An Architecture for Modelling Emergence in CA-Like Systems

Fiona Polack, Susan Stepney, Heather Turner, Peter Welch*, and Fred Barnes*

Department of Computer Science, University of York, Heslington, York, YO10 5DD, UK.
*Computing Laboratory, University of Kent, Canterbury, CT2 7NF, UK.
`fiona|susan|turner@cs.york.ac.uk`
`p.h.welch|f.r.m.barnes@kent.ac.uk`

Abstract. We consider models of emergence, adding downward causation to conventional models where causation permeates from low-level elements to high-level behaviour. We describe an architecture and prototype simulation medium for tagging and modelling emergent features in CA-like systems. This is part of ongoing work on engineering emergence.

Keyword: Cellular Automata, emergence, occam-pi , simulation

1 Introduction

This paper represents part of ongoing research to establish engineering principles for complex emergent systems. Various systems require several levels of description; for example, the behavioural descriptions of individual components and of some aggregate. In an *emergent system*, there is a discontinuity in the descriptions of these various layers. For example, the low-level components might be described as changing state, whereas the system description might be in terms of the movement of patterns. The upper, system, level describes the required emergent properties.

We consider complex emergent systems, comprising 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, and cellular automata (CAs).

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 are looking at compositionality and refinement. We start with simple emergent systems, specifically CAs, and derive more general guidance from our observations.

Elsewhere [9], we describe a 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.

In this paper, we explore extraction of a layered component model. The introduced layer maps from the component language towards the concepts of the emergent system. The layering approach explored here is derived from pure CA models; we deduce some characteristics of causal linkage among the system elements. We consider a system requiring emergence of specific gliders, and a case study simulating blood platelets.

2 Cellular Automata and Upward Causation

In a simple CA, such as Conway's Game of Life (GoL) [6], cell update rules and initial cell states completely determine the evolution of the CA. Emergence is detected when each cell state has a visual representation, and the repeated synchronous update of the cells reveals recognisable structures in space and time. When seeking to engineer emergence on such a CA, the three architectural elements are as follows.

1. Required emergent structures, such as gliders, described using relative motion concepts.
2. The CA, comprising many identical cell instances.
3. The representation, discretised space, to define cell neighbourhoods, and on which relative motion can be detected.

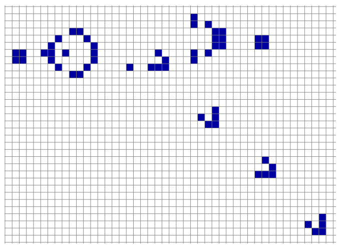

Fig. 1. A 2-D GoL Glider Gun

The CA has an *upward causal relationship* to the required emergence. For example, the upper part of figure 1 shows a GoL glider gun. In this part of the representation, we observe seemingly-random continuously-changing patterns. From the gun, a stream of gliders emerges, moving at a constant velocity, at 45 degrees from the vertical, down the screen. The glider gun is a simple result of applying the GoL rules to cells arranged in a 2-D regular grid, with a suitable arrangement of initial cell states. The high-level description of the observed behaviour of gliders does not have any role in the evolution of the CA; the described higher level behaviours are caused by the lower-level actions.

To be able to engineer emergent systems from high-level requirements, we need a more flexible and realistic causal model. First, we introduce our research case study; we then use some of its models to explore causality further.

3 The Case Study: Artificial Blood Platelets

Our working case study, a platform for specification, simulation and other emergent engineering aspects, is a system of artificial platelets. The desired emergent property is the sealing of breaks (wounds) in a tube or vessel.

The model is loosely based on the medical process of *haemostasis*. Real platelets are passive quasi-cells carried in the bloodstream. A platelet becomes active when a balance of chemical suppressants and activators shifts in favour of activation, usually due to damage to cells or vessel linings. With sufficient stimulus, platelets become sticky and form clusters. This is the first phase in limiting blood-loss and healing a wound.

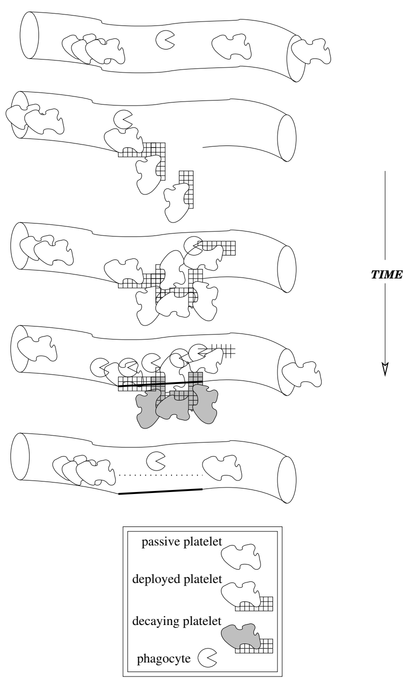

Fig. 2. Schematic of the artificial platelets

Our artificial platelet model, figure 2, assumes that the platelets can complete the entire wound-closing process. Our goal system might resemble Freitas' vision [5] of some 10^8 mechanical platelets of two microns diameter circulating in the blood, each carrying a fibre mesh. At a wound site, the mesh deploys, revealing sticky sections that bind other platelets and seal the wound; when the wound is healed, the mesh disperses.

