}^{1,m}}$ actions taken in $\mathcal{S}(c)$. Following this observation, condition 2b of gs-simulation is defined to prevent the blockage of global-synchronous actions.

To see this, consider $(\gamma, l, g) \in \hat{\mathcal{G}}(c-1) \times L \times \mathcal{G}(c)$ such that $(\gamma, l, g) \in \sim_{gs}$, and consider $g' \in \mathcal{G}(n)$ such that $g'.i = g.i$ for each $i \in \mathcal{A}^{1,c}$ and $g'.i = l$ for each $i \in \mathcal{A}^{c+1,n}$. Now suppose that $\gamma \rightarrow_{X*} \gamma^1 \rightarrow_a \gamma^2$ for some $a \in GS$. Condition 2 gives $g \rightarrow_{Y*} g^1 \rightarrow_a g^2$. Then, as no GS action is performed in the path from g to g^1, $\mathcal{S}(n)$ can reach a state g'^1 from g' (by performing the same sequence of actions in the path from g to g^1) such that $g'^1.i = g^1.i$ for each $i \in \mathcal{A}^{1,m}$ and $g'^1.i = l$ for each $i \in \mathcal{A}^{c+1,n}$. Then condition 2b implies that every agent in $\mathcal{A}^{c+1,n}$ can reach the template local state of some agent in $\mathcal{A}^{1,c}$ in g'^1 thereby enabling the action a. Given this we obtain the following.

Theorem 2. *If $\hat{\mathcal{S}}(c-1) \leq_{gs} \mathcal{S}(c)$, then c is a swarm cutoff.*

The above is our main theoretical result. If the gs-simulation can be established on a given SS $\mathcal{S}$ for an m-indexed formula $\phi[m]$, then by Theorem 2 the PMCP can simply be solved by checking the finite number of concrete systems $\mathcal{S}(m), \ldots, \mathcal{S}(c)$.

4 Analysing the Alpha Algorithm

We implemented the technique described in the previous section into a novel version of the experimental toolkit checker MCMAS-P (Kouvaros and Lomuscio 2013b). MCMAS-P takes as input an SS and its corresponding abstract SS described in PISPL (Kouvaros and Lomuscio 2013b). We chose PISPL as the input language as it provides all the essential features, including templates, that we require to describe a swarm. Given an SS $\mathcal{S}$ and an m-indexed specification $\phi[m]$ to check, MCMAS-P constructs the abstract system $\hat{\mathcal{S}}(m)$ and encodes it symbolically. The underlying model-checker MCMAS (Lomuscio, Qu, and Raimondi 2009) is called to verify the abstract system against the $\phi[m]$. Following this test, if $\phi[m]$ is satisfied by $\hat{\mathcal{S}}(m)$, we can then conclude (via Theorem 1) that $\phi[m]$ holds on a concrete system of any size. If, however, $\phi[m]$ is not satisfied by $\hat{\mathcal{S}}(m)$, then the cutoff procedure is invoked. For each step $x = 0, 1, 2, \ldots$, the procedure encodes symbolically the concrete system $\mathcal{S}(m+x+1)$ and the abstract system $\hat{\mathcal{S}}(m+x)$. The BDD encodings of the transition relations are used in a depth-first strategy to check the conditions of the gs-simulation. Upon a successful termination of the cutoff procedure, MCMAS is called to verify the concrete systems $\mathcal{S}(m), \ldots, \mathcal{S}(c)$, for a cutoff c, against the specification $\phi[m]$. The results of these checks enable the user to conclude, by means of Theorem 2, whether or not the specification holds for a system of any size.

We encoded the swarm system $\mathcal{S}_{AA}$ of the alpha algorithm in PISPL and constructed its abstract system $\hat{\mathcal{S}}_{AA}$. We chose an instance of alpha algorithm where we instantiated the sensor range to the value 1 and the alpha parameter to 2. We also fixed the environment to be a 5×5 square grid. This instance is of particular interest given its failure to satisfy the connectedness property, as previously shown in (Dixon et al. 2012), when three robots constitute the swarm. However, given the nature of swarms, an emer-

#Robots	#States	Time (s)	Memory (KiB)	ϕ_{AA} ?
1	423	1	12016112	No
3 (Cutoff)	**177243**	**15**	**29925984**	No
5	2.76×10^{34}	197	50101616	No
7	TIMEOUT	TIMEOUT	TIMEOUT	TIMEOUT

Table 1: Verification results for the alpha algorithm.

gent behaviour may be exhibited when additional agents are present and may continue to be exhibited in any bigger system. Thus, we proceeded to investigate whether the connectedness property (expressed by the 1-indexed formula $\phi_{AA} = \forall_{\{v\}} K_v GF con_v$) is satisfied in the presence of additional agents.

