Rule Migration: Exploring a design framework for emergence

HEATHER R. TURNER*, SUSAN STEPNEY, FIONA A. C. POLACK

Department of Computer Science, University of York, UK

Received 31 October 2005

We propose a framework for engineering emergent behaviour that allows system specification and design at the level of the emergence. We discuss how rule migration can be used to translate this high-level multi-layer design (*mobile process* model) into an equivalent simple *cellular automaton* model with only local rules, and emergent behaviour. In this paper, we use a case study in 2D to illustrate the process involved in deriving a migration rule for a simple random walk. We discuss how the mobile process architecture may be extended to model systems with multiple levels of emergence.

Key words: Rule migration, engineered emergence, cellular automata, mobile processes, occam-pi, random walk.

1 INTRODUCTION

To secure the future of practical applications of complex emergent systems, research needs to establish appropriate engineering principles. Engineering is a quality-enhancing activity, and is essential for the safe exploitation of emergence in nature-inspired computational systems; the engineered emergent system would be robust, with assurance of functionality and safety. In exploring emergent systems engineering, we need to consider architectural issues, as well as developmental issues such as compositionality and refinement.

* email: turner@cs.york.ac.uk

We consider complex emergent systems, which comprise many simple components. Often-cited examples of complex emergent systems include network navigation by ants (real or simulated), construction by termites, swarming and flocking, for example by birds or their simulated equivalent, boids. In this work, we focus on systems akin to cellular automata (CAs).

In developing a system, several levels of description are needed. For example, separate behavioural descriptions are needed for individual components and for larger structures or aggregates. In an *emergent system*, there is a discontinuity in the descriptions of the various levels. For example, in CAs the low-level components are described as changing state, whereas the system description might be in terms of the movement of patterns. The higher, system, level typically describes the required emergent properties.

In this paper, we use a simple case study in 2D to illustrate the process involved in deriving a migration rule, enabling high-level system specifications to be migrated into low-level system implementations with the required emergent behaviour.

2 CONTEXT AND RELATED WORK

Elsewhere [8], we describe a generally-applicable system architecture to underpin engineering of complex emergent systems. We identify *three key elements*: the high-level description of the required system; the specification of the components that form the lowest level of the system; and the specification of the representation that integrates the first two elements. Conventional development approaches, relying on a linear reduction in non-determinism (data and process refinement; model-driven development, etc) are applicable within each element, but the low-level system components cannot be systematically derived from the system specification. The components are fundamentally different from the overall system, and cannot be described using the same language concepts.

Subsequently [9], we explore the extraction of a layered component model, derived from pure CA models, and use this to deduce some characteristics of causal linkage among the system elements. We show how macro-scale descriptive layers can be used to express control, motion and relative characteristics in appropriate models or language, whilst appropriate micro-scale models can be used at the component layer. We can introduce and experiment with models related to the proposed system architecture, such as models of environmental interaction (at the macro-level) and of local chemical diffusion and other features of the immediate context of components (at the

component level). Lower-level features are monitored by the higher-layer structures, which also communicate control signals to the CAs.

In engineering terms, we use layering and causality to devise architectural patterns for the design of emergent systems. This will allow the incorporation of good engineering practices, such as validation, testing and safety argumentation to the development process associated with this layered architecture.

2.1 Our background research

To test the validity of the ideas behind our postulated system architecture and causality framework, we are exploring two CA-like applications. The first, based on a pure CA, relates to the tracking of emergent patterns such as gliders. The second, and more detailed application, is a very simple model of artificial blood platelets (see [9] for details).

As part of a wider research exercise[⋆], we initially focused on a one-dimensional (1D) representation of the artificial blood platelets model. A pure CA was used to model the movement and aggregation of platelets[†]. Probabilistic CA rules determine whether an update rule fires in a particular situation. This capacity was subsequently *migrated* to a higher level. [9] describes a model for the resulting two-layer system, using *mobile processes* (MPs) to perform higher-level monitor and control functions on the sites of the "CA" that hold platelets.

The engineering advantage of a model that uses layering, such as using MPs to control movement and aggregation, is that a model of each layer can be built using the abstractions and concepts appropriate to that layer. This has the potential to improve both modelling and verification of models. Interaction between layers is essentially in terms of control signals (passed from higher to lower behaviours) and state information (from lower to higher levels).

2.2 Extending the case studies

We are working towards models of artificial blood platelets in three dimensions, that are capable of representing the local environment of blood vessels, and aspects of the global environment such as vessel constriction and wounding. In this paper, we extend the *rule migration* aspect first presented in [9] to a 2D scenario.

⋆ The TUNA collaboration, EPSRC grant EP/C516966/1

† Specified and analysed in CSP by S. A. Schneider and H. Treharne, as part of TUNA; with A. Cavalcanti and J. C. P. Woodcock, TUNA has also produced CSP models for more elaborate platelet models.

