



The United Nations
University

UNU-IIST

International Institute for
Software Technology

Policies and emergent behaviour

Hu Jun and J. W. Sanders

March 2009

UNU-IIST and UNU-IIST Reports

UNU-IIST (United Nations University International Institute for Software Technology) is a Research and Training Centre of the United Nations University (UNU). It is based in Macao, and was founded in 1991. It started operations in July 1992. UNU-IIST is jointly funded by the government of Macao and the governments of the People's Republic of China and Portugal through a contribution to the UNU Endowment Fund. As well as providing two-thirds of the endowment fund, the Macao authorities also supply UNU-IIST with its office premises and furniture and subsidise fellow accommodation.

The mission of UNU-IIST is to assist developing countries in the application and development of software technology.

UNU-IIST contributes through its programmatic activities:

1. Advanced development projects, in which software techniques supported by tools are applied,
2. Research projects, in which new techniques for software development are investigated,
3. Curriculum development projects, in which courses of software technology for universities in developing countries are developed,
4. University development projects, which complement the curriculum development projects by aiming to strengthen all aspects of computer science teaching in universities in developing countries,
5. Schools and Courses, which typically teach advanced software development techniques,
6. Events, in which conferences and workshops are organised or supported by UNU-IIST, and
7. Dissemination, in which UNU-IIST regularly distributes to developing countries information on international progress of software technology.

Fellows, who are young scientists and engineers from developing countries, are invited to actively participate in all these projects. By doing the projects they are trained.

At present, the technical focus of UNU-IIST is on formal methods for software development. UNU-IIST is an internationally recognised center in the area of formal methods. However, no software technique is universally applicable. We are prepared to choose complementary techniques for our projects, if necessary.

UNU-IIST produces a report series. Reports are either Research **[R]**, Technical **[T]**, Compendia **[C]** or Administrative **[A]**. They are records of UNU-IIST activities and research and development achievements. Many of the reports are also published in conference proceedings and journals.

Please write to UNU-IIST at P.O. Box 3058, Macao or visit UNU-IIST's home page: <http://www.iist.unu.edu>, if you would like to know more about UNU-IIST and its report series.

G. M. Reed, Director



The United Nations
University

UNU-IIST

International Institute for
Software Technology

P.O. Box 3058

Macau

Policies and emergent behaviour

Hu Jun and J. W. Sanders

Abstract

A major current challenge in the area of complex multi-agent systems is the conceptual approach to, and practical treatment of, emergent behaviour. This paper indicates how Formal Methods may be applied to engineer multi-agent adaptive systems exhibiting emergence, by consideration of an ‘emergence predicate’ in the specification and the incremental derivation of interacting components that achieve that predicate. To perform that task, ‘policies’ are introduced as a novel technique for constraining agents in a hierarchical manner so that the multi-agent implementation as a whole exhibits the specified emergence. The ideas are demonstrated on models of the foraging behaviour of an ant colony, and used to analyse various designs appearing in the literature.

Hu Jun received an M.Sc. in Computer Application from Kunming University of Science and Technology, Kunming, China, and a Ph.D. in Computer Science and Technology from Zhejiang University, Hangzhou, China. In 2008 he was a postdoctoral fellow at UNU-IIST working on multi-agent systems; he is currently Associate Professor of Computer Science and Technology at Hunan University, Hunan, China. His main research interest focuses on Artificial Intelligence, multi-agent systems and software engineering. Very recently, it has also included areas such as policy-based management and trustworthy software and other emergent phenomena.

Jeff Sanders is Principal Research Fellow at UNU-IIST. His interests lie largely in Formal Methods.

Contents

1	Introduction	1
2	Related work	4
2.1	Design by science or engineering?	4
2.2	Parametrising agents	5
2.3	Capturing self-* properties	6
2.4	In denial	7
2.5	Two communities	8
3	A case study	8
3.1	Ant behaviour	9
3.2	Formalisation	9
3.3	Naive emergence	10
3.4	Trail dynamics	14
4	Policies and design patterns	16
4.1	Policies	18
4.2	Ants again	18
4.3	Gossip	19
4.4	Middle agents	20
4.5	Identifier-based methods	20
4.6	Fields and gradients	21
4.7	None of the above	22
5	Conclusion	23

1 Introduction

The ‘emergence’ of behaviour at the macro level, inexplicable in terms of behaviours at the micro level, seems long to have been more of interest to Philosophers than to Scientists. That is, until recently when the complex behaviours achievable by computer have forced scientists to confront it. These days the emergent patterns of simple fractal equations are admired in school, and the emergent services of the web, that complex ‘ensemble’ that has developed ungoverned, benefit the home. The Philosophy of Science has traditionally been concerned with the rich examples of emergence in the natural world: the flocking of birds, foraging of ants, phenomenon of language and, above all, consciousness and self-awareness of the human mind.

Emergence has been formalised with varying degrees of rigour and in various notations. But all approaches derive from the idea that system behaviour is emergent if it is not directly derivable from the behaviours of the individual system components viewed unilaterally. After all, emergent behaviour arises from the interactions between components, and those are not expressible in terms of the components in isolation.

Straddling both nature and information systems, the *paradox of emergence* is the same:

How can behaviour of the whole which is by definition not predicted on the basis of the constituents alone, be accounted for by the ‘reductionist method’?

Recall that method: abstraction is used to create a model of some complex phenomenon which is then revealed incrementally by the successive restoration of detail. In the end, no detail of interest is avoided. But in that top-down approach intermediate models (including the initial, most abstract, one —the specification) may prove just as useful as the ultimate description—the implementation. The importance of the paradox of emergence for Information Engineering is that the reductionist method corresponds with the top-down approach of a substantial branch of Formal Methods.

In Philosophical terms the question is: ‘how can emergence be consistent with reductionism?’ Applied to the human mind, that has of course polarised Philosophers of Science since the beginning of the subject. In Informatics terms the question becomes: is Formal Methods powerful enough for the engineering of information systems that exhibit emergence? Whilst reductionism in nature supports analysis (*i.e.* of existing systems), in Information Engineering it facilitates synthesis (of new ones). So the rigorous engineering of complex information systems requires firstly a reconciliation of emergence with Formal Information Engineering (*i.e.* of the paradox of emergence), and secondly a formalism for engineering systems that exhibit specified emergent behaviour.

A formal definition and resolution of the paradox of emergence has been given by Hu *et al.* [24] using levels of abstraction. In summary, system behaviours are captured by a predicate defining just those combinations of observables allowed in a behaviour. The *level of abstraction*

of a description comprises the observables (*i.e.* the free variables of the predicate). An agent's unilateral behaviour thus contains only the observables of that agent (typically reflecting a dynamic/open environment). But emergent behaviours, involving interactions between agents, are described by a predicate, called here an *emergence predicate*, that cannot be 'separated' into a conjunction of predicates each involving just observables of individual agents.¹ Resolution of the paradox is simply the observation that the global (non separable) predicate describing emergence is achieved (formally, is implied) by the behaviour of a design involving interactions between agents (formally, predicates involving observables of groups of agents). However the system is implemented by agents whose abstract unilateral behaviours match those described originally, but which now are guaranteed to have interactions achieving emergence due to group-wise behaviours at a lower level.

For example, in the case of flocking birds the specification combines the behaviours of the individual birds in isolation, together with a predicate describing flocking ([24] Section 3.2): the position and velocity of each bird converges to a value that results in a flock. But that emergence predicate gives no indication, as is usual in a specification, of how it is to be achieved. In fact it can be achieved by a distributed design in which each bird updates its own position and velocity to the averages of those of its neighbours [12]. Whilst the emergence predicate involves all birds, and so is 'global', that design (called here the bird's 'policy' for achieving emergence) is distributed because at any time each bird interacts with only its immediate neighbours.

That resolution of the paradox of emergence suggests an approach to the engineering of emergence which is the subject of the present paper. The first step consists, as usual in the top-down approach, of the specification including an emergence predicate. There follows a sequence of refinement steps differing from the usual only in that a particular type of design is being targeted: one which accounts for the emergence predicate by the groupwise interaction of agents. Of that there is considerable experience in distributed systems. The case of adaptive multi-agent systems is distinguished not by the *style* of that incremental development, but the nature of agents and the way in which they interact dynamically in response to their open environment. Before elaborating that topic (in the next section) it is helpful to have notation for the three levels of abstraction: the specification, interacting-design and unilateral-implementation levels.

In the Interlink programme [54], complex adaptive multi-agent systems exhibiting emergence have been called *ensembles*. There a presentation by Smith [45] has viewed the engineering of ensembles in terms, introduced by Zambonelli and Omicini's survey [58], of levels: the macro, micro and meso levels. The macro level corresponds to specification —including the emergence predicate; the micro level corresponds to the combination of all components viewed unilaterally; and the meso level corresponds to a design in which interactions between components is made explicit. But [45] left open the question of how the emergent behaviour might be engineered, although an AMAS example has been provided [46] of a top-down incremental development (as far as the meso level) of an adaptive dynamic clustering algorithm for mobile agents. Here we address that question by considering in greater generality and detail relevant meso-level designs, and the heuristic of 'policies' for implementing those designs at the micro, agent, level.

¹Levels of abstraction are merely conceptual, chosen to facilitate the process of incremental development. It is too restrictive to employ levels motivated entirely by naive 'granularity', as demonstrated by Ryan [43].

