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 enables 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 (Pettinari 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 $\phi_S$ 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 $T = (L, l, Act, P, t)$ and a template environment $E = (L_E, l_E, Act_E, P_E, t_E)$. A template agent
The evolution of the template local states is characterised by a transition function \( t : L \times \text{Act} \rightarrow L \) returning the next local state given the template’s local state and action. Similarly, \( E \) is associated with a nonempty set of local states \( L_E \), an initial state \( l_E \in L_E \), a nonempty set of actions \( \text{Act}_E \), a protocol \( P_E \), and a transition function \( t_E \). We include the “null” actions \( \epsilon \) and \( \epsilon_E \) to the sets of actions \( \text{Act} \) and \( \text{Act}_E \) 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 \( t_E(l) \) is similarly described.

We extend parameterised interleaved interpreted systems (Kouvaros and Lomuscio 2013b) to define a swarm system (SS) as a tuple \( S = (T, E, V) \), where \( V : L \rightarrow \mathcal{P}(\text{Act}) \) is a labelling function on the template agent’s states for a set \( \text{Act} \) of template atomic propositions. \( S \) gives a generic description of an unbounded collection of concrete swarm systems, each one obtained by setting the parameter pre- 

nomens to the number of agents in the system. The concrete syste 

In this case, 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 \( Y \in \{ \text{Act}, A, AE, GS \} \) and \( X \subseteq \mathcal{A}^{1:n} \), 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., \( A_{\mathcal{A}^{1:n}} = \bigcup_{a \in A_{\mathcal{A}^{1:n}}} A \). For \( a \in \text{Act}_{\mathcal{A}^{1:n}} \), let \( \text{Agent}(a) = \{ i \in \mathcal{A}^{1:n} : a \in \text{Act}_i \} \) 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 \( a = (a_1, \ldots, a_n, a_E) \in \text{Act}_1 \times \cdots \times \text{Act}_n \times \text{Act}_E \) and \( a \in \text{Act}_{\mathcal{A}^{1:n}} \) such that for all \( i \in \text{Agent}(a) \), we have that \( a_i = a \) and \( t_l(g, a_i) = g'.i \); and for all \( i \in \mathcal{A}^{1:n} \setminus \text{Agent}(a) \), we have that \( a_i = \epsilon \) and \( t_l(g, a_i) = g'.i \). In brief we write \( g \rightarrow_a g' \) if \( (g, g') \in \mathcal{R}(n) \) by means of action \( a \).

**Definition 2** (Concrete swarm system). Given \( S = (T, E, V) \) and \( n \geq 1 \), the concrete swarm system (CSS) \( S(n) \) is a tuple \( S(n) = (G(n), \iota(n), \mathcal{R}(n), \mathcal{V}(n)) \), where \( G(n) \subseteq L_1 \times \cdots \times L_n \times L_E(n) \) is the set of global states reachable via \( \mathcal{R}(n) \) from the initial global state \( \iota(n) = ((i_1), \ldots, (i_n), l_E(n)) \), and \( \mathcal{V}(n) : G(n) \rightarrow \mathcal{P}(\text{Act} \times \mathcal{A}^{1:n}) \) is a labelling function for the set \( \text{Act} \times \mathcal{A}^{1:n} \) 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 \( S \) gives a concise description of an unbounded collection \( \{S(n) : n \geq 1 \} \) of CSS, each composed of a different number of identical agents.
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 \(\alpha\) robots, for a threshold \(\alpha\); 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 \((\text{North}, \text{East}, \text{South}, \text{West})\), the motion mode \((\text{Forward}, \text{Coherence})\), the connectivity status \((\text{Connected}, \neg \text{Connected})\), 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 \(\text{syn}\) 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 \(\text{north}, \text{east}, \text{south}, \text{west}\), thereby choosing an initial random direction. In phase 2 each robot moves forward one step by performing the asynchronous action \(\text{move}\). 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 \(\neg \text{con} \cdot (x, y)\) if the sum of \(z\)’s in the range of \((x, y)\) is at least \(\alpha + 1\), otherwise it enables the agent-environment action \(\text{con} \cdot (x, y)\). Thus each robot can update its connectivity status by synchronising with the environment through the \(\text{con} \cdot \) 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.

