Formal development of self-organising systems

Graeme Smith[1]* and J.W. Sanders[2]**

[1] School of Information Technology and Electrical Engineering
The University of Queensland, Australia
[2] International Institute for Software Technology
United Nations University, Macao SAR China

Abstract. Attempts to engineer autonomic multi-agent systems, particularly those having large numbers of agents, have revealed the need for design structures and formalisms to support the construction of properties that emerge at the system level. Such emergence, like self-$*$ behaviour, relies typically on intricate inter-agent interactions. This paper shows how the top-down incremental approach of Formal Methods can be used satisfactorily in that situation, by considering a case study in which agents adapt and autonomously achieve a given configuration.

1 Introduction

Autonomic multi-agent systems achieve their important self-$*$ properties (of configuration, management and repair, for instance) by exploiting carefully configured agent interactions. But engineering such interactions, so that the correct self-$*$ properties emerge at the system level with a reasonable degree of trustworthiness, is not easy. New formalisms and design patterns appear necessary.

A first conceptual step has been taken by Zambonelli and Omicini [17] who, by distinguishing three levels of abstraction (the *macro* (or system) level, the *meso* (or agent-interaction) level, and the *micro* (or individual agent) level), have set the scene for a *reductionist* approach: one that reveals the intricacies level by level, in the manner familiar from Physical Science. This is one of the approaches taken by Formal Methods in the less ambitious case of conventional information systems. Is it also applicable to autonomic multi-agent systems?

Opinion is divided. In this paper we summarise the various arguments (in Section 2) then, in the body of the paper, consider a case study (Sections 3 to 6) chosen to exhibit typical self-$*$ behaviour. Focusing on the property of self-reconfiguration, we choose an example from robotics in which mobile 'atoms' must assemble autonomously to achieve a certain global shape.

As an example of autonomous multi-agent systems the case study is particularly stringent: agents must conform to a given spatial target. The role of

* The first author acknowledges support from Australian Research Council (ARC) Discovery Grant DP0987452, *Combining Time Bands and Teleo-Reactive Programs for Advanced Dependable Real-Time Systems.*

** The second author acknowledges support from the Macao Science and Technology Development Fund under the PEARL project, grant number 041/2007/A3.

agents is played by the atoms, the micro-level behaviour is that of the atoms individually, the meso-level behaviour—of primary importance in this paper—consists of inter-atom interactions, and the emergent macro behaviour is the formation of the given shape. The techniques involved, however, seem relevant in achieving the weaker properties of 'subgrouping' or 'clustering' required by many meso-level interactions. Techniques for achieving this can be expected to play the dynamic equivalent, for multi-agent systems, of the *design patterns* so useful in object-oriented systems. The algorithm we treat is a modification from a family of algorithms given by Støy [12]. In the current context its adaptability (or 'self-repair' in Støy's terminology) is important.

A feature of the incremental top-down approach taken here is that an operation considered to be atomic at one level may be split at the next level down. That idea is far from new. Our use of it has been partially motivated by the 'time bands' approach [7, 2] in which a split is seen as resulting from temporal considerations. An important by-product of our approach is the clarification of conditions that ensure correctness of the algorithm (or 'convergence' in robotics terminology). The way in which they emerge is particularly satisfying.

2 Related work

It has been suggested in the literature that the top-down approach of Formal Methods is not applicable to multi-agent systems. One argument is that the models produced are necessarily incomplete and therefore do not accurately reflect the real world[3]. This argument, based on Gödel's incompleteness theorem, applies equally to models for simulation as it does to mathematical models, and equally to single-component interactive systems as it does to multi-agent systems. Hence, it is not a new problem, and the usefulness of (possibly incomplete) models in these other areas indicates that they should also be useful for multi-agent systems. This point of view is supported by recent work on mathematical modelling of systems which has shown that such approaches are useful despite the necessary use of simplifying assumptions [16, 6, 3].

Another argument is that proofs over such models cannot be totally automated, nor can their subsequent development towards implementations be totally automated [5]. However, this does not preclude use of formal methods. It is well known that while automation can be used to support formal methods, high-level proof strategies and design decisions need to be developed by the engineer. Indeed, this is essential if we are fully to understand the design and proofs and hence have any real confidence in them.

Finally, a reluctance to adopt formal methods has arisen from the similarities between multi-agent systems and complex systems. Formal approaches have found less utility in the field of Complex Systems due to the emphasis on *observation* of existing systems (both natural and man-made, *e.g.*, the Internet)

[3] The argument is elaborated in [15] and referenced by several papers in the literature on multi-agent systems, *e.g.*, [17].

and the *prediction* of their global behaviour. When we model an existing system, unknown discontinuities in behaviour may not be modelled, and hence proof techniques may not be successful in uncovering emergent behaviour. Furthermore, it is claimed that some complex systems exhibit *strong emergence* [1] (*e.g.*, the mind) and therefore, by definition, proofs cannot be constructed of how their behaviour arises.