The emergence predicate must be ‘discharged’ to groups of agents interacting (at the meso level) in ways that are not described (unilaterally) at the micro level. The construction of systems exhibiting emergence then becomes the construction of meso designs able to achieve emergence conditions. As usual in Information Engineering, a methodology includes iteration over several levels, reinforced by feedback as it progresses. But the most useful contribution would be a formalism able to be exploited by a range of Formal Methods and that is what we propose: a notion of ‘policy’ is evaluated as a means of bridging the meso-micro levels, but free of commitment to any particular formalism.

It is to be expected of ensembles that their meso-level designs typically impose only approximate behaviour on their agents. The uncertainty in behaviour arises for two reasons. The first is that, because of the huge number of components, each is addressed only ‘statistically’, and that the corresponding requirements therefore hold only at a certain confidence level. The second is that agent interactions fostered at the meso level may take time to achieve a required pattern, in which case some kind of temporal or spatial limit constrains the agents. In both cases the result can be thought of as imposing *normative* behaviour on agents at the micro level. In other words, they can be thought of as policies guiding agent behaviour.

When the agents are human and the system a community, a policy might be viewed as a legal or ethical principle. Indeed they are the primary means by which unilateral agent behaviours are constrained in a community. The purpose of a policy, then, is simply to guide agent behaviour so that interactions at the meso level achieve the required emergent behaviour at the macro level. But here we shall consider agents that are far from sentient. In common with their use in other branches of Information Engineering (like mobile network technology and distributed-database security) policies constrain the behaviours of autonomous agents by taking into account dynamic interactions in response to an open environment. As stated, we refrain here from introducing specialised notation to express such policies (although we have studied one such language), using for convenience Object-Z when precision requires it. The intention is that policies be considered a formalism for use with a range of Formal Methods.

This paper is organised as follows. The next section surveys related work on the engineering of multi-agent systems; of particular interest is the claim by some authors that no resolution of the emergence paradox based on Formal Methods—and in particular the approach detailed in this paper—is possible! Section 3 contains a case study whose purpose is to demonstrate how standard Formal Methods suffice to cover the use of statistics and differential equations in engineering emergence. Section 4 elaborates the policy-based approach and shows how a selection of approaches of other authors is illuminated by its use. Section 5 concludes. In a subject beset by acronyms we continue the use of MAS for multi-agent system and AMAS for adaptive MAS (both in singular and plural).

2 Related work

The application of Formal Methods to the development of AMAS which exhibit emergent behaviour has a peculiar status in the literature. There are claims that it is impossible and there are treatments to indicate how it can be achieved! In this section we review that contradictory situation.

2.1 Design by science or engineering?

Work applying Formal Methods to the construction of MAS (and AMAS in particular) is less developed than that applying it to distributed systems in general. One reason is no doubt the substantial and imposing body of literature² for distributed systems which presumably must inform methods for MAS. Indeed there, local interactions are (routinely) used to establish system behaviour that would be considered ‘emergent’ in an MAS, although that term is not current in the theory of distributed systems. Another reason seems to be a wise reluctance to propose theories before a comprehensive range of examples has been constructed (a sample of which is surveyed in Section 4). Thus, as might be expected, engineering has preceded science.

Pertinently Fromm [18] discusses two fundamental approaches to the design of systems exhibiting emergent behaviour: (a) experimentation with, and observation of, examples; and (b) analysis using theoretical methods. Seeking a single uniform approach he writes

There is no formal approach or methodology which would be a solution to the problems of engineering emergence . . . Of course the scientific method is well-known and has been taught and applied for centuries. Yet it has not been applied directly to engineering problems. [18], Section 3.1.

In the past decade or so various ‘methodologies’ and ‘formalisms’ have been proposed. Here it is convenient to use ‘methodologies’ as synonymous with Formal Methods, like those mentioned in Footnote 2, to mean notations with semantics, a definition of refinement and preferably including sound laws; and ‘formalisms’ for less comprehensive notations that suggest a setting or approach that is typically to be used in combination with a methodology. For a fairly representative sample of methodologies from 2000, see the collection [47]; for formalisms, see Fromm [18]. A current ‘roadmap’ for research, featuring initiatives that span science and engineering, is given by Chen *et al.* [9].

²Methodologies for distributed systems include: process algebras like CSP, CCS, ACP and π -calculus; shared-variable concurrency like action systems, IOAutomata, ObjectZ and Event-B; Unity; BSP; TLA; Petri nets; abstract machines; concurrent transition systems and event structures, to name just a few. Naturally, particular (A)MAS provide rich territory for model checking. But our concern here is with the more comprehensive correctness of an implementation or design against its specification which typically contains conjuncts that are not readily model checked.

Moving from distributed systems to AMAS, several difficulties are encountered. The first is the potentially huge number of agents (that in the Theory of Distributed Algorithms tend to be of uniform kind) but with variations between them. Here application of techniques like inheritance, aggregation and instantiation from Object Orientation have been found useful, as have Formal Methods like Z that are particularly helpful in the systematic description of variations.

Further difficulties arise due to openness of environment. That requires dynamic response by the system which must therefore be self-* (a term that includes properties like adaptability, configuration, organisation and repair). The formalisation of such properties could be said to have begun with mobility (the π -calculus, brane calculus and their derivatives) but evidently much more is required in AMAS.

Early notations for MAS exploited ideas from object orientation, no doubt in the expectation that results would be deployed together with progress made on the more difficult, dynamic, issues of self-*. Progress is reviewed Sections 2.2 and 2.3 respectively. Each notation we consider is to be judged by the success with which it (a) extends the experience acquired in the methodologies for distributed systems, to features specific to AMAS, and (b) facilitates meso-level design and hence the engineering of AMAS.

2.2 Parametrising agents

Throughout the 1990s the Australian Artificial Intelligence Institute, AAIL, developed, studied and applied an object-oriented methodology. It used an ‘internal’ model to describe agents unilaterally and an ‘external’ one for system-wide behaviours. As a result one strength is its description of system dynamics. Of relevance to our approach is the internal model which endows agents with beliefs, desires and intentions comparable to our policies. However there is no particular support for top-down development or meso-level design. For a summary we refer to Kinny and Georgeff [32].

Luck and d’Inverno [25] use the state-based method Z to provide a setting for MAS specification. Its strength is the clarity with which it relates various kinds of entity in a MAS; its weakness is the lack of emphasis on dynamic evolution of the system. It might be concluded that the underlying method, Z, is not well suited to that purpose. However that is not the case, as demonstrated by Sanders and Turilli in [44] where Object-Z is used to specify and implement a dynamically-changing MAS.

Fisher and Wooldridge’s Concurrent METATEM [17] is a methodology. Agents communicate by asynchronous message passing, each specified in a first-order (linear) temporal logic called FML. System-wide properties are specified in a discrete linear temporal logic with unary belief connectives indexed by agent, called TBL. Their view is close to that adopted here: a system is specified by the combination of all unilateral agent behaviours together with macro-level emergent properties, there described in a particular temporal belief logic. However their formalism has been chosen to support model checking of particular properties rather than the top-down

incremental derivation of a design, and so it does not seem to confer any particular advantage to design at the meso-level. However such logical properties, and their extensions using the knowledge calculus [16], fit naturally into our setting of policies.

The GAIA formalism of Wooldridge, Jennings and Kinny [56] also derives much from OO. It is particularly suited to systems thought of as societies; an organisation is seen as comprising a set of agents ‘by role’ having: responsibilities (operations defined by safety and liveness), permissions (rights to resources), activities (unilateral behaviour), and protocols (for inter-role interactions), changing and interacting dynamically. It provides a formalism uncommitted to any particular methodology and has been designed with top-down (and hence, in passing, meso-level) design in mind. Their protocols are analogous to our policies.

Zhu *et al.* [59, 40], propose the Language SLABS and the scenario calculus. SLABS uses ‘Castes’ to describe families of agent by analogy with the way in which Classes are used in object oriented design. Since the scenario calculus provides criteria for establishing invariance and liveness of system properties, its utility is to be compared directly with the various formalisms mentioned in Footnote 2. The typical example, of a sorting network, is readily developed in many of those formalisms and requires almost no meso-level design. Thus the contribution of SLABS remains largely unvalidated in the AMAS setting.

2.3 Capturing self-* properties

Ball and Butler [3] use the methodology of event-B with decomposition to verify a MAS consisting of a bank whose customers query their balance. Their use of middleware in capturing interactions between bank and users in the context of a dynamic environment forms a meso-level design that might be expected to have further application. The example is realistic enough for a sequence of refinement steps to be found convenient. Their experiment is encouraging for the approach defended here, although it stops short of suggesting a general approach in terms of emergence or agents that adapt.

Use of specific notations can discharge the obligation of formality provided they have a semantics (and preferably sound laws to aid reasoning). The familiar combination of CSP and Z has been extended in the notation ForMAAD by Kacem and Kacem [30] and used with the method of successive refinements to describe an agent-oriented system as the parallel composition of agents and an environment. Each agent is described by a combination of CSP (for inter-agent interactions) and temporal Z (for agent-internal updating), and the resulting system is model checked using SPIN from a translation into PROMELA. The formalism is demonstrated on an Air Traffic Control system but with only two planes, so it is not yet clear what meso-level structures will be found convenient for more realistic applications.