To explore the process of rule migration, we present two simplified versions of the 2D platelet model. The first is a 2D CA. The second is a two-layered MP model. We explore the relationship between the two models, and establish their equivalence, constituting a migration rule between the two models. This illustrates how we can devise engineering procedures for mapping designs expressed in terms of higher-level emergent requirements into lower-level component rules.

2.3 Related work

The hierarchical structure of emergence has been identified and discussed by many researchers. A recent literature survey by Fernando [4] provides a useful introduction to both the theoretical and experimental research in this area. He focusses on *hierarchical complexity*, and a study of the modelling constraints that are necessary to evolve this structure.

There appears to be a limited amount of material available concerning higher levels of emergence. Mayer and Rasmussen [7] also use a variation on a CA in their *Lattice Molecular Automaton*. The state and dynamics of this model are highly complicated, and the rules are derived from the laws of physics. The authors claim that this detail is necessary to achieve the higher order structures. It would be interesting to discover if we can derive a similar model from the top down, using our MP architecture, and rule migration.

Another researcher who uses a multi-layer architecture is Capcarrere [2]. His model, Phuon, is constructed from two layers, the active cellular layer, and a passive environmental layer. Those layers correspond quite closely to the layers in our MP architecture, and we have chosen to name our lower layer in the same way for consistency. That model particularly considers the simulation of growth and development, but in addition incorporates motion at the cellular layer.

Other authors have considered how to model movement using just CA rules, although the explicit detail of how the movement is achieved in terms of the local rules is not always clear. Goel and Thompson [5] use movable finite automata in a specific biological case study. Wolfram [13] considers motion in a CA at the macro-level in terms of its ability to model fluid dynamics. That paper concentrates heavily on the mathematics of the macro-level behaviour, and the underlying rules for movement are not entirely clear.

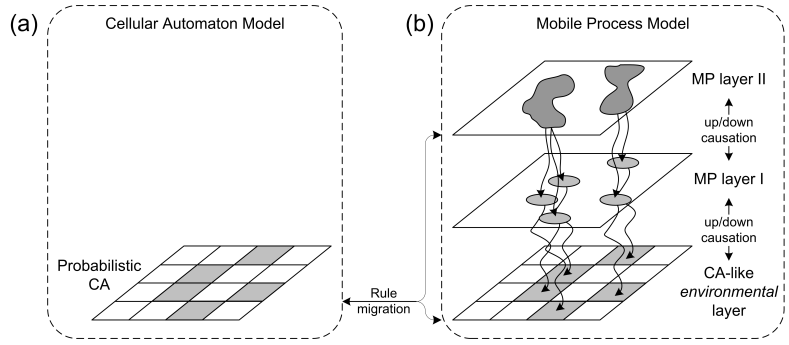

FIGURE 1
Relationships between and within (a) the CA model, and (b) the MP model.

3 RELEVANT MODELS

There are three distinct kinds of mobile process model that use rule migration. It is important to clarify the differences between them, and show how they relate to one another in the context of this research.

3.1 Upward versus downward causation

Our MP models have different levels of description. If all the behaviour is a result purely of the lowest level of description, this is *upward causation*: the upper layers merely record emergent properties of the lower layers, and have no causative influence on the model.

If, on the other hand, some of the behaviour of the lowest level is caused by higher level entities, this is *downward causation*: the upper layers are actively engaged in the behaviour. Whether or not this happens in physical systems is moot: it is certainly relevant in design models.

3.2 Tagging emergent properties

Mobile processes may be used to *tag* identified emergent properties in a model. For example, in a CA, a MP could be attached to a configuration of 'on' cells that appear to be 'moving' across the CA grid, thus identifying them as a glider, and capturing the emergent property of motion. This glider can then be considered as an entity in its own right, and discussed at a higher level than the CA rules. When used in this tagging way, the MP does not control the behaviour of the glider; all the rules are applied at the level of the

CA cells. The MP simply tags the consequences of emergence, responding to *upward causation*.

Given mechanisms for the automatic identification of gliders, and other objects in CAs, a MP layer would then provide a useful platform for discussing higher-level interactions between these entities. Additional MP layers could be added to identify any emergent behaviour resulting from these higher-level interactions. Figure 1 shows how the CA model (a) can be tagged as a three tier MP model (b), by tagging an environmental layer with two MP layers.

3.3 Upward rule migration

Once emergent properties have been *tagged*, we may discover aspects of the system behaviour, or elements of the control structure, that are easier to describe at the level of the MPs than in the low-level CA rules. We may want to modify the behaviour by adding new control structures, or just to simplify the model: the CA layer can have simpler states if it does not need to record decisions about the direction of movement (see Section 4.2). This can be achieved by migrating some of the rules upwards, from the CA to the MP layer. The modified system uses *downward causation*, allowing MPs to exert some control over the low-level environmental layer behaviour. The control might, for example, manifest itself as 'permission giving', or keeping track of group membership, or perhaps adding probabilistic choice.

Figure 1 also illustrates upward rule migration, as it is feature tagging, plus downward causation.

