Using Temporal Logic to Specify Emergent Behaviours in Swarm Robotic Systems

Alan FT Winfield **Jin Sa**
Intelligent Autonomous Systems Laboratory[1]
University of the West of England
Coldharbour Lane, Bristol BS16 1QY
{Alan.Winfield,Jin.Sa}@uwe.ac.uk

Mari-Carmen Fernández-Gago **Clare Dixon** **Michael Fisher**
Liverpool Verification Laboratory[2]
Department of Computer Science, University of Liverpool
Liverpool L67 7ZF
{M.C.Gago,C.Dixon,M.Fisher}@csc.liv.ac.uk

Abstract

It is a characteristic of swarm robotics that specifying overall emergent swarm behaviours in terms of the low-level behaviours of individual robots is very difficult. Yet if swarm robotics is to make the transition from the laboratory to real-world engineering realisation we need such specifications. This paper explores the possibility of using *temporal logic* to formally specify, and possibly also prove, the emergent behaviours of a robotic swarm. The paper makes use of a simplified wireless connected swarm as a case study with which to illustrate the approach. Such a formal approach could be an important step toward a disciplined design methodology for swarm robotics.

1. Introduction

In a previous paper (Winfield et al., 2005) we introduced the notion of a 'dependable swarm', that is a distributed multi-robot system based upon the principles of swarm intelligence upon which we can place a high degree of reliance. That paper concluded that, although some of the tools needed to assure a swarm for dependability exist, most do not, and set out a roadmap of the work that needs to be done before embodied swarm intelligence can make the transition from the research laboratory to real-world applications.

One of the defining characteristics of robotic swarms is that overall swarm behaviours are, typically, an *emergent* consequence of the interaction of robots with each other and their environment (Bonabeau et al., 1999). If future real-world robotic swarms are also to exploit emergence and self-organisation to generate desired overall system behaviours then we will need to be able to verify, or better still prove, that those behaviours are guaranteed to emerge (since few real-world applications would tolerate only some possibility of desired behaviour).

Within swarm robotics research relatively little work has been done in the direction of mathematical analysis and modelling; for a recent review see (Lerman et al., 2005). Perhaps the most successful approach to date is the work of (Martinoli et al., 2004), which uses a stochastic approach in which an ensemble of probabilistic finite state machines describe the overall structure of the swarm in terms of its microscopic parameters. Martinoli's work is concerned with modelling rather than specification, or formal proof. The work we present in this paper should therefore be seen as complimentary to existing approaches.

Recently, there has been some work in the area of applying formal methods to specifying and verifying swarm intelligent systems, notably within the NASA project 'Autonomous Nano-Technology Swarm' (ANTS) (Rouff et al., 2003, Rouff et al., 2004). That work evaluated and compared four formal specification techniques: Communicating Sequential Processes (CSP), the Weighted Synchronononous Calculus

[1]http://www.ias.uwe.ac.uk
[2]http://www.csc.liv.ac.uk/~liverlab

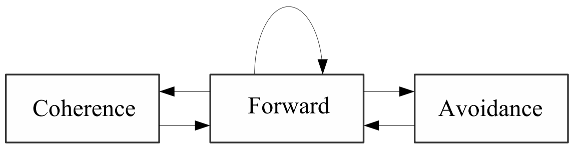

Figure 1: Robot Finite State Machine

of Sequential Systems (WSCCS), Unity Logic and X-Machines.

In this paper we shall explore the use of a *temporal logic* to formally specify and verify emergent behaviours of a robotic swarm system. Temporal logics have been shown to be useful for specifying dynamical systems that change over time (Manna and Pnueli, 1992), and we believe that this ability is essential for describing emergent behaviours. Indeed, in the world of multi-agent systems, temporal formalisms (often extended with modal logics) have been widely used for specification, verification, and even implementation (Fisher, 2005).

This paper proceeds as follows. In section 2 we introduce the wireless connected robotic swarm that forms the case study of this paper. Section 3 proposes a formal approach to swarm specification and verification, then introduces the linear-time temporal logic that we propose to use, and its notation. Section 4 then applies this formal approach to the wireless connected swarm of our case study. Finally, section 5 summarises the findings of the work to date.