Chen *et al.* [8] introduce ‘complex events’ for the description and analysis of emergent behaviours in MAS. The aim is to bridge the gap between descriptions at differing levels of abstraction. For the same reason, Turner *et al.* [52] ‘tag’ sets of lower-level states in the uniform setting

of cellular automata. In that setting there is some prospect of ‘migrating’ the transition rules associated with specified emergent properties of tagged phenomena to the lower level, where they are implemented, via a ‘meso’ level at which the tagged phenomena are realised. In the approach proposed here, that is achieved by the standard techniques of incremental development; they argue that that is not possible (as summarised in the next section). There is nothing that restricts ‘complex events’ and ‘tags’, to AMAS. In particular they might be compared with ‘system refinement’ in Back’s action systems [2], ‘development’ in Unity [7], or ‘event refinement’ in process algebra [1]. There is also a rich body of work that is directly relevant, in view of cellular automata’s use of space and time, from systolic design [34].

A formalism, OPERAS, has been proposed by Stamatopoulou, Kefalas and Gheorghe [48] as a setting in which the dynamics of individual agents is separated from system-wide dynamics involving system reconfiguration. In approach it resembles ours in proposing a formalism that can be used, in principle, with any Formal Method. In [48] it is used together with Eilenberg’s original idea of X-machines [15], with streams to represent reactivity, to model the AMAS ANTS (Autonomous Nano-Technology Swarm). That AMAS reconfigures itself with the creation and annihilation of agents and by adjustment of spatial and communication relationships. Of interest in further developments of OPERAS will be the manner in which system reconfiguration is discharged at the agent level, as it inevitably must be in an implementation. Again, that focuses attention on meso-to-micro design.

2.4 In denial

However some authors take the view that emergence is different in principle from other kinds of system behaviour and therefore its engineering requires a different treatment; in particular it cannot be treated by existing Formal Methods.

For example Stepney, Polack and Turner claim

... the high level abstract language H and the low level concrete language L employ distinct concepts. If these concepts are sufficiently distinct ... classical refinement techniques will not be able to establish the required connection ... [49], Section 4.2.

This is a curious view, since refinement is transitive, so a composition of small refinements is also a refinement (*i.e.* able to ‘establish the required connection’). Thus it is to be expected—and it is indeed the practice when performing refinements—that ‘sufficiently distinct’ levels are bridged by repeated small increments. Furthermore the distributed system considered in justifying that statement—Conway’s *Game of Life*—can be considered, for the purpose of its functional properties, a sequential system (on each iteration each cell is updated according to Conway’s transition rules; indeed that would be the approach taken using several of the formalisms mentioned in Footnote 2). In that case emergent behaviour becomes a consequence of the loop invariant. The claim becomes that refinement techniques (for sequential programs)

are unable to establish the connection between a loop and its invariant. Yet simulation techniques (presumably of the sort referred to above as ‘classical refinement techniques’) are complete for refinement of sequential systems [41].

Similar views are expressed by Polack and Stepney [39] and by Edmonds and Bryson [14]. In the terms of the latter authors, the approach adopted in the present paper is ‘formal specification strategy’, FSS, and most AMAS and ensembles would be termed ‘messy’ systems (to contrast them with simple ‘academic’ systems). Of messy systems they write ‘in such cases the FSS is, at best, in need of supplementary strategies and, at worst, inapplicable’ [14], Section 2. They propose instead a more experimental approach whose unfortunate casualty is ‘that we will have to rid ourselves of the illusion that we can *fully* understand our own code’, [14], Conclusion.

2.5 Two communities

There seems, then, to be a divide between communities: of the relatively sophisticated (but diverse) formal theories for describing and developing distributed systems, and the more naive attempts to provide the same kind of ‘service’ for the more complex AMAS that inevitably (as a result of dynamic response to environmental imposition) exhibit emergence. Our contribution of a formalism can be viewed as harnessing the rigour of the former for use in the latter. In the reverse direction, the Distributed Systems community benefits by extension of its usual domain of operation to self-* properties. Indeed the emergent behaviour exhibited by AMAS is typically described using limits: probability distributions and spatial or temporal limits. For example the flocking of birds involves both spatial and temporal adaptation (to capture eventual stability from perturbation). We study such an example in detail in Section 3.

Thus Formal Methods is to be used —as intended— on large discrete and, more conveniently, continuous and hybrid state spaces. Statistical specifications arise because individuals are too numerous to be addressed individually; differential equations specify emergent behaviour in terms of rates of change; and temporal specifications constrain normative system behaviour. Yet all are ultimately to be implemented by discrete agents. Thus statistical, temporal and hybrid reasoning seem inevitable. Perhaps that is the warning to be taken from the views of the previous section; otherwise the divide may appear too imposing for the incremental method to work. However let us see now how such reasoning can be included in standard Formal Methods.

3 A case study

One of the most popular examples of an AMAS is provided by the food foraging and transporting behaviour of an ant colony. The purpose of this section is to indicate how such behaviour is described using the approach advocated in this paper. Naturally our contribution is not biological. Instead it is the manner in which the AMAS is captured in the formalism being proposed in this paper, and its emergent behaviour ‘revealed’ (since, in the analysis of this

existing system, that is the analogue of its being ‘designed’ in a system under construction). Of particular interest is the use of DEs to capture emergence that is discharged by a discrete implementation.

We begin with a simple general model before specialising to the case of a steady-state foraging trail. For succinct summaries of ant behaviour we refer to the articles by Jackson and Ratnieks [26] and to Wikipedia.

3.1 Ant behaviour

A simple model of ant behaviour, known as *central-place food foraging* (see Sudd and Franks [51]), is as follows. Ants forage randomly for food, typically exploring up to 200 metres from the nest. Essentially they are blind, unable to smell food and have little memory (see Hölldobler and Wilson [23]). If they find food then they transport some of it (by following a pheromone trail if one presents itself but otherwise by retracing their own route by a combination of means) back to the nest, in doing which they lay down a pheromone trail for others to detect. A foraging ant which intercepts a pheromone trail is able to determine the direction of the food (using forking paths and antennation with other ants on the bidirectional trail). After depositing food at the nest, each ant recommences foraging. Since pheromones deteriorate after about six minutes, trails need continually to be updated. The speed of ants is about the same in either direction on a trail.

This simple model overlooks many things (not least the difference between kinds of ant), including obstacle avoidance, dropping of food, U-turns and competitors. But it provides an appealing MAS which is completely decentralised and whose agents/ants are essentially memory-free. Our first step is to show how it can be formalised.

3.2 Formalisation

An ant is assumed to have two states, F for ‘foraging’ and T for ‘transporting’, and, at any time, a location in some set Loc containing a distinguished location $nest$. Initially each ant is in the foraging state and at the nest. An ant can forage for food and it can transport food, but it cannot do both simultaneously.

In foraging, an ant is sensitive to the presence of pheromone (modelled as Boolean input $ph?$ being true) and food (Boolean input $fd?$ being true). In the absence of both food and pheromone it continues to forage according to the function *advance*; in the presence of food it changes state to transport some of it back to the nest; and in the absence of food but the presence of pheromone it follows the pheromone trail according to the function *follow*.

In transporting food back to the nest the ant is again sensitive to the presence of pheromone, which it itself now deposits. At the nest it resumes foraging; otherwise, in the presence of

pheromone it moves according to the function *follow* and, in its absence, according to the function *return*. For simplicity we suppose that those functions output pheromone in the transporting state.

For formalisations of the ant and the ant colony, with the three functions *advance*, *follow* and *return* left generic, see Figures 1 and 2 respectively. The notation Object-Z (with minor variations; see Duke and Rose [13]) is used to define each as an object: as having internal state supporting various methods. An ant, for instance, has state composed of *F* or *T* together with a location; initially its state is *F* with location at the nest. It has two operations; each changes state; and *forage* accepts two inputs whilst *transport* accepts one input and delivers one output; all I/O are Boolean. The transition predicate (involving state-before, input, state-after and output) defining each operation is given in the lower part of the schema box; state-before components are unprimed whilst state-after components are primed.

So far, no emergent behaviour has been incorporated in our model of the ant colony. The primary example of emergence in this paper appears in the context of a restriction of that model in Section 3.4. But let us first give a simple example in the model of Figure 2.

3.3 Naive emergence

In this section we consider a very simple case and return to a more realistic model in the next section. Our purpose here is to demonstrate in as simple a setting as possible the difference between specification, with emergent predicate, and implementation, where the predicate has been incorporated into the design of each agent (*i.e.* ant).

