Hierarchical Components and Entity-based Modelling in Artificial Life

Matt Webster[1] and Grant Malcolm[1]

[1]Department of Computer Science, University of Liverpool, Liverpool, L69 3BX, UK.
matt@liverpool.ac.uk

Abstract

We present notions of entity and entity-based model that are applicable to artificial life. We illustrate these notions by giving an abstraction of Langton's loops: loop-like structures that reproduce in a cellular automaton (CA). Our abstraction takes as entities persistent configurations of the cellular automaton, and shows how these entities may be combined to form more complex entities. The resulting entity-based model of Langton's loops describes the functionality and interrelationships of these components, abstracting from their actual realisation in a cellular automaton. As well as providing a basis for the study of ecologies of interacting entities in artificial life, our approach provides a useful intermediate level of abstraction that can relate top-down and bottom-up approaches to the study of life-like systems.

Introduction

One of the earliest results in the field of artificial life was given by von Neumann (1966), in which an automaton on a cellular automaton (CA) grid was shown to reproduce itself. Subsequent improvements by Codd (1968), Langton (1986), and others (Sipper, 1998) showed an intriguing approach to the modelling of life that differed from much of mainstream biology. In biology, abstract processes such as reproduction, metabolism and evolution are specified based on observation of existing life forms. We can characterise this as a top-down approach, in which we attempt to understand life and its systems through abstract models and classifications. The opposite approach is widely-adopted in artificial life, in which bottom-up models of life are generated. This difference hs been well-documented (Langton, 1986; Sipper, 1995; Bonabeau and Theraulaz, 1996), and is a key characteristic of much work within the field of artificial life. The question of which approach to use was also explored by Bedau (1998) with the question, 'Does the essence of life involve matter or form?'

The work by Rosen (1991) on the modelling of life has received attention recently in the artificial life community (Chu and Ho, 2006; Louie, 2007; Wolkenhauer, 2007; Chu and Ho, 2007). Rosen argued that reductionistic models of life were not adequate for giving a formal description of the organisation observed, i.e., life cannot be reduced to physical laws of the Universe. Instead, Rosen suggested a high-level description of life based on systems consisting of interacting components. This kind of approach has been adopted elsewhere in the artificial life community, including work by Adams and Lipson (2003) and ourselves (Webster and Malcolm, 2007b,a).

The apparent disparity between top-down and bottom-up approaches to modelling life presents a problem for the field of artificial life. Both approaches have been proven to be valuable, and many interesting results have been obtained in both directions. However, the question of how these differing approaches are related is still an open question. In this paper we present a way of relating these two levels. exemplified through a 'reverse engineering' of Langton's loops, a seminal example of implementing artificial life in a cellular automaton. We take a paradigmatic artificial life reproducer, Langton's loop, and show how its interacting components can be abstracted from the low-level state of the cellular automaton. We show that these components can be tied together using formal constraints that are based on observation of the cellular automaton itself. The result is a high-level formal description of Langton's loop, which abstracts the functionality from the particular implementation details of the loop, and captures the interaction of various components within the reproducing whole. We describe how the high-level abstract model can be fitted to a variety of low-level models through a refinement process.

This relationship between high- and low-level models is inspired by work in the field of software engineering, in which high-level specifications of a software system are refined to produce a low-level implementation. A typical example of this would be the transition from a software design specification in an abstract modelling language (e.g., UML), which is refined to a software specification in a high-level language (e.g., C++), which is then refined further to a low-level language implementation (e.g., using Intel 64 assembly language) during the compilation process. There may be several different low-level language implementations that match the higher-level specifications. Simi-

larly, the high-level component-based specification of Langton's loop can be refined to a number of different low-level specifications, in which the implementation details (such as the transition rule, or number of states of the cells in the cellular automaton) vary, but which all satisify the high-level 'reverse-engineered' component-based description of a reproducing loop. These refinements need not be trivial alternatives, such as re-labelled states, as the high-level component-based specification would essentially describe any reproducing loop satisfying certain constraints. Therefore, this specification could be refined to different cellular automata (with different states, topologies and transition rules), or even non-discrete examples of cellular computation, in which the data-paths of Langton's loop are continuous or noisy communication channels in the vein of those described by information theory (Shannon, 1948).

In the following section, we review the construction of Langton's loops, and present abstract entities that capture the functionality of the various components that build up these loops. We also illustrate how our approach allows for a dynamic system in which entities are created and change relationship between one another.

