Evolution in Asynchronous Cellular Automata

Chrystopher L. Nehaniv

Adaptive Systems Research Group
Faculty of Engineering & Information Sciences
University of Hertfordshire
Hatfield Herts AL10 9AB
United Kingdom

Abstract

Building on the work of Von Neumann, Langton, and Sayama among others, we introduce the first examples of evolution in populations of self-reproducing configurations in *asynchronous* cellular automata. Reliance on a global synchronous update signal has been a limitation of all solutions since the problem of achieving self-production in cellular automata was first attacked by Von Neumann half a century ago. Results of the author obviate the need for this restriction.

We review our simple constructive mechanism to transform any cellular automata network with synchronous update into one with essentially the same behavior but whose cells may be updated randomly and asynchronously. The generality of this mechanism is guaranteed by a general mathematical theorem that any synchronous cellular automata configuration and rule can be be realized asynchronously in such a way the the behavior of the original synchronous cellular automata can be completely recovered from that of the corresponding asynchronous cellular automaton, in which temporal synchronization locally stays within small tolerances.

It follows that most results on self-reproduction, universal computation and construction, and evolution in populations of self-reproducing configurations in cellular automata that have been obtained in the past carry over to the asynchronous domain using the method described here.

Here we discuss requirements for evolutionary systems in cellular automta and describe implemented examples of our procedure applied to a variety of self-reproducing systems (Byl, Reggia et al., Langton, Sayama).

In particular, we have implemented Sayama's evoloop system asynchronously, giving the first example of evolution in asynchronous cellular automata.

Introduction

From the beginnings of the study of self-reproduction in artificial systems initiated by John Von Neumann in 1948, the primary formal model has been synchronous cellular automata in which configurations develop that eventually may include an unbounded number of copies of the original. The models constructed by Von Neumman and his successors have amply demonstrated that self-reproduction is indeed possible in artificial systems. Furthermore, Von Neumann's work on his automata models even anticipated the important transcription ("blind copying") and translation ("executability") properties of genetic material later found for DNA.

Naturally, the different possible mechanisms for achieving self-reproduction have implications for our understanding of the origin of life, the nature of organic life and evolution, and for the possibilities of life as it may exist elsewhere in the universe. Szathmáry (1999) has proposed a classification of replicators that is applicable to natural and artificial systems, but this has by no means settled the controversies and questions regarding what types of self-replication are, in principle, possible, or regarding what modes of self-replication are necessary or minimal for evolvability. Thus, further study of replication and possible bases for heritability are called for, as self-reproduction is an indispensible prerequisite for any independent evolutionary process. Moreover, self-reproduction and self-repair (or self-maintenance) are often closely related in biology, and an understanding of self-reproduction can thus contribute to our ability to create self-repairing, self-maintaining hardware and software. Sending information, instructions on how to build copies of desired structures using local materials, into an environment rather than sending all necessary materials into that environment represents more economical methods of space exploration and colonization. A NASA report edited by Freitas and Gilbreath (1980) discusses further potential examples and applications of self-reproduction to space science, e.g. proposed self-replicating and self-maintaining lunar factories.

Cellular automata models of self-reproduction have always relied on global synchronous updates. However, the need for an assumption of synchronous update in these spaces may be questioned. In building an artificial self-reproducing entity is it really necessary to have a single global synchronization signal that reaches all parts of the entity simultaneously (or at least within a well-defined tolerance)? If local parts of the configuration are ready to change their state, is it realistic and practical to assume that they must wait until all other parts of

the cellular space are also ready to update their states? Will our assumption of synchronous update unnecessarilty constrain and limit the types of self-reproducing systems we are able to build?

In (Nehaniv, 2002 (in press)), we presented a method to free all cellular automata models of self-reproduction (as well as all cellular automata models of evolution, universal computation, and universal construction) from the need for synchronous update. This is accomplished by an elegant simple mechanism that allows one to construct an asynchronous cellular automaton that is capable of emulating the behavior of a given synchronous cellular automaton. State updates in the asynchronous model may be produced by any asynchronous update mechanism whatsoever (e.g. updates may be random, sequential, locally Poisson-distributed, partially simultaneous, etc., or even synchronous). This result for cellular automata is a special case of a more general mathematical theorem for automata networks with inputs, due to the author (Nehaniv, 2002 (accepted)). Nehaniv (2002 (in press) also gave the first implemented examples of self-reproduction in asynchronous cellular automata by applying this method to Langton's example of synchronous self-reproduction in cellular automata (Langton 1984).

In this paper, we first review the concepts of cellular automata and our construction for making any cellular automata computation asynchronously realizable. In the companion paper (Nehaniv 2002 (in press)) appear illustrations of implementations of this method to (1) Conway's Game of Life (which entail that, in principle, universal computation can be achieved by an asynchronous version of Life) and to (2) the first examples of self-reproduction in cellular automata. Going beyond these results, in this paper we describe additional, implemented models of self-reproduction in asynchronous cellular automata. Moreover, we exhibit the first implemented example of evolution occurring in asynchronous cellular automata.