In this paper, we consider the development of a model of platelet movement and clustering, the basis for a computer simulation. The high-level description is of platelets moved by blood flow through a vessel, with no independent means of locomotion. When platelets merge with other platelets, they form a slow-moving cluster. This description of platelet behaviour is at the same level of abstraction as the high-level glider description.

3.1 Upward Causation Model of Artificial Blood Platelets

Our first platelet model represents a blood vessel as a one-dimensional grid. Figure 3 shows eight time steps of a purpose-built CA running on this representation, to simulate the flow of platelets, the formation of clusters, and the movement of clusters. Here, we see two clusters merging, and then free platelets joining the large cluster from behind.

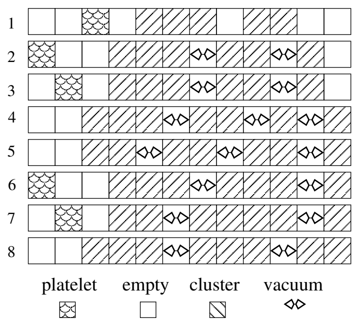

Fig. 3. 1D platelet CA

The CA rules are given in the Appendix. To achieve the required behaviour with pure CA rules, the first platelet in a cluster can move (non-deterministically), creating a vacuum; the cluster moves by successively passing the vacuum backwards. All free platelets move in each time step.

This is a pure CA, with a stochastic rule that determines whether a cluster moves in any step. There is only local communication, and only upward causation from the CA to the required emergent clusters.

4 Downward Causation and Rule Distribution

The simple CA platelet model is not ideal. The rate of movement of a cluster is very much slower than that of a free platelet, at most one cell per update, because cells cannot communicate throughout a cluster. Furthermore, a model where platelet locations control platelet movement by upward causality is not an adequate model of reality; we know that platelet aggregation influences the flow of blood and the flow of blood influences the aggregation of platelets. Such causal links are well-known in biological and other emergent systems [1].

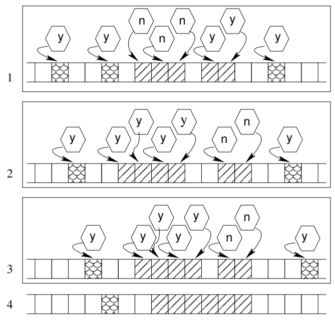

Fig. 4. Two-layer platelets

A revised model retains the one-dimensional CA structure at the lowest level, to simulate discrete physical locations that may contain a platelet. A more abstract level sits above the CA to model clustering characteristics. In this model, only the higher level "knows" that a particular platelet is part of a cluster. Figure 4 shows four time steps. The free platelets still move in every time step; each cluster either moves or stays put, depending on the decision made in the higher layer.

Conventional CA rules still determine whether a cell could change state, but the actual change requires permission to be communicated from the platelet layer, a *downward causation* from the higher level to the CA. The permission is used to co-ordinate the movement of platelets in a cluster, depending on the (non-deterministic) movement of the front platelet.

The higher layer can also be used to add further cluster behaviour, such as cluster breakdown and dispersion.

5 Discussion of Rule Migration

In the platelet models, the glider model [9], and other CA-based systems, the CA model has no inherent awareness of the structures that may emerge. In general, modelling is simplified if there are extra modelling layers that capture concepts expressed in the system-level language. Thus, the gliders are specified in terms of velocity not CA cell states; they can be identified by monitoring at a higher level, over many time steps. Similarly, platelet behaviour (here) is specified in terms of clustering characteristics; clusters are initially identified from the CA, but their persistent characteristics are modelled in the higher layer.

In the platelet model, the clustering rules have been taken out of the CA and migrated to the higher layer; the downward causation (permission) maintains integrity between the layers. We observe that, if the low-level rules are very compact, as in the GoL without the added requirement of structure identification, there may be no rule migration that makes the model simpler. The simplest additional layer provides "tagging", with no downward causation, for example as an aid to the detection of emergence. Thus, in the glider model, a higher level might be used to detect and highlight gliders; this is analogous to experiments in nature that use markers for tracking to collect data. At a level more akin to that of the platelet model, we might then wish to exercise control over, for example, what happens when two gliders collide; this would be accomplished via downward causation from the higher level to the relevant cells of the CA.