3.4 Downward rule migration

In contrast to the applications described above, downward rule migration is used in the *engineering* of systems with emergent behaviour. The tiered structure enables models to be designed at a level where the rules are simple, and easy to define, giving a multi-layered design using abstractions, and concepts appropriate at each layer. Such a model is not directly implementable as a pure agent system or CA system, however, requiring as it does potential global control and communication to implement the downward causation. Migrating the rules *downwards* from higher layers into the lower layers can then create an equivalent implementable system with only upward causation, in which the MPs provide only *tagging*. This structure is also illustrated by Figure 1.

3.5 The challenge

Developing the framework to enable downward rule migration is the main focus of this paper. Here, we take an initially layered system, and implement

it with a CA. For most real applications, however, migrating the rules down to a CA would be inappropriate, and the lowest implementation level might correspond, for example, to a multi-agent system.

Migrating the rules down to the CA level in this way, will almost certainly result in a CA with a large state space. This appears to oppose one of the key properties of a CA: its ability to produce complex behaviour from a very small state space. However, consider von Neumann's first attempt at a self-replicating CA [12]. His CA had 29 states, and served to prove that such behaviour could be achieved. In the years that followed, many others devoted time and effort to discover simpler CAs that still exhibit self-replication. Moshe Sipper provides an excellent summary of the developments in this area [10], from which we have taken the following key developments: Codd reduced the number of states required to 8 [3]; Langton used a smaller self-replicating structure in an automaton with the same size of state space [6], which was then improved upon by Byl, reducing the number of states required to 6 [1]. Here, we are concentrating on proof-of-concept, rather than efficiency and conciseness.

Reverse-engineering the CA rules which exhibit particular *emergent* behaviour is a challenging task. When we consider how it might be achieved, we expect to require some exploration, and adjustment to the source and target models (see Section 5 for more discussion). It would be impractical to have to perform this from scratch every time a new system is to be engineered, and also redundant, as many different systems will share common aspects of behaviour. Returning to our CA example, *movement* is an emergent property of a CA, and is likely to be required in many different CA models. If we can define an alignment between a MP model of movement, and a CA model of movement, it can be used as a translation rule to migrate between such MP models and their CA equivalents. The same is true for other forms of behaviour such as aggregation, and signalling. Movement itself is a non-trivial form of behaviour, incorporating, for example, diffusion, random-walks, biased random-walks, and deterministic directed motion. As mappings are created between MP models and corresponding CA models, we can add these to a translation dictionary, to facilitate rule migration.

A MP model displaying movement of aggregates can be designed as the three tier model shown in Figure 1b. The upper layer controls the aggregation, whilst the middle layer controls the movement. We hypothesise that rule migration will allow us to migrate all the control down through the layers by applying the appropriate translation rules (for details of how this might be achieved, see section 6).

In the following sections, we discuss a process for deriving the first translation rule, for simple movement. This paper describes one entry in the translation dictionary.

4 CASE STUDY: SIMPLE RANDOM WALK (SRW)

4.1 Motivation

The inspiration for this work on rule migration came from our attempts to model the emergent properties that blood platelets display when they encounter a change to their environment, the blood vessel. Simple behaviours that we identified include biased random movement (carried by blood flow), aggregation (formation of blood clots), and movement of aggregates (clots that have not yet attached to heal wounds). There are also other, more complicated behaviours, including the diffusion of signalling molecules that trigger the clotting behaviour.

The blood platelet case study will eventually use the downward rule migration process. Before we are able to do this, however, we must derive MP model to CA model translation rules for, at the very least, biased random movement, and aggregation. With this goal in mind, we first explore a more general translation rule for the simple random walk (SRW), unbiased random movement. In the descriptions that follow, we use the general term 'agent' to refer to entities that are perceived to 'move' through an environment. In the artificial blood platelet model, the agents would be the platelets.

The simple random walk on a square grid allows an agent to move to any one of its eight neighbouring sites with equal probability. The situation is made interesting by the presence of multiple agents; these act as obstacles to movement, and can compete for environmental real-estate.

In the subsections that follow, we discuss two different models of the simple random walk by agents on a 2D square grid, one a CA, one an MP model. The models have already gone through a process of alignment (see Section 5.1), and are therefore equivalent.