2. Case study: A Wireless Connected Swarm

We have developed a class of algorithms which make use of local wireless connectivity information alone to achieve swarm aggregation (Nembrini et al., 2002, Nembrini, 2005). These algorithms use situated communications in which connectivity information is linked to robot motion so that robots within the swarm are wirelessly 'glued' together. This approach has several advantages: firstly the robots need neither absolute nor relative positional information; secondly the swarm is able to maintain its coherence (i.e. stay together) even in unbounded space and, thirdly, the connectivity needed for, and generated by, the algorithms means that the swarm naturally forms an ad-hoc communications network. Such a network would be a significant advantage in many swarm robotics applications. In this case study we make use of the simplest (alpha) algorithm. The basic premise of this algorithm is that each robot has range-limited wireless communication which, for simplicity, we model as a circle of radius r_w with the

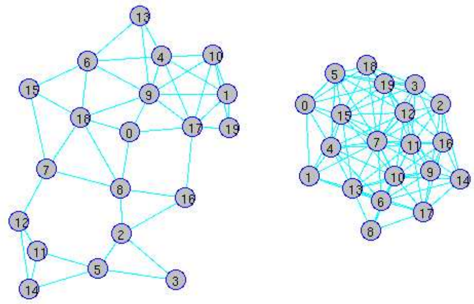

Figure 2: Swarm with $\alpha = 5$ (left) and $\alpha = 10$ (right)

robot at its centre. The boundary of the circle represents the threshold beyond which another robot is out of range. Each robot also has collision avoidance sensors with a range r_a, where $r_a < r_w$. The basic algorithm is very simple. The default behaviour of a robot is forward motion. While moving each robot periodically sends an 'are you there' message. It will receive 'yes I am here' messages only from those robots that are in range, namely its neighbours. If the number of a robot's neighbours should fall below the threshold α then it assumes it is moving out of the swarm and will execute a $180°$ turn. When the number of neighbours rises above α (when the swarm is regained) the robot then executes a random turn. This is to avoid the swarm simply collapsing in on itself. In the interests of simplicity we can consider each robot as having three basic behaviours, or states: move forward (default); avoidance (triggered by the collision sensor); and coherence (triggered by the number of neighbours falling below α). Figure 1 shows the finite state machine (FSM) for the individual robots in the swarm.

The alpha algorithm achieves useful swarm coherence in which a larger value of α results in a smaller more highly connected swarm and a smaller value of α in a larger more loosely connected swarm, as shown in figure 2.

3. A Formal Method for Swarm Development

It is the contention of this paper that formal methods can be usefully applied to swarm robotic system specification and development, especially in relating emergent behaviours to individual robot behaviours. We propose the following formal approach:

1. Formally specify the individual robots, including their *safety* and *liveness* properties.

2. Formally specify the swarm by combining the specifications of individual robots.

3. Formally specify any anticipated or desired emergent behaviours.

4. Carry out proofs to determine if the swarm satisfies any of the emergent behaviours.

Safety[1] and liveness are defined as follows. The safety property specifies the set of legal actions, i.e the set of actions that are allowed. If the robot performs actions from within this set, it will not make the system unsafe. The liveness property specifies the dynamic behaviour, i.e. the set of eventualities that *will* occur. If we only have the safety property, we cannot guarantee that anything will happen at all. If we only have the liveness property, we cannot guarantee that what is happening is safe. So we need to establish both safety and liveness.

The 4 steps proposed above can be applied iteratively. The outcomes of each iteration - typically 'proven', 'not-proven' or 'unable to determine either way' - will provide feedback to the swarm developer. Based on these outcomes, modifications to individual robot specifications may be carried out. Expectations of overall emergent behaviours may also be adjusted.

3.1 A Linear Time Temporal Logic

Temporal logic is an extension of classical logic, whereby time becomes an extra parameter when considering the truth of logical statements (Emerson, 1990). The variety of temporal logic we are particularly concerned with is based upon a discrete, linear model of time, having both a finite past and infinite future, i.e.,

$$\sigma \;=\; s_0,\, s_1,\, s_2,\, s_3,\, \ldots$$

Here, a model (σ) for the logic is an infinite sequence of states which can be thought of as 'moments' or 'points' in time. As we use a first-order temporal logic, associated with each of these states is a first-order structure.

The temporal language we use is that of classical logic extended with various modalities characterising different aspects of the temporal structure above. Examples of the key operators include '$\bigcirc\varphi$', which is satisfied if the formula φ is satisfied at the *next* moment in time, '$\Diamond\varphi$', which is satisfied if φ is satisfied at *some* future moment in time, and '$\Box\varphi$', which is satisfied if φ is satisfied at *all* future moments in time.