In engineering terms, the migrated rules are used to produce more natural, comprehensible models. Whether to choose the pure CA or a multi-level model is a *modelling decision*. The downward causation layers introduce control, and can be used to bring the power of the models closer to the full environmental interactions of real systems. In general, as the number of control aspects modelled in the abstracted layers increases, the behavioural similarities of the single-layer and multi-layer models become less apparent.

5.1 Emergence and Relativity

In physics, there is no absolute space; all motion is *relative*. It is also the case that all emergence is relative. Consider a single GoL glider; viewed from a sufficient distance, a glider moving across the screen is indistinguishable from a screen window being scrolled past a stationary glider. To perceive motion, there needs to be a frame of reference within the window. This could be a visible grid, a stationary (or slower-moving) CA structure, or other gliders. The glider is then seen to be moving *relative* to the other contents of the window.

In migrating CA rules upwards, we often move from an absolute to a relative perspective, taking the design nearer to the context in which the emergence is detectable. We would like to be able to abstract away from artificial representations, to use natural descriptions of these high-level rules. For example, when modelling the layers on top of a CA, we should ignore the absolute grid representation, describing the emergence (gliders and other CA structures) abstractly, and without reference to lower level rules. We can then connect the high- and low-level elements by suitable causation links, to engineer the required emergence. The next section shows how a layered design can be implemented, preserving the relativity of the higher layer and the absolute lower-layer concepts.

6 An Implementation of the Layered Platelet Design

To explore simulations that demonstrate rule migration, we use a mobile extension of a traditional concurrent language. occam-π [1] is a small language that implements the communication strengths of Hoare's CSP[7] and the mobile aspects of Milner's π-calculus[8]. It takes the well-grounded semantics of these specification calculi, and provide a programming environment to support an engineering approach to the underlying mathematics[2, 10].

The implementation of the platelet model uses occam-π static processes to represent the underlying CA, and mobile processes to model the activation and clustering of platelets. Downward causation is programmed as the mobile processes stimulating change in a CA cell. Upward causation is the reading of cell state by the mobile channels.

We can associate various visualisations to the simulation. An absolute-space model can be observed if static processes communicate their location and state to a display. A relative-location visualisation is achieved if mobile processes communicate their size and relative location to a display.

6.1 The occam-π Design

The occam-π model has a one-dimensional cell array, as before. Each cell is a static *server* process.

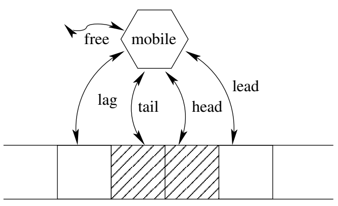

Fig. 5. A two-platelet cluster

One approach to simulating the two-layer platelet model it to associate a mobile process to each cell containing a platelet. The mobile process (figure 5) holds the size of the cluster and *client*-ends of four (multi-way) channels: `head` and `tail` connect to the first and last cells in a cluster; `lead` and `lag` connect to cells immediately ahead of and behind the cluster, acting as *feelers* to the cells round the cluster. The mobile process also holds the *server*-end of the `free` channel, used to merge adjacent clusters. The channel protocols allow two-way communication for the

[1] See http://frmb.org/kroc.html for the latest implementation.

setting and retrieving of data and channel ends. Each cluster deposits the *client*-end of its `free` channel in the cell connected through `tail`. When the cluster moves, the `free` channel is thus dragged along the cells in turn.

When a platelet or cluster `head` enters the cell immediately behind another, the front mobile process (`m1`) detects its presence via an enquiry on the `lag` channel. The back process (`m2`) also detects that it is adjacent to another cluster via an enquiry on its `lead` channel, which points to the same cell as `m1`'s `tail`.

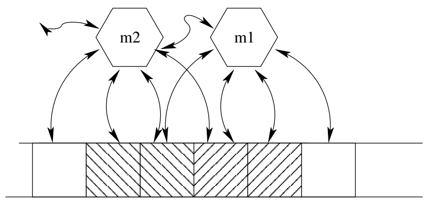

Fig. 6. Two coalescing clusters

Figure 6 shows the results of the `m2` process using its `lead` channel to acquire the client-end of `m1`'s `free` channel, resulting in direct communication between the two clusters. Once the link between the mobile processes is established, `m2` communicates its size, the client-ends of its `tail` and `lag`, and the server-end of its `free` channel to `m1`; `m2` then terminates. `m1` adds the received size to its own size, and overwrites its `tail`, `lag`, and `free` channel ends with those that it receives. `m1` has now assumed control of the combined cluster, shown in figure 7.