4.2 Cellular automaton model: emergent movement

The CA model of a SRW adheres to many of the properties that we consider to be the defining characteristics of a CA. Space is modelled by a regular grid of sites that update in synchrony at discrete time intervals. Each site is in one of 17 possible states. The update rules, which define how the sites change state based on their own state, and those of their neighbours, are defined for the Moore neighbourhood of size 9. The automaton is homogeneous, that is,

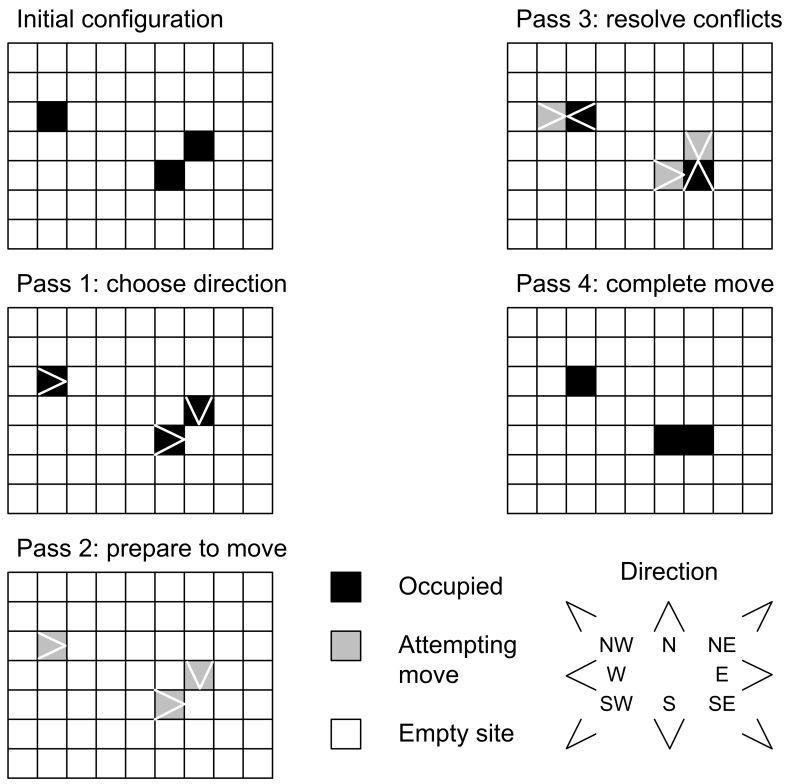

FIGURE 2
One time step in the evolution of the CA. Four different passes update the states of the grid sites, comprising **type**, indicated by the colour of the grid square, and **direction**, identified by the arrows. By applying the simple local rules, we simulate observable random motion.

all the sites in the grid use the same rules. In contrast to the usual definition of a CA, the rules are probabilistic.

The rules for simple CAs are usually given as a lookup table, mapping the current state of a site's neighbourhood to its corresponding next state. For this CA, with 17 states (illustrated in Figure 2; direction is not used in empty cells) and a neighbourhood of size nine, there would be 17^9 ($> 10^{11}$) entries in the lookup table. Instead the rule is described as a compressed algorithm.

The 17 states of the CA are derived from two variables, **type** with three

values, and **direction** with eight (not all combinations are possible). The **type** can take the value *black*, *white*, and *grey*, corresponding to the concepts *occupied*, *empty*, and *attempting move*, respectively. **Direction** indicates N, NE, E, SE, S, SW, W, and NW, corresponding to neighbouring sites in each of the compass positions. The value of **direction** is irrelevant when **type** is *white* (empty). The different states are shown in Figure 2 by combining the appropriate symbols from the key to the diagram.

The occupied sites represent the agents. They have the ability to choose their desired direction of motion, based on the availability of empty sites in their neighbourhood. CA sites can update only their own state, and so have no direct means of telling the chosen destination site that it has been selected. A more indirect approach is needed. We use a series of *passes* in the CA rule to allow the sites to indicate their preference, verify selection, and perform the necessary state updates.

We now describe in detail how this strategy can be employed for our SRW model. Figure 2 shows a possible initial configuration for the CA grid, with different snapshots, or passes, identifying different synchronisation phases of the CA rule.

At the beginning of each time step, all sites are either *black* (occupied), or *white* (empty). Each *black* site (agent) is given the opportunity to try to "move". It first evaluates the states of the neighbouring sites, to establish a current view of its surroundings. The agent may move only to a cell that is empty during this evaluation. Having determined the availability of empty sites in the eight neighbouring directions, one is selected, with uniform probability, if any are available. The *black* site indicates its chosen destination by setting its direction state appropriately. This is the situation shown in Figure 2 as Pass 1.

The agent cannot just *move* in the chosen direction, as that would involve updating the state of its neighbouring site (forbidden by CA rules). In Pass 2, the *black* sites change their **type** to *grey* to indicate that their contents are ready to move.

In Pass 3, the *white* (empty) sites evaluate their neighbourhood, and determine whether any agents want to move into them. They evaluate both the **type** and **direction** of their neighbouring sites, counting any prospective new occupants. They select with uniform probability any neighbouring agent wanting to move, and set their **type** to *black* (indicating the move into this site), and **direction** to point at the selected agent. Otherwise the site is unchanged.