More formally, the semantics of the language can be defined with respect to the model (σ) in which the statement is to be interpreted, and the moment in time (i) at which it is to be interpreted. Thus, the semantics for the key temporal operators is given as follows.

$$\begin{array}{lll}
\langle\sigma,\, i\rangle \models \bigcirc A & \text{iff} & \langle\sigma,\, i+1\rangle \models A \\
\langle\sigma,\, i\rangle \models \Box A & \text{iff} & \text{for all } j \geq i.\ \langle\sigma,\, j\rangle \models A \\
\langle\sigma,\, i\rangle \models \Diamond A & \text{iff} & \text{exists } j \geq i.\ \langle\sigma,\, j\rangle \models A
\end{array}$$

We also allow standard first-order quantifiers, such as '$\exists$' and '$\forall$' and arithmetical operators.

Such a logic is widely used in the specification of concurrent and distributed systems, in both Computer Science (Manna and Pnueli, 1992) and Artificial Intelligence (Fisher et al., 2005a).

Note: as abbreviations later, we will often use formulae such as

$$\bigcirc p = p$$

meaning that the value of the variable 'p' remains the same between the current and next state. This is actually short-hand for the (legal) first-order temporal formula

$$\exists v.\ (p = v)\ \wedge\ \bigcirc(p = v)$$

i.e. 'p' has exactly the same value in the next state as it has now.

4. Applying our Formal Approach

In this section we formally specify a simplified version of the wireless connected swarm (the alpha algorithm) outlined in section 2. Section 4.1 describes the behaviours of the individual robots and the possible emergent behaviours. Section 4.2 defines the specification of the individual robots. Section 4.3 combines the specifications of the individual robots. Section 4.4 specifies some possible emergent behaviours and 4.5 outlines a route to proving the emergent behaviours.

4.1 A simplified alpha algorithm

For simplicity, we discretise the robot space so that the robots move in a grid world, and make the following assumptions.

1. The bearing of each robot will have only one of these four values: N, S, E, and W.

2. The maximum connected distance between two robots is r_w units.

3. At each time step a robot moves a units ($a \ll r_w$).

4. A robot can move forward, turn $90°$ left, $90°$ right or $180°$ back.

5. Given a robot i in position x, y, if another robot j is in the shaded area shown in Figure 3, then robots i and j are 'connected'.

[1] In this paper, we adopt the convention that the safety property defines the set of valid actions. In some literatures, the safety property is defined as the set of invalid actions. Both approaches can be used to achieve the same effect.

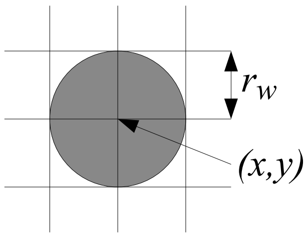

Figure 3: Area of Connectivity

6. We simplify the FSM of figure 1 by omitting the avoidance state.

7. We assume a value of $\alpha = 1$ so that the loss of *any* connection triggers the coherence state.

Given the above assumptions, the behaviour of each robot can be described as follows. Each robot can be in one of two motion states: forward or coherence. The connectivity of each robot can also be in one of two states: connected or not connected. The combination of the motion states and the connectivity states give us four possible actions:

In the forward state, when connected $\rightarrow$ move forward

In the forward state, but not connected $\rightarrow$ turn $180°$ and change the motion state to 'coherent'

In the coherent state, but not connected $\rightarrow$ move forward

In the coherent state, when connected $\rightarrow$ perform a random turn (i.e. either left or right) and change the motion state to 'forward'.

Now, given a swarm of robots with the above behaviours, there may potentially be the following (desirable) emergent behaviours:

- Property 1: It is repeatedly the case that for each robot, we can find another robot so that they are connected.
- Property 2: Eventually it will always be the case that every robot is connected to at least k robots, where k is a pre-defined constant.

4.2 Specification of individual robots

Before defining the specification of individual robots, we need some auxiliary definitions to make the specification more readable.

The following local variables and global constants are used in the subsequent specifications:

x_i, y_i: position of $robot_i$.

θ_i: bearing of $robot_i$. This can be N, S, E or W.

$motion_i$: flag indicating whether $robot_i$ is in the forward or coherence state.

M: total number of robots in the swarm.

r_w: connectivity range.

a: distance of one move.

π_i: $robot_i$ will transition from the current state to the next state if π_i is $true$.