With multi-agent systems, however, the emphasis changes from observation of existing systems to *engineering* of new systems and from prediction to *assurance* of their global behaviour. When we engineer new systems, we are not trying to prove the existence of emergent behaviours. Rather we start with the emergent behaviour we require (which may include the avoidance of undesirable behaviours), and develop a design which ensures it. To provide assurance that a design is sufficient, the emergent behaviour must be a consequence of the component interactions within the design. Hence, in this context we are interested only in systems which exhibit *weak emergence* [1]. This approach has been suggested and discussed in the context of the InterLink[4] programme [8, 9].

There is little work on a general approach to formal assurance of multi-agent systems. The most mature work in the area focuses on particular systems [16, 6, 3]. Other papers suggest more general strategies, but do not apply them to significant case studies [10, 18]. This paper is a first step in bridging this gap.

3 Case study

In this section, we give an informal description of our case study: a collection of components referred to as 'atoms' which can configure to form 3-dimensional shapes through local interaction. The description is based on a family of solutions to this problem studied by Støy (and Nagpal) [13, 14, 12, 11].

3.1 Atoms

Atoms in the system are capable of local communications with immediate neighbours, of movement over or around their neighbours, and of storing information including:

– a representation of the shape to be formed (referred to as the *target*);
– their position within the target, should they reach it;
– their status which is either *stationary* or *moving*;
– their distance from an atom in the target requiring a neighbour;
– their distance from a 'seed' atom used to start the shape-forming process.

Initially, only the seed atom stores the target and its position within it. It uses this information to decide at which of its neighbouring positions it needs atoms, and broadcasts accordingly. Atoms which respond to its broadcast for neighbours

[4] InterLink is an international coordination activity of ERCIM (European Research Consortium on Informatics and Mathematics).

are given the target and their position in it by the seed atom when they take up target positions. They then broadcast for the neighbours they require; and so on.

An atom has *stationary* status unless it is moving, or has decided to move, in response to a broadcast. Only stationary atoms store the distance from an atom requiring neighbours and from the seed atom. When an atom starts moving it will delete any such values it has stored. How these values are set up and utilised is described below.

3.2 Recruiting neighbours

An atom moves towards the source of a request for neighbours by following a *recruitment gradient* established by the broadcaster. This gradient consists of an integer value at each atom, denoting the distance of the atom from the source of a broadcast.

An atom which broadcasts its need for neighbours sets its value of the recruitment gradient to 0 and sends this value to each of its neighbours. When an atom receives such a value, if it is either moving or decides to start moving in response to the receiving the value, it does nothing further with the value. Otherwise, if it does not have a value for the recruitment gradient, or has a value greater than that received, it sets its value to one more than the value received. It then sends this new value to all of its neighbours. If, on the other hand, it already has a recruitment gradient value less than that received, it ignores the received value.

This results in stationary atoms storing a value representing the shortest distance to an atom whose broadcast for neighbours has reached it. If it is able to follow a gradient to successively smaller values, a moving atom can reach the source of a broadcast and join the target shape; it then becomes stationary and remains so.

3.3 Connectedness

Initially, the atoms form a connected mass. This guarantees that a communication from one atom can reach any other atom in the system via a sequence of communications between neighbours. To ensure that the mass stays connected, each stationary atom stores an integer value denoting its distance from the seed atom. An atom is allowed to move only when it is certain that none of its neighbours rely on it to remain connected. That is, an atom can move only if its neighbours' distances from the seed do not exceed its own.

This *connection gradient* is set up by the seed atom before it begins broadcasting for neighbours. At this point in time, all atoms are stationary. The seed atom sets its own value to 0, and broadcasts this value to all of its neighbours. On receiving such a value, if an atom has no connection gradient value or a value greater than the received value, it sets its value to one more than the received value, and broadcasts this new value to its neighbours. If it has a value less than the received value, it ignores the received value.

3.4 Ensuring progress

To ensure progress towards the final shape, we require that all broadcasts for neighbours reach available atoms. Connectedness alone does not ensure this as the recruitment gradient decreases only to the nearest source of a broadcast. We require also that, once an atom has all the neighbours it needs, its broadcast for neighbours is 'cancelled' (thus allowing other broadcasts to propagate). One way of doing that is to have all recruitment gradient values 'dissipate' after a certain time limit, and have broadcasters reestablish their individual gradients (after the same time limit) only if they still require neighbours.