For the moment we assume that locations lie on a planar integer lattice, $Loc = \mathbb{N} \times \mathbb{Z}$, with the nest at $(0, 0)$ and we suppose that for each ant, the function *advance* chooses between the two forward neighbours of its argument

$$advance(x, y) \in \{(x, y+1), (x, y-1)\}.$$

Thus the set of points able to be occupied by an ant is

$$A = \{(x, y) : \mathbb{N} \times \mathbb{Z} \mid (|y| \leq x) \wedge even(x + y)\},$$

and after m time steps an ant has reached a point $(m, y) \in A$ on the vertical line $x = m$. See Figure 3. With a total of n ants, the fraction of ants at the point (m, y) is (writing $\#$ for cardinality)

$$\beta(m, y) = \frac{1}{n} \# \{a : E \mid a.(x, y) = (m, y)\}.$$

The colony is specified to consist of a bag (*i.e.* multi-set) of ants and the emergent behaviour we want is that for each m , ants are roughly normally distributed along (the alternate points of)

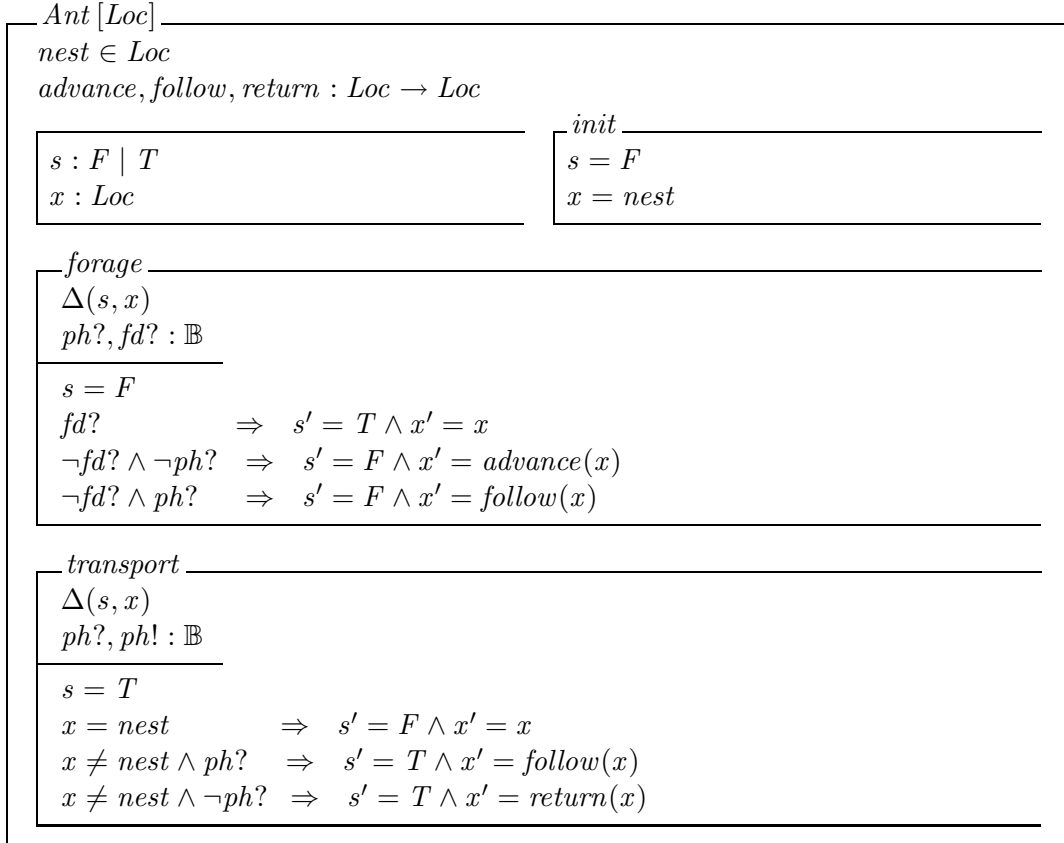


Figure 1: An abstract description of an ant in terms of the generic functions *advance*, *follow* and *return* which determine how the ant's position changes with time. *Loc* is some generic set of locations with a distinguished element *nest*. States *F* and *T* represent foraging and transporting, respectively. Inputs *fd?* and *ph?* are Booleans representing the presence of food and pheromone respectively; output *ph!*, which is returned by the functions *follow* and *return*, deposits pheromone. The precondition of action *forage* has state $s = F$ and that of action *transport* has state $s = T$.

$x = m$ with mean 0 and variance $2m$ (as in Figure 3). The emergence behaviour is formalised by the predicate

$$\text{Normal}(E, m_0, \varepsilon) = \forall m \leq m_0 \cdot (m, y) \in A \Rightarrow |N(0, 2m)(y) - \beta(m, y)| < \varepsilon,$$

where $N(0, 2m)(y)$ denotes the normal distribution, with mean 0 and variance $2m$, at the point y . The values of m_0 and ε determine the accuracy of approximation. (The constant m_0 is necessary because ε is not m -dependent; without it the approximation would become unrealistically close for large m .)

The resulting *Specification* is given in Figure 4. In summary, the unilateral ant descriptions have been augmented with the emergence predicate *Normal* which captures behaviour described in terms of *all* ants via the free variable E . In the sense of reflecting all the system components, it is a typical emergence predicate. Its purpose is to exclude, to within the stated confidence, ‘biased’ or ‘extreme’ choices of move otherwise permitted by the specification.

The most obvious way to meet that specification is by empowering each ant with the ability to make a fair choice at each move. Then, with all ants following the same policy, their distribution after a given number of steps is binomial and, in the limit, normal (by a simple version of the central-limit theorem; see for example the text by Bertsekas and Tsitsiklis [4]). Thus even without agents exhibiting any form of cooperation, the MAS exhibits its emergent behaviour: the spatial distribution of foraging agents after a given number of steps is approximately normal. Again, see Figure 4.

That implementation is unusual because each component in the specification has been (autonomously) strengthened to ensure that the implementation bag E satisfies the specification’s emergence predicate. Thus the emergence predicate has become incorporated entirely ‘locally’. More typically that can be achieved only at the meso level, midway between the individuals and the system as a whole, often by grouping of components to ensure intermediate properties which in turn achieve the global emergence predicate.

Why is that limiting behaviour referred to as *emergent* if it can be realised without any ant interactions? Because in a language based on the state and actions of just a single ant, expectations and probabilities cannot be expressed: formally, it makes no sense to talk about the distribution of a single ant’s trajectory, for which more sophisticated expressions involving many ants are required.³

Thus prepared, it is time to consider our main example of emergence in which rates of change are necessary.

³Remember that the argument here is a formal one; to express distribution of outputs, for example, requires further observables.

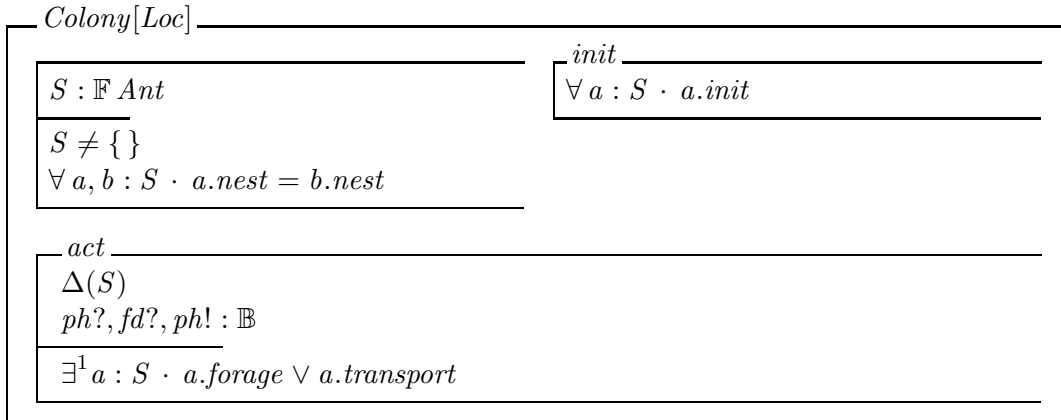


Figure 2: An abstract description of an ant colony as a nonempty finite set of ants sharing the same nest. (By abuse of notation, all ants share the same location *Loc*.) Its initialisation consists of initialisation for each ant and its single action comprises an internal choice promoting one (since their preconditions are disjoint) of the two ant actions.

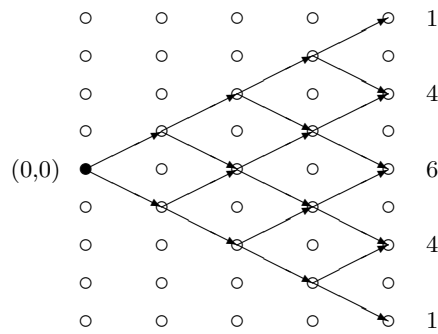


Figure 3: From each point an ant moves with equal probability to one of its two forwards neighbours. After a given number of steps, ants are distributed binomially, and hence approximately normally. The conclusions concerning emergence of the limiting distribution apply equally to the more accurate discrete approximation to Brownian motion.

3.4 Trail dynamics

We now return to the model of the ant colony from Section 3.2 and restrict attention to the movement of ants, in one dimension, along an established trail between a supply of food and the nest. Thus after leaving the nest each ant detects and follows the pheromone trail to a supply of food. It then returns to the nest following the trail.

But having made that simplification we can follow Johnson and Rossi [29] and, with analogue time, see how two waves of movement emerge, surprisingly, in either direction, with distinct phase velocities. Suppose that *Loc* is the real interval $[0, 1]$, with *nest* = 0 and food at $x = 1$. Assuming that a trail has been established along the unit interval and is being used in the steady state; input *ph?* holds in both actions *forage* and *transport*, which simplify as a result. Furthermore, the input *fd?* can be ignored, since it can be inferred from the invariant *fd?* = ($x = 1$); and the output *ph!* can be ignored since it can be inferred from *ph!* = ($s = T$). Finally, in the steady state the value *follow*(x) of an ant in state $s = F$ is simply $1 - \text{follow}(x)$ for an ant in state $s = T$. Incorporating time explicitly, now, in the function *follow*, the resulting simplifications of *forage* and *transport* from Figure 1 are shown in Figure 5.