Preliminaries on Cellular Automata

In this section we review the necessary background on synchronous and asynchronous cellular automata.

Cellular Automata: Definitions

Cellular automata were introduced by J. Von Neumman and S. Ulam to model natural physical and biological phenomena, in particular, for Von Neumann's pioneering studies of self-reproduction.

A *graph* Γ is a set of *nodes* V and a set of *edges* E. An edge $e \in E$ is an unordered pair of nodes $\{v, v'\} \subseteq V$. The *neighborhood* of a node is the set of nodes directly connected to it in the graph by an edge. That is, node v is a neighbor of v' if $\{v, v'\}$ is an edge in the graph. We shall write $v' \in nbhd(v)$ if node v' is a neighbor of node v.

A deterministic *finite state automaton* $\mathcal{A}$ is finite set of states Q, a finite set of inputs X and a transition function $\delta : Q \times X \to Q$.

A *cellular automaton* is a finite or infinite network of identical deterministic finite state automata and a graph structure such that:
(1) each node has the same (finite) number of neighbors,
(2) at each node we have a fixed ordering on the neigbhor nodes (e.g. north, south, east, and west if there are exactly four neighbors to every node), and
(3) the next state of an automaton at node v is always the same function δ of its current state and the current states of its neighbors. (Thus the ordered list of states at neighbor nodes is the input to the automaton at node v.)
We shall denote the ordered list of the states of the neighbors of node v by $(q_w)_{w \in nbhd(v)}$.

Although cellular automata are more general, for our purposes, we may assume that the graph can be realized in some N-dimensional Euclidean space, for example as the set of points with integer coordinates, with edges connecting points that differ by at most exactly 1 in exactly one position ("the Von Neumman neighborhood") or, alternatively by at most 1 in each coordinate ("the Moore neighborhood"). For example, in a two-dimensional case the point (2,3) has neighbors (1,3), (3,3), (2,4), and (3,3) if our cellular automaton uses a Von Neumann neighborhood, and has additional neighbors (1,2), (1,4), (3,2), (3,4) if we instead have a cellular automaton with Moore neighborhoods. We may allow "wrapped around" or a "toroidal topology" by identifying nodes which differ by a fixed vector. For example, in a 25×50 node toroidal topology with Von Neumann neighborhood, node (1,1) has neighbors (1,2), (2,1), (25,1) and (1,50) since (25,1) is identified with node (0,1) and (1,50) is identified with (1,0).[1]

Such cellular automata network topologies are specific examples, but the methods presented here and the supporting mathematical results apply to all cellular automata networks regardless of the details of their particular topologies.

In addition there is usually assumed to be a *quiescent state* $q_0 \in Q$ in the local finite state automata such that if the automaton at node v is in state q_0 and all the its neighbors are in state q_0, then in the next time step the automaton at node v will still be in state q_0. (We will not have a strictly quiescent state in our asynchronous cellular automata.)

A *configuration* is any assignment of local state values to the set of automata at nodes in a finite subgraph of Γ.

[1] We remark that these definitions of Von Neumann and Moore neighborhoods, and that of toroidal topology, are applicable to any dimension N.

Synchronous vs. Asynchronous Update Rules

Usually a cellular automaton $\mathcal{A}$ is required to update the state of the finite automata at all of its nodes simultaneously and in discrete steps. Thus, for all discrete times $t \geq 0$, if at discrete time step t each node v is in some state $q_v(t)$ then at the next discrete time step $t+1$ node v is in its next state $q_v(t+1)$. In the notation introduced above,

$$q_v(t+1) = \delta(q_v(t), (q_w(t))_{w \in nbhd(v)}).$$

Thus the new state $q_v(t+1)$ at node v is given by the local update rule as a function of $q_v(t)$, the current state of v, and the (finite) list $(q_w(t))_{w \in nbhd(v)}$ of all current states $q_w(t)$ of all nodes w in the neigborhood of v. In this case of globally simultaneous update, we say that the cellular automaton is *synchronous*. The *global state* of the cellular automaton $\mathcal{A}$ at time t is comprised of the states $q_v(t)$ of all its nodes at time t and can be regarded as a function from nodes V to local states Q.

If the updates of the local component automata are not required to take place synchronously, but each one will be updated to its next state an unbounded number of times as (locally discrete) time goes on, then we speak of an *asynchronous* automata network. The updates are otherwise unconstrained, e.g. they may be deterministic, non-deterministic, random, sequential, etc., or even synchronous. For further discussion of the relevance of synchronous and asynchronous cellular automata to the modelling of biological systems see for example (Schönfisch and de Roos, 1999). Prior to this paper, all published models of self-reproduction in cellular automata have used only synchronous cellular automata update rules.