In Pass 4 the *grey* sites update to indicate whether the agent successfully moved, or whether it failed at this particular time step. Each grey site checks

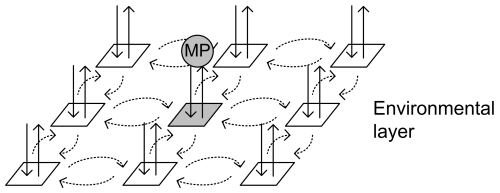

FIGURE 3
Mobile process model architecture. State information at the environmental layer reflects the state of the MPs in the upper layer.

that it has one *black* neighbour with its **direction** pointing back at the *grey* site. If so, the *grey* site changes its **type** to *white*, thus completing the agent's move out of this site. If the neighbouring states indicate that the move is not accepted, the *grey* site changes back to *black* (no move occurs); the agent can attempt to execute a move again in the next time step. The direction is no longer relevant.

The four pass structure of this algorithm is used for clarity; it is equivalent to one single rule that includes extra states that identify which pass the system is in.

The "movement" of the agent is an emergent property of this CA. Nothing has actually moved: sites merely change state. The rules have to be carefully designed to yield this emergent movement and conservation of agents. Deducing the rules for a non-trivial 2D CA is a complicated and time-consuming task. The emergent motion has to be reverse engineered into state-change rules for neighbourhoods, with all possible contingencies accounted for: the CA must not lose or gain agents.

4.3 Mobile process model: controlling the emergent movement

The MP model for this SRW example is built on a 2D cellular array, the lower layer. Sites occupied by agents are "tagged" with mobile processes at an upper layer. Some of the logic resides in the lower cellular layer, which we call the environmental layer after [2], and some is captured by the MPs, making the movement easier to model. This is an example of downward causation (because the MPs affect the behaviour of the environmental layer).

An important aspect of the two layer architecture is the ability of each layer to abstract away detail. Communication channels act like sensors into the environment, providing a means by which the MP can sense and interact

with its neighbourhood. The environment contains state information indicating whether each site has active connections to upper layer processes. In Figure 3, which illustrates the architecture, each MP represents one agent, and is connected to precisely one site at the environmental layer, so there is a direct equivalence between sites representing agents and processes representing agents: the MPs "tag" the agents in the environmental layer. Each site is connected to its eight immediate neighbours by two communication channels. Each site has two further channels, which can be connected to by any MP at the upper layer. A communication protocol permits the transmission of state information associated with neighbouring sites. An MP can move from one site to a vacant neighbouring site by dropping its channel connections to the environmental site, and picking up those of the neighbour. An implementation is described in [11], Appendix A.

At each time step, the MP tagging each agent submits a request to its environmental site for information about its neighbourhood. In this model, the MP performs the probabilistic selection of movement direction. To do this, the site at the environmental layer counts the available neighbouring sites and reports back to the MP. The MP then chooses one with uniform probability, and sends the 'move' request to its environmental site. The MP attempts to move to the chosen neighbouring environmental site, using a mechanism similar to that described in [11], Appendix A. If the movement is unsuccessful, because, for example, another process is chosen to move to the site, the MP will be unable to move. At this point, information about agent locations in the two layers is no longer equivalent: the environmental layer must update. Sites that have lost connections become *empty*, whilst those that find their channels newly connected become *occupied*.

We can compare this MP model to the corresponding CA model. In the MP model the tagged identity of the agents, and their explicit movement provides simplicity. In the CA model, movement is an emergent property: it is not explicit in the context of the rules, making them hard to engineer. In addition, a large number of states are required to perform the signalling needed to resolve the conflicts occurring with the probabilistic choice.

5 RULE MIGRATION

In the previous section, we describe two models of the SRW on a square grid, one a pure CA, and the other a MP model. The two implementations are equivalent, as we show here. At the beginning of a design process, models are derived independently, so are unlikely to have exactly the same behaviour. We

explore how the differences in the models' behaviours are resolved during the alignment process, and show the similarities and the differences in the final, aligned implementations.

5.1 Design and alignment

The first step in the design of a basic translation rule is the derivation of the two relevant models (here the MP model and CA model of SRW). The next step is to make the two models functionally equivalent, that is, produce the same result at each main time step when given the same initial configuration. This process can be seen as an alignment operation. We work from the MP model, and from the CA model, and make any necessary modifications and adjustments until the two models can 'meet in the middle', and be equivalent.

In our *initial* design of the MP model (which differs slightly from the implementation described in the previous section), the agent, as a MP, appears to have an identity, and it seems reasonable to offer 'it' flexibility in the decision making process. When two agents attempt to move to the same location, one is prevented from moving. In the MP model, it is quite easy to provide a 'second chance', and let the failed agent try another direction, at least until it runs out of options. The simplicity is achieved because the MP and its environmental site can freely exchange information without requiring the complicated state changes that would be required in the CA model. In the MP architecture, when the environmental site fails to fulfil a move request, it can send a message to the MP, and ask it to pick a different direction. The MP can reply with a new choice, and another move attempted.

The pure CA model does not have this richer functionality, as it does not fit so naturally with the CA style of rules. In the process of aligning these two models, it is simpler to scale down the functionality of this more complicated MP model. The 'second chance' for the MP is removed from the model, and the alignment is complete. This is the situation in the algorithm described in Section 4.3.

