



The United Nations
University

UNU-IIST

International Institute for
Software Technology

Position paper: Ensemble engineering and emergence (and ethics?)

Hu Jun, Zhiming Liu,
G. M. Reed and J. W. Sanders

December 2007

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

Position paper: Ensemble engineering and emergence (and ethics?)

Hu Jun, Zhiming Liu,
G. M. Reed and J. W. Sanders

Abstract

The complex systems lying at the heart of ensemble engineering exhibit—and are perhaps even characterised by—emergent behaviour: behaviour that is not explicitly derived from the functional description of the components of the ensemble at the level of abstraction at which they are provided. A typical example from Artificial Life comprises an ensemble consisting of a flock of birds; the components are the individual birds, thought of autonomously, and the emergent behaviour is that of the flock. Emergent behaviour *can* be understood by expanding the level of abstraction of description of the components to augment their functional behaviour; but that is infeasible in specifying ensembles of realistic size (although it is the main implementation method) since it amounts to consideration of an entire implementation. An alternative must be taken.

(continued next page)

It is proposed that to specify an ensemble, the functional behaviour of its components instead be augmented by a system-wide predicate (or conjunction of ‘policies’, which may be seen as kinds of weak ‘ethical principles’, when the components are agents) pertaining to the collective behaviour of its components and accounting for emergence. An implementation, however, ranges in distribution from being centralised to being fully distributed, depending on the degree to which the emergent behaviour is incorporated in the components. So the next step in this work consists of identifying implementation designs. A further step consists of establishing criteria for the conformance of an implementation against a specification, and the final step applies those ideas to case studies using model checking. Important application of these ideas lies in ensembles whose dynamics is controlled by artificial agents, for which a satisfactory theory of ethical behaviour can be given that is not based on free will, and takes the form of policies. The tension between emergence and reductionism, that is felt in moving from a specification to an implementation, is resolved by making explicit the level of abstraction of a description.

Hu Jun is a Postdoctoral Fellow at UNU-IIST. He has a PhD in multi-agent systems, artificial intelligence and software engineering from Zhejiang University, and currently holds a position at Hunan University, Changsha.

Zhiming Liu is a Research Fellow at UNU-IIST. His research interests include the theory of computing systems, emphasising sound methods for specification, verification and refinement of fault-tolerant, real-time and concurrent systems and formal techniques for OO development. His teaching interests are Communication, Concurrent and Distributed Programming, Internet Security, Software Engineering, Formal Specification and Design of Computer Systems.

Mike Reed is the Director of UNU-IIST. He is an Emeritus Fellow of St Edmund Hall, Oxford University, where from 1986 to 2005, he was the General Electric Company Fellow in Computation. His previous experience includes terms as a Senior Research Associate at NASA Goddard Space Flight Center and the US Naval Research Laboratory, and as a Manager of Postdoctoral Programs for the US National Science Foundation. He is a former Research Fellow of the American Mathematical Society and former Professor of Mathematics and Computer Science at Ohio University, where he was also the Associate Director of the Institute for Medicine and Mathematics. On three occasions in the 1970's, he was an Exchange Scholar to Eastern Europe (Poland and Czechoslovakia) for the US National Academy of Sciences. He has been a Visiting Professor at the University of Maryland, the US Naval Academy, Tulane University, and the University of Paris. He has given over two hundred research presentations at Universities, Research laboratories, and International research meetings. In addition, he has been the organizer of several international conferences in both Mathematics and in Computer Science. His current work includes the design and analysis of fault-tolerant embedded systems, and automated support for reasoning about computer security. He holds a Doctorate in Pure Mathematics from Auburn University (USA) and a Doctorate in computation from Oxford University (UK)...

Jeff Sanders is Principal Research Fellow at UNU-IIST, with interests lying largely in Formal Methods.

Contents

1	Introduction	1
2	Ensembles	2
2.1	Levels of abstraction	2
2.2	Emergence	3
2.3	Ensembles	4
2.4	Reductionism	5
3	Examples	7
3.1	Coin tossing	8
3.2	Flocks	10
3.3	Machine learning: MENACE	11
4	Emerging ethics	13
4.1	Ethics without free will?	13
4.2	Emergence and conformance	14
5	Conclusion	15
6	Acknowledgements	16

1 Introduction

The large complex systems that currently exist, either by explicit design or by accretion, have been called *ensembles* by the Interlink Working Group on software intensive systems and new computing paradigms (see the Interim Management Report [30]). Examples include the power grid, the internet and large systems of agents (including swarms *etc.*). Naturally there is healthy debate about the characteristic properties of an ensemble, amongst which are included: a massive number of components and behaviour that is open and adaptive (as a result of being situated in the real world) and emergent and statistical (rather than being able predominantly to be addressed at the individual level).

To assist the process of classification, the Interlink group has divided ensembles into two kinds, physical and societal. Examples of the former are: very large adaptive sensor or robot systems; systems composed of programmable molecules; advanced manufacturing systems; the internet. Examples of the latter are: large traffic systems; swarm or colony behaviour; systems of interacting businesses; the stockmarket. Typically ensembles are systems of systems that were not necessarily designed to be composed but adapt, reconfigure and self-organise. Common to all should be a Theory of Ensembles, and Ensemble Engineering.