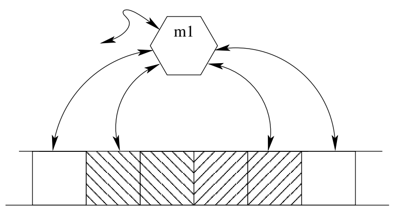

Fig. 7. The completed merger

Coordination of cluster merging and movement is safely and efficiently managed by *barrier synchronisation.* A two-phase cluster cycle is divided by barriers. Phase one detects when one cluster has other adjacent clusters (on one or both sides) and handles all the resultant cluster merges. In phase two, mobile processes determine the movement of their clusters. Barrier synchronisation in occam-π is extremely cheap (see [3]). All memory for terminated processes and discarded mobile channels is automatically freed (without garbage collection); there can be no memory leaks and the model runs indefinitely.

6.2 Extending the Platelet and Glider Simulations

The occam-π simulation allows us to explore the use of higher layers in a CA-based model, and to explore platelet simulation with additional control factors from environmental models. The mobile process tagging will also be used on a GoL simulation to facilitate automatic detection of incipient gliders. An element of downward causation could be added to the GoL, perhaps "clearing" the neighbours ahead of a new glider to prevent its being absorbed by background "noise".

The mobile processes used to tag gliders and link platelets implement relative location (i.e. connections to neighbouring cells); this information is held locally.

Rules at the higher level refer only to relative properties, not absolute properties of the current grid location. The lowest level still uses an absolute grid, but this is accessed only to display the result of each update cycle; individual cells are unaware of their absolute grid location.

7 Related Work

We are not alone in recognising that representations are often layered, such that point events at one level correspond in some approximate but definable way to actions with extent at lower levels. The point events are often invisible at higher levels. We are aware of at least three other research initiatives, in areas as diverse as model-driven architectures and real-time systems, which are discovering that layering is a key concept; no work has yet been published on these discoveries.

There are some similarities between our extra levels to control and interpret the CA behaviour and other CA-based research programmes; the difference is that others do not explicitly use their interpretation layers for downward causation. For example, in Fredkin's digital philosophy, readings of various parameters at various of six defined phases of a two-time-layer, 3-D CA are interpreted as physical properties. The CA simulates the laws of physics [4]. Wuensche's work, interpreting the time-series of CA updates and detecting attractors [11, 12], also provides implicit interpretation layers. This work is potentially important for engineering emergence, since design is likely to be considerably facilitated if the attractors of an emergent feature can be established.

8 Conclusion

Our work exploits notions of layering and causality in emergent systems to improve our ability to engineer required properties and to enhance the expressive power of our simulations.

Having introduced layers for migrated rules, in the platelet model we can exploit the higher layer for more natural control laws, and the implemented platelet simulation could easily be extended to a two- or three-dimensional representation that is a better model of a blood vessel. We can introduce and experiment with models of environmental interaction, and, having abstracted platelet control from the CA grid, we could introduce local diffusion CAs on the absolute grid, modelling the chemical environment that acts on and is affected by the platelets. Further local CA rules could model flow features such as the effect of proximity to the vessel boundary on speed. These new features would be monitored by the higher-layer structures, which would also communicate "chemical" signals to the CAs.

In engineering terms, we are using layering and causality to devise architectural patterns for the design of emergent systems. We also seek to introduce good engineering practices, such as validation, testing and safety argumentation to the development process associated with this layered architecture.

9 Acknowledgements

H. R. Turner is funded jointly by EPSRC and BAE Systems. The work presented here is part of the TUNA feasibility study, EPSRC grant EP/C516966/1. Thanks are due to S. A. Schneider for the inspiration for the vacuum CA model.