Additionally, for progress we require that it is always possible for at least one atom not positioned in the target to reach the source of a broadcast. This is possible if the target has spaces through which atoms can move. Støy presents two algorithms to deal with that situation. In the first, a particular *scaffold* structure is used to tile (an approximation to) the target shape [12, 11]. In the second, atoms of *varying resolution* are used, lowest first, to approximate the target [13]. Our treatment assumes instead that the target has sufficient space around it to allow the uninhibited movement of atoms.

Finally, we require that there are enough atoms to form the desired target shape. We assume that the number of atoms in the system are known by the user who chooses a target with exactly that number of atoms.

4 Macro-level specification

In this section, we specify the case study at the macro-level. We choose the granularity of our initial level (or time band) so that the entire process of forming the desired target shape is accomplished in one event. The outcome of this event is that all atoms lie within the target. We then present two lower levels (*i.e.*, time bands with finer granularity) which progressively bring our specification to a form more suited for exploring a meso-level strategy.

4.1 Level 1

As a preliminary to our specification, we define a type *Position* denoting (absolute) positions in space. This type is not constrained in any way. We could be more precise and define position to consist, for example, of the set of all triples of real numbers. For our purposes, however, the more abstract definition suffices.

The (macro-level) system includes a *target* which is a finite set of positions (*i.e.*, of type $\mathbb{F}\,Position$) and a second finite set of positions, *atoms*, denoting the current position of the atoms. At this level we abstract the internal details of atoms, as well as any meso-level concepts such as gradients. For simplicity, we constrain the number of atoms to be equal to the number of positions in the target configuration (but return to consider an alternative in Section 6). We also have a variable, *placed*, to represent the set of atoms fixed in a location in the target.

*System*1

$$target, atoms, placed : \mathbb{F}\, Position$$

$$\#target = \#atoms$$
$$placed \subseteq (target \cap atoms)$$

We then define the operation *Event*1 to capture our desired goal: placement of an atom at each target position. (ΔS is a standard Z abbreviation to introduce the pre-state and post-state variables of an operation on state S, the latter being primed; thus the pre-state of the target is *target* and the post-state is *target*′.)

*Event*1

$$\Delta System1$$

$$placed' = target'$$
$$target' = target$$

At this level a single event occurs: *Event*1. In one step the atoms conform to the target (since the constraints imply that in fact $atoms' = target$).

4.2 Level 2

As a first step in our formal development, we introduce a lower level of granularity in which with each event some (but not necessarily all) of the atoms move into a position of the target. The final configuration is hence reached after several events. Once atoms are part of the target, they remain so.

We replace *Event*1 in the previous specification with *Event*2. This operation specifies that the set of atoms fixed in the target before the operation (*placed*) forms a proper subset of those after the operation.

*Event*2

$$\Delta System1$$

$$placed \subset placed'$$
$$target' = target$$

One way to think of *Event*2, following the time-bands approach, is that the single time step of Level 1 is partitioned into finer time steps and *Event*2 concerns the atoms that are placed within those finer steps. According to that view, for this to be a valid development step, we need to show that a finite repetition of *Event*2's eventually achieves the post-state of *Event*1: for some $k : \mathbb{N}$,

$$Event1 \; = \; (Event2)^k \, . \tag{1}$$

That follows since either (i) the result has already been achieved and the iteration is vacuous, or (ii) since *Event*2 strictly increases *placed* (and *target* is finite) within a finite number of iterations, *atoms* equals *target*. The formalisation of this proof (like later proofs in the paper) is straightforward and omitted.

4.3 Level 3

At the next level of granularity, each event corresponds to a single atom joining the target configuration.

We replace *Event*2 with *Event*3. This operation specifies that a position in *target* but not in *placed* (*i.e.*, in $\mathit{target} \setminus \mathit{placed}$) is added to *placed*.

$$\begin{array}{|l}
\hline
\mathit{Event}3 \\
\Delta \mathit{System}1 \\
\hline
\exists\, p : \mathit{target} \setminus \mathit{placed} \bullet \mathit{placed}' = \mathit{placed} \cup \{p\} \\
\mathit{target}' = \mathit{target} \\
\hline
\end{array}$$

We need to show that a sequence of occurrences of *Event*3 corresponds to a single *Event*2 occurrence. This follows since *Event*3 is an instantiation of *Event*2 where the increase in the size of *placed* is 1. More precisely,

$$\mathit{Event}1 \;=\; (\mathit{Event}3)^k \tag{2}$$

where $k : \mathbb{N}$ is the difference in size between *target* and the (so far unspecified) initial value of *placed*.

It is possible in Z to adopt (from process algebra) an *interleaving* semantics where events that may occur in any order may be implemented concurrently. Such a semantics would be necessary for this specification to allow implementations where finite numbers of *Event*3's occur in parallel.