Ensembles are engineering products, too recent for appropriate supporting theories to have arisen. Such theories would provide the right abstractions for specifying, developing, reasoning about and programming ensembles. Without them, the state of the art will remain at the engineering level; with them there is the prospect of controlling and thus further exploiting ensembles. For standard systems, this is the domain of Formal Methods.

The group ended its second workshop having made considerable progress in demarking areas of interest for future medium and longer-term work, but with some uncertainty concerning emergent behaviour. Clearly, it felt, emergence is a unifying theme across the spectrum of examples. Yet if an ensemble exhibits behaviours not predictable from those of its components, what part can Formal Methods play in Ensemble Engineering? After all, the utility of Formal Methods lies in the specification of systems and the verification of implementations or designs against their specifications. Do these systems lie outside the scope of Formal Methods? And what exactly is Ensemble Engineering?

Those are the topics addressed and answered in this paper. Its purpose is: to clarify the place of emergence in the types of system quoted above (Section 2); to consider typical examples and be guided by them (Section 3); and to suggest an agenda for laying a foundation of Ensemble Engineering (Section 5). On the way it is observed that in the special case of ensembles of agents, the emergent behaviour can profitably be thought of as the result of ethical protocols of the agents, imposed at a societal level. That leads, if the agents are artificial, to an interesting theory of ethics weaker than the usual theory (for sentient agents, based on free will); it is discussed in Section 4.

2 Ensembles

A definition of ‘ensemble’ based on any of the quantitative features like those mentioned in the previous section (number of components and so on) is not going to support a very interesting theory,¹ regardless of the number of exemplars it has. This section proposes instead to study the important place of emergence in such systems by abstracting all other properties and *defining* an ensemble to be a system exhibiting emergent behaviour. That way any conclusions apply to all the examples above.

But that approach presupposes a definition of emergence, an established term about which there is rough consensus but nonetheless a little debate. Again, and for the same reason, this paper takes a ‘minimalist’ view and restricts the definition of ‘emergence’ to just ‘system behaviour not derivable (at the stated level of abstraction) from the behaviour of its components alone’. This suppresses any ‘element of surprise’ sometimes discussed in the philosophy of emergence [5]. The details are as follows.

2.1 Levels of abstraction

The concept of emergence has been associated with complex systems—and in particular those arising in ‘artificial life’—for some time [22, 4].² According to Pepper over 80 years ago,

The theory of emergence involves three propositions: (1) that there are levels of existence ... (2) that there are marks which distinguish these levels from one another ... (3) that it is impossible to deduce marks of a higher level from those of a lower level ...

S. C. Pepper, [25].

That is the view followed here. However the familiar notation of Formal Methods is used to interpret Pepper’s ‘levels of existence’ as ‘levels of abstraction’. The result will be a definition of emergence parameterised by ‘level of abstraction’ (LoA). But first it is necessary to recall that notion, on which Formal Methods is (implicitly?) based.

A formal description of a system consists, regardless of the notation used, of a predicate whose free variables are the system observables, and which therefore determine the LoA of the description. For analogue, differentiable, systems the observables are rates of change of system parameters and the predicate corresponds to the solution of a differential equation which yields the system state at any given time. In that case the standard concepts of Differential Analysis complement those of Computer Science to facilitate description and analysis of such systems.

¹‘Theory’ here is interpreted in the formal mathematical sense of ‘consequences of the definition’. Thus interest focuses on a definition strong enough to support an interesting and appropriate theory whilst being weak enough to apply to the range of desired examples.

²A fine general treatment of emergence is given in the Stanford Encyclopedia of Philosophy [28].

But if the system is discrete, so that the observables assume only finitely-many values, then the predicate can be expressed in terms of system state, input and output.³ There the notations and concepts of Formal Methods are required to structure (particularly, large) descriptions and provide them with semantics. For hybrid systems a combination of both those styles is to be expected. Either way, the result is a predicate whose free variables determine the LoA of the system description.

2.2 Emergence

Now emergence is simply explained: the LoA at which the system is observed differs from that at which the components are specified—it lies at a lower level. In the case of a flock of birds, for example, the components are the birds specified unilaterally at a LoA sufficient for just that purpose; but the flock is observed at a LoA consisting of the previous one augmented by further observables relating to flock behaviour (for example, ‘location of a bird in the flock’ makes sense only at the flock level—although a distributed implementation might enforce it by providing each bird with a (bird dependent) ‘strategy’ for its location within the flock). More detailed behaviour is now able to be observed at the (lower) flock level: the required emergent behaviour situates birds correctly.

For a system to exhibit emergence, not all behaviour possible at the lower level may satisfy the desired criterion for emergence. For example there are ‘potential flocks’ that position birds incorrectly, and so do not conform to the required (emergent) definition of flock. Otherwise, the low-level observables just introduced would not serve to discriminate any behaviours and all low-level behaviour would appear emergent. But then all low-level behaviour would be anticipated in the high-level behaviour, contrary to the desired meaning of ‘emergence’. This apparently subtle point forms an essential part of the definition.

Definition (Emergence). Suppose a system is observed (specified, described) at two LoAs as follows. At the high (or abstract, or component-based) level the observables form a vector $a : A$. At the low (or concrete, or system-wide) level the observables $c : C$ are obtained by augmenting the abstract variables with fresh observables $b : B$ to form $c = (a, b) : A \times B = C$. Thus at the system level, new observations are possible.

