Rocha, Luis M. [2004]. "Evolving Memory: Logical Tasks for Cellular Automata". Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9). Boston, Massachusetts, September 12-15th 2004. In Press. LAUR 04-0628.

Evolving Memory: Logical Tasks for Cellular Automata

Luis Mateus Rocha
Modeling, Algorithms, and Informatics Group
Los Alamos National Laboratory, MS B256
Los Alamos, NM 87545, USA
e-mail: rocha@lanl.gov, URL: http://www.c3.lanl.gov/~rocha

Abstract

We present novel experiments in the evolution of Cellular Automata (CA) to solve nontrivial tasks. Using a genetic algorithm, we evolved CA rules that can solve non-trivial logical tasks related to the density task (or majority classification problem) commonly used in the literature. We present the particle catalogs of the new rules following the computational mechanics framework. We know from Crutchfield et al (2002) that particle computation in CA is a process of information processing and integration. Here, we discuss the type of memory that emerges from the evolving CA experiments for storing and manipulating information. In particular, we contrast this type of evolved memory with the type of memory we are familiar with in Computer Science, and also with the type of biological memory instantiated by DNA. A novel CA rule obtained from our own experiments is used to elucidate the type of memory that one-dimensional CA can attain.

1. Background

An important question for both Cognitive Science and Artificial Life is that of the origin of symbols from the dynamic interaction of many components. By symbols we mean memory structures which can be used to store and manipulate information used to produce and re-produce some behavior. There are currently two main camps in Cognitive Science and Artificial Life with very distinct approaches to the concepts of symbols, representations and even information: the representationalist and dynamicist camps. The first regards information as the most important feature of Life and Cognition, emphasizing genotype/phenotype relations (e.g. Langton, 1989) and internal representations of the environment (e.g. Pinker, 2002), respectively. The second, in its radical form, regards information as an unnecessary concept to explain Life and Cognition. Instead, explanations based solely on dynamical systems theory are preferred (e.g. [Beer, 1995]).

This feud has been discussed in detail in (Rocha and Hordijk, 2004) where we emphasized that both of these camps, while choosing to work either with symbols or equations of dynamics, fail to approach the study of the origin of memory, symbols, representations, information, and the like, from dynamics. As also detailed (Ibid) the biological organization clearly uses the genotype as a type of memory which can be accessed very much like Random Access Memory in a computer. Thus, studying the origin of memory from a dynamic milieu should be a fundamental goal of Artificial Life.

Indeed, we proposed that using the known living organization as a guideline, artificial life can become the ideal laboratory to study the problems of origin of memory (Ibid), as well as the relative advantages of alternative forms of implementing memory in evolving systems (Rocha, 2001). This research program follows directly from previous work on complex systems, where Mitchell (1998) and Rocha (1998, 2000) have proposed a set of experiments with Cellular Automata as paradigmatic examples of the process of emergence of representations from a dynamical substrate.

2. Evolving Cellular Automata

2.1 Nontrivial Tasks

One-dimensional cellular automata (CA) consist of a one-dimensional *lattice* of N identical cells, each a state-determined automaton with k possible states; here k=2. Let $s_i(t)$ denote the state of cell i at time t, with $s_i \in \{0,1\}$. Each cell is "connected" to $2r$ other cells which we think of as its neighborhood of *radius* r. Usually, periodic boundary conditions are employed, i.e., cells 1 and N are each other's neighbor. In homogeneous CA, each cell's automaton is defined by the same update rule ϕ which takes as input the cell's neighborhood state, $\mu_i = (s_{i-r}(t),..., s_i(t),..., s_{i+r}(t))$, and outputs the new state of the cell at time $t+1$: $s_i(t+1) = \phi(\mu_i)$.

The initial conditions for a CA are defined by a particular *initial configuration* (IC) of (typically random) cell states. In discrete time steps, all the cells subsequently update their state synchronously according to the update rule ϕ. This update rule can be represented by a *lookup table* with one entry for each of the 2^{2r+1} possible neighborhood configurations μ, and their corresponding output values for $s(t+1)$. Here we use CA rules with r=3, thus the lookup table contains 128 entries: there are 2^{128} such rules.