Following the general method and theorem of (Nehaniv, 2002 (accepted)), which also applies to more general types of automata networks than cellular automata, for each synchronous cellular automaton $\mathcal{A}$ on graph Γ, we construct another particular cellular automaton $\mathcal{A}'$ on the same graph. Moreover, if the local finite automata in $\mathcal{A}$ have n states then component automaton in $\mathcal{A}'$ at each node v will have $3n^2$ states. The author's theorem guarantees that if $\mathcal{A}'$ is updated by any asynchronous method whatsoever, for each time t in the computation of $\mathcal{A}$, the global state of $\mathcal{A}$ at t is completely determined by a "continuous spatio-temporal section" of the behavior of $\mathcal{A}'$ (see below).

This mathematical theorem implies that all computations that can be carried out on any synchronous automata network can be recovered from the computation of an asynchronous automata network with no constraints on how the updates actually occur in the latter.

Construction of Equivalent Asynchronous Models

The construction of the local automata of the asynchronous cellular automaton $\mathcal{A}'$ from local automata of the synchronous cellular automaton $\mathcal{A}$ is extremely straightforward: Suppose the local automaton of $\mathcal{A}$ has states $Q = \{q_0, \ldots, q_{n-1}\}$ with q_0 quiescent and update function $\delta : Q \times X \to Q$.

The states of the local automaton in $\mathcal{A}'$ are the $3n^2$ states $Q \times Q \times \{0, 1, 2\}$.

For $r \in \{0, 1, 2\}$, we say the neighborhood of a node v in the asynchronous cellular automaton $\mathcal{A}'$ is $ready(r)$ if none of v's neighbors is in a state with third component equal to $r+2 \pmod 3$.[2] Recall that we write $w \in nbhd(v)$ if w is a node in the neighborhood of v.

The update rule of this local automaton is given as follows: suppose node v is in state (q, q', r) with q and q' in Q, and $r \in \{0, 1, 2\}$, and has neighborhood with list of states $(q_w, q'_w, r_w)_{w \in nbhd(v)}$, then, if $r = 0$, the next state of node v is

$$\delta'((q, q', 0), (q_w, q'_w, r_w)_{w \in nbhd(v)})$$

$$= \begin{cases} (\delta(q, (q''_w)_{w \in nbhd(v)}), q, 1) & \text{if the neighborhood of } v \text{ is } ready(0) \\ (q, q', 0) & \text{otherwise,} \end{cases}$$

$$\text{where } q''_w = \begin{cases} q_w & \text{if node } w \text{ is in a state of the form } (q_w, q'_w, 0) \\ q'_w & \text{if node } w \text{ is in a state of the form } (q_w, q'_w, 1). \end{cases}$$

(Since the neighborhood is $ready(0)$, these are the only possibilities. Note the use of the original local transition function δ of the synchronous cellular automaton $\mathcal{A}$ in this case in the definition of δ'.)
For the remaining cases with $r \in \{1, 2\}$, the next state is

$$\delta'((q, q', r), (q_w, q'_w, r_w)_{w \in nbhd(v)})$$

$$= \begin{cases} (q, q', r+1 \pmod 3) & \text{if the neighborhood of } v \text{ is } ready(r) \\ (q, q', r) & \text{otherwise.} \end{cases}$$

The state (q, q', r) can be thought of as encoding the following information: The first coordinate q shows the "visible" state of the corresponding network $\mathcal{A}$ at this node. The second coordinate q' is "hidden" and is used to remember the most recent old state of this node, in case any neighbor needs to refer to it. The third coordinate is used to locally synchronize updates of the visible and hidden nodes.

[2] For any integer n, "$n \pmod 3$" denotes the least nonnegative integer k such that $n - k$ is divisible by 3. Of course k must then be one of $\{0, 1, 2\}$.

The above rule δ' only allows a local state of $\mathcal{A}'$ to change if no node in the neighborhood will get more than one step behind if the update were to be made, otherwise it allows no change at all to the current state. The important $ready(0)$ and $r = 0$ case occurs exactly when the asynchronous cellular automaton is locally ready to emulate the transition of the synchronous cellular automata at this local node using the local transition function of $\mathcal{A}$. Intuitively, the third component of a node v's state can locally distinguish 'present', 'future', and 'past' for neighboring nodes, respectively, by whether they have modulo 3 value equal to, one more than, or one less than the value of the third component of v's state.

Properties of the Emulation

The asynchronous cellular automaton has two important properties established with detailed mathematical proofs in the main theorem of (Nehaniv, 2002 (accepted)):

Suppose the synchronous cellular automaton $\mathcal{A}$ is started in configuration C with node v in state $C(v)$ and all other cells quiescent. Also suppose that asynchronous cellular automaton $\mathcal{A}'$ is started with each node v of the configuration active in state $(C(v), C(v), 0)$ and all other nodes in state $(q_0, q_0, 0)$.