5.2 Comparing the models

The result of the alignment process is shown in Figure 4. It illustrates how the equivalent functionality is achieved by different means in each algorithm. The details are discussed below. To further clarify what we consider to be equivalence, the configuration of the environmental layer of the MP model, and the grid of the CA model, should be equivalent at the beginning and end of each main time step. In the intermediate processing, the CA model makes use of additional state information, and the MP model sends messages via

its communication channels, and changes their connections. Figure 5 shows the different phases of execution of the two models as images of the grids, along with brief descriptions of what is happening in each model, at different stages. The intermediate steps in the different models are not synchronised with each other; they are shown side-by-side to highlight the equivalence of the initial and final configurations. The models are synchronised at each main time step.

In Figure 4, we show the finished product of the alignment process. The two models are functionally equivalent, and we can show where they perform similar operations by different means. In the diagram, numbered arrows link the parts of the algorithm that perform equivalent operations. These are described in more detail below:

1. In both models, the agents' environments are evaluated.
2. In both models the agents select a direction in which to move.
3. In the CA model the agents that are trying to move change state so that they can be distinguished in later passes from agents that have successfully moved. There is no equivalent operation in the MP model.
4. In both models, the agents ascertain whether they were successful in their requests to move.
5. In the CA model, all empty sites in the grid are evaluated to determine whether any agents want to move into them. Many empty cells will not be near agents and so do not change state. In the MP model there is no equivalent as computation is carried out only in the locality of the MPs.
6. Much of the CA algorithm is covered in one step of the MP model. In the MP model, this step corresponds to the *actual* movement, and the communication channels are rearranged at this point. In the CA model, we have the simple case where there is no competition for the destination site and the movement effectively occurs (although the source site must be updated later, as identified by the second arrow marked with a 6). In the more complicated case where there is a conflict, the destination site detects that more than one agent is trying to move into it, and chooses one at random (this is not dissimilar from the random element to the process scheduling that determines which MP moves first). The successful agent *moves* at this point.
7. In both models, the agent did not get to its destination before it was occupied by another. The agent does not move. In the CA model, the agent location from the first pass is reactivated.
8. In the MP model, the environmental layer is updated to reflect the new locations of the MPs. No equivalent operation is needed in the CA model.

We hypothesise that the functional equivalence described here for the SRW can be generalised and used with other behaviours. By considering more examples, and through further experimentation, we propose to find patterns of operation in an MP model that correspond consistently to other patterns of behaviour in an equivalent CA model. When we have built a dictionary of such correspondences (migration rules), we will have accomplished a major step towards an architecture for exploring emergence in CA-like systems.

6 DISCUSSION AND FUTURE WORK

We have shown how to achieve desired emergent behaviour in a CA model, by using rule migration for an MP design model. We describe two levels in an emergence hierarchy in terms of our MP architecture. Then, by applying derived migration rules we build a low-level CA that displays emergent behaviour. The MP model can include moving, aggregating objects, and these can be programmed explicitly. After migration, the rules for the CA model produce the same behaviour, but this will be difficult to discern from the CA rules themselves. The resulting behaviour may also be difficult to explain in terms of the low-level CA rules; it will require description in a higher-level language. We argue that although there is much controversy surrounding the definition of *emergence*, the behaviour just described would be classified as emergent.

Extending this idea, we plan to develop migration rules between MP layers, not just from MP models to CA models. Ultimately we want to be able to model systems with multiple levels of emergence. By describing these systems in terms of MPs at the highest level with downward causation, we can migrate the rules downwards to produce an MP model at a lower level, with upward causation to the highest level. If the lower-level MPs also tag, and influence (by downward causation), behaviour at a CA layer, then the rules can be migrated downwards again, until eventually the rules are migrated to a pure CA model with only upward causation. The MP to CA rule migration is the special case, while the more general MP to MP migration rules will permit us to model, at least in principle, an arbitrary hierarchy of emergence.

Our next step is to generate more migration rules. We will specialise the SRW to provide a migration rule for a biased random walk, and design a migration rule for aggregation. In addition, we will explore the *combination* of simple properties such as movement and aggregation, into a more complete model.

We expect it to be possible during the design process to either combine the

MP properties and then migrate the rules to a CA model, or to perform the rule migration on the individual properties, before combining them. We hope that properly designed rules will have a useful degree of commutativity (see Figure 6), although we do not expect the relationships to be simple.

Once we have a satisfactory dictionary of migration rules, we can apply them to model systems. We will begin with the artificial blood platelets, by attempting to combine movement rules with aggregation rules. We expect to require a three layered structure, the top layer representing blood clots (aggregation of platelets), the next layer representing the platelets themselves, and controlling their movement, and a final layer providing the model for the implementation. The MPs representing clots may need to be connected to (i.e. have control over) *all* of the MPs that make up the aggregate, as shown in Figure 1b, or they may connect only to some *key* MPs. Movement of aggregates may be achieved in the same way that a glider moves across a CA, by translation of a fixed configuration, facilitated by intermediate state changes.