The system is said to exhibit *emergence*, as expressed by predicate $em(c)$ at the lower LoA, if the truth of $em(c)$ is not determined at the higher LoA:

$$(1) \quad \exists a : A \cdot \exists b, b' : B \cdot em(a, b) \wedge \neg em(a, b').$$

The predicate $em(c)$ is called the *emergence predicate* of the system. □

³Of course there is a trade-off between state and input-output history; hence ‘can be expressed’.

For instance in the case of the flock of birds, $a : A$ consists of the states of the birds, independent of each other but parameterised to make them individual; so A is a (large) product space with one component for each bird. The type B includes a component for ‘location within the flock’. The predicate em includes a conjunct placing each bird in its correct (though perhaps approximate, depending on the exact nature of the description) location. On the other hand, introduction of an observable corresponding to ‘flock sleep’ does not produce emergent behaviour if it occurs exactly when all individual birds sleep.

The following contrived but simple example clarifies the importance of condition (1).

Example. A system is designed to have abstract behaviours consisting of $a : \mathbb{B}$ (where \mathbb{B} denotes the type of Booleans) and concrete behaviours having type $c = (a, b) : \mathbb{B} \times \mathbb{B}$. Three putative emergence predicates are defined:

$$\begin{aligned} em_0(a, b) &\hat{=} \neg b \\ em_1(a, b) &\hat{=} true \\ em_2(a, b) &\hat{=} \neg b \vee a. \end{aligned}$$

Neither em_0 nor em_1 can be considered emergent because neither uses the augmenting variable b to specialise behaviours. In either case the behaviour could be modelled, by suitable interpretation, at just the abstract level. That is not true of em_2 , just because it satisfies condition (1). \square

In some Formal Methods, like alphabetised process algebra, the two descriptions $em(a, b)$ and its abstraction $\exists b : B \cdot em(a, b)$ are deemed incomparable, exactly because the types of their free variables differ. Such theories are therefore not obvious candidates as the basis for a theory in which passage from components to the whole ensemble (or *vice versa*) is required. In others like data refinement, translation via a simulation relation is required before the two levels of behaviour can be directly compared. That latter approach is adopted here.

The setting chosen above for the definition of emergence uses the simplest common relationship between the high and low-level systems: the low-level system is a restriction (by the emergence predicate) of an extension (by the augmenting variables) of the high-level system. More complicated settings are possible. One natural setting is for the low-level system to be a restriction of a data refinement of the high-level system. The setting chosen will depend on what seems simplest and most natural for the system being described.

2.3 Ensembles

Having clarified the definitions of ‘level of abstraction’ and ‘emergence’, the definition of ‘ensemble’ is now straightforward.

Definition (Ensemble). A system forms an ensemble iff it has emergent behaviour: its components are described abstractly whilst its system-wide behaviour is described as the combination of the abstract components augmented by variables and, in terms of them, an emergence predicate. \square

Thus a system forms an ensemble if its behaviour is not derivable (at the stated LoA) from that of its components alone. How is an ensemble to be described? In Section 3 the notation ObjectZ [6] is used simply because it allows the components to be described in a modularised manner, and an emergence predicate to be added.

It is important to appreciate that the definition depends on the LoA: a system may exhibit emergence and so form an ensemble when described at one level (or, strictly, pair of levels) but not at another. This property of the definition is absolutely crucial for Ensemble Engineering, as will appear. Also it resolves, at a more fundamental level, much of the tension between emergence and reductionism.

2.4 Reductionism

Evidently the concept of emergence is coupled not only with that of system complexity but also with that of reducing the behaviour of the whole to that of its parts: with reductionism [10]. But how is that to be reconciled with Formal Methods, which after all relies on the decomposition of a complex system into formalised components. If that methodology does not capture all system behaviour then it is seriously flawed. The next section provides a reconciliation.

The tension between emergence and reductionism is long standing and has been extremely well documented since Descartes. Much of the confusion can be clarified by making explicit the level of abstraction of a description (as described in the previous section) [7].

As will be seen from the examples in Section 3 and, as already pre-empted, at the specification level there is typically insufficient state in the components to account for emergent behaviour of the ensemble. To expand component state would be tantamount to describing an implementation; but then what was emergent in the specification would no longer be emergent in the implementation. What is emergent at one level of abstraction (for us, the abstract level of specification) may not be at another (for us, the implementation level).

Formal Methods supports the top-down incremental derivation of a system from its specification. At intermediate stages the resulting construct, usually called a design, is part specification and partly executable (already code). It is to be expected, then, that derivation will yield intermediate levels of abstraction in which the ensemble's functionality is being captured in an efficiently executable manner—as usual—but also that the emergent behaviour is gradually being accounted for.

A design thus represents a step towards ensuring that the specified components are fit for ensuring the emergent behaviour. At the specification level (exhibiting emergence) the components by themselves (*i.e.* before incorporating the emergence predicate) are not fit; at the implementation level (since no emergence remains) they are fit; and in between they are being modified to make them fit. Throughout this paper ‘fit’ refers to just that conformance.

