

On Formal Specification of Emergent Behaviours in Swarm Robotic Systems

Alan FT Winfield¹; Jin Sa¹; Mari-Carmen Fernández-Gago²; Clare Dixon² & Michael Fisher²

¹Intelligent Autonomous Systems Laboratory, University of the West of England, Coldharbour Lane, Bristol BS16 1QY

email:{Alan.Winfield,Jin.Sa}@uwe.ac.uk

²Liverpool Verification Laboratory, Department of Computer Science, University of Liverpool, Liverpool L67 7ZF

email:{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 use of 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.*

Keywords: *swarm robotics, formal specification, temporal logic.*

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), (Şahin, 2005). If future real-world robotic swarms are to exploit emergence and self-organisation to generate desired 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 analytical modelling 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 (individual robot) parameters. Martinoli’s work is concerned with modelling rather than specification, or formal proof.

(Kiriakidis and Gordon-Spears, 2002) modelled a multi-robot system as a Discrete Event System (DES) and went further than modelling, toward formal verification, using automata-theoretic model checking. However, the multi-robot system of Kiriakidis and Gordon-Spears employs a central supervisory controller which assures the behaviour of the overall system by constraining the actions of individual robots. By contrast the work of this paper is concerned with fully distributed robotic swarms in which there is no centralised command and control structure. Thus we are faced with the difficult problems of, firstly, designing and specifying individual robot behaviours such that when the robots of the swarm interact with each other and their environment the desired overall swarm behaviours will emerge and, secondly, verifying those emergent behaviours.

There has been recent work in the area of applying formal methods to specifying and verifying swarm intelligent systems, notably

¹<http://www.ias.uwe.ac.uk>

²<http://www.csc.liv.ac.uk/~liverlab>

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 Synchronous Calculus of Sequential Systems (WSCCS), Unity Logic and X-Machines and concluded that none of those approaches is - on its own - sufficient for the task of formal specification and verification of robotic swarms.

By contrast, in this paper we shall explore the use of a *temporal logic* to formally specify and verify emergent behaviours of a robotic swarm system. The recent theoretical results that have led to the mechanisation of fragments of first-order temporal logic are exciting new results that, for the first time, open up the possibility of applying first order temporal theorem proving in areas such as swarm verification. 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

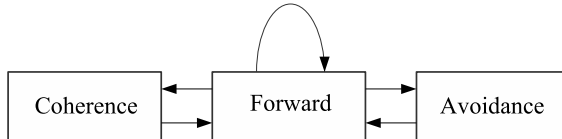


Figure 1: Robot Finite State Machine

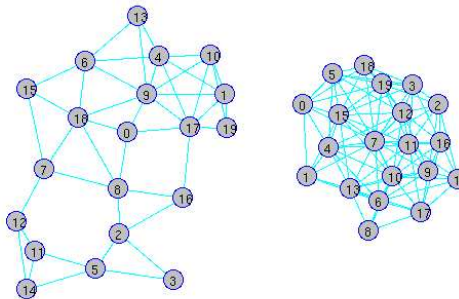


Figure 2: Swarm with $\alpha = 5$ (left) and $\alpha = 10$ (right)

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 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 broadcasts an ‘I am here’ message. The message will of course be received only by those robots that are within wireless range - 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 methodology:

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 specification satisfies any of the emergent behaviours.

Safety³ 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 four 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 designer. 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, in which 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.,

$$\sigma = s_0, s_1, s_2, s_3, \dots$$

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

³In 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 the same effect.

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 ‘ $\square\varphi$ ’, 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:

$$\begin{aligned} \langle \sigma, i \rangle \models \bigcirc A & \text{ iff } \langle \sigma, i+1 \rangle \models A \\ \langle \sigma, i \rangle \models \square A & \text{ iff for all } j \geq i. \langle \sigma, j \rangle \models A \\ \langle \sigma, i \rangle \models \diamond A & \text{ iff exists } j \geq i. \langle \sigma, j \rangle \models A \end{aligned}$$

We also allow standard first-order quantifiers, such as ‘ \exists ’, ‘ \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.

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.

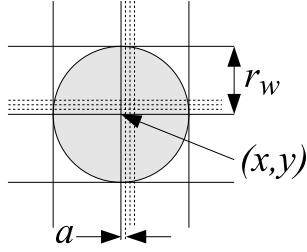


Figure 3: Area of Connectivity, showing movement grid

3. At each time step a robot moves a units ($a \ll r_w$).
4. A robot can move forward, turn 90° left before making a move, turn 90° right before making a move or turn 180° back before making a move.
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’.
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.