Langton's Loops

Langton's loops provide a simple example of self-reproduction in cellular automata (Langton, 1984). The question of whether an automaton could reproduce, thereby exhibiting at least one life-like trait, was first studied by von Neumann (1966). He was able to provide a positive answer by exhibiting a *universal constructor*: an automaton which, given as input a description of any automaton, could construct that automaton and copy the input description as that automaton's input. Therefore, given a description of itself, the universal constructor reproduces by building a copy of itself with its own self-description. Von Neumann's universal constructor required cellular automata with 29 states, and was later simplified by Codd (1968). While universality is an interesting feature from the point of view of computability, it is not, as Langton (1984) noted, a necessary feature of reproduction, and indeed it seems unlikely to figure in biological reproduction. Langton's loops were the result of his search for a simpler example of a reproducer, though not so simple as to become trivial. In particular, Langton required that the process of reproduction be 'actively directed' by the reproducer, and that the reproducer store information directing its reproduction that is both interpreted (as instructions) and uninterpreted (as data that is copied or transcribed).

Langton states[1] that 'the idea for this simple self-reproducing configuration came out of a study of the components of Codd's universal constructor', and it is our goal to elucidate in what sense both Langton's and Codd's configurations can be said to have 'components'.

A cellular automaton is an example of a *deterministic* transition system: at a given moment each cell in the grid is in a particular state; its state at the next moment of time is determined by its own state and the states of its neighbours (the 5-cell neighbourhood for Langton's loops) according to a *transition function*, which it is usually convenient to present as a table listing transitions for all possible configurations of a neighbourhood's states. Langton's loops are realised on a cellular automaton where each cell is in one of eight states. Throughout the remainder of this paper we will refer to this state set as $\mathsf{S} = \{0, 1, 2, 3, 4, 5, 6, 7\}$. Each of these states has a particular function: for example, state 0 represents 'blank space', though it also provides direction for instructions such as state 7, which causes a 'data-path' to be extended; these data-paths are the basic structure of Langton's loops, as they were also in Codd's universal constructor. They are formed from two rows of sheath cells with a row of 'core cells' between, as pictured in Figure 1.

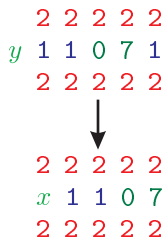


Figure 1: A data-path. The data signal '7 0' travels along the data-path. The state of cell x in the updated state is dependent on the state of cell y. If $y = 1$, then in the updated state $x = 1$.

The function of a data-path is the transmission of a data signal along its length. For example, in Figure 1 the '7 0' signal has been shifted one cell to the right. This process continues as long as the data-path does. data-paths can be 'capped' using a cell in state 2, as seen in Figure 2.

Figure 2: A capped data-path.

The cap allows the data signal to effect the extension of the data-path. This extension process is what allows Langton's loop to extend an 'arm', which loops around to complete the act of reproduction.

Another kind of data-path functionality in Langton's loop is the T-connection, as seen in Figure 3. The data-path sends a signal to a point where two other data-paths are connected. The intersection of the three paths is at a particular cell.

[1] *op. cit.*, p.137

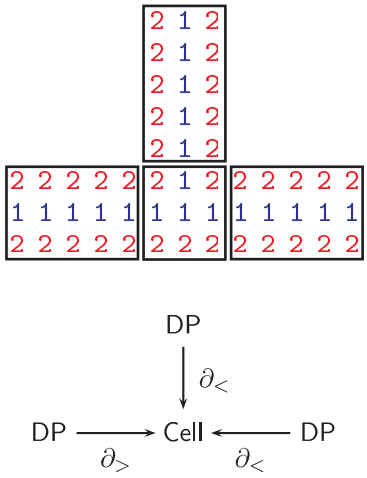


Figure 3: A T-connection. The image at the top shows a T-connection as realised in a cellular automaton; the diagram below describes the same connectivity in the schematic form described in the following subsection.

When the data signal arrives at the cell, it is duplicated and sent out through each of the other two connected data-paths.

Components

We want now to abstract, as far as we can, the functionality described above from its realisation in a particular cellular automaton. Our goal is to be able to identify individual components, such as data-paths, capped data-paths and T-connections, and capture the way they interact.