Freedom from Deadlocks. At each node v of the asynchronous network, if the state of v has third component $r \in \{0, 1, 2\}$ eventually the neighborhood of v will be $ready(r)$, and the third component will change value to $r + 1 \pmod 3$.

Existence of Continuous Spatio-Temporal Sections. The state $q_v(0)$ at time 0 of node v of $\mathcal{A}$ is equal to the first component of the state of node v of $\mathcal{A}'$ in its initial configuration. By mathematical induction, one shows that the state $q_v(t)$ of node v in $\mathcal{A}$ at time t is equal to the first component of the state of node v in $\mathcal{A}'$ on the t^{th} time that the third component of node v becomes 1.

The latter property implies that it is possible to completely recover the behavior of $\mathcal{A}$ from the behavior of $\mathcal{A}'$ by simply recording the first components in the initial configuration of $\mathcal{A}'$ and recording the first component of each node v whenever its local automaton makes a transition so that the third component changes from 0 to 1.

Temporal Waves

To an observer, time as it occurs in the synchronous cellular automata may seem to pass at different rates in different parts of the asynchronous cellular space. In fact, locally all neighboring cells are guaranteed to show a visible state (first component) that occurs at most one unit of time in the past or future relative to the corresponding state in the synchronous automaton. Thus, time of the synchronous cellular automaton is emulated in a manner such that it can never get very far out of sync locally in the emulating asynchronous cellular automaton.

This results in waves of temporal update in the cellular space with continuous wavefronts all in the same state of temporal synchronization (i.e., all with the same third component) representing the same moment in the synchronous cellular automaton.

In the figures below the relative temporal synchronization of component is indicated by differences in shading. Another example is illustration of temporal waves occurs in (Nehaniv 2002 (in press)) for the example of an asynchronous version of John Conway's famous "Game of Life". Note that the possibilty of implementation of Conway's Game of Life in an asynchronous cellular automaton entails that universal computation is possible in a two-dimensional asynchronous cellular automata running the modified rules (Nehaniv 2002 (in press)).

Self-Reproduction in Asynchronous Cellular Automata

Detailed background on self-reproduction in cellular automata, including a survey of proposed definitions and problems with these can be found in (Nehaniv 2002 (in press)), so we give only a brief overview before illustrating our various implementations of self-reproduction in asynchronous cellular automata. Discussions of the relationship of self-reproduction to individuality and the heritability of fitness can also be found there.

Some Synchronous Implementations

Langton (1984) implemented and studied the first example of feasible self-reproduction in cellular automata, using an 8-state cellular automaton with an initial configuration of 86 cells, that produces a first offspring after 151 time steps and then proceeds to fill up available space with copies. To avoid trivialities while avoiding the complexity of Von Neumann's model, *Langton's criterion* (1984, 1986) was proposed as a necessary condition on self-reproduction and requires that information is treated in two ways: as instructions that are *executed* ('translation') and as data which are blindly *copied* ('transcription'). These properties are also present in and abstracted from Von Neumann's and later Codd's (1968) examples, and by that time also known to be characteristic of biological self-reproduction. Subsequent examples of Byl (1989) and Reggia et al. (e.g., Reggia, Armentrout, Chou, and Peng (1993), Lohn and Reggia (1997)) simplified the self-replicating loop of Langton with fewer states or less cells in the initial configuration. Subsequently, various researchers kept Langton's requirement for self-reproduction, but have added more and more computational power to the relatively small self-reproducing cellular automata configurations (in comparison to Von Neumann's solution). These trends are surveyed by Lohn (1999), who also describes the evolution of cellular automata rules that support self-reproduction (see also Lohn and Reggia (1997)).

A fairly complete and up to date annotated bibliography with links to various relevant on-line resources can be found at Moshe Sipper's Artificial Self-Replication page. H. Sayama (1998b, 1999) has constructed variants of the self-reproducing Langton loop which exhibit self-dissolution once they can no longer reproduce, thus freeing up space for reuse by progeny, and most interestingly, another similar variant called "evo-loop" which exhibits heritable variability in loop size and is subject to evolution via interaction among descendants of a common ancestor acting as a selective force (Sayama 1998a, 1999). Heritability, variability, and differential survival in an environment with limited resources are present in his evo-loop when run in finite spaces. Thus evo-loop appears to be the first convincing example of an evolutionary process occurring in cellular automata. An asynchronous version is given below.