Following Johnson and Rossi we consider, in each of an ant's states F and T , a pair of analogue functions of position $x : [0, 1]$ and time $t : \mathbb{R}$. The first represents the ant's velocity, v_F or v_T , and so is given by the derivative with respect to time of *follow*

$$v_F = \frac{\partial}{\partial t} \text{follow} = -v_T.$$

The second represents the density of ants in either direction, ρ_F or ρ_T , reflecting the combined effects of pheromone concentration, antennation and so on.

Johnson and Rossi show that by representing the functions as deviations (having superscript (1)) from their mean values (superscript (0)),

$$\rho_F = \rho_F^{(0)} + \rho_F^{(1)} \quad \text{and} \quad \rho_T = \rho_T^{(0)} + \rho_T^{(1)},$$

the velocity functions can be eliminated with the result that ρ_F and ρ_T satisfy a pair of second-order coupled partial-differential equations

$$\begin{aligned} \frac{\partial^2}{\partial t^2} \rho_F^{(1)} - 2V \frac{\partial^2}{\partial x \partial t} \rho_F^{(1)} + V^2 \frac{\partial^2}{\partial x^2} \rho_F^{(1)} + k_F \rho_F^{(0)} \frac{\partial^2}{\partial x^2} \rho_T^{(1)} &= 0 \\ \frac{\partial^2}{\partial t^2} \rho_T^{(1)} + 2V \frac{\partial^2}{\partial x \partial t} \rho_T^{(1)} + V^2 \frac{\partial^2}{\partial x^2} \rho_T^{(1)} + k_T \rho_T^{(0)} \frac{\partial^2}{\partial x^2} \rho_F^{(1)} &= 0 \end{aligned}$$

(for appropriate constants k_F, k_T, V ; see [29] (6a), (6b)), which can in turn be replaced by a pair of coupled second-order ODEs. Solving the resulting 4×4 linear system they deduce that in both directions the density differs from a constant by a Fourier series composed of two pairs of contrary-moving waves, featuring the same pair of explicitly determined phase velocities (one always greater than the other). Thus in both cases (F and T) the density has the form

$$\rho^{(0)} + \text{Re} \sum_n (a_n e^{ik_n(x-ct)} + a'_n e^{ik_n(x-c't)}) + (b_n e^{ik_n(x+ct)} + b'_n e^{ik_n(x+c't)})$$

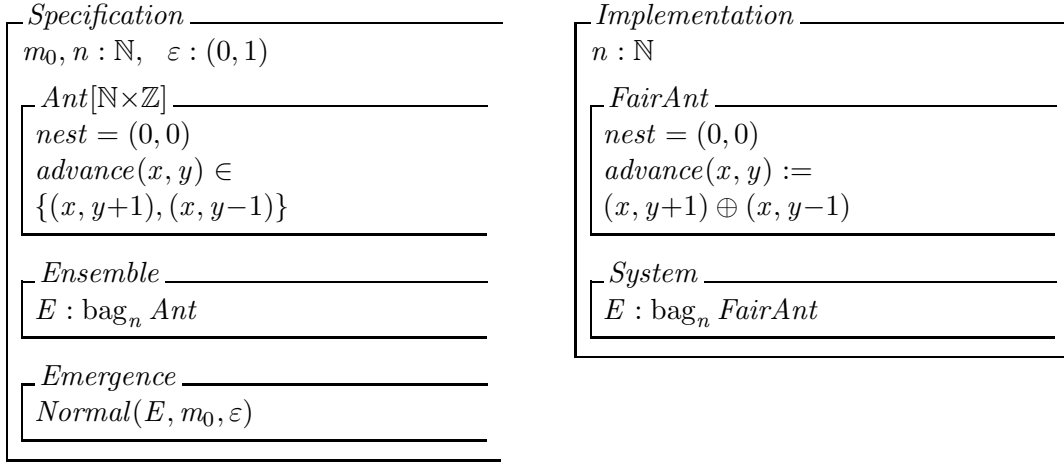


Figure 4: Systems capturing the specification and implementation of a simple stream of ants. The type $bag_n T$ denotes the bags of T of size n . The emergence predicate $Normal(E, m_0, \varepsilon)$ ensures that amongst the first $m \leq m_0$ moves, the distribution of ants along $x = m$ is normal to within error ε . The expression $A \oplus B$ denotes a fair choice between outcomes A and B , as might be achieved, for instance, by use of a random number generator. Correctness of the implementation follows by the binomial approximation to the normal distribution (*i.e.* a simple case of the central-limit theorem).

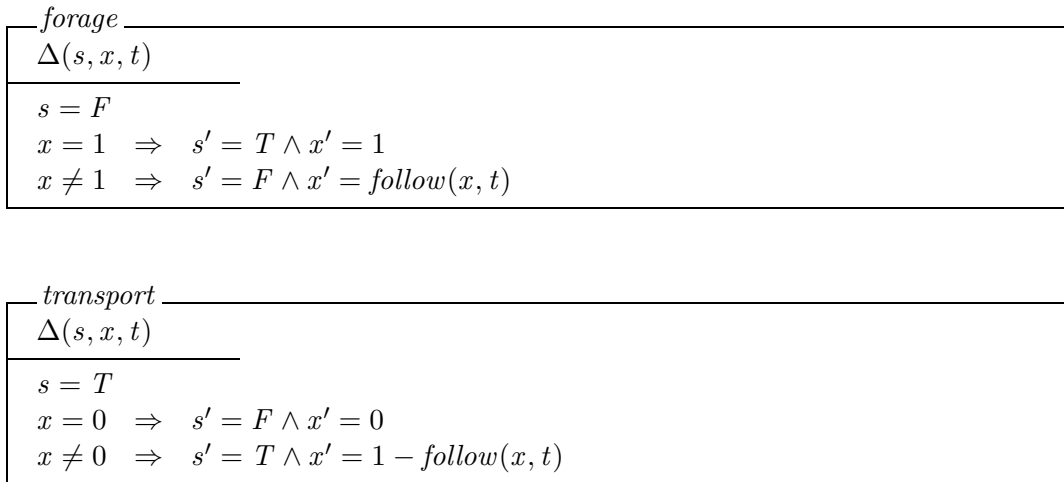


Figure 5: Simplifications of an ant's two actions in the steady state of an established trail along $Loc = [0, 1]$. Time is now explicit in the function $follow(x, t)$ and input and output are ignored since now their values can be inferred from those of x and s .

(where Re returns the real part of its complex argument) for constants $c' < 1 < c$ (*i.e.* independent not just of n but also of F and T). They confirm this surprising behaviour experimentally on a 29-centimetre trail.

Our concern here is to use those results to distinguish specification, with an emergence predicate, from design and implementation and yet to show how all can be incorporated in current Formal Methods. We define the predicate $pde(\rho_F, \rho_T, k_F, k_T, V)$ to consist of the two coupled second-order partial-differential equations above. Its importance is that it captures the stable state of the trail, by encapsulating the obvious constraints involving the observables *follow*, ρ_F and ρ_T . It thus provides the means for specifying the behaviour of the ant colony.

Secondly we define the predicate $phased(\rho_F, \rho_T, c, c')$ to express the coupled phase phenomenon described above (for frequencies n lying in some set E and some choice of value constant in x and t but not n)

$$\begin{aligned} & phased(\rho_F, \rho_T, c, c') \\ & = \\ & \exists \rho_F^{(0)}, \rho_T^{(0)} : \mathbb{Z} \cdot \exists E \subseteq \mathbb{Z} \cdot \forall n : E \cdot \exists k_n, a_n, a'_n, b_n, b'_n, c_n, c'_n, d_n, d'_n : \mathbb{Z} \cdot \\ & \quad \rho_F = \rho_F^{(0)} + Re \sum_{n \in E} (a_n e^{ik_n(x-ct)} + a'_n e^{ik_n(x-c't)} + (b_n e^{ik_n(x+ct)} + b'_n e^{ik_n(x+c't)}) \\ & \quad \rho_T = \rho_T^{(0)} + Re \sum_{n \in E} (c_n e^{ik_n(x-ct)} + c'_n e^{ik_n(x-c't)} + (d_n e^{ik_n(x+ct)} + d'_n e^{ik_n(x+c't)}). \end{aligned}$$

Its importance is that it describes explicitly the emergent behaviour of the colony. The behaviour cannot be imposed at the level of abstraction of a single ant (as for instance ‘fairness’ was imposed on a single coin in Figure 4), requiring instead the monitoring of several ants. But equally it is not global because only a sample of ants need be considered in ensuring it. Thus *phased* represents a local condition —to be achieved hierarchically at a level intermediate between individual ants and the whole colony— which is therefore the basis of an implementation, whilst *pde* is global and so forms the basis of a specification. See Figure 6.

4 Policies and design patterns

In the design of object-oriented systems ‘design patterns’, like those proposed by the Gang of Four [19], play an important role. The purpose of this section is to indicate how the approach of the present paper may help to overcome the difficulty of that task in the (A)MAS case, thus making feasible a systematic library of design patterns at the meso level for helping designers to achieve various kinds of emergence at the macro level.

To date, various meso-level designs have appeared in experiments with the large number of AMAS platforms.⁴ In this section we review a cross section of them, indicating how they may be viewed in terms of agent policies at the meso level achieving emergence at the macro level

⁴Examples of what might be termed *third generation* AMAS platforms relevant here include ADELFE, DIET, GAIA, JADE (with FIPA-compliant middleware), TOTA and TROPOS.