The most important functional unit is the data-path, consisting of a series of core cells sandwiched between two rows of sheath cells. The function of such a component is to transmit signals: it acts as a queue for the data in the core cells, which move, cell by cell, along the data-path. To capture this, we define a *data-path* to be a transition system (which we shall call DP) whose states are pairs $(n; d)$, where $n > 0$ and d is an n-tuple of states, $d = d_1, \ldots, d_n$ with $d_i \in \mathsf{S}$; and with the following schematic transition rule:

$$(n;\ d_1, \ldots, d_n) \longmapsto (n;\ x, d_1, \ldots, d_{n-1}) \ . \qquad (1)$$

I.e., all the data values in the path move up one space (left to right), and a new value x enters at the start-cell. Note that this is non-deterministic: there are no constraints on what this new value is, beyond $x \in \mathsf{S}$. Thus, we might have both

$$(5;\ 1, 1, 0, 7, 1) \longmapsto (5;\ 1, 1, 1, 0, 7) \ ,$$

as in Figure 1, and

$$(5;\ 1, 1, 0, 7, 1) \longmapsto (5;\ 7, 1, 1, 0, 7) \ .$$

Insofar as a data-path represents an actual sequence of core cells in the cellular automaton, the value that 'enters' the data-path will be determined by the values of the cells in the neighbourhood of the start-cell d_1, but that is exactly what we are abstracting from: functionally, a data-path allows arbitrary signals to be transmitted.

We can *ground* a data-path in a cellular automaton by means of a mapping $h : \mathsf{CA} \to \mathsf{DP}$ of the state-space of the cellular automaton to the state-space of the data-path. That is, if the state-space of CA consists of configurations $\vec{s} \in \mathsf{S}^{pq}$, with pq the number of cells in the grid (which we could allow to be infinite), then such a mapping would project a pq-tuple $\vec{s}$ to an n-tuple $(n;\ s_{i_1}, \ldots, s_{i_n})$. This would allow projecting a configuration to a tuple of states of an arbitrary collection of cells (i.e., they need not be contiguous), which wouldn't capture the intention of picking out a particular data-path in the configuration. However, we impose the condition that the mapping *preserve transitions*, i.e., if $\vec{s} \longmapsto \vec{s'}$ is a transition of the cellular automaton (viewed as a transition system), then $h(s) \longmapsto h(s')$ in DP. This means that the mapping must pick out a tuple of cells that acts as a data-path: we must be able to observe signals moving cell-by-cell along those tuples.

The basic building block of all components, including data-paths, is the individual cell. As a constituent part of the cellular automaton grid, its functionality is dependent on the states of the cells in its neighbourhood, so at an abstract level, its functionality is simply to be in a particular state. We define a *cell* to be a transition system (call it Cell) with state set S and universal transition relation; i.e., $s \longmapsto s'$ for all $s, s' \in \mathsf{S}$. A *grounded cell* is a transition-preserving map $h : \mathsf{CA} \to \mathsf{Cell}$. Note that the requirement that h preserve transitions is trivial, because any state can make a transition to any state in Cell.

This gives us a means of glueing data-paths together. Define $\partial_<, \partial_> : \mathsf{DP} \to \mathsf{Cell}$ by

$$\begin{aligned} \partial_<(n;\ d_1, \ldots, d_n) &= d_1 \\ \partial_>(n;\ d_1, \ldots, d_n) &= d_n \end{aligned}$$

so that $\partial_<$ picks out the 'start cell' of a data-path, and $\partial_>$ picks out the 'end cell'. Again, these maps are trivially transition-preserving.

Now, to join together two data-paths, let a *data-path connection*, $\mathsf{DP} \Rightarrow \mathsf{DP}$, be the transition system whose state-set consists of pairs (p_1, p_2), with each p_i a DP-state such that $\partial_>(p_1) = \partial_<(p_2)$, i.e., p_1 and p_2 communicate by sharing a cell which is both the end-cell of p_1 and the start-cell of p_2. Let the transitions of $\mathsf{DP} \Rightarrow \mathsf{DP}$ be given by

$$\begin{gathered} ((m;\ d_1, \ldots, d_m), (n;\ d_m = e_1, \ldots, e_n)) \longmapsto \\ ((m;\ x, \ldots, d_{m-1}), (n;\ d_{m-1}, d_m = e_1, \ldots, e_{n-1})) \end{gathered}$$

This effectively constrains the non-determinism in (1): the new value that enters the second data-path p_2 is determined by the values at the end of the first data-path p_1. For exam-

ple, we have

$$((3;\ 1,0,7),\ (4;\ 7,1,1,0)) \longmapsto$$
$$((3;\ 7,1,0),\ (4;\ 0,7,1,1))\ .$$

Effectively, we can think of this as one long data-path, in which the above transition could be rewritten as

$$(6;\ 1,0,7,1,1,0) \longmapsto (6;\ 7,1,0,7,1,1)\ .$$

As with the previous components, a data-path connection can be grounded in a cellular automaton by providing transition-preserving morphisms from the cellular automaton to the individual subcomponents: the two data-paths and their shared cell. Note that there is no requirement that grounded connected data-paths be orthogonal.