Whilst emergence has been seen here as the explicit structuring of phenomena at one LoA in terms of those at another, there is a more extreme position in the study of Mind according to which, for example,

... human level intelligence is too complex and little understood to be correctly decomposed into the right subspecies at the moment and that even if we knew the right subspecies we still wouldn't know the right interfaces between them. Furthermore, we will never understand how to decompose human intelligence until we've had a lot of practice with a simpler level intelligence. R. A. Brooks, [1].

There is, of course, considerable support for this conservative position concerning the all-encompassing notion of intelligence. But the last sentence might be seen as suggestive that simpler forms of intelligence and more restricted interfaces be studied first. The hope would then be that a hierarchy of incremental LoAs be used to understand the more detailed behaviour. That is an approach with which Formal Methods has substantial experience.

It has been suggested by Pattee [23] that emergent behaviour may be able to be explained, in our terms, by including in the interface enough state to permit each agent to store the interactions with its environment. Moreover

The question that should motivate AL research is how usefully we can simulate the process of measurement in artificial environments. As I have argued, the answer to this question will require a theory of measurement. H. H. Pattee, [23], page 391.

The notion of LoA appears to be sufficient at the gross level for supporting that program. Langton [15] extends the biological notions of genotype and phenotype to GTYPE and PTYPE respectively so that those terms apply equally to Artificial Life. Thus GTYPE refers to a low-level implementation mechanism or *behavior* whilst a PTYPE refers to a higher-level *behavioural* structure which emerges as a result of those mechanisms interacting. Langton, [15] section 5, discusses the subtleties of inferring the latter from the former. His model provides an important instance of the use of a descripton containing two LoAs, as used here: one for GTYPE observables and the other for PTYPE observables.

In the general setting of complex systems Gell-Mann [9] has suggested that the study of such phenomena be called *plectics* and treats it using an idea he calls granularity, which is conveniently formalised using levels of abstraction. Damper [5] also discusses reductionism and emergence from the perspective of levels of abstraction. Although levels of abstraction do clarify much of the Philosophical discussion, they of course leave open the fundamental question of whether or not there exist ensembles whose emergent behaviour is reducible at any level of abstraction; the example usually quoted is that of *the mind*, and the emergent phenomenon is consciousness. Fortunately in Computer Science our examples are well-defined.

3 Examples

The emergent behaviour of an ensemble with a large number of components may be (partly) statistical in nature. That is something with which ‘traditional’ Software Engineering has little experience in spite of the availability of probabilistic algorithms [21] and societal algorithms [24] from Computer Science. So the purpose of the first example, a fair coin, is to consider in as simple a case as possible the essence of statistical emergence, stripped of the attendant functionality that would make a realistic ensemble more complex. It is to be expected that the statistical ‘ingredient’ must, by the standards of Statistics, be trivial. The purpose of the example is thus to clarify: ‘how does a statistical property emerge, how is it to be specified (using formal methods) and how is it to be implemented?’

The second example, a flock of birds, sketches a dynamically changing ensemble seen from the viewpoint of emergence. A standard from Artificial Life, it contains less detail because the points made—concerning distributed versus centralised control in a design that conforms to the specification of an ensemble—do not lie in the detail.

The final example considers a simple case of machine learning, something to be expected in any agent-based system. It indicates the feasibility of viewing as emergent the result of machine learning, and leads to the discussion of ethics of artificial agents in the following section.

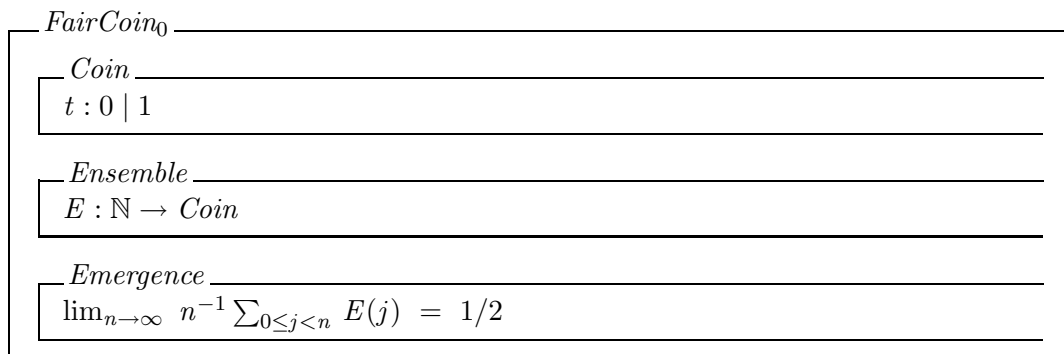


Figure 1: Specification of the ‘probabilistic’ fair coin.

3.1 Coin tossing

By abstracting almost all the functionality in an agent-based ensemble, we are led to the following example of a simple ensemble exhibiting statistical behaviour consisting of (a large number of) tosses of a coin. The abstracted components of the ensemble are identical: each is a single coin toss. The ensemble, however, consists of the (large number of) tosses, and the emergent behaviour is the bias—in this case none—of the coin. Thus there is no way to infer the emergent behaviour from any collection of single components.

In the first specification, the ensemble is countably infinite, permitting expression of the usual criterion that two events 0 and 1 (representing *heads* and *tails*) to be equally probable. In the notation of ObjectZ [6], emergent behaviour may be described as if it were liveness of a state-based system. See Figure 1. The system is named *FairCoin₀*. It has three local constituents: a component named *FairCoin₀.Coin* whose observable *t* is either 0 or 1; a countable collection of such components, called *FairCoin₀.Ensemble*; and an emergence predicate *FairCoin₀.Emergence*. The first is used simply in order to define the second, whilst the third is conjoined with the second to yield the system behaviours.