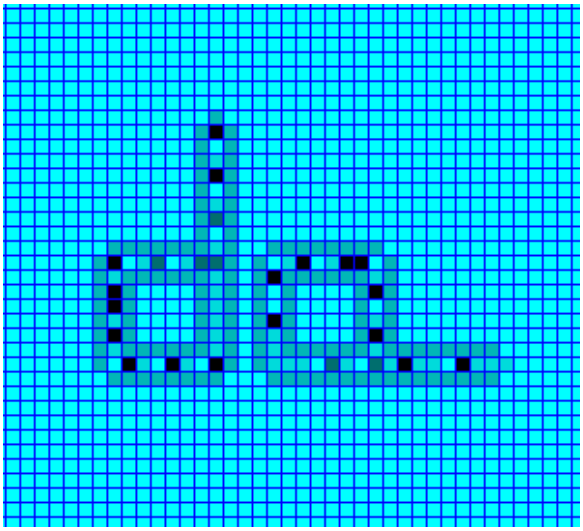

Figure 1: Asynchronously Self-Replicating Loop after First Reproduction. (from (Nehaniv 2002 (in press)).

Asynchronous Langton, Byl, Reggia et al. Self-Reproducing Loops

Using the method above it is now straightforward to construct asynchronous cellular automata that exhibit self-reproduction. Taking any of the models of Von Neumann (1966), Codd (1968), Langton (1984, 1986), Byl (1989), Reggia et al. (1993), Sayama (1998a, 1998b, 1999), etc., we merely apply the construction above. The first implemented example of self-reproduction in asynchronous cellular automata was given by applying our construction to Langton's self-reproducing loop (Nehaniv 2002 (in press)); see Figure 1. In this paper, we give another asynchronous implementation of Langton's example together with asynchronous examples of Byl's, Reggia and Chou's, and Sayama's examples.

Using the rule of Langton (1984) for a synchronous cellular automaton exhibiting self-reproduction, we derive by the method above an asynchronous cellular automaton with $192 (= 3 \times 8^2)$ states possible in the local automata at each node. Using random asynchronous update of the nodes we achieved an implementation of self-reproduction in an asynchronous cellular automaton. Figure 1 shows the state of the cellular automaton near the end of the first reproduction cycle: a sheath of cells encloses an asynchronously counterclockwise circulating stream of instructions to extend a construction arm and turn left. The instructions are copied as they flow through a fork in the sheath. After this stream has been executed four times, an offspring asynchronous loop is present in the cellular space (right in Figure 1).

Simplifications of Langton's loop by Byl (1989) and Reggia et al. (1993) reduced the size of the self-reproducing loop and are illustrated with asynchronous implementations in Figure 2.

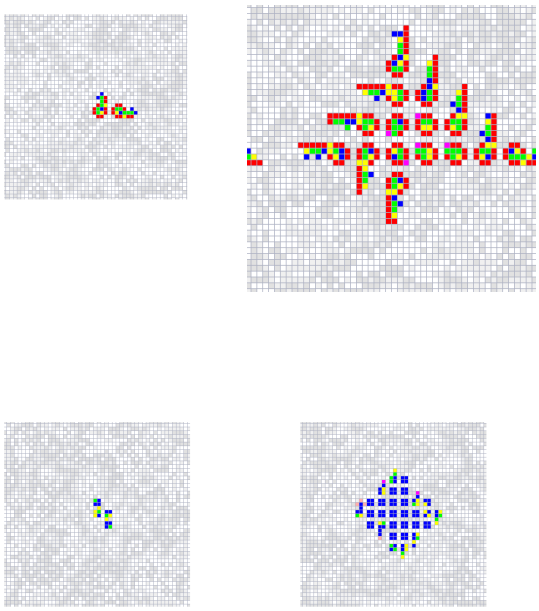

Figure 2: Asynchronous Versions of Byl's (top panels) and Reggia et al.'s (lower panels) Small Self-Replicating Loops after One (left) and Several Reproductive Cycles (right). Differences in shading are indicative of temporal waves in the asynchronous cellular space (see text).

Evolution in Asynchronous Cellular Automata

In this section we review some obstructions to evolution in cellular automata models such as Langton's self-reproducing loop, and Sayama's introductions of (1) self-dissolution (apoptosis: "programmed cell death" or "suicide" by individual loops that cannot replicate anymore) and of (2) variability into cellular automata replicators. We discuss the how this properties relate to evolvablility and give illustrations of the first asynchronous implementations of these as we proceed.

Finite Resources and an Obstruction to Turn-over of Generations

In Langton's original example (1984) a finite space is filled with loops which then become inert, and the space cannot be reused by descendants of the original ancestral loop.

Figure 3 shows a space in which self-reproduction is about to halt, since all available space will soon be filled with the inert husks of replicators, that are no longer active. The structure is reminiscent of a coral reef, and could have continued to expand indefinitely in an unbounded space. However, if as in this case, the world is finite, all available space is filled and cannot be reclaimed for further replication to occur. This occurs for the asynchronous implementation of Langton's loop (see Figure 3).