Transcription of the data within Langton's loops is achieved by branches in the data-paths: as signals move through a T-connection, they are copied along the two branches, as in Figure 3. Abstractly, we have three data-paths p_1, p_2, and p_3, all sharing a single cell at their intersection, which is the end-cell of p_1 and the start-cell of both p_2 and p_3 — a diagrammatic representation is given in the lower half of Figure 3. Thus, a *T-connection* is set up by two data-paths $\partial_<$-connected to the end-cell of a single data-path, and we define $\mathsf{DP} \Rightarrow \mathsf{DP}|\mathsf{DP}$ to be the transition system whose states are triples (p_1, p_2, p_3) with $\partial_>(p_1) = \partial_<(p_2) = \partial_<(p_3)$. The transitions are given by the fact that both (p_1, p_2) and (p_1, p_3) are data-path connections: in brief, $(p_1, p_2, p_3) \longmapsto (p_1', p_2', p_3')$ iff both $(p_1, p_2) \longmapsto (p_1', p_2')$ and $(p_1, p_3) \longmapsto (p_1', p_3')$ as data-path connections. It should be clear that the effect of these definitions is to capture the data-flow and copying of the T-connections pictured in Figure 3. Again, a T-connection is grounded in a cellular automaton by grounding its subcomponents.

Clearly, we can specify a great variety of structures built from data-paths and shared cells in a diagrammatic way along the lines of the T-connection shown in Figure 3. For example, a *loop* is a loop of connected data-paths, as shown in Figure 4. In more detail, we have data-paths p_1, p_2, p_3 and p_4 with structural constraints

$$\begin{aligned}
\partial_>(p_1) &= \partial_<(p_2) \\
\partial_>(p_2) &= \partial_<(p_3) \\
\partial_>(p_3) &= \partial_<(p_4) \\
\partial_>(p_4) &= \partial_<(p_1)\ .
\end{aligned}$$

As with T-connections, the transitions are determined by the fact that each pair p_i, p_{i+1} (mod 4) is a data-path connection. It should be clear that the data in these paths continually circles around the loop. As with the other components described above, a loop is grounded by grounding the subcomponents, as illustrated in Figure 4.

In fact, we have not quite modelled Langton's loops: the essential missing ingredient is the arm that 'interprets' the data in the loop, extending into the grid, turning, and eventually looping back on itself. Let us call such an interpreting arm a *capped data-path*, and provide it with the structure of a transition system, which we shall call CDP. The states of CDP are the states of DP, but we shall use the suggestive notation $(n;\ d\,]$, $n > 0$ and $d = d_1, \ldots, d_n$ a data-path of length n. The transitions of CDP should capture the effects of interpreting the signals in the data-path. For example, the signal '7' extends the path:

$$(n;\ d_1, \ldots, d_{n-1}, 7\,] \longmapsto (n{+}1;\ x, d_1, \ldots, d_{n-1}, 1\,] \quad (2)$$

Note also that this transition allows data to move along the capped data-path, just as for data-paths themselves; in particular, a new value x non-deterministically 'enters' the data-path. As with data-paths, capped data-paths can be $\partial_<$-connected to data-paths, giving a transition system $\mathsf{DP} \Rightarrow \mathsf{CDP}$, and so may also form T-connections. We shall give further transitions in the following subsection; we end this subsection by giving the abstract, functional form of Langton's loops: an *L-loop* is a tuple (p_1, p_2, p_3, p_4, c), where the p_i form a loop of data-paths, and c is a capped data-path $\partial_<$-connected to p_4:

$$\begin{aligned}
\partial_>(p_1) &= \partial_<(p_2) \\
\partial_>(p_2) &= \partial_<(p_3) \\
\partial_>(p_3) &= \partial_<(p_4) \\
\partial_>(p_4) &= \partial_<(p_1) = \partial_<(c)\ .
\end{aligned}$$

With the definitions given above, we could fill an L-loop with '7' signals, and the transitions allow the capped data-path to extend indefinitely.