Though statistically standard, that infinite ensemble is of mere theoretical interest. A more realistic ensemble contains only a finite number of (unordered) tosses. Then the limit is replaced by some agreed statistical approximation: the fraction of heads in the total number of tosses deviates from half by an amount interpreted as fair at a certain confidence level (based on the binomial distribution and, for a large ensemble, its approximation to the normal distribution using the central limit theorem).

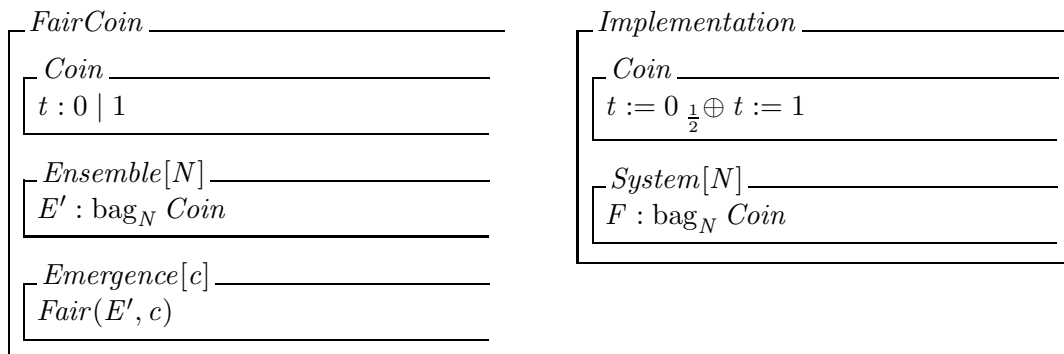


Figure 2: Specification and implementation of the fair coin.

Treating the size N of the ensemble and the degree $c : [0, 1]$ of confidence as parameters (although the latter is typically expressed as a percentage), write $\text{Fair}(E, c)$ to mean that the bag E of tosses is fair at the c -confidence level: the difference

$$| N^{-1} \sum_{e \in E} e - 1/2 |$$

lies within the bound dictated by the confidence level c . See Figure 2.

The obvious implementation consists of the iterated toss of a coin which is fair by construction. The program $P \frac{1}{2} \oplus Q$ is equally likely to be P or Q [18], and may be implemented directly using a random-number generator. The statistical behaviour of the implementation is now not emergent because it is inferred directly from the iterated components. Again, see Figure 2.

Conformance of *Implementation* to *FairCoin'* then follows by standard Statistics (or probability theory, actually, via the binomial distribution). Indeed the criterion is the standard one: the behaviour of *Implementation* is a special case of that described by *FairCoin'*.

By concentrating on just emergent statistical phenomena, that example has been devoid of interesting state. However it is easy to incorporate a mechanism. For example in interactive science museums can often be found a peg board with pegs configured in such a way to deflect balls into columns that exhibit a binomial (or other) distribution. Such a mechanism is readily modelled as a loop in which, on each iteration, its Boolean variable is assigned 0 (a ball goes to the left) or 1 (to the right) with equal probability (say); and that loop is iterated N times.

The *FairCoin* is representative of a well-known family of examples of emergence in which more interesting distributional behaviour (than binomial) emerges as the result of a more subtle generating mechanism. The family includes Zipf's law, Benford's law and so on. A typically thorough but old treatment of such laws is given by Knuth [12]; see more recently [3, 17].

For instance many realistic random variables are lognormally distributed (that is, the logarithm of the variable is normally distributed); that skews them to the right. Examples are heights (usually assumed to be normal), particle sizes, ore deposits, incomes across the population and so on. These days the most convincing explanations for lognormal genesis are founded on the mere process of classifying data. However an elegant mechanism was given by Kolmogorov [13]. An attribute (income, or ore) is dispersed in the population by a series of steps; dispersal is proportional to the power of some constant, but random at each step; so taking logarithms the central limit applies to produce a normal distribution. Such a mechanism can be coded like the peg board above, and produces an example combining both nontrivial state in each component, but statistical emergence in the ensemble.

3.2 Flocks

An ensemble in which more truly dynamic behaviour emerges is that of a flock of birds (swarm of bees, school of fish, crowd of people *etc.*); see for example [2]. The components are the individuals, described autonomously (now statistical behaviour is abstracted, to concentrate on distribution; adaptability of the birds is also abstracted, to be considered in the next example). Such a description suffices to specify the actions of the birds independent of other birds but *en masse*. It does not account for behaviour of the ensemble as a whole, like flocking, which is thus emergent.

The ensemble is specified much as above; for an outline see Figure 3. The specification of the unilateral individuals is given first, with some parameters to account for differences between individuals. Then a collection of individuals, the colony, is defined (a set suffices in this case if individuals are unique). Finally emergence is described. Now it is more subtle, since the behaviour of the colony may result in dynamic reconfigurability: a flock may temporarily divide when confronted by predator and afterwards recombine. Techniques must be developed to describe such reconfigurability elegantly and in a structured manner (but little seems to have been done [27]).