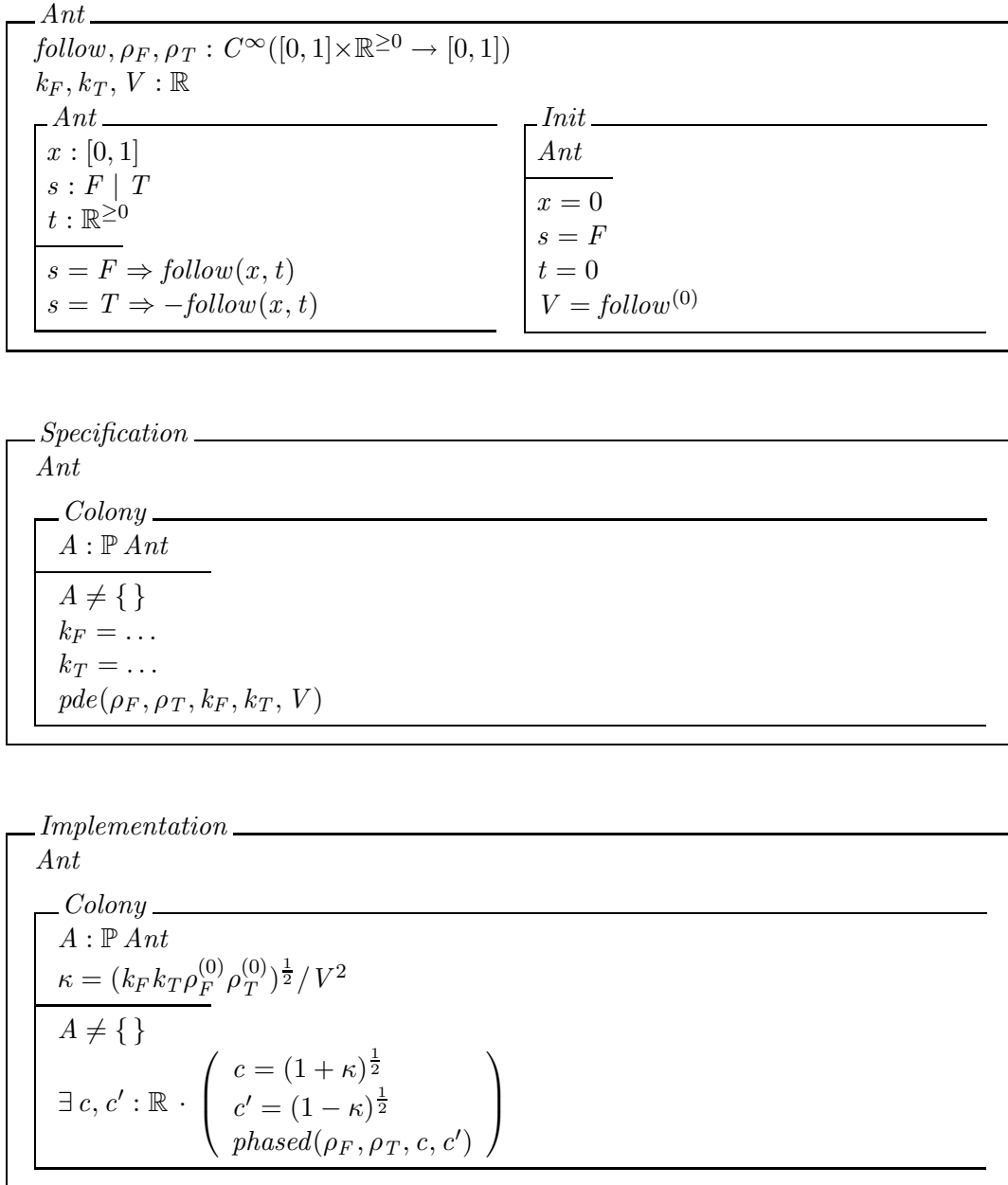


Figure 6: Systems capturing the continuous behaviour of the ant colony at the specification and implementation levels. For subsets S of Euclidean space and T of \mathbb{R} , $C^\infty(S \rightarrow \mathbb{R})$ denotes the type of continuously differentiable functions from S to \mathbb{R} . The constants k_F and k_T are determined by the relationship between *follow* and the density functions. In the specification, the partial differential equation acts as an emergence predicate to specify, globally, the required emergent behaviour. In the implementation (where $\kappa : \mathbb{R}$ is simply a convenient abbreviation) such behaviour is assured by the (locally-defined) predicate *phased*, using results of Johnson and Rossi [29].

yet susceptible to implementation at the micro level. We have chosen to concentrate on those that support efficient peer-to-peer interactions in the presence of adaptive cooperation and agent mobility, since that complements the standard case of distributed systems.

But first we introduce the notion of ‘policy’ that provides a systematic technique to bridge the meso-micro divide.

4.1 Policies

The behaviour of each agent is in general a combination of its unilateral behaviour and its interactions with other agents, and the emergent behaviour of the system derives from the latter. An agent’s behaviour, ensuring the system property that has been viewed as emergent, is thought of as the agent’s *policy*. Evidently that is ‘view-dependent’ and so it acts as a design heuristic. The word ‘policy’ is chosen to reflect the fact that an agent’s interactions depend on time, prevailing conditions and perhaps probability, as a result of dynamic adaptability to an open environment. That view is an integral part of the way in which agents are viewed in agent-oriented software engineering; for a survey see Jennings [28].

If a policy is viewed as being state based, its state consists of some dynamically changing data structure whose transition function is constrained by the policy. However a policy may be described more abstractly using various forms of logic, including temporal and modal logics like epistemic logic [16], qualitative choice logic [6], and implemented using various designs, like Belief-Desire-Intention [20], probabilistic algorithms [38], and a variety of *ad hoc* style a sample of which is surveyed in this section.

But first we return to the simple example of emergence and ants.

4.2 Ants again

Recall the implementation of emergence from Figure 4, termed ‘simple’ emergence because it can be implemented without agent interactions. Each ant, at any time x , from position y chooses with equal probability to move to $y - 1$ or $y + 1$. That is each ant’s policy —of equal probability— for implementing the nondeterministic specification. An alternative policy might be for the move to $y - 1$ to be twice as likely as that to $y + 1$; or the probability of choice might be ant dependent.

Let us now consider the more complex case of Figure 6. A policy is introduced hierarchically, at a level of organisation between the meso and micro levels. For example in a ‘societal system’ [57], agents are grouped into societies each with its own society-specific policy to be imposed on groups of individuals. In the present example, a discrete design for the implementation of Figure 6 would meet the constraint imposed by the density functions ρ by policies involving buffers at the nest and at the food source, to ensure the appropriate flow of ants, enabling an

ant to benefit from antennation and stigmergy, to revise its interactions given environmental change, and so on. The collection of such policies provides the colony with a meso-level design that results from the micro level and at the macro level achieves emergence.

Stigmergy provides a means of agents interacting indirectly, by accessing a dynamically changing data structure; by comparison antennation is direct interaction. Those two paradigms play an important role in the design patterns to follow. The patterns are evaluated for their functionality and interpreted in terms of policies, and are arranged in increasing specialisation.

4.3 Gossip

Jelasity [27] observes that ‘gossip’, or communication amongst only close friends, results in the remarkably efficient dissemination of information society wide. Furthermore the idea is so firmly rooted in the theory of distributed algorithms that scarcely a single (distributed) algorithm fails to use it. ‘Close friends’ are neighbours in the adjacency relation of the communication graph of the system and social dissemination of information corresponds to the global acquisition of knowledge, achieved by only local communications. The proposal is thus a ‘template’ to be tailored to particular applications.

Two examples are provided in [27]: (a) establishing a communication network (by learning the addresses of other agents); and (b) acquiring global information (in the form of data aggregation). In fact the two are equivalent: (a) is an instance of (b) in which the information being acquired corresponds to the communication graph of the network; and (b) is an instance of (a) in which the network is used to disseminate the information globally. Connectivity of the network is assumed in order to attain globality. Gossip has the advantage of being adaptable in the sense that if the network changes at run time, globality is still achieved (given connectivity).

In our terms, the emergence predicate describes the global state achieved by gossip, and the policy followed by each agent determines its (dynamic) choice of neighbours and what information it passes to them. In particular, Jelasity’s two examples are seen as follows. In (a) an ‘overlay’ network emerges. It is specified (the emergence predicate) as a function assigning to each pair of agents the address of the second from the point of view of the first. It emerges as a result of agents invoking their **selectPeer** operations which embody agent preference and which is biased towards network proximity (agent policy). In (b) what emerges is each agent’s evaluation of a global function, like maximum, minimum or some form of average. The emergence predicate is thus that function, from agents to values. An agent’s method of achieving it is encapsulated in its **selectPeer** operation; Jelasity considers randomisation, but that is just one policy decided by the agent.

In general, each agent’s state contains a component $info(a)$. The emergence predicate records that eventually that component contains the global information $global$. In terms of modal logic, $\diamond\Box\forall a : Agents \cdot info(a) = global$.

4.4 Middle agents

Whilst gossip is suitable in an MAS whose agents are reasonably homogeneous, efficiency often requires specialisation of agents. In the *selective information dissemination* of Koubarakis *et al.* [33] agents seeking to share data are assisted by ‘superpeers’. Each agent is assigned a superpeer for the purpose of advertising the information it can offer and the information it seeks; the superpeers form a peer-to-peer network. The system must adapt to agents moving between superpeers and even being absent from the system for some time without missing any information.