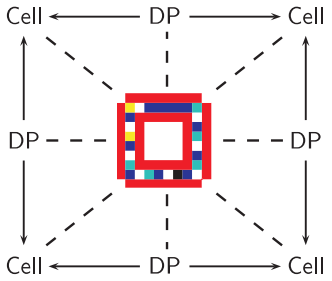


Figure 4: Connections between data-paths and cells for Langton's loop. The outer square denotes the structure of the component data-paths, while the dotted lines indicate a grounding of the subcomponents of the loop in a cellular automaton.

Introduction of New Components

We have seen how we can model components of Langton's loops, building larger components from subcomponents in a diagrammatic, hierarchical way. The configurations we have seen are all static, in the sense that components are

related in a fixed way that never alters. Obviously, since the purpose of Langton's loops is to demonstrate reproduction by the creation of one loop by another, we need to allow for components that are created or destroyed, or even change their structural relationships.

In the case of Langton's loops, all the structural changes are brought about by the actions of the interpreting arm: the capped data-path attached to the loop. We saw in (2) above how we could capture the effect of a '7' signal at the cap of the data-path, namely by extending the length of the path. The other relevant signal for Langton's loop is a '4' signal, which is interpreted as an instruction to create a left-hand turn in the data-path. In fact, we shall simplify things here, as a left-hand turn in Langton's cellular automaton is created by *two* '4' signals. However, this simplification is quite in keeping with our goal of abstracting the functionality of the relevant components. In accordance with this simplification, when a '4' signal reaches the cap of the interpreting arm, the left-hand turn is achieved by turning the capped data-path into a data-path, with a capped data-path $\partial_<$-connected to its end-cell. In our abstract, functional view, two things happen. The first is that the diagram representing the component changes from

$$\mathsf{Cell} \xleftarrow{\partial_<} \mathsf{CDP} \xrightarrow{\partial_>} \mathsf{Cell}$$

to

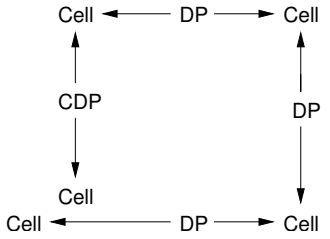
$$\mathsf{Cell} \xleftarrow{\partial_<} \mathsf{DP} \xrightarrow{\partial_>} \mathsf{Cell} \xleftarrow{\partial_<} \mathsf{CDP} \ .$$

The second thing that happens is that the states of the transition system denoted by this diagram makes the corresponding change:

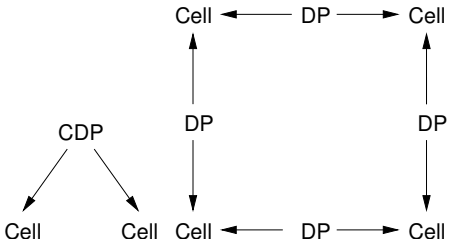
$$(n;\ d_1 \dots d_{n-1}, 4] \longmapsto ((n;\ x, d_1, \dots, d_{n-1}), (1;\ d_{n-1}])$$

This transition turns a capped data-path into a data-path and a new capped data-path; i.e., extending the notation above, the transition system CDP becomes $\mathsf{DP} \Rightarrow \mathsf{CDP}$.

A CA grid imposes constraints on what may happen in components that are grounded in it, but there are purely topological constraints that may be imposed on components, whether or not they are grounded in a cellular automaton. For example, and here we consider functionality rather than topology, we may wish to state that, after turning three times, an interpreting arm is heading back towards itself, and will sever the umbilical cord that ties the parent to the child loop. When it meets its original arm, the loop is closed and the original data-path is split: the first part becomes a capped data-path, while the remainder becomes a part of the child loop. Diagrammatically, the state



becomes



In accordance with this, the states of the relevant components change as follows:

$$((m;\vec{a}), (n;\vec{b}), (m';\vec{c}), (n;\ \vec{d}]) \longmapsto$$
$$(m-m';{}_{m-m'}|\vec{a}], ((m';\vec{a}|_{m-m'}), (n;\vec{b}), (m';\vec{c}), (n;\vec{d}))$$

provided that $m' < m$, and where ${}_p|\vec{a}$ denotes the first p elements of $\vec{a}$, and $\vec{a}|_p$ denotes the 'remainder' of $\vec{a}$ from the pth position on. Here again we simplify matters: in Langton's automaton, the capped data-path belonging to the parent loop bears a '5' signal that will close off the capped data-path; also, the data-path in the child loop contains a '6' signal that will cause a new capped data-path to be created, causing the process of reproduction to begin anew. This extra functionality can be captured straightforwardly, using the above techniques, so we omit the details. The main point is that, even without these details, we have captured reproduction of loops in an abstractly functional, entity-based model.