5 Meso-level specification

In this section, we extend the macro-level specification of Section 4 to include meso-level design. The final macro-level specification has focused our attention on a single atom moving to a position in the target. We introduce a lower level of granularity in which this event corresponds to two events: an atom within the target broadcasting its need for a neighbour, and an atom not within the target responding to the broadcast by moving towards the atom in the target.

At a lower level, we then introduce Støy's notions of gradients to realise the broadcast and moving events. It is at this level that the formal approach brings to the fore the necessary constraints on the system. To prove the development step is valid, it is necessary that the atoms form a connected structure, and that the target configuration allows atoms to move about within it.

5.1 Level 4

So far we have abstracted the mechanism by which atoms are placed.

At this level, we introduce local communication between atoms. The definition of 'local' depends on whether we are talking about programmable matter, where only neighbours in contact can communicate, or swarm systems, with wireless communication. For generality, we define a relation $\mathit{in\_range} : \mathit{Position} \leftrightarrow$

Position such that an atom at position p can (directly) communicate with an atom at q if, and only if, $(p, q) \in \mathit{in\_range}$. From now on, 'neighbour' means 'neighbour with respect to *in_range*'.

We introduce a new state variable $\mathit{waiting} : \mathbb{F}\, \mathit{Position}$ to consist of those placed atoms having communicated that they require one or more neighbours.

$$\begin{array}{|l}
\hline \mathit{System}4 \\
\mathit{System}1 \\
\mathit{waiting} : \mathbb{F}\, \mathit{Position} \\
\hline
\mathit{waiting} \subseteq \mathit{placed} \\
\hline
\end{array}$$

For the algorithm to work, we need at least one placed atom initially. We assume there is exactly one such 'seed' atom. Initially, this atom has not broadcast its need for neighbours, and hence *waiting* is empty.

$$\begin{array}{|l}
\hline \mathit{Init}4 \\
\mathit{System}4 \\
\hline
\#\mathit{placed} = 1 \\
\mathit{waiting} = \varnothing \\
\hline
\end{array}$$

We replace *Event*3 with *Move*4 which strengthens the former operation by ensuring that the atoms move next to a 'waiting' atom in the target.

$$\begin{array}{|l}
\hline \mathit{Move}4 \\
\Delta \mathit{System}4 \\
\hline
\exists\, p : \mathit{waiting};\ q : \mathit{target} \setminus \mathit{placed} \bullet \\
\qquad \mathit{in\_range}(p, q) \land \mathit{placed}' = \mathit{placed} \cup \{q\} \\
\mathit{target}' = \mathit{target} \land \mathit{waiting}' = \mathit{waiting} \\
\hline
\end{array}$$

That operation is enabled, however, only if an atom in the set *waiting* has a target neighbour not in *placed*. That fact is recorded in Z as the precondition of the operation *Move*4 (fat parentheses denote relational image):

$$\begin{array}{|l}
\hline \mathit{pre}\ \mathit{Move}4 \\
\mathit{System}4 \\
\hline
\mathit{in\_range}\, (\!|\, \mathit{waiting}\, |\!) \cap (\mathit{target} \setminus \mathit{placed})\ \neq\ \varnothing \\
\hline
\end{array}$$

Hence, in order to ensure that *Move*4 becomes enabled, we need an additional operation, *Broadcast*4, to add positions to *waiting*.

$$\begin{array}{|l}
\hline \mathit{Broadcast}4 \\
\Delta \mathit{System}4 \\
\hline
\exists\, p : \mathit{placed};\ q : \mathit{target} \setminus \mathit{placed} \bullet \\
\qquad \mathit{in\_range}(p, q) \land \mathit{waiting}' = \mathit{waiting} \cup \{p\} \\
\mathit{target}' = \mathit{target} \land \mathit{atoms}' = \mathit{atoms} \land \mathit{placed}' = \mathit{placed} \\
\hline
\end{array}$$

That is in turn enabled whenever a placed atom has a target neighbour not in *placed*:

$$in\_range \,(\!|\, placed \,|\!)\cap (target \setminus placed) \;\neq\; \varnothing\,. \qquad (3)$$

*Broadcast*4 does not change the state variables of the previous specification, *atoms*, *target* and *placed*. Hence, it is equivalent to a 'skip' operation at that level. Therefore *Event*3 is achieved by sufficiently many *Broadcast*4 events followed by a consequently enabled *Move*4. Writing '$\fatsemi$' for sequential composition and * for its transitive closure, $Event3 = (Broadcast4)^* \fatsemi (Move4)$. So from (2)

$$Event1 \;=\; ((Broadcast4)^* \fatsemi (Move4))^k \qquad (4)$$

where $k = \#target - 1$ is the difference in size between *target* and the initial value of *placed*.