To do this we invoked the verification procedure with input $\mathcal{S}_{AA}$, appropriately encoded in PISPL, and $\hat{\mathcal{S}}_{AA}$. The procedure first established that $\hat{\mathcal{S}}_{AA}(1)$ does not satisfy ϕ_{AA}. Indeed, $\hat{\mathcal{S}}_{AA}$ is able to simulate the concrete system of 3 agents, where, as described in (Dixon et al. 2012), 2 agents may initially go East and the remaining agent may initially go West. In this case the pair of agents initially going East are afterwards always connected, in forward mode, and moving East, whereas the agent initially going West is afterwards is never connected, in coherence mode, and moving East. Following this, the verification procedure continued to establish that $\hat{\mathcal{S}}_{AA}(2) \leq_{gs} \mathcal{S}_{AA}(3)$, thereby identifying a cutoff equal to 3. Finally, the procedure checked whether $\mathcal{S}_{AA}(x) \models \phi_{AA}$, for $x \in \{1, 2, 3\}$, returning true in the case of $x = 2$ and false in all other cases. The case of most interest is the cutoff case ($x = 3$) which allows us to conclude that the connectedness property is not satisfied by the alpha algorithm for any $x > 3$

The simulation test took approximately 9 minutes and required 65 MB of memory on an Intel Core i7 machine clocked at 3.4 GHz, with 7.7 GiB cache, running 64-bit Fedora 20, kernel 3.16.6. Model checking the cutoff system against ϕ_{AA} required approximately 15 seconds. In comparison, traditional model checking techniques would not be able to solve the problem as they would have to consider an unbounded number of concrete systems. Additionally, any conclusion that can be drawn by traditional approaches is necessarily limited to the specific case investigated. But as we build bigger systems the verification step quickly becomes intractable, as shown in Table 1.

5 Conclusions

In this paper we presented a novel approach for the verification of robot swarms independently of the number of robots in the swarm. Specifically, we put forward a sound methodology for identifying cutoffs with respect to expressive temporal-epistemic specifications. In many cases this allows us to check swarms of arbitrary size by checking systems of a limited number of components. The implementation that we presented was used to show that the Alpha algorithm for swarms is indeed not correct with respect to its intended specifications. This was shown in the absence of a collision avoidance mechanism. Collision avoidance mechanisms cannot be investigated by our methodology as it assumes a finite environment only. We leave this for future work.

Related work. A number of proposals have previously been put forward to verify swarms via model checking (Konur, Dixon, and Fisher 2012; Dixon et al. 2012; Winfield et al. 2005). These approaches attempt to analyse a swarm by verifying concrete instances of the system. While these techniques can validate the properties of the particular swarm under analysis, no guarantees can be drawn in principle on the behaviour of the swarm when a different number of agents is present. In contrast, the methodology here put forward aims to verify the system irrespective of the number of agents present in the swarm.

Parameterised model checking techniques for a fragment of the specification language here considered have previously been studied in the context of the analysis of networking protocols (Emerson and Namjoshi 1995; 1995; Aminof et al. 2014) and cutoff detection techniques have been discussed in the context of parameterised model checking for reactive systems (Clarke, Grumberg, and Browne 1989; Emerson and Kahlon 2000; Clarke et al. 2004; Emerson and Kahlon 2000; Hanna, Basu, and Rajan 2009). In particular (Emerson and Namjoshi 1996; Clarke, Talupur, and Veith 2008; German and Sistla 1992; Sun et al. 2009) use counter abstraction, as we do here, to represent any concrete system of arbitrary size by means of an abstract counter system. All these approaches differ from the one here put forward in that they focus on the particular network topology analysed, e.g., rings. Instead, we here base our analysis on the way the synchronisation between the agents of the system can happen. Additionally they are limited to temporal properties only, whereas we here consider temporal-epistemic specifications. Because of this it is difficult to compare the results even for the fragment of temporal logic that all approaches include.

In previous work (Kouvaros and Lomuscio 2013a; 2013b) we introduced a generic methodology for the verification of unbounded multi-agent systems. However, the abstraction methodologies we used there are entirely different; specifically, (Kouvaros and Lomuscio 2013a) does not consider pairwise synchronisation whereas (Kouvaros and Lomuscio 2013b) assumes that an agent's action can be performed at exactly one local state. As a result, robot swarms cannot be analysed in either of these approaches.

While this paper is related to the techniques we discussed, we are not aware of any work in the literature that would enable us to draw conclusions on the properties of the swarm independently of the number of agents present. Much remains to be investigated in this line. Most importantly the conditions we assumed on the models enable us to investigate only a small fraction of all possible swarm systems. In future work we plan to investigate the conditions under which some of these restrictions can be overcome.

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

References

Aminof, B.; Jacobs, S.; Khalimov, A.; and Rubin, S. 2014. Parameterized model checking of token-passing systems. In *Proceedings of VMCAI'14*, volume 8318 of *LNCS*, 262–281. Springer.