Entities and Some Technical Details

In the preceding subsections, we have tried to avoid going into too many technical details, preferring to keep the exposition at an intuitive level. From our use of diagrams and transition-preserving mappings, it may be clear that category theory provides a natural setting for our constructions. Indeed, the constructions of transition systems such as $\mathsf{DP} \Rightarrow \mathsf{DP}$ are limit constructions in a category of transition systems and transition-preserving morphisms, as shown in (Malcolm, 2006). This is very much in keeping with the slogan 'Behaviour is Limit' from the Categorical Systems Theory of Goguen (1992). In this view, mappings between transition systems can be seen as constraints: for example, the two mappings in the configuration $\mathsf{DP} \Rightarrow \mathsf{DP}$ state that the end-cell of one data-path is the start-cell of the other, and as noted above, this constrains the non-determinism in (1). The limit of a configuration then consists of all possible behaviours that meet the given constraints; in the example of $\mathsf{DP} \Rightarrow \mathsf{DP}$, the limit gives all possible behaviours where the

'output' of the first data-path is the 'input' of the second'. Thus our approach is quite general in that the limit construction captures all the ways in which components may interact.

In our functional model of Langton's loops, we take an *entity* to be a component, such as a data-path, which is a transition system *in a diagram*, such as the diagram of Figure 3. We might say that an entity is necessarily 'situated': it stands not alone (except in the case of a trivial diagram) but in relationships with other entities, particularly any subcomponents it may have, and may share with other entities. Since complex objects are created by limit constructions from subcomponents, technically, our entities are *sheaves*: see Malcolm (2008) for details. Again, this is in keeping with the slogan 'Objects are Sheaves' of Goguen (1992).

Conclusion

As we described in the introduction, the aim of our work is to create a formal bridge between top-down and bottom-up models of life forms. We have shown how an abstract model of cellular automaton-based artificial life forms can be constructed, based on a formal description of various interacting components of the life form. We presented an entity-based model of Langton's loops that abstracts the functionality of the various components from their actual realisation in a cellular automaton. Interactions between data-paths and cells are formalised in terms of constraints on their transition systems. Our approach allows for entities to be constructed hierarchically from subcomponents, and to interact with each other through shared subcomponents: for example, two data-paths communicate values through a shared cell in the $\mathsf{DP} \Rightarrow \mathsf{DP}$ configuration. We can also model dynamic configurations where entities may be created and adopt varying relationships with other entities. This is useful in modelling reproduction, as it is often the case that the reproduction process involves creating various components of the offspring. We gave an example of the introduction of new components for Langton's loops, in the case where a data-path approaches a cell, and then branches off in a new direction. This is a repeating process that results eventually in the creation of a second loop, showing that our abstract component-based model is able to model reproduction.

Therefore, we can see the different entities in the model as different behaviours, which may combine to form more complex behaviours. For example, a data-path is an entity that propagates information; this can be combined with other data-paths, cells and a capped data path in order to form a conglomerate entity capable of reproduction.

We showed how these abstract models can be related ('grounded') to a particular implementation of a loop, through a mapping from the transition system of the former, to the transition system of the latter. This notion of grounding allows an actual realisation of the model in a cellular automaton to be viewed as a refinement of the model. Conceptually, grounding can also be seen as the imposing of constraints on the model by the cellular automaton and its topology: in much the same way, a model grounded in a real-life system would be constrained by the laws of physics.

Related Work

In this paper we have described an approach to the development of entity-based models of artificial life systems. A recent report by Wheeler et al. (2002) highlighted many of the challenges in the area of artificial life modelling.

Entity-based models of reproduction, such as those described earlier, have been used before in artificial life. For example, Adams and Lipson (2003) give a formal universal framework for reproduction based on 'subsystems' within an environment. The subsystems are analogous to the entities or components discussed in this paper. One possible property of a subsystem is reproduction. Since Adams and Lipson do not preclude the possibility of subsystems consisting of other subsystems, we could re-frame the discussion of Langton's loop with each component as a subsystem. The system consisting of these subsystems, i.e., the loop, has the property of reproduction. Another example is the work by Hordijk et al. (1998) on embedded-particle models, which model emergent functionality in evolved cellular automata. The approach is similar to our own in that an abstract description of organised behaviour is formed, although the emphasis is more on abstracting higher-level behaviour from an existing CA configuration than on providing a general means of describing the behaviour of hierarchical systems.