These assumptions may appear to be severe but, we argue, do not compromise the thesis of this paper. For example, the avoidance behaviour, while necessary in a real-world swarm, is not essential to the emergent swarm behaviours we seek. Similarly, a swarm with $\alpha = 1$ may have limited practical value, but our contention that temporal logic can be used to specify swarm properties, including emergence, is uncompromised by this assumption.

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 state transitions:

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.

N : 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:

$$robotSet := \{1, \dots, N\} \quad (1)$$

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:

$$inRange(i, j) := (|\mathbf{x} - \mathbf{y}| < r_w) \quad (2)$$

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

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

Movements

First, we 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:

$$moveN(i) := (\bigcirc x_i = x_i) \wedge (\bigcirc y_i = y_i + a) \quad (4)$$

And for directions south, east and west respectively:

$$\text{moveS}(i) := (\bigcirc x_i = x_i) \wedge (\bigcirc y_i = y_i - a) \quad (5)$$

$$\text{moveE}(i) := (\bigcirc x_i = x_i + a) \wedge (\bigcirc y_i = y_i) \quad (6)$$

$$\text{moveW}(i) := (\bigcirc x_i = x_i - a) \wedge (\bigcirc y_i = y_i) \quad (7)$$

We now specify the ‘move forward’ action. If the current direction is north, in the next time step, the robot remains in the same direction and moves north by a units. If the current direction is south, in the next time step, the robot again remains in the same direction and moves south by a units, etc. Thus:

$$\begin{aligned} \text{moveF}(i) := & \\ & (\theta_i = N) \wedge (\bigcirc \theta_i = N) \wedge \text{moveN}(i) \vee \\ & (\theta_i = S) \wedge (\bigcirc \theta_i = S) \wedge \text{moveS}(i) \vee \\ & (\theta_i = W) \wedge (\bigcirc \theta_i = W) \wedge \text{moveW}(i) \vee \\ & (\theta_i = E) \wedge (\bigcirc \theta_i = E) \wedge \text{moveE}(i) \quad (8) \end{aligned}$$

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

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

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

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

State transitions

We can now specify the four possible combinations of motion states and connectivities, described in section 4.1, as follows. $\text{forwardConnected}(i)$ specifies that the robot is in the forward motion state and is connected:

$$\begin{aligned} \text{forwardConnected}(i) := & \\ & (\text{motion}_i = \text{forward}) \wedge \text{connected}(i) \quad (11) \end{aligned}$$

and for the other three state transitions:

$$\begin{aligned} \text{forwardNotConnected}(i) := & \\ & (\text{motion}_i = \text{forward}) \wedge \neg \text{connected}(i) \quad (12) \end{aligned}$$

$$\begin{aligned} \text{coherentNotConnected}(i) := & \\ & (\text{motion}_i = \text{coherent}) \wedge \neg \text{connected}(i) \quad (13) \end{aligned}$$

$$\begin{aligned} \text{coherentConnected}(i) := & \\ & (\text{motion}_i = \text{coherent}) \wedge \text{connected}(i) \quad (14) \end{aligned}$$

Idle situation

Finally, we specify the ‘idle’ situation.

$$\begin{aligned} \text{idle}(i) := & \\ & (\bigcirc x_i = x_i) \wedge \\ & (\bigcirc y_i = y_i) \wedge \\ & (\bigcirc \theta_i = \theta_i) \wedge \\ & (\bigcirc \text{motion}_i = \text{motion}_i) \quad (15) \end{aligned}$$

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

4.3 Specification of Robot i

Let Robot_i denote the specification of robot i :

$$\text{Robot}_i := \square(\text{Safety}_i \wedge \text{Liveness}_i) \quad (16)$$

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

4.3.1 Specification of safety for Robot i

The safety properties specify the valid actions. In our specification, concurrency is modelled through interleaving. So, for each robot, we need to specify two cases: one for the robot taking an action and another for the robot not taking an action. The first case we label the component action and the second the environment action. The component action defines what robot i is allowed to do. The environment action defines what the environment is allowed to do to robot i . Now the environment of robot i consists of all the other robots, but the environment action does not need to specify all possible actions of the other robots, only those that affect robot i . In our particular case study, when the environment is taking an action, there will be no changes to robot i ; robot i will thus be idle.