4.2.1 Auxiliary definitions

Set of Robots

$robotSet$ denotes the set of robots in the collection:

$$robotSet := \{1, ..., M\} \tag{1}$$

Detection of Connectivity

Two robots i and j are within the connection range if the Euclidian distance between their x, y coordinates is less than the connection distance, thus:

$$inRange(i, j) := (\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} < r_w) \tag{2}$$

Robot i is connected to some other robots in the collection if there exists another robot within its connection range, thus:

$$connected(i) := \exists j \in robotSet \setminus \{i\}.inRange(i, j) \tag{3}$$

Movements

First, we specify the 'move forward' action. If the current direction is north, in the next time step, x_i remains unchanged, y_i is incremented by a units. If the current direction is south, in the next time step, x_i remains unchanged, y_i is decremented by a units, etc. Thus:

$$\begin{aligned} &moveF(i) := \\ &\quad (\theta_i = N \wedge (\bigcirc x_i = x_i) \wedge (\bigcirc y_i = y_i + a)) \vee \\ &\quad (\theta_i = S \wedge (\bigcirc x_i = x_i) \wedge (\bigcirc y_i = y_i - a)) \vee \\ &\quad (\theta_i = W \wedge (\bigcirc x_i = x_i - a) \wedge (\bigcirc y_i = y_i)) \vee \\ &\quad (\theta_i = E \wedge (\bigcirc x_i = x_i + a) \wedge (\bigcirc y_i = y_i)) \end{aligned} \tag{4}$$

We now specify the 'move north' action. If in the next time step robot i moves in the direction north by a units, the value of the x-coordinate of robot i in the next state is the same as the value of the x-coordinate now; and the value of the y-coordinate of robot i in the next state is the same as the value of the y-coordinate now plus the distance a. Thus:

$$moveN(i) := (\bigcirc x_i = x_i) \wedge (\bigcirc y_i = y_i + a) \tag{5}$$

And for directions south, east and west respectively:

$$moveS(i) := (\bigcirc x_i = x_i) \wedge (\bigcirc y_i = y_i - a) \tag{6}$$

$$moveE(i) := (\bigcirc x_i = x_i + a) \wedge (\bigcirc y_i = y_i) \tag{7}$$

$$moveW(i) := (\bigcirc x_i = x_i - a) \wedge (\bigcirc y_i = y_i) \tag{8}$$

The $turn90Move()$ action specifies that in the next step robot i turns $90°$ randomly and moves a units in the new direction:

$$\begin{aligned}
&turn90Move(i) := \\
&\quad (\theta_i = S) \wedge (\bigcirc\theta_i = W) \wedge moveW(i) \vee \\
&\quad (\theta_i = S) \wedge (\bigcirc\theta_i = E) \wedge moveE(i) \vee \\
&\quad (\theta_i = W) \wedge (\bigcirc\theta_i = N) \wedge moveN(i) \vee \\
&\quad (\theta_i = W) \wedge (\bigcirc\theta_i = S) \wedge moveS(i) \vee \\
&\quad (\theta_i = E) \wedge (\bigcirc\theta_i = N) \wedge moveN(i) \vee \\
&\quad (\theta_i = E) \wedge (\bigcirc\theta_i = S) \wedge moveS(i) \vee \\
&\quad (\theta_i = N) \wedge (\bigcirc\theta_i = E) \wedge moveE(i) \vee \\
&\quad (\theta_i = N) \wedge (\bigcirc\theta_i = W) \wedge moveW(i)
\end{aligned} \tag{9}$$

The $turn180Move()$ action specifies that in the next step robot i turns $180°$ and moves a units in the new direction:

$$\begin{aligned}
&turn180Move(i) := \\
&\quad (\theta_i = S) \wedge (\bigcirc\theta_i = N) \wedge moveN(i) \vee \\
&\quad (\theta_i = W) \wedge (\bigcirc\theta_i = E) \wedge moveE(i) \vee \\
&\quad (\theta_i = N) \wedge (\bigcirc\theta_i = S) \wedge moveS(i) \vee \\
&\quad (\theta_i = E) \wedge (\bigcirc\theta_i = W) \wedge moveW(i)
\end{aligned} \tag{10}$$