Abstract, high-level models of life-like phenomena have also been explored in biology. Lazebnik (2002) describes the fundamental differences between the language of biology and engineering, and posits that the formalisms of engineering permit a greater understanding of complex systems such as life. Discrete, top-down models of biological processes have also been described, e.g., the approach of Laubenbacher and Mendes (2006) to modelling biochemical networks.

The work in this paper may be seen as an 'intellectual progeny' of our earlier work on formal affordance-based modelling of reproduction, in which we described a reproduction system in terms of a labelled transition system, with entities present in various states, and affordances relating the entities which cooperate in various parts of the reproductive process (Webster and Malcolm, 2007b,a). These affordance-based models are therefore entity-based, and are in this way related to the entity-based models presented here.

Our work is also influenced by the work of Rosen (1991, 1999) on modelling life-like processes. Rosen argues that a reductionistic 'machine metaphor', in which life is seen simply as the result of the underlying physical laws of the Universe, is insufficient to capture the full complexity of life: self-organisation, reproduction and so on. Rosen suggests that a more holistic model of life is needed, and that a sufficient model for life might be obtained at a natural level of

abstraction. For example, if a large conglomeration of cells (e.g., an animal) appears to reproduce, then the natural way to view that process is not in terms of the cells, but in terms of the sub-components of the animal that enable reproduction, e.g., sex organs, nervous system, etc. Therefore, our attempt to model artificial life forms like Langton's loop is in the same vein. Of course, Langton's loop can be described completely by its formal definition; however, it may also be interesting to describe formally the same system at a different level of abstraction, as the aim of abstraction is to provide greater insight than can be obtained through individual case studies.

Future Work

We have presented Langton's loops as a case study in our approach to modelling hierarchical artificial life systems. An obvious question that arises is: what other systems can usefully be captured in this way? There are several candidate systems in artificial life and in biology that seem to have a natural hierarchical structure, and one thrust of future research would be to construct entity-based models of these.

An example from artificial life is the mechanical reproducing robots described by Zykov et al. (2005), in which a number of modular robots interact in order to reproduce a conglomerate multi-robot entity. The robots are constrained to connect at certain points, mirroring the cells at which one or more data-paths may be connected.

Another interesting application of our work would be to model the component behaviour of more complex forms of reproduction, including evolution (Sayama, 1999) and sexual reproduction (Oros and Nehaniv, 2007). These seem particularly feasible examples in that they are based on the same conceptual mechanisms as Langton's loops. In the case of sexual reproduction, there is a link to our earlier work (Webster and Malcolm, 2007b,a) on reproduction modelling based on Gibson's theory of affordances (Gibson, 1979). Sexual reproduction can be seen as a collaborative arrangement, in which each partner affords the other the act of reproduction, and therefore there is a possibility of extending the component-based models presented in this paper to include a notion of affordances.

In the field of biological systems, we would expect to be able to model simple ecologies such as those of bacteriophage viruses and host cells. There are obvious entities or components which we may identify, such as viral RNA, generation of offspring based on proteins, cell walls, and so on. We have given a simplified model of the T4 bacteriophage lifecycle in (Webster and Malcolm, 2008), and we expect the extension of this to a more realistic hierarchical model to be reasonably straightforward. Other interesting biological examples with significant hierarchical structure would include meiosis, and possibly tissues: for example, hepatic tissue has a rich structure that provides a rich functionality (Teutsch, 2004).

A useful feature of our entity-based models is that they are formal. In principle, it should therefore be possible to prove formally that a certain group of entities in a given model is capable of the act of reproduction. One way of doing so would be to construct a 'minimal' model of reproduction, and show that the given model is a refinement of it. A minimal model would be purely schematic, consisting of just two states, the first of which contains just one entity, and which has a transition to the second state, which contains just two entities: notionally, these would be the original entity and its offspring. A refinement of this by another model would relate these two entities to entities in the other model, for example a loop and its offspring loop in the case of Langton's loops. In addition, the transition from the first schematic state to the second would be related to a path in the other model that leads from a state containing the progenitor to a state containing the offspring. In general, a model may refine this minimal model in more than one way, picking out different entities that reproduce: as a ready example from biology, it may be possible to view an organism as reproducing itself, while another view of the same process may see the organism's genetic material as the entity that is reproduced. While the intuitions are clear, the technical details of refinement for our hierarchical, entity-based models still need to be spelt out.

References

Adams, B. and Lipson, H. (2003). A universal framework for self-replication. In *European Conference on Artificial Life (ECAL'03)*, pages 1–9.