5.2 Level 5

At this level, we introduce recruitment gradients explicitly and model the mechanisms for propagating and following them. We add a state variable *grad* to denote the strength of the recruitment gradient at each atom to which it has propagated. Thus *grad* is a partial function (denoted by the symbol $\pfun$) on *atoms*.

As the variable *waiting* is no longer required at this level, we extend the state *System*1 rather than *System*4.

$$\begin{array}{|l}
\multicolumn{1}{l}{System5} \\ \hline
System1 \\
grad : Position \pfun \mathbb{N} \\ \hline
\mathrm{dom}\, grad \subseteq atoms \\ \hline
\end{array}$$

Initially, no gradients have been propagated and so the gradient is undefined at each atom.

$$\begin{array}{|l}
\multicolumn{1}{l}{Init5} \\ \hline
System5 \\ \hline
\#placed = 1 \\
grad = \varnothing \\ \hline
\end{array}$$

If a placed atom at position p requires one or more neighbours, it creates a new gradient having strength 0 at p. The precondition for being able to do so is simply (3). That strength replaces any previous gradient strength at p (described below using Z's functional overriding operator $\oplus$).

$$\begin{array}{|l}
\multicolumn{1}{l}{CreateGrad5} \\ \hline
\Delta System5 \\ \hline
\exists\, p : placed;\; q : target \setminus placed \bullet \\
\qquad in\_range(p, q) \wedge grad' = grad \oplus \{p \mapsto 0\} \\
target' = target \wedge atoms' = atoms \wedge placed' = placed \\ \hline
\end{array}$$

An atom at position p can propagate a gradient to a neighbouring atom at q provided q has no gradient (*i.e.*, is not in the domain of *grad*) or has a gradient strength less than the gradient strength at p plus one.

$$
\begin{array}{|l}
\hline
\mathit{PropGrad5} \\
\Delta \mathit{System5} \\
\hline
\exists\, p : \mathit{placed};\ q : \mathit{atoms} \bullet \\
\qquad \mathit{in\_range}(p, q) \land \\
\qquad (q \notin \operatorname{dom} \mathit{grad} \lor \mathit{grad}(q) < \mathit{grad}(p) + 1) \land \\
\qquad \mathit{grad}' = \mathit{grad} \oplus \{q \mapsto \mathit{grad}(p) + 1\} \\
\mathit{target}' = \mathit{target} \land \mathit{atoms}' = \mathit{atoms} \land \mathit{placed}' = \mathit{placed} \\
\hline
\end{array}
$$

An atom at $p \notin \mathit{placed}$ can follow a decreasing gradient towards its origin (with the aim of becoming a neighbour of the atom which created the gradient). In the following description the atom at p is initially a neighbour of that at q, whose position afterwards, p', makes it a neighbour of the atom at r which is a neighbour of that at q but having lower gradient strength.

$$
\begin{array}{|l}
\hline
\mathit{FollowGrad5} \\
\Delta \mathit{System5} \\
\hline
\exists\, p : \mathit{atoms} \setminus \mathit{placed};\ q, r : \operatorname{dom} \mathit{grad} \bullet \\
\qquad \mathit{in\_range}(p, q) \land \mathit{in\_range}(q, r) \land \mathit{grad}(r) < \mathit{grad}(q) \land \\
\exists\, p' : \mathit{Position} \setminus \mathit{atoms} \bullet \mathit{in\_range}(p', r) \land \mathit{atoms}' = (\mathit{atoms} \setminus \{p\}) \cup \{p'\} \\
\mathit{target}' = \mathit{target} \land \mathit{placed}' = \mathit{placed} \land \mathit{grad}' = \mathit{grad} \\
\hline
\end{array}
$$

Since this operation does not change *placed*, another operation is required to fix atoms in the target. An atom not in *placed* can join as a neighbour of an atom with gradient strength zero. If the latter atom does not require further neighbours, it is removed from the gradient ($\lhd$ is Z's notation for domain subtraction).

$$
\begin{array}{|l}
\hline
\mathit{JoinTarget5} \\
\Delta \mathit{System5} \\
\hline
\exists\, p : \mathit{atoms} \setminus \mathit{placed};\ q : \operatorname{dom} \mathit{grad} \bullet \\
\qquad \mathit{in\_range}(p, q) \land \mathit{grad}(q) = 0 \land \\
\qquad (\exists\, p' : \mathit{target} \setminus \mathit{placed} \bullet \\
\qquad\qquad \mathit{in\_range}(q, p') \land \\
\qquad\qquad \mathit{atoms}' = (\mathit{atoms} \setminus \{p\}) \cup \{p'\} \land \\
\qquad\qquad \mathit{placed}' = \mathit{placed} \cup \{p'\}) \land \\
\qquad (\nexists r : \mathit{target} \setminus \mathit{placed}' \bullet \mathit{in\_range}(q, r)) \Rightarrow \mathit{grad}' = \{q\} \lhd \mathit{grad} \land \\
\qquad (\exists\, r : \mathit{target} \setminus \mathit{placed}' \bullet \mathit{in\_range}(q, r)) \Rightarrow \mathit{grad}' = \mathit{grad} \\
\mathit{target}' = \mathit{target} \\
\hline
\end{array}
$$

The invariant $\#\mathit{target} = \#\mathit{atoms}$ and the fact that *target* does not change ensure that either $p' = p$ or p' is not already occupied.