The four possible states and movements

We can now specify the four possible actions described in section 4.1, as follows. $forwardConnected()$ specifies the next action when the robot is in the forward motion state and is connected:

$$\begin{aligned}
&forwardConnected(i) := \\
&\quad (motion_i = forward) \wedge connected(i) \wedge \\
&\quad (\bigcirc motion_i = forward) \wedge moveF(i)
\end{aligned} \tag{11}$$

When the robot is in the forward motion state but not connected:

$$\begin{aligned}
&forwardNotConnected(i) := \\
&\quad (motion_i = forward) \wedge \neg connected(i) \wedge \\
&\quad (\bigcirc motion_i = coherent) \wedge turn180Move(i)
\end{aligned} \tag{12}$$

When the robot is in the coherence state and not connected:

$$\begin{aligned}
&coherentNotConnected(i) := \\
&\quad (motion_i = coherent) \wedge \neg connected(i) \wedge \\
&\quad (\bigcirc motion_i = coherent) \wedge moveF(i)
\end{aligned} \tag{13}$$

And when the robot is in the coherence state and connected:

$$\begin{aligned}
&coherentConnected(i) := \\
&\quad (motion_i = coherent) \wedge connected(i) \wedge \\
&\quad (\bigcirc motion_i = forward) \wedge turn90Move(i)
\end{aligned} \tag{14}$$

Idle situation

Finally, we characterise the 'idle' situation.

$$idle(i) := (\bigcirc x_i = x_i) \wedge (\bigcirc y_i = y_i) \wedge (\bigcirc\theta_i = \theta_i) \tag{15}$$

The above formula specifies that in the next step robot i does not make any change.

4.2.2 *Specification of Robot i*

Let $Robot_i$ denote the specification of robot i:

$$Robot_i := \Box(Safety_i \wedge Liveness_i) \tag{16}$$

The above formula states that the behaviour of $Robot_i$ will always satisfy its safety properties and liveness properties.

4.2.3 *Specification of safety for Robot i*

The safety properties specify the valid actions. To allow for concurrent composition of all the robots, the specification of each robot consists of two parts: the component part and the environment part. The component part defines what actions robot i is allowed to do. The environment part defines what other robots are allowed to do from the point of view of robot i. In our specification, concurrency is modelled through interleaving. Therefore the specification needs to ensure that at any time, only one robot is taking an action. We use the proposition π_i to label robot i's actions. Therefore π_i is true if robot i is taking an action.

$$\begin{aligned}
&Safety_i := \\
&\quad \pi_i \wedge CompAction_i \wedge \forall j \in robotSet \setminus \{i\}.idle(j) \\
&\quad \vee \\
&\quad \neg\pi_i \wedge EnvAction_i \wedge idle(i)
\end{aligned} \tag{17}$$

The $Safety_i$ formula above specifies what robot i is allowed to do itself (i.e. a component action) and what the environment (i.e. the other robots) are allowed to do. If robot i is executing an action, all other robots must be idle. If the environment is executing an action, robot i must be idle.

The $CompAction_i$ formula specifies four allowed actions:

$$\begin{aligned}
CompAction_i \;:=\; & forwardConnected(i) \vee \\
& forwardNotConnected(i) \vee \\
& coherentNotConnected(i) \vee \\
& coherentConnected(i)
\end{aligned} \tag{18}$$

$EnvAction_i$ specifies that one of the other robots is allowed to make a move in one of the four directions:

$$\begin{aligned}
EnvAction_i \;:=\; & \exists j \in robotSet \setminus \{i\}. \\
& (moveN(j) \vee moveS(j) \vee \\
& moveE(j) \vee moveW(j))
\end{aligned} \tag{19}$$

4.2.4 Specification of liveness for Robot i

We can now specify liveness for robot i. The $liveness_i$ formula specifies that if, in the current step, robot i has performed a detection action and it is in the forward state, and it is connected to another robot, robot i *will* move forward; if robot i is in the forward state and it is not connected to another robot, it *will* do a $180°$ turn and change the motion state to be 'coherent', and so on:

$$
\begin{aligned}
& Liveness_i := \\
& \quad (\pi_i \wedge (motion_i = forward) \wedge connected(i) \Rightarrow \\
& \quad\; moveF(i)) \\
& \quad \wedge \\
& \quad (\pi_i \wedge (motion_i = forward) \wedge \neg connected(i) \Rightarrow \\
& \quad\; (turn180Move(i) \wedge \bigcirc motion_i = coherent)) \\
& \quad \wedge \\
& \quad (\pi_i \wedge (motion_i = coherent) \wedge \neg connected(i) \Rightarrow \\
& \quad\; moveF(i)) \\
& \quad \wedge \\
& \quad (\pi_i \wedge (motion_i = coherent) \wedge connected(i) \Rightarrow \\
& \quad\; (turn90Move(i) \wedge \bigcirc motion_i = forward))
\end{aligned} \tag{20}
$$