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

Baldan, P.; Corradini, A.; and König, B. 2008. A framework for the verification of infinite-state graph transformation systems. *Information and Computation* 206(7):869–907.

Beni, G. 2005. From swarm intelligence to swarm robotics. In *Swarm Robotics*, volume 3342 of *LNCS*. Springer. 1–9.

Berman, S.; Halász, Á.; Hsieh, M. A.; and Kumar, V. 2009. Optimized stochastic policies for task allocation in swarms of robots. *IEEE Transactions on Robotics* 25(4):927–937.

Brambilla, M.; Pinciroli, C.; Birattari, M.; and Dorigo, M. 2012. Property-driven design for swarm robotics. In *Proceedings of AAMAS'12*, 139–146. IFAAMAS.

Clarke, E.; Talupur, M.; Touili, T.; and Veith, H. 2004. Verification by network decomposition. In *Proceedings of CONCUR'04*, volume 3170 of *LNCS*. Springer. 276–291.

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

Clarke, E.; Grumberg, O.; and Peled, D. 1999. *Model Checking*. The MIT Press.

Clarke, E.; Talupur, M.; and Veith, H. 2006. Environment abstraction for parameterized verification. In *Proceedings of VMCAI'06*, volume 3855 of *LNCS*, 126–141. Springer.

Clarke, E.; Talupur, M.; and Veith, H. 2008. Proving ptolemy right: The environment abstraction framework for model checking concurrent systems. In *Proceedings of TACAS'08*, volume 4963 of *LNCS*, 33–47. Springer.

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

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

Emerson, E., and Namjoshi, K. 1995. Reasoning about rings. In *Proceedings POPL'95*, 85–94. Pearson Education.

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

Engelbrecht, A. P. 2006. *Fundamentals of computational swarm intelligence*. John Wiley & Sons.

Fagin, R.; Moses, Y.; Halpern, J. Y.; and Vardi, M. Y. 2003. *Reasoning about knowledge*. The MIT Press.

German, S. M., and Sistla, A. P. 1992. Reasoning about systems with many processes. *Journal of the ACM* 39(3):675–735.

Hanna, Y.; Basu, S.; and Rajan, H. 2009. Behavioral automata composition for automatic topology independent verification of parameterized systems. In *Proceedings of ESEC/FSE'09*, 325–334. ACM.

Kaiser, A.; Kroening, D.; and Wahl, T. 2010. Dynamic cutoff detection in parameterized concurrent programs. In *Proceedings of CAV'10*, volume 8 of *LNCS*, 645–659. Springer.

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

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

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

Lomuscio, A.; Penczek, W.; and Qu, H. 2010. Partial order reductions for model checking temporal-epistemic logics over interleaved multi-agent systems. *Fundamenta Informaticae* 101(1):71–90.

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

Martinoli, A.; Ijspeert, A. J.; and Mondada, F. 1999. Understanding collective aggregation mechanisms: From probabilistic modelling to experiments with real robots. *Robotics and Autonomous Systems* 29(1):51–63.

Pettinaro, G. C.; Kwee, I. W.; Gambardella, L. M.; Mondada, F.; Floreano, D.; Nolfi, S.; Deneubourg, J.-L.; Dorigo, M.; Kwee, I. W.; Kwee, I. W.; et al. 2002. *Swarm robotics: A different approach to service robotics*. IEEE.

Pnueli, A.; Xu, J.; and Zuck, L. 2002. Liveness with (0, 1,∞)-counter abstraction. In *Proceedings of CAV'02*, volume 1855 of *LNCS*, 93–111. Springer.

Şahin, E., and Winfield, A. 2008. Special issue on swarm robotics. *Swarm Intelligence* 2(2):69–72.

Spears, W. M.; Spears, D. F.; Heil, R.; Kerr, W.; and Hettiarachchi, S. 2005. *An overview of physicomimetics*, volume 3342 of *LNCS*. Springer.

Sun, J.; Liu, Y.; Roychoudhury, A.; Liu, S.; and Dong, J. S. 2009. Fair model checking with process counter abstraction. In *Proceedings of FM'09*, volume 5850 of *LNCS*, 123–139. Springer.

Tan, Y., and Zheng, Z.-y. 2013. Research advance in swarm robotics. *Defence Technology* 9(1):18–39.

Winfield, A. F.; Sa, J.; Fernández-Gago, M.-C.; Dixon, C.; and Fisher, M. 2005. On formal specification of emergent behaviours in swarm robotic systems. *International journal of advanced robotic systems* 2(4):363–370.

Winfield, A. F.; Liu, W.; Nembrini, J.; and Martinoli, A. 2008. Modelling a wireless connected swarm of mobile robots. *Swarm Intelligence* 2(2-4):241–266.

Zafar, K.; Qazi, S. B.; and Baig, A. R. 2006. Mine detection and route planning in military warfare using multi agent system. In *Proceedings of COMPSAC'06*, 327–332. IEEE.