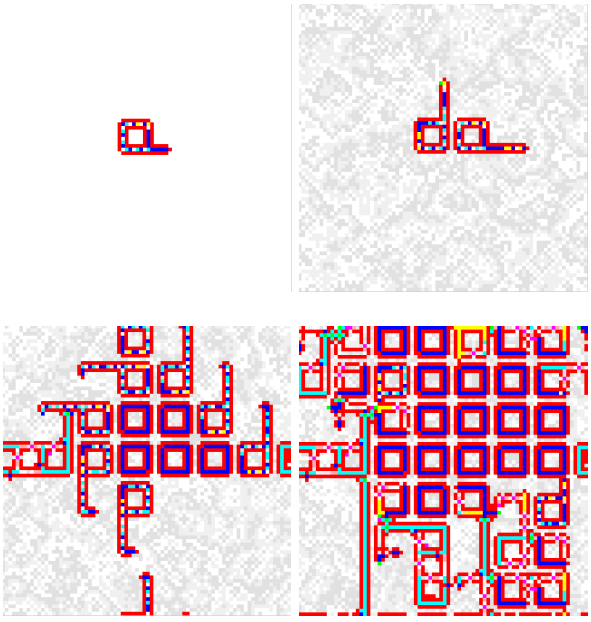

Figure 3: Asynchronously Self-Replicating Loop Population Nearly Exhausting Available Space Resources (Langton Loop, 4 snapshots from a single run; toroidal topology). Replicants with monochromatic 'cores' have ceased to reproduce due to lack of available space, but persist and thus obstruct their descendants from reproducing.

Programmed Cell Death

Sayama (1998) introduced self-dissolution into the study of self-reproduction in cellular automata by modifying Langton's construction. This allows a loop to dissolve – i.e. its cells return to a quiescent state – if the available surronding space is already occupied so that no more offspring can be produced near the loop. A new "dissolution" state is added to the local finite automata states which the local automaton moves into if reproduction is no longer possible and triggers cell-death in neighoring cells as it moves around within the sheath; this structural dissolution of the entire loop. Thus space can be freed up and later reused by descendants of the ancestral (or potentially other) self-reproducers loops. This occurs for the asynchronous implementation of Sayama's structurally dissolvable loop (see Figure 4).

Any evolutionary process without such turn-over of generations in a finite space would necessary halt. Sayama's introduction of self-dissolution eliminates this obstruction to evolution in cellular automta.

Evolvability, Robustness, and Hertiable Variability

Conrad (1980) discussed the importance of genetypic robustness for evolvability in relation to mutational buffering and extra-dimensional. Kirschner and Gerhart (1998) have discussed the importance for biological evolvability of robustness to environmental and phenotypic variation.

Sayama also extended Langton's cellular automata rules and his own structural dissolution rule to allow replication and survival of the self-reproducing loops to occur in many more possible situations than was possible for Langton's case. At the same time, he introduced simple variability in size and contents. Combined with the self-dissolution mechanism, the added robustness transformed Langton's recipe for self-replication into an system that was evolution ready, "Evo-loop" (Sayama 1999). This result depends on free-up finite resources (space) to permit a turn-over of generations and on the introduced robustness of replicators to environmental, phenotypic, and heritable variation. Our asynchronous version is shown in Figure 5.

In a finite cellular space, this provides all the requirements, identified by Darwin, for a rudimentary evolutionary process: populations of self-replicators with heredity, variability, turn-over of generations with bounded resources (Sayama 1999). Since replicated offspring resemble their parent and the environment has the same cellular automata "physics" as the environment of the parent, reproductive success tends to be heritable as

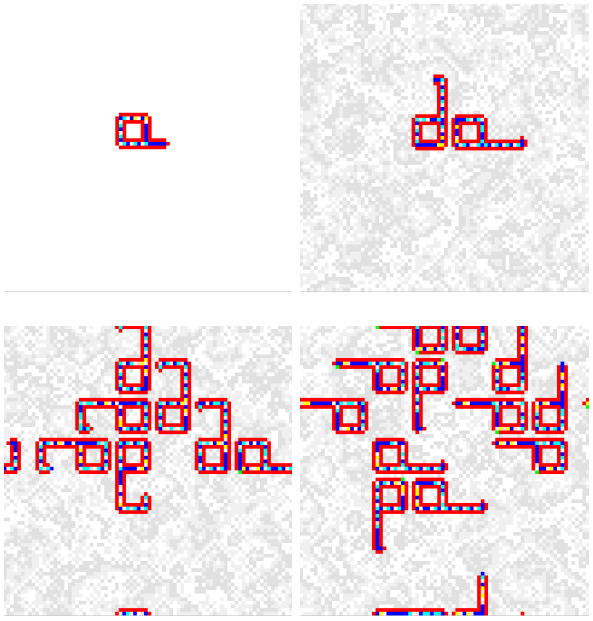

Figure 4: Asynchronous Version of Sayama's Structurally Dissolvable Self-Reproducing Loop. Space is liberated by "programmed cell death" and can be reused by descendents of the original loop (4 snapshots of a single run; toroidal topology).

well in the Evo-loop model. However, due to the changing state of the cellular space, with loops of various sizes reproducing and colliding within it, the "biotic" enviroment of offspring may differ from that experienced by a parent. Thus, as for organic evolution, reproductive success in Evo-loop is generally but not completely heritable. In Sayama's evoloop model, hertiable variation may arise when collisions occur between loops giving rise to new self-reproducing loops which may differ in size from the 'parents' involved in the collision. That is, direct interaction of the phenotypes produces heritable variability. Interestingly, given the cellular update rule and initial configuration in a cellular array, this example of evolution is completely deterministic, although Sayama has also experimented with stochastic variants. [3]

The implementation described here and illustrated in Figure 5 thus comprises the first example of evolution occurring in an asynchronous cellular automata space.