To distinguish between a component step and a environment step for robot i , we use the proposition π_i to label robot i 's actions. Therefore π_i is true if robot i is taking an action; it is false if it is not taking an action. Thus:

$$\begin{aligned} \text{Safety}_i := & \\ & \pi_i \wedge \text{CompAction}_i \vee \\ & \neg \pi_i \wedge \text{idle}(i) \quad (17) \end{aligned}$$

The $Safety_i$ formula above specifies what robot i is allowed to do when it is taking an action (i.e. a component action) or that it remains idle when it is not taking an action. This idle period leaves the space for the environment to take some actions.

The $CompAction_i$ formula specifies three allowed actions:

$$\begin{aligned} CompAction_i &:= moveF(i) \vee \\ &\quad turn90move(i) \vee \\ &\quad turn180move(i) \end{aligned} \quad (18)$$

4.3.2 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 is active ($\pi_i = true$), and it is in the forward state, and it is connected to another robot, then it *will* move forward and remain in the forward state; 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 ‘coherent’, and so on. Thus:

$$\begin{aligned} Liveness_i &:= \\ &(\pi_i \wedge forwardConnected(i) \Rightarrow \\ &(moveF(i) \wedge \bigcirc motion_i = forward)) \wedge \\ &(\pi_i \wedge forwardNotConnected(i) \Rightarrow \\ &(turn180Move(i) \wedge \bigcirc motion_i = coherent)) \wedge \\ &(\pi_i \wedge coherentNotConnected(i) \Rightarrow \\ &(moveF(i) \wedge \bigcirc motion_i = coherent)) \wedge \\ &(\pi_i \wedge coherentConnected(i) \Rightarrow \\ &(turn90Move(i) \wedge \bigcirc motion_i = forward)) \end{aligned} \quad (19)$$

4.4 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:

$$\begin{aligned} Swarm &:= \\ &Robot_1 \wedge Robot_2 \wedge \dots \wedge Robot_N \wedge \\ &\quad \square(\pi_1 \oplus \pi_2 \oplus \dots \pi_N) \end{aligned} \quad (20)$$

The exclusive or condition ensures that at any moment in time, if π_i is true, only robot i is taking an action, all the other robots are idling. For example, if π_1 is true, π_2 to π_N are false. Therefore according to the safety property specification, the behaviour of robot 1 is specified by $compAction_i$, i.e. one of the three actions: $moveF(i)$, $turn180Move(i)$

and $turn90Move(i)$. The behaviour of all the other robots are specified by their environment action, i.e. idling.

It is important to recognise that the specification of the swarm above specifies the set of all possible orderings of interleaved actions of all robots. The swarm specification must not be taken to imply that the actions of robot 1, robot 2, etc, are interleaved in order 1, 2... N .

4.5 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.

$$property1 := \square \diamond (\forall i \in robotSet. connected(i)) \quad (21)$$

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.

$$\begin{aligned} property2 &:= \\ &\diamond \square (\forall i \in robotSet. \\ &\quad (\exists j_1 \in robotSet \setminus \{i\}. inRange(i, j_1) \wedge \\ &\quad \exists j_2 \in robotSet \setminus \{i\}. inRange(i, j_2) \wedge \\ &\quad \dots \\ &\quad \exists j_k \in robotSet \setminus \{i\}. inRange(i, j_k) \wedge \\ &\quad distinct(j_1, j_2, \dots, j_k))) \end{aligned} \quad (22)$$

where $distinct()$ is defined as,

$$distinct(i_1, i_2, \dots, i_k) := |\{i_1, i_2, \dots, i_k\}| = k \quad (23)$$

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 N robots might split into a number of connected subswarms each with $k + 1$ robots (which, in fact, can happen with the alpha algorithm). However, the purpose of this paper is not to derive a full specification, but to demonstrate the validity 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.

4.6 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.

$$\text{Swarm} \Rightarrow \text{property1} \quad (24)$$

$$\text{Swarm} \Rightarrow \text{property2} \quad (25)$$

Currently we are experimenting with mapping specifications for the swarm and the emergent behaviours into a *monodic* first-order temporal logic (FOTL) so that a monodic⁴ first-order temporal prover 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 TeMP to carry out such proofs. Note that, whereas the scenario described in previous sections for the swarm of robots assumes infinite domains, we have used finite domains for the re-written specification. In particular, we have assumed a finite number of robots involved as well as a finite grid. This is necessary in order to map the specification into the monodic form. 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.