References

1. P. B. Andersen, C. Emmeche, N. O. Finnemann, and P. V. Christiansen, editors. *Downward Causation. Minds, Bodies and Matter.* Åarhus University Press, 2000.
2. F.R.M. Barnes and P.H. Welch. Communicating Mobile Processes. In I. East, J. Martin, P. Welch, D. Duce, and M. Green, editors, *Communicating Process Architectures 2004*, volume 62 of *Concurrent Systems Engineering Series*, pages 201–218. IOS Press, September 2004.
3. F.R.M. Barnes, P.H. Welch, and A.T. Sampson. Barrier synchronisations for occam-pi. In *Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'2005)*, pages 20–26. CSREA press, 2005.
4. E. Fredkin. Digital mechanics: An informational process based on Reversible Universal CA. *Physica D*, 45:254–270, 1990.
5. R. Freitas. Clottocytes: artificial mechanical platelets. Technical Report IMM Report Number 18: Nanomedicine, Foresight Update 41, 2000.
6. M. Gardner. Mathematical games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223:120–123, 1970.
7. C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.
8. R. Milner. *The Pi Calculus.* Cambridge University Press, 1999.
9. F. Polack and S. Stepney. Emergent properties do not refine. In *REFINE 2005, BCS-FACS Refinement Workshop*, ENTCS. Springer, April 2005.
10. P.H. Welch and F.R.M. Barnes. Communicating mobile processes: introducing occam-pi. In A.E. Abdallah, C.B. Jones, and J.W. Sanders, editors, *25 Years of CSP*, volume 3525 of *LNCS*, pages 175–210. Springer, 2005.
11. A. Wuensche. Finding gliders in cellular automata. In A. Adamatzky, editor, *Collision-Based Computing.* Springer, 2002.
12. A. Wuensche. Discrete Dynamics Lab: Tools for investigating cellular automata and discrete dynamical networks. In A. Adamatzky and M. Komosinski, editors, *Artificial Life Models in Software.* Springer, 2004.

A Appendix: CA Model of Platelets

This is one possible CA rule set to simulate platelet clustering. The cell design has boolean `state`, `next` and `vacuum` variables. The `nd` variable is set to determine whether a cell containing a platelet loses its platelet in the next time step. It takes values 0, 1 and 2, where 0 represents "no change", 1 represents "change" and 2 is assigned if the resolution of non-determinism is a decision to change.

The value of `nd` is deterministic for all cells except the first cell of a cluster and the cell immediately behind (most) vacuums. The value is set using the first set of CA rules. The `next` variable of each cell is then calculated in the second phase. Calculations within a phase can be concurrent, as can the actual update where each `state` is reset to the cell's `next`.

A.1 The First Phase

The first pass visits only cells having `cell[i].state = TRUE`.

Rule A. A platelet cannot move because it is blocked by platelets ahead.
Rule B. A platelet cannot move because it is located ahead of a vacuum.
Rule C. A platelet between two vacuums cannot move.
Rule D. A platelet before a vacuum can decide whether or not to move.
Rule E. The platelet at the front of a cluster can decide whether or not to move.
Rule F. A singleton platelet must move.

The tabular summary gives the rule name, then the applicable values of the current cell and its neighbours, and the resulting `nd`. An occupied cell is labelled, T; an empty cell, -; and a vacuum, V.

RULE				`nd`
A	T	T	T	0
	-	T	T	0
	V	T	T	0

RULE				`nd`
B	V	T	-	0
C	V	T	V	0

RULE				`nd`
D	T	T	V	nd
	-	T	V	nd
E	T	T	-	nd

RULE				`nd`
F	-	T	-	1

A.2 The Second Phase

In the second pass, any cell that contains a platelet and has an `nd` value of 0 is unchanged. The `next` state for cells with `nd` > 0 is calculated to take account of vacuums. Cells that do not contain platelets have their state calculated.

Rule S.1. A cell with `nd` $= 2$, and a platelet behind, becomes a vacuum.
Rule S.2. A cell with `nd` $= 2$, which is at the back of a cluster, becomes empty.
Rule S.3. A cell holding a singleton platelet becomes empty.
Rule S.4. An empty cell, with an empty cell before it, does not change.
Rule S.5. An empty or vacuum cell, with a preceeding platelet having `nd` $= 0$, does not change. The CA design disallows a vacuum with an empty cell after it.
Rule S.6. A cell whose `nd` value is 0 does not change.
Rule S.7. An empty or vacuum cell with a platelet behind it having `nd` > 0, becomes occupied.

In the summary tables, each rule number is followed by the applicable states of the current cell and its neighbours; the last column is the `next` value of the current cell. Where the fact that the `nd` value was set non-deterministically is important, the resolved value is shown (eg $\boxed{\text{T,2}}$).

RULE				`next`
S.1.	T	T,2	-	V
	T	T,2	V	V
S.2.	-	T,2	V	-
S.3.	-	T	-	-

RULE				`next`
S.4.	-	-	-	-
	-	-	T	-
S.5.	T,0	-	T	-
	T,0	-	-	-
	T,0	V	T	V

RULE				`next`
S.6.	T	T,0	T	T
	-	T,0	T	T
	V	T	T	T
	V	T	-	T
S.7.	T,2	-	T	T
	T,2	-	-	T
	T,2	V	T	T