When the gradient to a particular atom is no longer required (since the atom has all of its neighbours), the gradient must be removed from the system. This is necessary to enable other gradients to be followed; yet operation *JoinTarget*5 removes only a source. Another operation is required to allow non-zero gradients to dissipate over time whenever they are not reinforced by *PropGrad*5 events.

$$\begin{array}{|l}
\text{\_\_ } \mathit{Dissipate5} \text{ \_\_\_\_\_\_\_\_} \\
\Delta \mathit{System5} \\
\hline
\exists\, p : \operatorname{dom} \mathit{grad} \bullet \mathit{grad}(p) \neq 0 \wedge \mathit{grad}' = \{p\} \lhd \mathit{grad} \\
\mathit{target}' = \mathit{target} \wedge \mathit{atoms}' = \mathit{atoms} \wedge \mathit{placed}' = \mathit{placed} \\
\hline
\end{array}$$

Verification of this development step requires a data refinement [4] relating the variable *grad* to the variable *waiting* of the previous specification: a position p in which $\mathit{grad}(p) = 0$ is related to a position in *waiting* which still requires neighbours.

$$\begin{array}{|l}
\text{\_\_ } \mathit{Rel} \text{ \_\_\_\_\_\_\_\_} \\
\mathit{System4} \\
\mathit{System5} \\
\hline
\forall\, p : \mathit{Position} \bullet \\
\qquad p \in \operatorname{dom} \mathit{grad} \wedge \mathit{grad}(p) = 0 \\
\qquad \Leftrightarrow \\
\qquad p \in \mathit{waiting} \wedge (\exists\, q : \mathit{target} \setminus \mathit{placed} \bullet \mathit{in\_range}(p, q)) \\
\hline
\end{array}$$

Subject to *Rel*, *CreateGrad*5 performs the same task as *Broadcast*4:

$$\mathit{Broadcast4} \;=\; (\mathit{Rel} \wedge \mathit{CreateGrad5} \wedge \mathit{Rel}') \setminus (\mathit{grad}, \mathit{grad}') \tag{5}$$

where Rel' is *Rel* with all declared variables primed, $S \wedge T$ is the schema formed from schemas S and T by merging their declarations and conjoining their predicates, and $S \setminus (x, y)$ is the schema S with the variables x and y hidden.

Similarly, *JoinTarget*5 performs the same task as *Move*4. To ensure that *JoinTarget*5 is enabled whenever *Move*4 is, however, we need to show that the other operations, *PropGrad*5, *FollowGrad*5 and *Dissipate*5 (which correspond to 'skip' operations at the previous level) lead to *JoinTarget*5 being enabled; the intention is to replace the right-hand side of (4) by

$$(\mathit{Rel} \wedge (\mathit{Init5} \mathbin{⨾} \mathit{CreateGrad5} \mathbin{⨾} \mathcal{O})^k \wedge \mathit{Rel}') \setminus (\mathit{grad}, \mathit{grad}') \tag{6}$$

where $\mathcal{O}$ is a parallel composition of a single *JoinTarget*5 with a finite number of iterations of *CreateGrad*5, *PropGrad*5, *FollowGrad*5 and *Dissipate*5 operations. This will be true if (i) gradients propagate to atoms that are not yet in the target, and (ii) atoms can follow these gradients to their origin.

Attempts to prove (i) will immediately run into a problem. Propagation relies on atoms being neighbours and there is no guarantee that initially the set *atoms* is connected. Furthermore, there is no guarantee that a connected collection of atoms remains connected after an atom changes position. To ensure it, we could add an invariant to *System*5 stating that between any two atoms there lies a sequence of atoms, consecutive pairs of which are neighbours. Expressed in terms of transitive closure, the extra property of *System*5 is:

$$\mathit{connected} \quad \equiv \quad (\,(\mathit{in\_range})^* = \mathit{atoms} \times \mathit{atoms}\,)\,. \tag{7}$$

While this suffices to prove (i) (taking into account that gradients will eventually dissipate and allow other gradients to propagate when (ii) holds) it does not reflect an implementation in terms of local interactions. Although we could ensure the invariant holds initially, we need a local mechanism to ensure that it continues to hold when atoms move. Støy's solution to this problem is to add another, *connection*, gradient. An atom is allowed to move only when the connection gradient strengths of its neighbours do not exceed its own. If that were not the case, neighbours having higher gradient strengths might rely on the atom for connection to the rest.