Das et al (1994) such CA rules using genetic algorithms (GA) to solve several non-trivial computational tasks, such as the *density classification* task (a.k.a majority classification problem). Each CA rule is encoded in the GA as a 128 bit string, where each bit encodes the outcome of each entry in the rule's lookup table. The goal of the density task is to find a CA that decides whether or not the IC contains a majority of 1s (i.e., has high density). Let ρ_0 denote the density of 1s

in the IC. If $\rho_0 > 1/2$, then within M time steps the CA should reach the fixed-point attractor configuration of all 1s (i.e., all cells in state 1 for all subsequent iterations); otherwise, within M time steps it should reach the fixed-point configuration of all 0s. Since the CA cells have access only to local interactions (with other cells within radius r), this task requires the CA to propagate information across the lattice in order to achieve global coordination. In this sense, this is a nontrivial task.

The unbiased performance $\mathcal{P}_{N,I}(\phi)$ of a CA rule ϕ on a given task is defined as the fraction of I randomly generated ICs for which ϕ reaches the desired behavior within M time steps on a lattice of length N. Here, we employ $N = 149$, $M = 2N$ and $I = 10^5$.

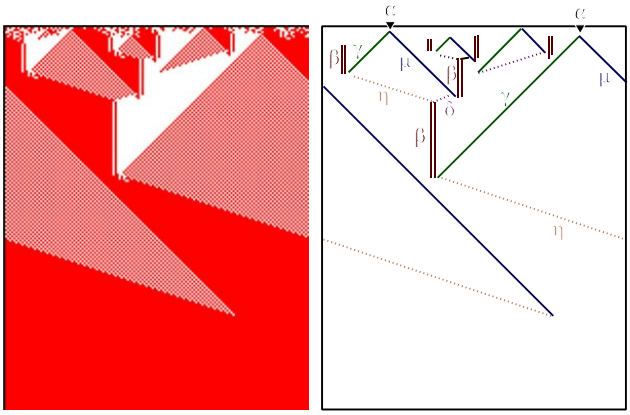

Figure 1: (a) Space-time diagram for ϕ_{DMC} given a random IC with a majority of dark cells. The rule correctly classifies the IC in 141 iterations. (b) Space-time diagram with regular domains filtered out, depicting particles and their interactions after the initial transient is removed.

Figure 1.a shows the space-time diagram of one of the CA rules evolved by Das et al (1994): ϕ_{DMC}. This rule is defined by a 128-bit string as discussed above, in hexadecimal: 0504058705000F77037755837BFFB77F. The lattice is started with a random IC (0 is denoted by white, 1 by dark). Each row in the space-time diagram shows the CA lattice at a particular time step t; time increases down the page.

2.1 Particle Interactions

The large regular, relatively stable regions in the space-time diagram are called *regular domains*. Examples in figure 1.a are the all white, all dark, and checkerboard (alternating white and dark) regions. Crutchfield et al (2002) refer to the boundaries between domains as *particles*. Domains and particles were defined formally in the *computational mechanics* framework (Hanson and Crutchfield, 1992), which provides a way of suppressing (by way of filtering out) the domains in a space-time diagram, making the particles more explicit. An example of this filtering process for ϕ_{DMC} is shown in figure 1.b.

Particles are localized patterns that behave according to certain rules. For example, they have a certain constant velocity at which they move through the lattice. Velocity is defined as the number of cells the particle moves at each iteration of the CA; it is positive if the movement is to the right of the lattice, and negative to the left. Particles also interact with one another according to deterministic rules. These rules and the velocities of particles are referred to as a *particle catalog* for a given CA. Typically, such a catalog is based on a small number of particles, α, β, δ, γ, η, and μ, and a small number or rules such as: $\beta + \gamma \rightarrow \eta$, meaning that when particles β and γ collide, the η particle results. The two-particle interaction catalog for ϕ_{DMC} is shown on table I.

Table I: Catalog of regular domains, particles and particle interactions for rule ϕ_{DMC}

Regular Domains	$\Lambda^0 = \{0+\}$, $\Lambda^1 = \{1+\}$, $\Lambda^2 = \{(01)+\}$	
Particles (velocities)	$\alpha \sim \Lambda^0 \Lambda^1$ (–), $\beta \sim \Lambda^1 \Lambda^0$ (0), $\gamma \sim \Lambda^0 \Lambda^2$ (-1), $\mu \sim \Lambda^2 \Lambda^1$ (1), $\delta \sim \Lambda^2 \Lambda^0$ (-3), $\eta \sim \Lambda^1 \Lambda^2$ (3)	
Observed Interactions	**decay**	$\alpha \rightarrow \gamma + \mu$
	react	$\beta + \gamma \rightarrow \eta$, $\mu + \beta \rightarrow \delta$, $\eta + \delta \rightarrow \beta$

Particles transfer information about properties of local regions across the lattice to distant sites. Crutchfield, et al (2002) defend that particle collisions are the loci of information processing and result in either the creation of new information in the form of other particles or in annihilation.