Bedau, M. A. (1998). Four puzzles about life. *Artificial Life*, 4:125–140.

Bonabeau, E. W. and Theraulaz, G. (1996). Why do we need artificial life? In Langton, C. G., editor, *Artificial Life: An Overview*. MIT Press.

Chu, D. and Ho, W. K. (2006). A category theoretical argument against the possibility of artificial life: Robert Rosen's central proof revisited. *Artificial Life*, 12:117–134.

Chu, D. and Ho, W. K. (2007). The localization hypothesis and machines. *Artificial Life*, 13:299–302.

Codd, E. (1968). *Cellular Automata*. Academic Press, New York.

Gibson, J. J. (1979). *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston. ISBN 0395270499.

Goguen, J. A. (1992). Sheaf semantics for concurrent interacting objects. *Mathematical Structures in Computer Science*, 11:159–191.

Hordijk, W., Crutchfield, J. P., and Mitchell, M. (1998). Mechanisms of emergent computation in cellular automata. In *Proceedings of the Fifth International Conference on Parallel Problem Solving From Nature—PPSN V*. New York: Springer.

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

Langton, C. G. (1986). Studying artificial life with cellular automata. *Physica D*, 22:120–149.

Laubenbacher, R. and Mendes, P. (2006). A discrete approach to top-down modeling of biochemical networks. In Kriete, A. and Eils, R., editors, *Computational Systems Biology*. Elsevier Academic Press.

Lazebnik, Y. (2002). Can a biologist fix a radio? — or, what I learned while studying apoptosis. *Cancer Cell*, 2(3):179–182.

Louie, A. H. (2007). A living system must have noncomputable models. *Artificial Life*, 13:293–297.

Malcolm, G. (2006). Sheaves and structures of transition systems. In Futatsugi, K., Jouannaud, J.-P., and Meseguer, J., editors, *Algebra, Meaning and Computation: Essays dedicated to Joseph A. Goguen on the Occasion of his 65th Birthday*, volume 4060 of *Lecture Notes in Computer Science*, pages 405–419. Springer.

Malcolm, G. (2008). Sheaves, objects, and distributed systems. *Electronic Notes in Theoretical Computer Science*. To appear.

Oros, N. and Nehaniv, C. L. (2007). Sexyloop: Self-reproduction, evolution and sex in cellular automata. In *Proceedings of the 2007 IEEE Symposium on Artificial Life (CI-ALife 2007)*.

Rosen, R. (1991). *Life Itself*. Columbia University Press.

Rosen, R. (1999). *Essays on Life Itself*. Columbia University Press. ISBN: 978-0231105118.

Sayama, H. (1999). A new structurally dissolvable self-reproducing loop evolving in a simple cellular automata space. *Artificial Life*, 5:343–365.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656.

Sipper, M. (1995). An introduction to artificial life. *Explorations in Artificial Life (special issue of AI Expert)*, pages 4–8.

Sipper, M. (1998). Fifty years of research on self-replication: An overview. *Artificial Life*, 4:237–257.

Teutsch, H. (2004). Modular design of the liver of the rat. In Malcolm, G., editor, *Multidisciplinary Approaches to Visual Representations and Interpretations*, volume 2 of *Studies in Multidisciplinarity*, pages 63–67, Elsevier.

von Neumann, J. (1966). *Theory of Self-Reproducing Automata*. University of Illinois Press. Edited by Arthur W. Burks.

Webster, M. and Malcolm, G. (2007a). Reproducer classification using the theory of affordances. In *Proceedings of the 2007 IEEE Symposium on Artificial Life (CI-ALife 2007)*, pages 115–122. IEEE Press.

Webster, M. and Malcolm, G. (2007b). Reproducer classification using the theory of affordances: Models and examples. *International Journal of Information Technology and Intelligent Computing*, 2(2).

Webster, M. and Malcolm, G. (2008). Classifying and relating affordance-based models of reproduction. Submitted for publication.

Wheeler, M., Bullock, S., Paolo, E. D., Noble, J., Bedau, M., Husbands, P., Kirby, S., and Seth, A. (2002). The view from elsewhere: Perspectives on ALife modelling. *Artificial Life*, 8:87–100.

Wolkenhauer, O. (2007). Interpreting Rosen. *Artificial Life*, 13:291–292.

Zykov, V., Mytilinaios, E., Adams, B., and Lipson, H. (2005). Self-reproducing machines: A set of modular robot cubes accomplish a feat fundamental to biological systems. *Nature*, 435(7038):163–164.