We could specify such a connection gradient and its propagation just as we have specified the recruitment gradient. It would be created by the initial 'seed' atom, but would not dissipate. The formal details are routine and are omitted.

Attempts to prove (ii) will also lead to a problem. It is not guaranteed that an atom can follow a recruitment gradient through the existing target where the atoms are unable to move. Støy solves this problem in [12, 11] by ensuring that a target is in the form of a 'scaffold' with spaces around the atoms already in fixed positions.

We could capture this formally by including an initial condition on *target*. Since *target* is unchanged by any operation, this condition is invariant:

$$\mathit{lacunose} \quad \equiv \quad \forall\, p : \mathit{target} \bullet \exists\, q : \mathit{Position} \setminus \mathit{target} \bullet \mathit{in\_range}(p, q)\,. \tag{8}$$

Such a condition ensures that the algorithm succeeds in filling the target (in Støy's notation, is *convergent*).

That completes the development at the meso level. The development of the micro level is far more detailed since atoms must contain sufficient state to keep track of their position, the target, their neighbours, the gradients, avoiding collisions, and so on. However it does not add to our understanding of atomic interactions, the focus of the current paper, and so is not considered here.

6 Adaptivity

In this section we evaluate the suitability of our approach for expressing adaptability, a feature that is even more important in the multi-agent system context than in Støy's original setting of reconfigurable robots.

An adaptive algorithm must respond to dynamic changes in its environment. We consider here the removal of target positions: at run time some target positions are blocked by the environment. If such a position already contains an atom, the atom is removed from the system. Atoms can sense when a neighbouring position is blocked and are unable to move into that position.

The new system, $System5a$, differs from $System5$ by satisfying the weaker invariant $\# target \leq \# atoms$ (since an atom is not always removed when a target position is blocked).

$$\begin{array}{|l}
\text{— } System5a \text{ ————————————} \\
target, atoms, placed : \mathbb{F}\, Position \\
grad : Position \mathrel{+\!\!\!\!\rightarrow} \mathbb{N} \\
\hline
\# target \leq \# atoms \\
placed \subseteq (target \cap atoms) \\
\operatorname{dom} grad \subseteq atoms \\
connected \\
lacunose \\
\hline
\end{array}$$

Nonetheless, the condition $placed' = target'$ of $Event1$ correctly describes what the system should now achieve, although its other condition, $target' = target$, must be weakened to an inclusion. The result fills the target with atoms but may leave some, connected to the others, outside the target.

$$\begin{array}{|l}
\text{— } Event1a \text{ ————————————} \\
\Delta System1 \\
\hline
placed' = target' \\
target' \subseteq target \\
\hline
\end{array}$$

Adaptation is modelled by a revision of (6), operating instead on $System5a$ and including an operation $Block$ (in parallel with the other operations of $\mathcal{O}$ but the only one to change the target) which removes a set $blocked$ of target positions and updates the system components accordingly. Explicitly:

$$\begin{array}{|l}
\text{— } Block \text{ ————————————} \\
\Delta System5a \\
blocked : \mathbb{F}\, Position \\
\hline
blocked \subseteq target \\
target' = target \setminus blocked \\
atoms' = atoms \setminus blocked \\
placed' = placed \setminus blocked \\
grad' = blocked \mathbin{\lhd\!\!\!-} grad \\
\hline
\end{array}$$

It is enabled iff $blocked \subseteq target$ and $atoms \setminus blocked$ remains connected (since (7) is invariant). Since it only removes target positions, (8) still holds.

After $Block$ occurs the recruitment gradient may be temporarily disrupted (since some sources may have been removed and other placed atoms may find they no longer need certain neighbours). However in spite of the inconsistency between the representation of the target generated by the seed and that which now pertains, the recruitment gradient regenerates and succeeds in filling the new target, since the physical conditions *connected* and *lacunose* are preserved. The same holds even with multiple invocations of *Block*. Adaptability is captured by the claim that $Event1a$ is achieved by the revised version of (6).

Adaptability to an increased target or decreased atom set is handled similarly. If a situation is too severe for Støy's algorithm (*e.g.*, an increased target violating (8)), the deficiency is revealed; if an extension is possible, $placed'$ emerges.

7 Conclusion

We have presented a case study to support the view that formal methods can be used in a top-down incremental manner to account for behaviour which emerges at the system level of a multi-agent system. One strength of the approach has been the ease with which the physical conditions of connectedness and lacunosity arise as necessary for correctness. Another has been the manner in which the meso-level design structures (the recruitment and connectivity gradients) have been revealed incrementally to facilitate inter-agent interactions and hence to achieve the required emergent behaviour.