An implementation - at least a design for one - can, as in any distributed system, vary in its degree of distribution. At one extreme it is centralised; at the other it is entirely distributed. A distributed design would incorporate extra information in each individual to account for behaviour of the colony - perhaps with a relatively small amount of randomness in response to the environment. A centralised design—which would seem less appropriate in this case—includes extra components (omnipotence, with access to the state of each individual) to account for the emergent behaviour of the colony. Techniques for the description of distributed designs are well developed in Computer Science.

The criterion for conformance of a design to a specification is again that each behaviour exhibited by the design is allowed by the specification. But sufficient conditions, respecting the modularisation of the ensemble, must be developed.



Figure 3: The shape of a specification of a flock, clarifying the part played by emergence.

3.3 Machine learning: MENACE

Consider, for simplicity, one of the earliest examples of machine learning due to Michie [19]—so early, in fact, that it was implemented with a pile of matchboxes containing coloured beads! MENACE (Matchbox Educable Noughts and Crosses Engine) is a system which learns to play noughts and crosses (a.k.a. tic-tac-toe) by repetition of many games.

MENACE plays O and its opponent plays X; so it suffices to concentrate entirely on plays of O. Initially the board is empty with O to play. Taking into account symmetrically equivalent positions, there are three possible initial plays for O. The state of the game consists of the current position of the board. There is no need to augment that with the name, O or X, of the side playing next since the board is considered only when O is to play. All together there are some three hundred such states; MENACE contained a matchbox for each. In each box were beads to represent the plays O can make from that state. At most nine different plays are possible and MENACE encoded each with a coloured bead. Those which cannot be made (because the squares are already full in the current state) were removed from the box for that state. That provided MENACE with a built-in knowledge of legal plays. (In fact MENACE could easily be adapted to start with no such knowledge and to learn it.)

O's initial play is made by selecting the box representing the empty board and choosing from it a bead at random. That determines O's play. Next X plays. Then MENACE repeats its method of determining O's next play. After at most five plays for O the game ends in either a draw, a win for O or a win for X. Now that the game is complete, MENACE updates the state of the (at most five) boxes used during the game as follows. If X won, then in order to make MENACE less likely to make the same plays from those states again, a bead representing its play from each box is removed. If O drew, then conversely each bead representing a play is duplicated; and if O won each bead is quadruplicated. Now the next game is played.

After enough games it becomes unlikely or even impossible for the random selection of O's next play to produce a losing play. MENACE has learnt to play (which, for noughts and crosses, means never losing). The initial state of the boxes was prescribed for MENACE. Assume merely that it contains sufficient variety of beads for all legal plays to be made; for then the frequency of beads affects only the rate at which MENACE learns.

The state of MENACE (as distinct from the state of the game) consists of the state of each box, the state of the game and the list of boxes which have been used so far in the current game. Its transition rule consists of the probabilistic choice of play (*i.e.* bead) from the current state box; that evolves as the states of the boxes evolve.

MENACE can be considered at various levels of abstraction. At the specification level the 'learning' mechanism of the boxes is abstracted and the components of the ensemble are games in which the choice of beads is random. The state of the ensemble consists of a sequence of games, and the probability of MENACE winning is emergent. In other words its adaptiveness, improving its performance as the sequence of games increases, emerges.

The description above of MENACE provides an implementation. At this level of abstraction the update of the contents of the boxes becomes visible and so the tendency to win is no longer emergent: it is derivable from the system given. In other words, the behaviour that was previously emergent is now captured in the functional behaviour of the system (though necessarily conditioned by environmental run-time behaviour).

This example makes two points. Firstly that machine learning (ML) [20] may be specified as the combination of standard functionality together with emergence. Such specifications will greatly improve the intelligibility and reuse of ML systems. Secondly, as for the other examples, that emergence depends on the level of abstraction at which a system is considered. (That distinction, between being able to observe code or not, is essentially the distinction between open source and proprietorial software.)

It is to be emphasised that the three examples have abstracted different behaviours. Birds, for instance, in fact exhibit probabilistic and learning behaviour on top of that considered here. Adaptability is to be considered not just in the initial phase of operation of an ensemble, but to guide component agents back to satisfying the stable ensemble state (and in particular to satisfy emergence).

By considering this simple example of ML, the topic of artificial agents has been broached. Some of the most interesting ensembles are expected to consist of such agents, whose deviation from 'ensemble fitness' may be interpreted, in this approach, as deviation from emergence; and whose conformance may be interpreted as their return to fitness. This may be achieved by techniques from ML, though the result is a form of self stability.

In the case of ensembles of artificial agents, the emergent behaviour can be viewed as an ethical constraint on agent behaviour, as is now considered.

4 Emerging ethics

4.1 Ethics without free will?

An important case of a multi-agent ensemble is that in which the agents are artificial. Whilst there is no (mathematical) definition of that term (it is ‘agent’ which is contentious, not ‘artificial’!), it seems to be accepted that such an agent must be interactive, autonomous and adaptable [8]. In other words, only very special programs are agents. A typical example is provided by reactive software considered at a level of abstraction at which, typically by employing machine learning and making probabilistic choices, it adapts to interactions with its environment. Reconfigurability may be a particular feature; but anyway as a result of adaptability, the ensemble exhibits emergence.

It is important that such systems be specified. Otherwise their behaviour as they adapt is unpredictable. But there seems to be almost no experience of that: such systems seem to be considered merely at the implementation level.