4.3 Specification of the overall swarm

The swarm of robots consists of all robots executing concurrently. Therefore our specification for the whole collection is defined as the logical 'and' of all the individual robots. As we use interleaving to model concurrency, we also need to ensure that only one robot is taking an action at a time. This mutual exclusion is specified by the exclusive 'or' ($\oplus$) condition. Thus,

$$
\begin{aligned}
& Swarm := \\
& \quad Robot_1 \wedge Robot_2 \wedge ... \wedge Robot_N \wedge \\
& \quad \Box(\pi_1 \oplus \pi_2 \oplus ...\pi_N)
\end{aligned} \tag{21}
$$

4.4 Specification of emergent behaviours

In this subsection, we demonstrate how we can use the same notation to describe possible emergent behaviours.

Property 1: It is infinitely often the case that, for each robot, we can find another robot so that they are connected.

$$property1 := \Box\Diamond(\forall i \in robotSet.\ connected(i)) \tag{22}$$

Property 2: Eventually it will always be the case that every robot is connected to at least k distinct robots, where k is pre-defined.

$$
\begin{aligned}
& property2 := \\
& \quad \Diamond\Box(\forall i \in robotSet. \\
& \qquad (\exists j_1 \in robotSet\{i\}.\ inRange(i, j_1) \wedge \\
& \qquad \exists j_2 \in robotSet\{i\}.\ inRange(i, j_2) \wedge \\
& \qquad ... \\
& \qquad \exists j_k \in robotSet\{i\}.\ inRange(i, j_k) \wedge \\
& \qquad distinct(j_1, j_2, ..., j_k)))
\end{aligned} \tag{23}
$$

and $distinct()$ is defined as,

$$distinct(i_1, i_2, ..., i_k) := |\{i_1, i_2, ..., i_k\}| = k \tag{24}$$

Thus $property1$ specifies that each robot has just 1 connected neighbour; $property2$ is stronger, and specifies that each robot in the swarm has k connected neighbours. However, $property2$ does still admit the possibility that our swarm of M robots might split into a number of connected subswarms each with $k+1$ robots. Given that this paper is reporting a work-in-progress we offer the specifications of emergence here not as complete and sufficient, but as illustrative of the approach we are advocating.

Since the disposition and connectivity of our swarm experiences a time evolution we also need to assume that $property1$ and $property2$ are true at the initial moment, in other words that our robots are initially tightly swarmed and fully connected. Over time the swarm will tend to disperse but the purpose of the alpha algorithm is to maintain swarm connectivity. Thus we seek to prove that $property1$ and $property2$ remain true.

4.5 Potential for proving emergent swarm properties

Our goal for this stage in our formal approach is to prove (or disprove) that the swarm of robots satisfies the emergent behaviours, i.e.

$$Swarm \Rightarrow property1 \tag{25}$$

$$Swarm \Rightarrow property2 \tag{26}$$

Currently we are experimenting with mapping specifications for the swarm and the emergent behaviours into a monodic first-order temporal logic so that a monodic first order temporal prover (Degtyarev et al., 2004, Konev et al., 2005) can be used to prove if the swarm robotic system satisfies the anticipated emergent behaviours. Our initial study has indicated that by rewriting the problem specification and the emergent behaviours into a monodic first order temporal specification, we are indeed able to use the temporal prover (Hustadt et al., 2004) to carry out such proofs. Although mapping to the monodic temporal logic produces a large number of clauses and the time taken for the proof is relatively long, this is a first step towards a solution for designing dependable swarm robotic systems that will guarantee certain emergent behaviours.

It should be noted that the finite domains used in the specification, e.g. finite numbers of robots, finite grid, finite actions, etc., all help to reduce the complexity of the description required. However, they greatly increase the size. Our aim is to revise the specification in order to take more first-order cases into consideration, thus allowing techniques such as in (Fisher et al., 2005b) to be utilised.