3. Emergent Memory in Evolving Automata

Most CA rules evolved tackle the density task by block-expansion, that is, by expanding large neighborhoods of either "1" or "0" states in the initial configuration. But, unlike rule ϕ_{DMC}, they lack the ability to integrate local information to produce an accurate global result. Indeed, the performance of block-expansion rules is quite inferior to ϕ_{DMC}, which grants an obvious evolutionary advantage to latter.

Furthermore, whereas the ϕ_{DMC} rule maintains a similar level of performance for larger lattices, block expansion rules performs very close to random guessing. Thus, the CA rule ϕ_{DMC} is indeed capable of effectively integrating information from local areas of large lattices, whereas block-expansion rules are not (Crutchfield and Mitchell, 1995).

3.1 Memory and Communication

The CA space-time domains, being regions that are "space- and time-translation invariant" (Crutchfield et al ,2002, page 17) can be seen as *memory* structures. Each domain is defined

by a cyclic repetition of strings (words) from its regular language (the 0's and 1's of the CA) in space and time. Unless otherwise perturbed, these domains retain their cycles in space and time. For instance, for the CA rule ϕ_{DMC} (see figure 1) we observe the three domains specified in Table I. Λ^0 and Λ^1 refer to the two desired outcomes for the density task, while Λ^2 refers to an intermediate domain used in the process of integrating lattice information and producing the final outcome.

Indeed, the introduction of intermediate domains in CA with intricate particle systems, is their key difference from block-expansion rules, which simply propagate the final outcome domains Λ^0 and Λ^1. Here we define CA with intricate particle systems, as those that employ at least one intermediate domain.

Domains interact by one taking over the other or by establishing an inalterable border. In either case, their interaction defines the particles described in section 2. In the first case, we obtain particles (e.g. μ and γ in figure 1.b) which propagate in the direction of the receding domain, at greater or lesser velocity, while in the second case we obtain a particle (e.g. β in figure 1.b), with zero velocity, which maintains the same lattice position in time, creating a vertical line in the space-time diagram.

The CA with intricate particle systems use the intermediate domains as *memory* stores for intermediate results, and the particles to *communicate* these results across the lattice. Furthermore, the particle interaction rules are used to *integrate* the information stored in the various intervening domains to ultimately produce a final homogeneous lattice state. The inclusion of an additional memory state in ϕ_{DMC}, establishes a more effective means to solve the density task in a distributed manner.

3.2 Building up Memory: Logical Tasks

The role of domains as emergent memory structures used for distributed information processing via the particle interaction scheme can be further appreciated as we notice that memory can be built upon in order to solve more complicated tasks. Rocha (1998,2000), conducted some additional experiments to evolve CA's which solve more than one task. The goal was the evolution of CA rules with radius 3 to solve both the density task and some related, but more complicated, logical tasks (Ibid).

From a machine learning perspective, the idea of evolving CA rules to solve more than one task, especially tasks that at times depend on conflicting demands as discussed below, is rather odd. But the idea of these experiments was to see if one could evolve CA rules that can use the evolved particle system as a more flexible computational system. Such motivation has been previously discussed in detail (Ibid). Here we present a novel analysis of the particle systems evolved for these tasks.

To implement logical tasks the CA lattice is functionally divided in two halves (the center cell is not used): *A* and *B* (figure 2.a). Here we describe results for the logical AND task only, which depends on the density value of the *A* and *B* lattice halves. Each half is interpreted as a separate logical variable in traditional logical operations. A variable is "1" if there is a majority of "1" cells in its respective lattice half, and "0"otherwise. Notice that since the boundary conditions of the lattice are periodic, this lattice has two boundaries between the two variables (halves) *A* and *B* (figure 2.b). The cells on the neighborhood of these boundaries compute their values from cells in both halves. However, since we are looking for global integration across the lattice, the local errors at the boundaries are not too relevant, especially as lattices grow in size.