TeMP (Hustadt et al., 2004) is a resolution-based theorem-prover for first-order temporal logic (FOTL). This logic is very expressive and, consequently, problems expressed in FOTL are often difficult (and sometimes impossible) to prove automatically. However, the restriction to *monodic* FOTL (Hodkinson et al., 2000) helps us considerably. Monodic FOTL has many appealing properties, for instance it is often possible to automatically prove monodic FOTL formulae when an appropriate first-order basis is used. In order to build tools that achieve this, the clausal resolution method for propositional temporal logic (Fisher et al., 2001) has been extended to monodic FOTL (Degtyarev et al., 2004, Konev et al., 2005). This has been implemented as TeMP, which currently utilises Vampire (Riazanov and Voronkov, 2001) (the fastest prover for first-order logic) for the base first-order proving activities. Specification using monodic FOTL, and then proof of properties in TeMP, has

⁴A monodic first-order temporal logic is one in which any subformulas beginning with a temporal operator contain at most one free variable.

already been used in a number of areas, including security verification (Gago et al., 2005) and distributed protocols (Fisher et al., 2005b).

It should be noted that in its re-written finite, form, the finite-state descriptions of the robots could also be verified using *model-checking* techniques (Clarke et al., 2000). However, 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. Furthermore, we have mapped the specification to a form that can be proven in an automated prover, and validated its suitability.

This work is developing and we have made a large number of simplifying assumptions. Our example specification thus falls well short of specifying even the simple alpha algorithm of our case study. We have argued, however, that our simplifying assumptions do not compromise the contention of this paper, that temporal logic can be used to specify swarm behaviours, including emergent behaviours. We would further argue that the logic proposed here is perfectly valid for real-world problems. Our discretisation of robot movement, for instance, while necessary to reduce the number of logic clauses to a manageable level does not compromise real-world applicability since we can reduce the movement and time steps to arbitrarily small values to specify smooth motion. Similarly, the time interleaving approach needed to model the swarm does not compromise the validity of the approach, since the specification makes no assumptions about the order of interleaving and, indeed, describes all possible orderings. Note also that our swarm specification is expressed from an external point-of-view in which we (the designers) are allowed to know each robot's position and bearing at any time. However, we seek here a swarm specification not a controller design, and the individual robot controllers do not need to have this global knowledge. Thus the principles of

swarm intelligence remain uncompromised by our approach.

We are confident that there is 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 and, if we combine formal specification using temporal logic with a method for provable design of the individual robots of the swarm (Harper and Winfield, 2005), then we have a methodology for verifiable design at all levels of the robotic swarm.

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 discussion and feedback from Kryisia Broda of the Department of Computing, Imperial College London, and the constructive and insightful comments of the reviewers. 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.
- Clarke, E., Grumberg, O., and Peled, D. (2000). *Model Checking*. MIT Press.
- Degtyarev, A., Fisher, M., and Konev, B. (Accepted April 2004). Monodic temporal resolution. *ACM Transactions on Computational Logic*, to appear.
- Emerson, E. (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., Dixon, C., and Peim, M. (2001). Clausal Temporal Resolution. *ACM Transactions on Computational Logic*, 2(1):12–56.
- Fisher, M., Gabbay, D., and Vila, L., (Eds.) (2005a). *Handbook of Temporal Reasoning in Artificial Intelligence*, volume 1 of *Advances in Artificial Intelligence*. Elsevier.
- 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 (VIS-SAS)*.
- Gago, M.-C. F., Hustadt, U., Dixon, C., Fisher, M., and Konev, B. (Accepted 2005). First-order temporal verification in practice. *Journal of Automated Reasoning*, to appear.
- Harper, C. and Winfield, A. (Accepted October 2005). A methodology for provably stable behaviour-based intelligent control. *Robotics and Autonomous Systems*, to appear.
- Hodkinson, I., Wolter, F., and Zakharyashev, M. (2000). Decidable fragments of first-order temporal logics. *Annals of Pure and Applied Logic*, 106:85–134.
- 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.
- Kiriakidis, K. and Gordon-Spears, D. (2002). Formal modeling and supervisory control of reconfigurable robot teams. In *Formal Approaches to Agent-Based Systems*, pages 92–102. LNCS 2699, Springer-Verlag.
- 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. 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 systems: A case study in collaborative distributed manipulation. In *Int. Journal of Robotics Research*, 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.
- Riazanov, A. and Voronkov, A. (2001). Vampire 1.1 (system description). In *Proc. IJCAR 2001*, pages 376–380. LNAI 2083, Springer-Verlag.
- 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.
- Şahin, E. (2005). Swarm robotics: From sources of inspiration to domains of application. In Şahin, E. and Spears, W., (Eds.), *Swarm Robotics Workshop: State-of-the-art Survey*, number 3342, pages 10–20. Springer-Verlag.
- 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. Springer-Verlag.