5. Conclusions and Further Work

This paper has proposed the use of a formal method, which would normally be used to specify and prove properties of a software system, in swarm robotics. We have argued that a linear time temporal logic formalism can be applied to the specification of swarm robotic systems, because of its ability to model concurrent processes and - we maintain - robots in a swarm can usefully be modelled as concurrent processes in a highly parallel system. We have applied this temporal logic schema to the specification of a wireless connected swarm, starting with the specification of individual robots and building up to the overall swarm. This work is at a very early stage and we have made a large number of simplifying assumptions. Our example specification thus falls well short of fully specifying even the simple alpha algorithm of our case study. We are, however, confident that there is potential merit in the approach proposed in this paper. We believe that such an approach could be an important step toward a disciplined design methodology for swarm robotics.

Further work will include:

1. development of the case study to reduce the number of simplifying assumptions and hence improve the fidelity of the formal specification of our wireless connected swarm;

2. further work to understand the scope and implications of the use of the temporal prover to prove the emergent properties of the swarm, and

3. work to extend and generalise this approach to other types of robotic swarm and hence determine whether the approach has merit as a generic tool in swarm engineering.

Acknowledgements

The authors gratefully acknowledge the constructive and insightful comments of the referees. This work was partially supported by the EPSRC under research grant GR/R45376.

References

Bonabeau, E., Dorigo, M., and Théraulaz, G. (1999). *Swarm Intelligence: from natural to artificial systems*. Oxford University Press.

Degtyarev, A., Fisher, M., and Konev, B. (Accepted April 2004). Monodic temporal resolution. *ACM Transactions on Computational Logic*, To appear.

Emerson, E. A. (1990). Temporal and Modal Logic. In van Leeuwen, J., (Ed.), *Handbook of Theoretical Computer Science*, pages 996–1072. Elsevier.

Fisher, M. (2005). Temporal Development Methods for Agent-Based Systems. *Journal of Autonomous Agents and Multi-Agent Systems*, 10(1):41–66.

Fisher, M., Gabbay, D., and Vila, L., (Eds.) (2005a). *Handbook of Temporal Reasoning in Artificial Intelligence*, volume 1 of *Advances in Artificial Intelligence*. Elsevier Publishers, North Holland.

Fisher, M., Konev, B., and Lisitsa, A. (2005b). Practical infinite-state verification with temporal reasoning. In *Workshop on Verification of Infinite State Systems and Security (VISSAS)*.

Hustadt, U., Konev, B., Riazanov, A., and Voronkov, A. (2004). **TeMP**: A temporal monodic prover. In Basin, D. A. and Rusinowitch, M., (Eds.), *Proceedings of the Second International Joint Conference on Automated Reasoning (IJCAR 2004)*, volume 3097 of *LNAI*, pages 326–330. Springer.

Konev, B., Degtyarev, A., Dixon, C., Fisher, M., and Hustadt, U. (2005). Mechanising first-order temporal resolution. *Information and Computation*, 199(1-2):55–86.

Lerman, K., Martinoli, A., and Galstyan, A. (2005). A review of probabilistic macroscopic models for swarm robotic systems. In Şahin, E. and Spears, W., (Eds.), *Swarm Robotics Workshop: State-of-the-art Survey*, number 3342, pages 143–152, Berlin Heidelberg. Springer-Verlag.

Manna, Z. and Pnueli, A. (1992). *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag.

Martinoli, A., Easton, K., and Agassounon, W. (2004). Modeling swarm robotic rystems: A case study in collaborative distributed manipulation. In *Int. Journal of Robotics*, volume **23**(4), pages 415–436.

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

Nembrini, J., Winfield, A., and Melhuish, C. (2002). Minimalist coherent swarming of wireless connected autonomous mobile robots. In *Proc. Simulation of Artificial Behaviour (SAB'02)*. Edinburgh.

Rouff, C., Hinchey, M., Truszkowski, T., and Rash, J. (2004). Formal methods for autonomic and swarm-based systems. In *1st International Symposium on Leveraging Applications of Formal Methods (ISoLA 2004)*. Cyprus.

Rouff, C., Truszkowski, W., Rash, J., and Hinchey, M. (2003). Formal approaches to intelligent swarms. In *IEEE/NASA Software Engineering Workshop, 2003*, pages 51–57. IEEE press.

Winfield, A., Harper, C., and Nembrini, J. (2005). Towards dependable swarms and a new discipline of swarm engineering. In Şahin, E. and Spears, W., (Eds.), *Swarm Robotics Workshop: State-of-the-art Survey*, number 3342, pages 126–142, Berlin Heidelberg. Springer-Verlag.