What emerges is the knowledge, by each agent, of where (*i.e.* from which other agents) to obtain the data in which it is interested. The policy of each agent is firstly to forward to its current superpeer a *profile* of its interests and a *notification* of information it possesses; secondly to migrate so as to acquire efficiently the data it requires, as a result of information obtained from its superpeer. However it is the policy of the superpeers that is the key to the efficiency of the system. In order to minimise the circulation of profiles and notifications, superpeers can exploit sophisticated algorithms to suppress less informative documents ([33], Section 3). They must also decide what information to store in order to cater for agent mobility. Finally superpeer policy must also deal with the failure of other superpeers and agents.

An alternative approach is provided by Wang [53] who calls superpeers ‘middle agents’. Each middle agent discharges its operation by performing three actions: *search*, *award* and *exchange*. In *searching* the middle-agent network to meet an enquiry, the middle agent’s policy consists of its search strategy: how many and which other middle agents to contact if the enquiry cannot be met on its own. If an enquiry cannot be met locally, agents are swapped between middle agents in order to preserve efficiency. The result is a grouping, by middle agent, of agents having similar interests (as reflected in their enquiries and notifications). In order to swap one of its own agents that requires information for one of the receiving middle agent’s agents that does not require the same kind of information, a system of agent awards is maintained by the middle agents. What emerges is the accumulation, by middle agents, of agents having similar interests.

In assigning an *award*, the middle agent’s policy is determined by the protocol and values it uses. In performing an *exchange* a middle agent’s policy employs those awards to make the swap. Experiments in [53] show that the number of queries per user, to achieve the required emergent property of grouping like-interested agents together, scales linearly with number of agents up to 5000.

4.5 Identifier-based methods

The first peer-to-peer MASs, like Gnutella [21] and Freenet [10], focused on file sharing. The second generation aimed to provide efficient, adaptable, scalable platforms for file-sharing amongst other applications and in doing so formed the first systematic meso-level AMAS designs. One

family of such systems⁵ dynamically groups agents by proximity of their identifiers, which are used to determine routing in a manner motivated by the neighbour-to-neighbour routing in a hypercube (there achieved by changing one bit of the address at a time but here by matching successively longer prefixes of the target address). For example Rowstron and Druschel [42] construct a system, Pastry, whose emergent property is the routing of a query to a (live) agent responsible for information associated with the query key —achieved by the agent’s address being nearest to the key— in spite of agent additions and deletions. It is accomplished by an overlay network; each agent’s routing policy results in a query being routed directly to the desired address or forwarded to an agent able to discriminate more accurately a prefix of the key. An alternative randomised policy has also been suggested to overcome a small number of malicious and failed agents. Policies are also required for new agents, and for handling deleted agents.

A quite different design is that of *Tags*, due to Hales *et al.* [22]. Each agent is endowed with a bit string that contains one place reserved for the bit representing the agent’s policy when playing the prisoner’s dilemma (PD) against other agents, (call it the PD-bit) and a tag that affects a partition of agents under the relation of ‘equal tags’. Within each equivalence class pairs of (distinct) agents play the PD, with policy determined by the PD-bit, until all agents have competed. An agent in a singleton equivalence class competes with another, randomly chosen, agent. Then agents are replicated in proportion to the reward accruing from the PD; finally with low probability a mutation is applied to the PD-bit and tag, and the procedure repeated. In [22] a networked variant is also studied.

Cooperation rapidly emerges in the population of agents because it is rewarded in the PD (provided, it turns out, that the level of mutation is an order of magnitude greater in the tag than in the PD-bit, a condition that is ensured if mutation occurs independently at each bit of a sufficiently long tag). That is in contrast to the Nash equilibrium for the straight PD (in which players defect). Thus by changing parameters (length of the tag, probability of mutation and PD payoff) some control can be achieved over the degree of coordination. That is the basis for each agent’s policy in seeking to cooperate with others.

The mechanism of tags provides a subtle and interesting tradeoff between cooperation being achieved ergodically and groupwise. Related work, providing further concepts to describe the results of meso-level designs, includes [11].

4.6 Fields and gradients

The usual algorithms for consensus amongst agents in a distributed system pay scant regard to context. However in a MAS there is a family of systems whose emergent properties are achieved using contextual fields.

Mamei *et al.* [35] introduce the idea of *co-fields* (for ‘computational fields’) generated by both

⁵Examples include Accordeon, Bamboo, CAN, Chord, Kademia, Kelips, Pastry and SkipNet.

agents and their environment to provide such context. Establishment and use of a field are obviously application dependent. Their primary example is that of urban traffic management. The field is generated by (static) sensors embedded in the environment (at every street corner) and by (mobile) vehicle sensors. Typical applications include balancing of traffic, coordinating a meeting between vehicles, and maintaining distance between patrolling vehicles. In each case the policy for calculating the field differs. (We refer to [35], Section 5.4 for details.)

A related use is one in which agents achieve a target configuration in 3-space, one agent per location, using only local information and in spite of run-time interruptions. Several such algorithms are considered by Støy [50]. Initially a sufficient number of agents are supplied, including a ‘seed agent’ which knows the target configuration and is located within it at a location it knows. Agents know the directions of their neighbours and so, as communications spread from the seed, can determine their own locations and whether or not they lie within the target. If it lies within the target, an agent probes its neighbouring location to determine whether or not it needs neighbours. If so it initialises a ‘recruitment field’ to 0 at its location, and the field propagates from neighbour to neighbour by incrementing its value so that the field decreases strictly towards a source. Agents outside the target can follow the field towards a source and eventually occupy a vacant target position. Agents whose vacant target neighbours become filled cancel their field. By maintaining connectivity of agents (by use of a second field) and under the assumption that the target contains enough nontarget locations around it for agents to move uninhibitedly, eventually all agents are placed in the target.

What emerges is thus the positioning of agents, one per location, within the target. An agent within the target has a policy that determines how it probes its neighbours, propagates its own and other recruitment fields, and eventually cancels its own field. Other agents have a policy of how they follow the recruitment field to a source, maintaining connectivity and avoiding collisions. Only minor variations are required to those policies to adapt to environmental interference [46].

Further variations include the work on Stigmergy (Karuna *et al.* [31] and Mamei and Zambonelli [37]) and *Tuples on the air*, TOTA, Mamei and Zambonelli [36].

4.7 None of the above

Further important meso-level design patterns currently being investigated include: the organisational/societal metaphor of Zambonelli, Jennings and Wooldridge [57]; mobile grid services including emergence of properties like authentication, authorisation, integrity, confidentiality, permissions and protection [55]; more general security properties, and swarms [5] (of interest in the recent application to satellites).

5 Conclusion

AMAS are expected to respond dynamically to environmental change, which is why they exhibit emergent behaviour. They achieve it by carefully configured inter-agent interactions at the meso level. Therein lies the fascination and subtlety of engineering AMAS. The need for accountably correct AMAS, in the presence of a large number of carefully considered notations and methodologies for AMAS, means that it would be useful to have formal support for the incremental derivation of MAS for use by a range of methodologies.

In this paper a formalism has been proposed for the incremental top-down development of AMAS. It consists of specifying the AMAS by conjoining to the combined unilateral agent behaviours a ‘global’ emergence predicate, and introducing appropriate designs in terms of agent policies at the meso level, for eventual implementation at the micro level. But how can that approach of Formal Methods, which has been used for several decades after all, be used when the formalisation of emergence might require statistical description or specification by differential equation? Our case study addresses both those issues, and shows how the notion of policy is helpful in formulating an implementation design.

However to be fundamental, the notion of policy ought to be apparent in current work on meso-level design. We have surveyed what we hope to be a representative cross-section of such work, and in each case found our understanding to be informed by use of the idea of agent policy.

Further work therefore seems justified. A compendium of meso-level design patterns would be as helpful in designing self-* AMAS as it has been in designing object-oriented systems. Application of the technique proposed here to the meso-level steps of a case study like free-flight would be interesting. Elucidation of sufficient criteria for the refinement of statistical and differential behaviours is also important.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant No.60773208; the Specialized Research Fund for the Doctoral Program of Higher Education under Grant No.20070532075; also by the Macao Science and Technology Development Fund, under the PEARL project, grant number 041/2007/A3.

References

- [1] L. Aceto and M. Hennessy. Towards action-refinement in process algebras. *Information and Computation*, **103**(2):204–269, 1993.