Summary and Conclusions

Previous results (Nehaniv 2002 (in press)), exhibited the first implemented example of self-reproduction in asynchronous cellular automata, and demonstrated that,

[3] On-line illustrations and movies of implementations of Langton's and Sayama's synchronous can be accessed at Sayama's webpage:
http://necsi.org/postdocs/sayama/sdsr/.

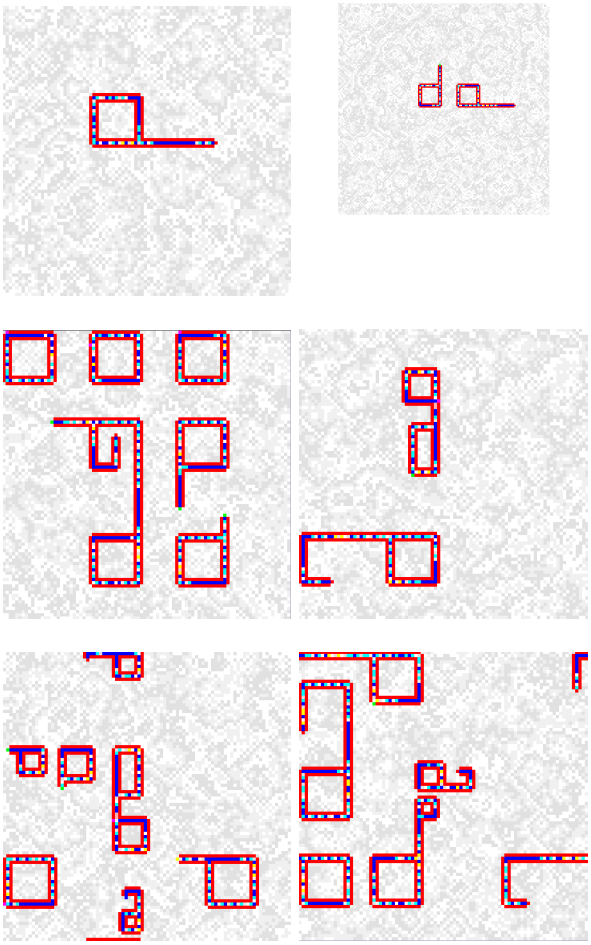

Figure 5: Evolution in Asyncrhonous Cellular Automata: Asynchronous Version of Self-Reproducing Evoloop (6 snapshots (not all from the same run); toroidal topologies). Heritable variability of characteristics of individuals (e.g. loop size) entails that this is an evolutionary system.

in principle, other cellular models of self-reproduction, universal computation and universal construction can be implemented in asynchronous cellular automata by applying our construction to Von Neumann's self-reproducing configuration. Moreover, universal computation can also be realized by our method has we have implemented Conway's Game of Life using asynchronous cellular automata.

Here, we have demonstrated the systematic implementation of asynchronous cellular automata that fully emulate the behavior of any synchronous cellular automata. This was used to give numerous examples of self-reproduction in asynchronous cellular automata. Similarly, implementing the asynchronous version of Sayama's evo-loop we also have created the first instance of evolution in a population of self-replicators in an asynchronous cellular automaton.

In general, most results about synchronous cellular automata carry over automatically now to the asynchronous realm. Thus these methods free those using cellular automata models of computation, self-reproduction, and evolution from the restriction of synchronous global update. With asynchronous (e.g. random) update, the same results are attained as guaranteed by the theorem of (Nehaniv, 2002 (accepted)), as described above.

Fault-tolerance and self-repair in the asynchronous cellular automata is a natural next direction to explore. A generalization of the methods presented here to cellular automata networks of variable size and shape, i.e. allowing dynamic growth or changing topology of the cellular space, would also a desirable development as it would help us study whether it is possible for an asynchronous cellular automata space to expand under the activity of, for example, evolving self-reproducers. This may provide the basis for physical realizations of asynchronous self-replication, and related scientific and technological applications.