Specifications for Robot Swarms

We express properties of robot swarms in the temporal-epistemic logic \(\text{ACTL}^* \K X\) (Lomuscio, Penczek, and Qu 2010). This logic combines the epistemic logic \(S5\) with the temporal logic \(\text{ACTL}^* \setminus X\) (the universal fragment of \(\text{CTL}^*\) without the \(X\) 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 \(\phi\). We verify SS against properties of the form

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

\(^1\)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.
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 $S(n)$, $\forall_V \phi(V)$ denotes a specific ACTL*K\X formula expressing the conjunction of all its ground instantiations under any assignment for $V$ from the domain $A_1^{m,n}$. For example, consider the connectedness property (Dixon et al. 2012) for the AA “each robot knows that It will be connected in-{

distinct values in $\phi$ thus can be evaluated by considering only its ground instantiation $m$], obtained by assigning to the variables in $\phi$.

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

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

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.

\textbf{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 $A_1^{m,n}$ (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 $\phi_{AA} = \forall_V K_i GF con\_{m,n}$ can be evaluated simply by considering its single conjunct $\phi_{AA}[l] = K_i GF con\_1$. 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 $S(n)$ ($n > m$), and then we define the abstract SS as the system built from the abstract agent, the abstract environment, and $m$ concrete agents.

\textbf{Definition 4} (Abstract agent). Given an SS $S = \langle T, E, V \rangle$, the abstract agent $\hat{a} = \langle L, l, \hat{\sigma}, \hat{\Pi}, \hat{t} \rangle$ is defined as follows.

- $\hat{L} = \mathcal{P}(L) \setminus \emptyset$ is the set of states; $\hat{l} = \{v\} \in \hat{L}$ is the initial state;
- $\hat{\sigma} = \hat{A} \cup \hat{AE} \cup \hat{GS} \cup \hat{\xi}$ is the set of actions, where $\hat{A} = (A \times L \times \{\downarrow, \uparrow\})$ and $\hat{AE} = (AE \times L \times \{\downarrow, \uparrow\})$;
- $\hat{\Pi} : \hat{L} \rightarrow \mathcal{P}(\hat{\sigma})$ is the protocol defined as $\hat{\Pi}(l) = \{(a, l, \uparrow), (a, l, \downarrow) \in \hat{A} \cup \hat{AE} : l \in \hat{l} \text{ and } a \in \hat{P}(l) \} \cup \{a \in \hat{GS} : \forall l \in \hat{l} \text{ we have that } l \in P(l)\}$;
- $\hat{t} : \hat{L} \times \hat{\sigma} \rightarrow \hat{L}$ is the evolution function defined as $\hat{t}(l, x) = l'$ if $x \in P(l)$ and one of the following holds: (i) if $x \in GS$, then $l' = l \in L$; (ii) if $x = (a, l, \uparrow)$, then $l' = l \cup \{l(a)\}$; (iii) if $x = (a, l, \downarrow)$, then $l' = (l \setminus \{l(a)\}) \cup \{l(a)\}$.

Following the definition of the PMCP, we would like to check whether a specification 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 $A_1^{m,n}$ may interfere with the agents in $A_1^{m,n}$ 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 $A_1^{m,n}$ may be in in a global state. A template action in $\hat{\sigma}$ represents all of its concrete instantiations that the agents in $A_1^{m,n}$ 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 $(a, l, \uparrow)$, whereas the latter is captured with the action $(a, l, \downarrow)$. The former action may cause the abstract state to “shrink” (↓) and not include $l$, whereas the latter action may cause the abstract state to “grow” (↑) and include the template state $l'$ as specified by the template transition $t(l, a_\uparrow) = l'$. Under this light, $\hat{P}$ and $\hat{t}$ are defined accordingly.

The abstract environment is defined as the concrete environment $E(m)$, but with additional agent-environment actions added for synchronising with the abstract agent. Formally, $E(m) = \langle L_E(m), \iota_E(m), Act_E(m), P_E(m), t_E(m) \rangle$, where $L_E(m) = L_{E_1}(m), \iota_E(m) = \iota_E(m)$ and $Act_E(m) = Act_{E_1}(m) \cup AE$. The abstract protocol and transition function are defined for $AE$ as follows: $(a, l, x) \in P_E(m)(l)$ if $a \in P_E(l)$ and $t_E(l_E(a, l, x)) = l_E'$ if $t_E(l_E(a, l, x)) = l_E'$ for $x \in \{\downarrow, \uparrow\}$.

The abstract SS $\hat{S}(m) = (\hat{G}(m), \hat{m}, \hat{R}(m), \hat{V}(m))$ is constructed by composing $m$ concrete agents, the abstract agent and the abstract environment. The abstract valuation function $\hat{V}(m) : \hat{G}(m) \rightarrow \mathcal{P}(AP \times A_1^{m,n})$ is given by $\hat{p}_i \in \hat{V}(m)(\gamma) \text{ if and only if } p \in V(\gamma,i)$. 