- [2] R.-J. R. Back and K. Sere. Stepwise refinement of action systems. In *Mathematics of Program Construction, LNCS, 375*:115–138, Springer Verlag, 1989.
- [3] E. Ball and M. J. Butler. Using Decomposition to Model Multi-agent Interaction Protocols in Event-B, 2006.
fm06.mcmaster.ca/11ElisabethBall.pdf
- [4] D. P. Bertsekas and J. N. Tsitsiklis. *Introduction to Probability, 2nd Edition*. Athena Scientific, 2008.
- [5] G. Bonnet and C. Tessier. An incremental adaptive organization for a satellite constellation. In A. Artikis, J. Pitt, K. Stathis and G. Vouros, editors, *Post-proceedings of OAMAS 2008, LNAI 5368*, Springer Verlag, 2008.
- [6] G. Brewka, S. Benferhat and D. Le Berre. Qualitative Choice Logic. In *Proceedings of Principles of Knowledge Representation and Reasoning, KR-02*. 2002.
www.informatik.uni-leipzig.de/brewka.
- [7] K. M. Chandy and J. Misra. *Parallel Program Design: A Foundation*. Addison Wesley, 1988.
- [8] C-C. Chen, S. Nagl and C. Clack. A calculus for multi-level emergent behaviours in component-based systems and simulations. In *Proceedings of Emergent Properties in Natural and Artificial Complex Systems, (EPNACS'2007)*, part of the 4th European Conference on Complex Systems (ECCS'07), 35–51, 2007.
- [9] B. H. C. Cheng *et al.*, Software engineering for self-adaptive systems: a research road map. Dagstuhl seminar proceedings 08031, *Software Engineering for Self-Adaptive Systems*, 2008.
<http://drops.dagstuhl.de/opus/volltexte/2008/1500>.
- [10] I. Clarke, O. Sandberg, B. Wiley and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 311–320. ICSI, Berkeley, CA, USA. July 2000.
- [11] T. Condie, S. D. Kamvar and H. Garcia-Molina. Adaptive peer-to-peer topologies. In *International Conference on Peer-to-Peer Computing*, 53–62, IEEE Press, 2004.
- [12] F. Cucker and S. Smale. On the mathematics of emergence. *The Japanese Journal of Mathematics*, **2**:197–227, 2007.
- [13] R. Duke and G. Rose. *Formal Object-Oriented Specification Using Object-Z*. Macmillan Press, 2000.
- [14] 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, *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, 938–945, IEEE Computer Society, 2004.
- [15] S. Eilenberg. *Automata, Languages and Machines*. Academic Press, 1974.
- [16] R. Fagin, J. Y. Halpern, Y. Moses and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

- [17] M. Fisher and M. Wooldridge. On the formal specification and verification of multi-agent systems. *International Journal of Cooperative Information Systems*, **6**(1):37–65, 1997.
- [18] J. Fromm. On engineering and emergence. Arxiv preprint 2006. <http://arxiv.org/abs/nlin.A0/0601002>.
- [19] E. Gamma, R. Helm, R. Johnson and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley Professional, 1994.
- [20] M. Georgeff, B. Pell, M. Pollack, M. Tambe and M. Wooldridge. The Belief-Desire-Intention Model of Agency. In J. P. Muller, M. Singh and A. Rao, editors *Intelligent Agents V, LNAI 1365*, Springer Verlag, 1999.
- [21] Gnutella protocol specification, 2000. <http://dss.clip2.com/GnutellaProtocol104.pdf>.
- [22] D. Hales and B. Edmonds. Applying a socially inspired technique (tags) to improve cooperation in P2P networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, **35**(3):385–395, 2005.
- [23] B. Hölldobler and E. O. Wilson. *The Ants*. The Belknap Press of Harvard University, Cambridge, Mass., 1990.
- [24] J. Hu, Z. Liu, G. M. Reed and J. W. Sanders. Ensemble engineering and emergence. In [54], 162–178, 2008.
- [25] M. d’Inverno and M. Luck. Development and Application of a Formal Agent Framework. In *Proceedings of the First IEEE International Conference on Formal Engineering Methods*, 222–231, IEEE Press, 1997.
- [26] D. E. Jackson and F. L. W. Ratnieks. Communication in ants. *Current Biology*. **16**(15):570–574, 8 August 2006.
- [27] M. Jelasity. In B. Edmonds, N. Gilbert, S. Gustafson, D. Hales and N. Krasnogor, editors, *Proceedings of the Joint Symposium on Socially Inspired Computing*, AISB Convention, pages 123–126, 2005.
- [28] N. R. Jennings. On agent-based software engineering. *Artificial Intelligence*, **117**:277–296, 2000.
- [29] K. Johnson and L. F. Rossi. A mathematical and experimental study of ant foraging line dynamics. *Journal of Theoretical Biology*, **241**(2):360–369, 2006.
- [30] A. H. Kacem and N. H. Kacem. From formal specification to model checking of MAS using CSP-Z and SPIN. *International Journal of Computing and Information Sciences*, **5**(1):35–44, 2007.
- [31] H. Karuna, P. Valckenaers, C. B. Zamfirescu, H. Van Brussel, B. Saint Germain, T. Holvoet and E. Steegmans. Self-Organising in Multi-agent Coordination and Control Using Stigmergy. *Engineering Self-Organising Systems*, 105–123, 2003.

- [32] D. Kinny and M. Georgeff. Modelling and design of multi-agent systems. In J. P. Miller, M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents, III*, LNAI **1193**:1–20. Springer Verlag, 1997.
- [33] M. Koubarakis, C. Tryfonopoulos, S. Idreos and Y., Drougas. Selective Information Dissemination in P2P Networks: Problems and Solutions. *ACM SIGMOD Record, Special issue on Peer-to-Peer Data Management*, Karl Aberer (editor), **32**(3), September 2003.
- [34] C. Lengauer. A Personal, historical perspective of parallel programming for high performance. In G. Hommel, editor, *Communication-Based Systems (CBS 2000)*, 111–118. Kluwer, 2000.
- [35] M. Mamei, F. Zambonelli and L. Leonardi. Distributed motion coordination with co-fields: a case study in urban traffic management. *Proceedings of the Sixth International Symposium on Autonomous Decentralized Systems, ISADS'03*. IEEE Computer Society, page 63, 2003.
- [36] M. Mamei and F. Zambonelli. Self-Organization in Multi Agent Systems: A Middleware Approach. *Engineering Self-Organising Systems*, 233–248, 2003.
- [37] M. Mamei and F. Zambonelli. Programming stigmergic coordination with the TOTA middleware. *AAMAS*, 415–422, 2005.
- [38] R. Motwani and P. Raghavan. *Randomized Algorithms*, Cambridge University Press, 1995.
- [39] F. Polack and S. Stepney. Emergent properties do not refine. *Electronic Notes in Theoretical Computer Science*, **137**:163–181, 2005.
- [40] M. Randles, H. Zhu and A. Taleb-Bendiab. A Formal Approach to the Engineering of Emergence and its Recurrence. Proceedings of The Second International Workshop on Engineering Emergence in Decentralised Autonomic Systems (EEDAS 2007), Jacksonville, Florida, USA, June 11, 2007, pp12–21. Greenwich University Press, London, UK, 2007.
- [41] W.-P. de Roever and K. Engelhardt. *Data Refinement: Model-Oriented Proof Methods and their Comparison*. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1998.
- [42] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Middleware 2001*, LNCS **2218**:329–350, Springer Verlag, 2001.
- [43] A. J. Ryan. Emergence is coupled to scope, not level. *ArXiv Nonlinear Sciences*, 2006. <http://arxiv.org/abs/nlin/0609011>
- [44] J. W. Sanders and M. Turilli, Dynamics of control, In *Theoretical Aspects of Software Engineering (TASE 2007)*, 440–449, IEEE Computer Society, 2007. Expanded version available as UNU-IIST report 353 at <http://www.iist.unu.edu>.
- [45] J. W. Sanders and G. Smith. Formal ensemble engineering. In [54], 132–138, 2008.

- [46] G. Smith and J. W. Sanders. Formal development of self-organising systems. To appear in ATC 2009. Draft available as Research Report **405**, UNU/IIST, 2009.
- [47] S. F. Smith and C. L. Talcott (editors). *Formal Methods for Open Object-Oriented Systems IV*. Kluwer Academic Publishers, 2000.
- [48] I. Stamatopoulou, P. Kefalas and M. Gheorghe. OPERAS: A framework for the formal modelling of MAS and its application to swarm-based systems, 2007.
esaw07.iit.demokritos.gr/presentations/11.ppt
- [49] S. Stepney, F. Polack and H. Turner. Engineering emergence. *IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'06)*, 89–97, IEEE Computer Society, 2006.
- [50] K. Støy. Using cellular automata and gradients to control self-reconfiguration, *Robotics and Autonomous Systems*, **54**:135–141, 2006.
- [51] J. H. Sudd and N. R. Franks. *The Behavioral Ecology of Ants*. Chapman & Hall, New York, 1987.
- [52] H. Turner, S. Stepney and F. Polack. Rule migration: Exploring a design framework for emergence. *International Journal of Unconventional Computing*, **3**(1):49–66, 2007.
- [53] F. Wang. Self-organising communities formed by middle agents. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 1333–1339, 2002.
- [54] M. Wirsing, J.-P. Banâtre, M. Hölzl and A. Rauschmayer, editors. *Software-Intensive Systems and New Computing Paradigms: Challenges and Visions*, LNCS **5380**, Springer Verlag, 2008.
- [55] S.-W. Wong and K.-W. Ng. MGS: An API for Developing Mobile Grid Services. In *On the move to meaningful internet systems 2006*, LNCS **4276**:1361–1375, Springer Verlag, 2006.
- [56] M. Wooldridge, N. R. Jennings and D. Kinny. A methodology for agent-oriented analysis and design. In *Proceedings of the 3rd International Conference on Autonomous Agents (Agents 99)*, Seattle 69–76, 1999.
- [57] F. Zambonelli, N. R. Jennings and M. Wooldridge. Organisational rules as an abstraction for the analysis and design of multi-agent systems. *International Journal of Software Engineering and Knowledge Engineering*, **11**(3):303–328, 2001.
- [58] 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.
- [59] H. Zhu. Formal reasoning about emergent behaviours of multi-agent systems. *Proceedings of the 17th International Conference on Software Engineering and Knowledge Engineering*, 280–285, 2005.