Acknowledgements

The implementations here were carried out by author applying the method explained here for constructing corresponding asynchronous cellular automata to a (synchronous) java applet written by Eli Bachmutsky (Batchmutsky 1999), which gives a classical synchronous implementation of the rules designed by Byl, Reggia et al., Langton, and Sayama. The earlier asynchronous implementations of the Langton loop and Conway's Game of Life mentioned here and in (Nehaniv 2002 (in press)) were obtained using the same method as described in the text applied to adapting synchronous java applet code for Conway's Life by Andreas Ehrencrona and incorporating the synchronous Langton's rules as given in David I. Bell's `xca.c` implementation of Langton's loop.

The source and compiled code of the author's derived asynchronous implementations is available on-line at http://homepages.feis.herts.ac.uk/∼nehaniv/async/

References

Bachmutsky, E. Self-replicating Loops and Ant Java Applet, applet and source code available on-line at: `http://necsi.org/postdocs/sayama/sdsr/java/`, 1999.

Byl, J. "Self-Reproduction in Small Cellular Automata", *Physica D*, Vol. 34, pp. 295–299, 1989.

Burks, A. W. (ed.), *Essays on Cellular Automata*, University of Illinois Press, Urbana, Illinois, 1970.

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

Conrad, M. "The Geometry of Evolution", *BioSystems*, Vol. 24, 61–81, 1990.

Freitas, R. A., Jr., and Gilbreath, W. P. (eds.) *Advanced Automation for Space Missions, Proceedings of the 1980 NASA/ASEE Summer Study; sponsored by the National Aeronautics and Space Administration and the American Society for Engineering Education.* NASA Conference Publication 2255. On-line extracts at http://www.zyvex.com/nanotech/selfRepNASA.html and http://www.islandone.org/MMSG/aasm/ .

Kirschner, M. and Gerhart, J. "Evolvability", *Proc. National Academy of Sciences*, Vol. 95, pp. 8420–8427, July 1998.

Langton, C. G. "Self-reproduction in Cellular Automata", *Physica D*, Vol. 10, pp. 135–144, 1984.

Langton, C. G. "Studying Artificial Life with Cellular Automata", *Physica D*, Vol. 22, pp. 120–149, 1986.

Lohn, J. D. "Self-replicating Systems in Cellular Space Models". In C. L. Nehaniv (ed.), *Mathematical and Computational Biology: Computational Morphogenesis, Hierarchical Complexity, and Digital Evolution.* Vol. 26 in Lectures on Mathematics in the Life Sciences, Providence, Rhode Island, American Mathematical Society, pp. 11–30, 1999.

Lohn, J. D. and Reggia, J. A. "Automatic Discovery of Self Replicating Structures in Cellular Automata", *IEEE Transactions on Evolutionary Computation*, Vol. 1 No. 3, pp. 165–178, 1997.

Nehaniv, C. L. "Self-Reproduction in Asynchronous Cellular Automata", Proc. 2002 NASA/DoD Conference on Evolvable Hardware (15-18 July 2002, Washington DC, USA), IEEE Computer Society Press, 2002 (in press).

Nehaniv, C. L. "Asynchronous Automata Networks Can Emulate Any Synchronous Automata Network", *International Workshop on Semigroups, Automata, and Formal Languages (June 2002 – Crema, Italy)*, 2002 (accepted). *journal version in preparation.*

Reggia, J. A., Armentrout, S., Chou, H. H., Peng, Y., "Simple Systems that Exhibit Self-Directed Replication", *Science*, Vol. 259, pp. 1282–1288, 1993.

Sayama, H. *Constructing Evolutionary Systems on a Simple Deterministic Cellular Automata Space*, Ph.D. Dissertation, Department of Information Science, Graduate School of Science, University of Tokyo, December 1998.

Sayama, H., "Introduction of Structural Dissolution into Langton's Self-Reproducing Loop", *Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life*, C. Adami, R. K. Belew, H. Kitano, and C. E. Taylor, eds., pp. 114–122, 1998, MIT Press. On-line material at: http://necsi.org/postdocs/sayama/sdsr/

Sayama, H. "A New Structurally Dissolvable Self-Reproducing Loop Evolving in a Simple Cellular Automata Space", *Artificial Life*, Vol. 5, no. 4, pp. 343–365, 1999.

Schönfisch, B. and de Roos, A. "Synchronous and Asynchronous Updating in Cellular Automata", *BioSystems*, Vol. 51, 123–143, 1999.

Sipper, M. The Artificial Self-Replication Page http://www.cs.bgu.ac.il/∼sipper/selfrep/

Szathmáry, E . "Chemes, Genes, Memes: A Classification of Replicators". In C. L. Nehaniv (ed.), *Mathematical and Computational Biology: Computational Morphogenesis, Hierarchical Complexity, and Digital Evolution*. Vol. 26 in Lectures on Mathematics in the Life Sciences, Providence, Rhode Island, American Mathematical Society, pp. 1–10, 1999.

Von Neumann, J. *Theory of Self-Reproducing Automata*. Edited and completed by A. W. Burks, University of Illinois Press, Illinois, 1966.