\textbf{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).

### 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 $A_1^{m,n}$ (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 $\phi_{AA} = \forall_V K_i GF con\_{m,n}$ can be evaluated simply by considering its single conjunct $\phi_{AA}[l] = K_i GF con\_1$. 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 $S(n)$ ($n > m$), and then we define the abstract SS as the system built from the abstract agent, the abstract environment, and $m$ concrete agents.
A state $\gamma \in \mathcal{G}(m)$ represents any concrete state $g$ in which: (i) $g.i = g.i$ for each $i \in A^{1,m}$; (ii) $g.i \in \gamma.a$ for each $i \in A^{m+1,n}$; (iii) for each $l \in \gamma.a$, there is an $i \in A^{n+1,n}$ 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 $i \in A^{1,m}$; then $[g] \rightarrow_a [g']$. In the second case, $i \in A^{m+1,n}$ and there is only one agent in $A^{m+1,n}$ at template state $g.i$ in $g$; hence $[g] \rightarrow_{(a,g,i,l)} [g']$. In the third case, we have $i \in A^{m+1,n}$ and there are at least two agents in $A^{m+1,n}$ at template state $g.i$ in $g$; so $[g] \rightarrow_{(a,g,i,l)} [g']$. Thus we get the following.

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

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

**Global-Synchronous Simulation and Cutoff Identification**

We now define the notion of $gs$-simulation between an abstract system $S(c-1)$ and a concrete system $S(c)$. We show that if $S(c)$ $gs$-simulates $S(c-1)$, 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 $S$ and $m \in \mathbb{N}$, $c \in \mathbb{N}$ is said to be a swarm cutoff if the following holds:

- If $S(c) \models [\phi[m]]$ then for all $n \geq c$, $S(n) \models [\phi[m]]$ and
- If $S(c) \not\models [\phi[m]]$ then for all $n \geq c$, $S(n) \not\models [\phi[m]]$ 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 $S(c)$ (Emerson and Kahl 2000; Clarke et al. 2004; Huang, Bao, and Rajan 2009; Kaiser, Kroening, and Wahl 2010; Kouvaros and Lomuscio 2013b). Hence our verification procedure is as follows: (i) check whether or not $\mathcal{S}(m) \models [\phi[m]]$; (ii) if so, then terminate; (iii) otherwise, starting from $(S(m+1), \mathcal{S}(m))$, iteratively check if there is a pair of systems in $\left\{ (S(m+x+1), \mathcal{S}(m+x)) : x \geq 0 \right\}$ such that $S(m+x+1) gs$-simulates, as defined below, $\mathcal{S}(m+x)$; (iv) if such a pair $(S(c), \mathcal{S}(c-1))$ is found, for some $c > m$, then check whether $S(c), S(m+1), \ldots, S(c) \models [\phi[m]]$ 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 $\rightarrow$. We now define the global-synchronous simulation. A concrete system $S(c)$ is said to $gs$-simulate the abstract system $S(c-1)$ if $S(c)$ can simulate $S(c-1)$ be means of the abstract states in which an action in $\text{Act}_{A^{1,m}}$ is enabled.

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

1. $\gamma.i = g.i$, for $i \in A^{1,m}$;
2. If $\gamma \rightarrow_X \gamma^1 \rightarrow_X \gamma^2$, for $X = (\text{Act} \cup \text{Act}_{A^{1,m-1,c-1}}) \setminus GS$ and some $a \in \text{Act}_{A^{1,m}}$, then $g \rightarrow_X g^1 \rightarrow_X g^2$, where $Y = \text{Act}_{A^{1,c}} \setminus GS$, and the following hold.
   (a) If $a \notin GS$, then $(\gamma^2, l, g^2) \sim_{gs}$;
   (b) If $a \in GS$, then $h \rightarrow_{(A^{1,c}, \cup_{\text{Act}_{A^{1,c}}})} g^1$, where $h$ is as $g^1$ but $h.i = l$ for some $i \in A^{1,c}$, and $(\gamma^2, g^2, i, g^2) \sim_{gs}$.

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 in which $\text{ACTL}^* \setminus X$ specifications are satisfied on a relatively small instantiation (the cutoff) and remain so even if we add further agents to 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 $\alpha$.