In society, an ensemble composed of sentient agents, such emergence can be seen as the result of either laws or ethical principles. When the agents are human (subject to the usual exceptions involving mental immaturity due either to youth or mental state) and so exhibit free will, the field of Ethics provides normative principles by which that dynamic multi-agent ensemble, society, functions within the desired tolerances. But those principles (like deontologism, consequentialism, *etc.*) rely entirely on the agent possessing free will; and they tend largely to focus on the individual.⁴ So in its absence, for example in a multi-agent ensemble composed of artificial agents, an alternative foundation must be provided: new normative principles must be developed which do not depend on the agents possessing free will and which apply squarely to systems.⁵ Recently the assumption of free will has been weakened that way, and the corresponding (weaker) theory of ‘information ethics’ (or ‘artificial ethics’) investigated by the Information Ethics Group at Oxford [11].

⁴The ethical platforms of various companies and organisations make interesting reading; they all seem to be strongly influenced by individual (human) ethics, even to the treatment of take-overs.

⁵In so far as laws carry over to the artificial case, they are readily specified as functional properties, to be satisfied like any safety property. (Do any laws represent liveness properties?)

Such principles will at first seem strange from the point of view of Ethics, for precisely the reason that they are not founded on free will. In many cases they do not look particularly ‘ethical’, pertaining instead simply to functionality of the multi-agent ensemble. But their utility is to be measured by the way in which, like the principles of standard Ethics, they enable behaviour of agents (collective behaviour, in the case studied here) to be specified, implemented and analysed. They ensure fitness of agents.

‘Ground zero’ of such principles for multi-agent systems is the ‘principle of distribution’ [26] according to which each system action should be as distributed as possible. It finds application in many distributed systems, including those from socio-economics and politics! The case of an individual (artificial) agent is considered in [8]; further examples appear in [29]. Indeed ethical considerations, when interpreted in this suitably abstracted manner that does not involve free will, play an important part in motivating the designs of many distributed systems [27].

4.2 Emergence and conformance

Section 3 has demonstrated that it is both appropriate and convenient to specify an ensemble as a conjunction of components augmented by an emergence predicate (by taking the conjunction of several such). More generally an ensemble might be naturally expressed in terms of further ensembles. It would be of interest to consider laws that permit transformation to the ‘canonical form’ of a single ensemble.

It would be of interest to consider various kinds of emergence. There might be a temptation to say what an ensemble ‘ought’ to do. Indeed it is to be expected that deontic logic will provide an important notation for expressing types of emergence. But it must be stressed that a property is only of use if conformance to it can be established. It is as well to be specific, since new techniques are to be expected in Ensemble Engineering:

A specification is a system description against which conformance can be decided.

Such is the case, for instance, for a large family of probabilistic properties based on expectations. It might at first be thought otherwise, on the grounds that any finite behaviour is consistent with a given frequency being attained in the limit. In fact the highly successful theory [18] has been assumed implicitly in the examples in Section 3.

But for deontic logic the position seems to be far from successful. There appears to be no denotational semantics and the only (Kripke) semantics already assumes on the possible worlds a semantic notion of duty. Thus it is at present of little help to say, without further clarification, that an ensemble’s emergent behaviour is that it ought to perform some action.

Having decided a denotational semantics for ensembles exhibiting emergent behaviour, laws and criteria for conformance will be important.

5 Conclusion

The importance of societal engineering and ensemble engineering seems assured. This position paper has considered one aspect of both: the place of emergent phenomena in such systems and the manner in which it can be formalised and implemented. That seems sufficient to justify the following research agenda whose purpose is to clarify the importance of emergence. The result is expected to be a foundation for Ensemble Engineering in the Interlink programme.

1. *Description.* Developing notation for specifying ensembles and for expressing designs of ensembles. The use of ObjectZ here is a first attempt. What emergence predicates arise and how are they best expressed? The use of statistics demonstrated here seems fairly rudimentary. Is deontic logic useful? (And if so, does it have a denotational semantics that can be used for the verification of laws, as has been done for probabilism?) What specific emergent clichés arise (like distribution has here)? What intermediate designs arise in ensuring that a component agent conforms to emergent behaviour?
2. *Conformance.* Give criteria, and practical sufficient conditions, for one design to conform to a specification or another design. This necessarily includes a semantics for the notations developed in the previous part. For the standard and probabilistic descriptions used here that has already been done via ObjectZ and the probabilistic guarded command language *pGCL*; it nonetheless poses some difficulties due to the interaction between probabilism and nondeterminism. But for other emergence conditions? The semantics makes available sound laws, which justify the extent to which a conjunction of ensembles may be reduced to a single ensemble. That should be investigated.
3. *Case studies.* Consider a range of case studies, representative of realistic ensembles. Important examples include agent-based ensembles, dynamically reconfigurable ensembles, machine learning, and designs that account for emergence with varying degrees of distribution. In particular the specification of the environmental emergence exhibited by adaptive (machine learning) systems seems both interesting and important; is it in essence different from other kinds of emergence? Vital here is the ‘self-stability’ of such dynamic ensembles to return to their ‘stable’ system state after perturbations.
4. *Model checking.* Show that some of the case studies conform to their specifications by automated verification of the sufficient conditions established above.