There is much work to be done in this area, and many applications to which we can apply the different forms of rule migration. We believe that the most important contribution of this work will be in the study and design of systems with multiple levels of emergence.

7 ACKNOWLEDGEMENTS

H. R. Turner is funded by an EPSRC and BAE Systems CASE Studentship. The TUNA feasibility study is supported by EPSRC grant EP/C516966/1.

REFERENCES

[1] J. Byl. (1989). Self-reproduction in small cellular automata. *Physica D*, 34:295–299.

[2] M.S. Capcarrere. (2004). An evolving ontogenetic cellular system for better adaptiveness. *BioSystems*, 76:177–189.

[3] E.F. Codd. (1968). *Cellular Automata*. Academic Press.

[4] C. Fernando. On the evolution of hierarchical levels of organisation. Unpublished DPhil literature survey, University of Sussex. Available from: http://www.informatics.susx.ac.uk/users/ctf20/end_of_year_report_sc.ps, accessed May 2005.

[5] N.S. Goel and R.L. Thompson. (1988). Movable finite automata. In C. Langton, editor, *Artificial Life*, SFI Studies in the Sciences of Complexity. Addison-Wesley.

[6] C.G. Langton. (1984). Self-reproduction in cellular automata. *Physica D*, 10:135–144.

[7] B. Mayer and S. Rasmussen. (1998). Self-reproduction of dynamical hierarchies in chemical systems. In C. Adami, R.K. Belew, H. Kitano, and C. Taylor, editors, *Artificial Life VI: proceedings of the Sixth International Conference on Artificial Life*, pages 123–129. MIT Press.

[8] F.A.C. Polack and S. Stepney. Emergent properties do not refine. In *REFINE 2005, BCS-FACS Refinement Workshop, ENTCS*, volume 137, pages 163–181. Elsevier.

[9] F.A.C. Polack, S. Stepney, H.R. Turner, P.H. Welch, and F.R.M. Barnes. (2005). An architecture for modelling emergence in CA-like systems. In *ECAL 2005, LNCS*, volume 3630, pages 471–480. Springer.

[10] M. Sipper. The artificial self-replication page. Available from: http://www.cs.bgu.ac.il/ sipper/selfrep/.

[11] H.R. Turner and S. Stepney. (2005). Rule migration: Exploring a design framework for modelling emergence in CA-like systems. In C. Teuscher and A. Adamatzky, editors, *Workshop on Unconventional Computing*. Luniver Press.

[12] J. von Neumann. (1966). *Theory of Self-Reproducing Automata*. University of Illinois Press.

[13] S. Wolfram. (1996). Cellular automaton fluids: Basic theory. In S. Wolfram, editor, *Cellular Automata and Complexity: collected papers*. Addison-Wesley.