Following the above observations, if $\mathcal{S}(c-1) \leq_{gs} S(c)$, for a CSS $S(c)$, then conditions 1 and 2(a) of the $gs$-simulation entail that every abstract path can be mapped to a concrete path in $S(c)$. Condition 2(b) entails that an abstract path can always be mapped to a concrete path in any CSS $S(c')$, for $c' > c$. Consequently, these guarantees entail that $S(c)$ is a cutoff; therefore any behaviour exhibited by $S(c)$ 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 \( \text{Act}_{A_{1,m}} \) actions: whenever the abstract system can reach a state enabling an \( \text{Act}_{A_{1,m}} \) 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 \( \text{GS} \) actions) can be used to show that \( S(c) \) stuttering simulates \( S(c-1) \), denoted \( S(c-1) \leq_{ss} S(c) \). The latter can be used to show that \( S(n) \leq_{ss} S(c) \) for any \( n \geq c \). That is, if \( \phi[m] \) is satisfied by \( S(c) \), then \( \phi[m] \) is satisfied by every \( S(n) \) for \( n \geq c \). This is because if \( S(n) \leq_{ss} S(c) \), then \( S(c) \models \phi[m] \) implies \( S(n) \models \phi[m] \) (Lomuscio, Penczek, and Qu 2010). However, to show that \( c \) is a cutoff we also need to show that \( S(c) \) is \( \geq_{ss} S(n) \) 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 \( \text{GS} \) action. This means that \( S(n) \) may not be able to simulate the \( \text{Act}_{A_{1,m}} \) actions taken in \( S(c) \). Following this observation, condition 2b of \( \text{gs}-\text{simulation} \) is defined to prevent the blockage of global-synchronous actions.

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

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

The above is our main theoretical result. If the \( gs \)-simulation can be established on a given \( SS \) \( S \) 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 \( S(m), \ldots, S(c) \).

---

### 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 2014). 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 \( S \) and an \( m \)-indexed specification \( \phi[m] \) to check, MCMAS-P constructs the abstract system \( S(m) \) 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 \( S(m) \), 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 \( S(m) \), then the cutoff procedure is invoked. For each step \( x = 0, 1, 2, \ldots \), the procedure encodes symbolically the concrete system \( S(m+x+1) \) and the abstract system \( S(m+x) \). 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 \( S(m), \ldots, S(c) \), 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 \( S_{AA} \) of the alpha algorithm in PISPL and constructed its abstract system \( S_{AA} \). 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 \( 5 \times 5 \) 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, connectedness may emerge 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 \( \phi_{AA} = \forall v \phi_a \phi G\text{con}_v \)) is satisfied in the presence of additional agents.

To do this we invoked the verification procedure with input \( S_{AA} \), appropriately encoded in PISPL, and \( S_{AA} \). The procedure first established that \( S_{AA}(1) \) does not satisfy...
comes intractable, as shown in Table 1.

necessarily limited to the specific case investigated. But as an unbounded number of concrete systems. Additionally, any able to solve the problem as they would have to consider an

parison, traditional model checking techniques would not be

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 $S_{AA}(2) \leq_{gs} S_{AA}(3)$, thereby identifying a cutoff equal to 3. Finally, the procedure checked whether $S_{AA}(x) \models \phi_{AA}$, 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 alpha algorithm does not satisfy the connectedness property for any system with greater than 2 robots.

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 $\phi_{AA}$ 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.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
#Robots & #States & Time (s) & Memory (KIB) & $\phi_{AA}$ ? \\
\hline
1 & 423 & 1 & 12016112 & No \\
3 (Cutoff) & 177243 & 15 & 29925984 & No \\
5 & $2.76 \times 10^{34}$ & 197 & 50101616 & No \\
7 & Timeout & Timeout & Timeout & Timeout \\
\hline
\end{tabular}
\caption{Verification results for the alpha algorithm.}
\end{table}

\phi_{AA}. Indeed, $\hat{S}_{AA}$ 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 $S_{AA}(2) \leq_{gs} S_{AA}(3)$, thereby identifying a cutoff equal to 3. Finally, the procedure checked whether $S_{AA}(x) \models \phi_{AA}$, 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 alpha algorithm does not satisfy the connectedness property for any system with greater than 2 robots.

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 $\phi_{AA}$ 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.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
#Robots & #States & Time (s) & Memory (KIB) & $\phi_{AA}$ ? \\
\hline
1 & 423 & 1 & 12016112 & No \\
3 (Cutoff) & 177243 & 15 & 29925984 & No \\
5 & $2.76 \times 10^{34}$ & 197 & 50101616 & No \\
7 & Timeout & Timeout & Timeout & Timeout \\
\hline
\end{tabular}
\caption{Verification results for the alpha algorithm.}
\end{table}

\section{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.

\textbf{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.

\begin{itemize}
\item 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 Kahlton 2000; Clarke et al. 2004; Emerson and Kahlton 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.
\end{itemize}

\section*{Acknowledgments}

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

\section*{References}


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