More speculative topics, that are nonetheless of interest, include:

1. *Continuous ensembles?* With a very large number of similar components, is there a place for reasoning as if the ensemble were infinite and then using an approximation for large finite ensembles? That would permit the standard theory of differentiability to be used to reason about the limiting case, and afterwards use a discrete approximation to infer behaviour of the ensemble in hand.. If such an approach is of use, what is the place of hybrid ensembles?
2. *Game theory* In ‘strategic’ ensembles, whose component agents compete for advantage, it is to be expected that the best theories available for describing emergent behaviour are game theoretic. It would be interesting to have a realistic but feasible case study of this kind.
3. *Artificial ethics?* Is it useful to pursue the idea of the ethical responsibility of artificial agents, and to use emergent ‘ethical’ qualities in specifying them?

Are these topics already being considered? Do they seem relevant to the the concerns of societal and ensemble engineering? Is there any interest in working on them?

6 Acknowledgements

The authors are grateful to all participants in *Interlink WG 1: Software intensive systems and new computing paradigms* for the lively and productive discussions surrounding this topic, and particularly to Martin Wirsing’s deft guidance.

References

- [1] R. A. Brooks. Intelligence without representations. *Artificial Intelligence*, **57**:139–159, 1991.
- [2] <http://www.aridolan.com/ad/Alife.html>
- [3] G. Brown and J. W. Sanders. Lognormal Genesis. *Journal of Applied Probability* **18**(2):542–547, 1981.
- [4] P. Cariani. Emergence and artificial life. In [16], 775–797, 1991.
- [5] R. I. Damper. Emergence and levels of abstraction. Editorial for the special edition on ‘Emergent properties of complex systems’. *International Journal of Systems Science*, **31**(7):811–818, 2000.
- [6] R. Duke and G. Rose. *Formal Object-Oriented Specification Using Object-Z*. Macmillan Press, 2000.
- [7] L. Floridi and J. W. Sanders. The method of abstraction. In *Yearbook of the Artificial, vol. 2, Models in Contemporary Sciences*, M. Negrotti (ed.), Peter Lang, 2004.

- [8] L. Floridi and J.W. Sanders. On the morality of artificial agents. *Minds and Machines*, **14**(3), 349–379, 2004.
- [9] M. Gell-Mann. *The Quark and the Jaguar*. W. H. Freeman and Company, 1994.
- [10] H. Hendriks-Jansen. In praise of interactive emergence: or why explanations don't have to wait for implementations. In [14], 282–299, 1989.
- [11] Information Ethics Group, University of Oxford.
<http://web.comlab.ox.ac.uk/oucl/research/areas/ieg>
- [12] D. E. Knuth. *The Art of Computer Programming*, volume 2, Seminumerical Algorithms. Second edition, Addison-Wesley, 1981.
- [13] A. N. Kolmogorov. C.R. Dokl. Acad. Sci. URSS **30**:301-305, 1941.
- [14] C. G. Langton, editor. *Artificial Life: Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*. Santa Fe Institute Studies in the Sciences of Complexity, Proceedings 6. Addison-Wesley, Redwood City, California, 1989.
- [15] C. G. Langton. Artificial Life. Reprinted in [?], 39–94, 1996. Updated version the original of which appeared in [16].
- [16] C. G. Langton, C. Taylor, J. D. Farmer and S. Rasmussen, editors. *Artificial Life II*. Santa Fe Institute Studies in the Sciences of Complexity, Proceedings 10. Addison-Wesley, Redwood City, California, 1992.
- [17] W. Li. Random texts exhibit Zipf's-law-like word frequency distribution. *IEEE Transactions on Information Theory*, **38**(6):1842–1845, 1992.
- [18] A. K. McIver and C. C. Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer Monographs in Computer Science, 2005.
- [19] D. Michie. Trial and Error. *Penguin Science Survey 1961, part 2*, Ed. S. A. Barnett and A. McLaren, 129–145. Pelican books, 1961.
- [20] T. M. Mitchell, *Machine Learning*. McGraw Hill, 1997.
- [21] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- [22] E. Nagel. *The Structure of Science: Problems in the Logic of Scientific Explanation*. Routledge & Kegan Paul, London, 1961.
- [23] H. H. Pattee. Simulations, realizations and theories of life. In *The Philosophy of Artificial Life*. M. A. Boden, editor. Oxford readings in philosophy. Oxford University Press, Oxford, 379–393, 1996.
- [24] M. Pauly. *Logic for Social Software*. PhD. thesis, CWI Amsterdam, 2001.
- [25] S. C. Pepper. Emergence. *Journal of Philosophy*, **23**:241–245, 1926.
- [26] G. M. Reed and J. W. Sanders. The principle of distribution. To appear, *JASIST*, 2007.
- [27] J. W. Sanders and M. Turilli. Dynamics of Control. *Theoretical Advances in Software Engineering 2007, TASE2007*. IEEE Computer Society, 440–449, June 2007.

-
- [28] The Stanford Encyclopedia of Philosophy.
<http://plato.stanford.edu/entries/properties-emergent>
- [29] M. Turilli. Ethical protocols design, *Ethics and Information Technology*. **9**(1):49–62, March, 2007.
- [30] M. Wirsing (working group leader). InterLink WG 1 Interim Management Report (IMR), June 2007. WG 1: Software intensive systems and new computing paradigms.