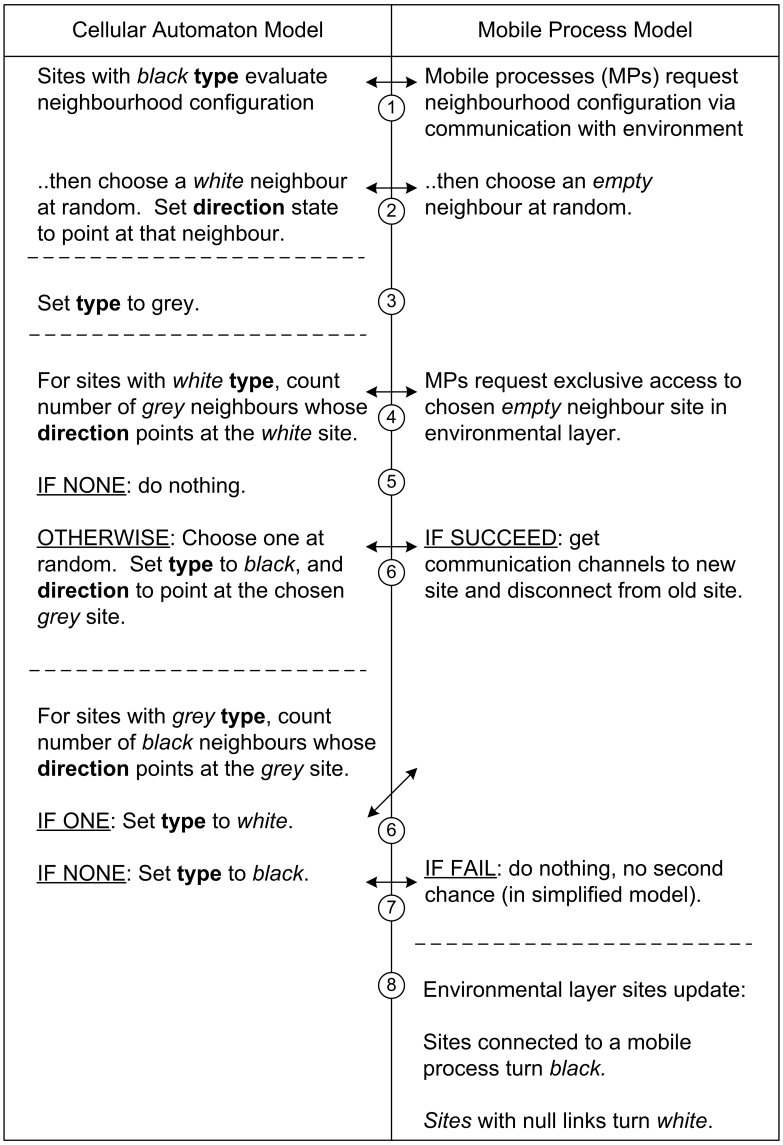

FIGURE 4
The finished product of the alignment process. Numbered arrows show where the algorithmic sub-units are equivalent, see text for more detail.

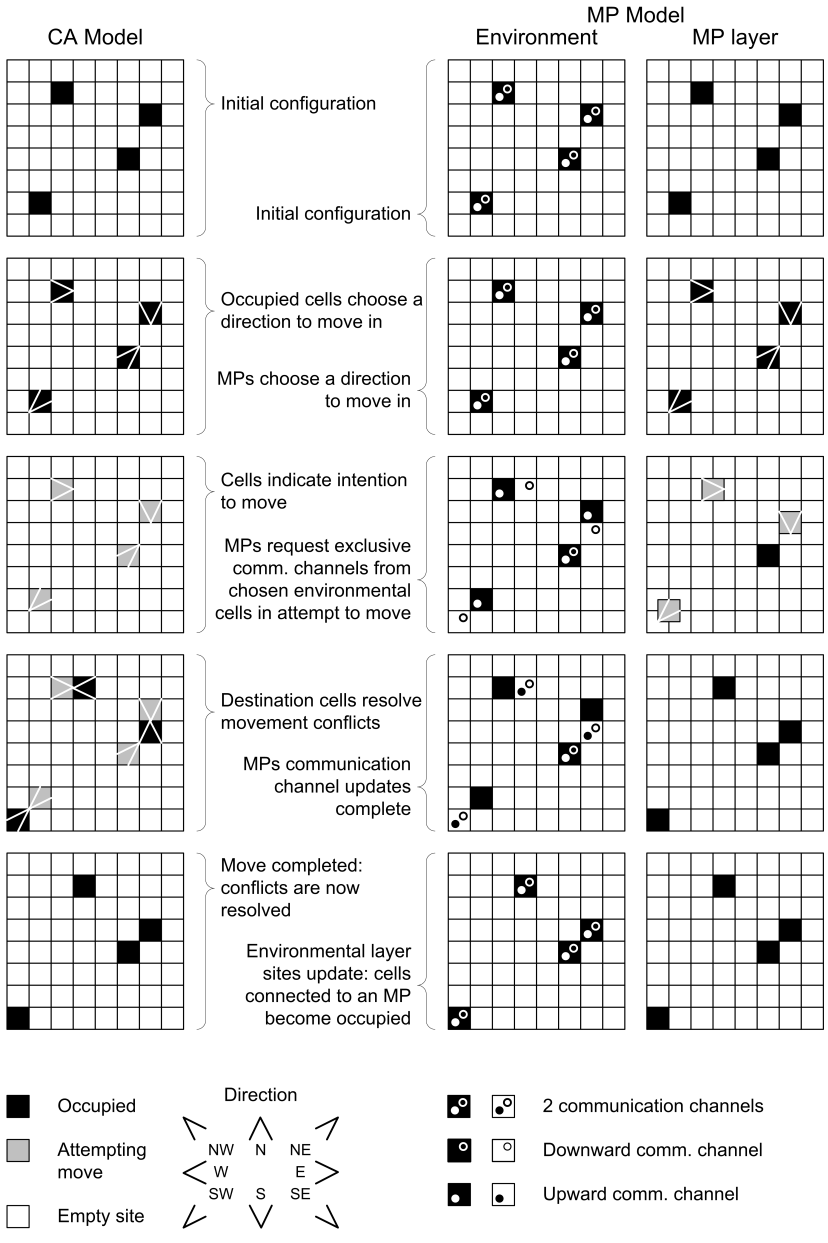

FIGURE 5
The two models are equivalent at the beginning and end of each iteration, but differing, and unsynchronised intermediate processing results in differing internal configurations.

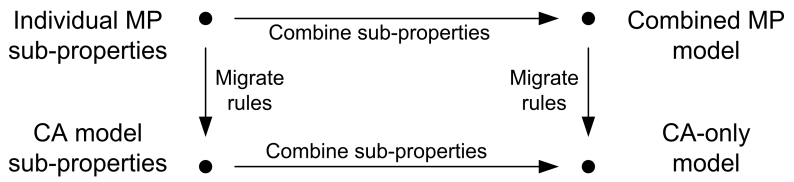

FIGURE 6
The hypothesised commutative nature of the migration and combination procedures.