Our case study indicates that emergence needs to be dealt with primarily at the boundary between the macro and meso levels. Conformance between specifications at these levels amounts to showing that a particular interaction paradigm gives rise to a desired global behaviour.

A further, crucial, benefit has been the ease with which the approach is able to incorporate adaptability. Whilst it makes no sense to discuss adaptability at our highest level of abstraction with its single, atomic operation, it becomes pertinent at the meso level where the environment may supervene with its own atomic operation. For the system to be entirely adaptable that operation should have no precondition; in our case, it must maintain connectedness and lacunosity.

We believe that adaptability needs to be dealt with primarily at the boundary between the micro and meso levels. Adaptability is modelled here as response to an environmentally-induced operation changing state. So, of interest is how meso-level strategies for obtaining global behaviour under such disturbances arise from changes in agent behaviour in response to those conditions.

To assist engineers in the development of multi-agent systems using our approach, we would ideally have a catalogue of "patterns", similar to those used in object-oriented design, to facilitate the modelling process at each level. Furthermore, we would have a catalogue of "strategies" for proving conformance between specifications at different levels of abstraction. Identifying and formalising such patterns and strategies is an area of future work.

References

1. M.A. Bedau. Weak emergence. In J. Tomberlain, editor, *Philosophical Perspectives: Mind, Causation, and World*, volume 11, pages 375–399. Blackwell Publishers, 1997.

2. A. Burns, I.J. Hayes, G. Baxter, and C.J. Fidge. Modelling temporal behaviour in complex socio-technical systems. *Technical Report YCS 390, University of York*, 2005.
3. F. Cucker and S. Smale. On the mathematics of emergence. *Japanese Journal of Mathematics*, 2:197–227, 2007.
4. J. Derrick and E. Boiten. *Refinement in Z and Object-Z: Foundations and Advanced Applications*. Springer-Verlag, 2001.
5. B. Edmonds and J. Bryson. The insufficiency of formal design methods — the necessity of an experimental approach for the understanding and control of complex MAS. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, pages 938–945. IEEE Computer Society, 2004.
6. H. Hamann and H. Wörn. A framework of space-time continuous models for algorithm design in swarm robotics. *Swarm Intelligence*, 2:209–239, 2008.
7. T.A. Henzinger, S. Qadeer, and S.K. Rajamani. Assume-guarantee refinement between different time scales. In *CAV 99: Computer Aided Verification*, volume 1633 of *Lecture Notes in Computer Science*, pages 208–221. Springer-Verlag, 1999.
8. Hu Jun, Zhiming Liu, G.M. Reed, and J.W. Sanders. Ensemble engineering and emergence. In M. Wirsing, J.-P. Banâtre, M. Hölzl, and A. Rauschmayer, editors, *Challenges for Software-Intensive Systems and New Computing Paradigms*, volume 5380 of *Lecture Notes in Computer Science*, pages 162–178. Springer-Verlag, 2008.
9. J.W. Sanders and G. Smith. Formal ensemble engineering. In M. Wirsing, J.-P. Banâtre, M. Hölzl, and A. Rauschmayer, editors, *Challenges for Software-Intensive Systems and New Computing Paradigms*, volume 5380 of *Lecture Notes in Computer Science*, pages 132–138. Springer-Verlag, 2008.
10. S. Stepney, F. Polack, and H. Turner. Engineering emergence. In *IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'06)*, pages 89–97. IEEE Computer Society, 2006.
11. K. Støy. Controlling self-configuration using cellular automata and gradients. In *8th International Conference on Intelligent Autonomous Systems (IAS-8)*, 2006.
12. K. Støy. Using cellular automata and gradients to control self-reconguration. *Robotics and Autonomous Systems*, 54:135–141, 2006.
13. K. Støy and R. Nagpal. Self-reconguration using directed growth. In *7th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, pages 149–160. ACM Press, 2004.
14. K. Støy and R. Nagpal. Self-repair through scale independent self-reconguration. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2062–2067. IEEE Press, 2004.
15. P. Wegner. Why interaction is more powerful than algorithms. *Communications of the ACM*, 40(5):80–91, 1997.
16. A. Winfield, W. Liu, J. Nembrini, and A. Martinoli. Modelling a wireless connected swarm of mobile robots. *Swarm Intelligence*, 2:241–266, 2008.
17. F. Zambonelli and A. Omicini. Challenges and research directions in agent-oriented software engineering. *Autonomous Agents and Multi-Agent Systems*, 9(3):253–283, 2004.
18. H. Zhu. Formal reasoning about emergent behaviour in MAS. In *International Conference on Software Engineering and Knowledge Engineering (SEKE'05)*. Knowledge Systems Institute, 2005.