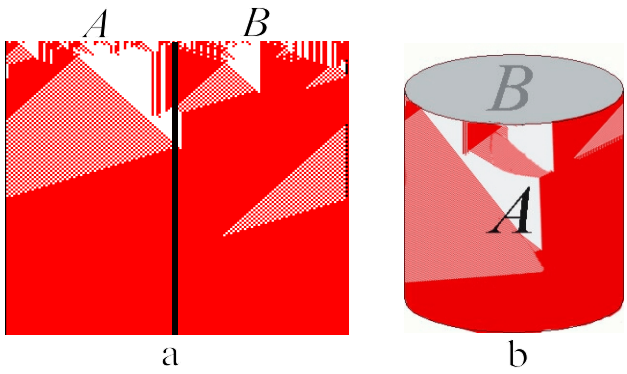

Figure 2: Implementation of logical tasks in one-dimensional CA. (a) The lattice is divided into two halves *A* and *B*, each interpreted as a separate logical variable whose value is "1" if it contains a majority of cells in state "1", and "0" otherwise. (b) The space-time lattice is periodic.

The AND task is thus related to the density. It differs from the density task for some of the cases when the two halves of the lattice have opposing densities (see details below). The gist of the logical tasks is that they should ideally perform the density task in each half, and then integrate the results appropriately.

Several rules were evolved with a GA, with elite selection, whose initial population of 100 individuals was composed of 20 individuals encoding some of the best rules evolved so far for the density task, including ϕ_{DMC}, $\phi_{ABK,}$ as well as a rule evolved by Juillé and Pollack (1998), and others, plus 80 randomly generated individuals. The fitness function used in this GA was calculated by presenting each rule with 100 different IC's, 50 to be analyzed by the density task, and the other 50 by the AND task. The 50 IC's presented to the density task had their density of "1's" uniformly distributed over the unit interval (just as the experiments of Das et al (2004)). The 50 IC's presented to the AND task were similarly biased to a uniform distribution of lattices where for 50% of lattices the density of both halves was "1", and for the other 50% the density of at least one of the halves *A* or *B* was "0". More specifically, in 50% of the lattices the density of "1's" in both halves is computed from a uniform distribution

on [0.5, 1], and for another 50% of the lattices, the density of "1's" in at least one half (*A* or *B*) is computed from a uniform distribution on [0, 0.5). In this last case, 1/3 of the time both halves have majority density "0", and 2/3 of the time only one of the halves has majority density "0". Notice that if we were to use an unbiased distribution of lattices, only 25% of the time would the case of both halves having density "1" be generated, thus making rules that always tend to "0" too favorable in the evolutionary process.

The AND task can only differ from the density task when the density of both halves is distinct, and the density of "1's" in the majority "1" half is greater than the density of "0's" in the majority "0" half. Without loss of generality, assume that the density of half *A* is predominantly 0, this means that the density of "0's" in half *A*, $d_A(0)$ is uniformly distributed in [0.5, 1]. Assume also that the density of half *B* is predominantly 1, this means that density of "1's" in half *B*, $d_B(1)$ is also uniformly distributed in [0.5, 1]. In this setting, the density task conflicts with the AND task only when $d_A(0) < d_B(1)$, which happens on average half the time each lattice half has opposed densities.

In our experiments, to compute fitness, 50% of the lattices were presented to the density task, and 50% to the AND task. The density of "1's" of lattices presented to the density task is uniformly distributed in the unit interval. Thus, the value of density used to generate each half of these lattices is equal and also uniformly distributed in the unit interval. Only when this value of density is in the near neighborhood of 0.5, would we find lattice halves with opposing densities – and only in half of those would the tasks conflict. Therefore, for the 50% of lattices presented to the density task, seldom will we find lattices where the density task should lead to a different result than the AND task.

Regarding the 50% of lattices presented to AND task to compute fitness, half are biased to produce both lattice halves with majority density "1". Of the remaining half, only 2/3 produce lattices with halves of opposing densities, that is a total of 1/3 of the lattices presented to the AND task. Since only in half of the latter do the rules conflict, in our experiment, only 1/6 of the lattices presented to the AND task conflict with the density task. In the case of our fitness function, this means on average 8.3 lattices.

Even though the tasks conflict in only a small fraction of lattices, the unbiased performance on the AND task of the best rules previously evolved for the density task was much smaller than that of the new CA rules evolved in our experiments. This means that solving the density task alone was not the best strategy found by the genetic algorithm – neither was solving the AND task alone. Indeed, several CA rules were evolved that can perform very well simultaneously on the density task and on the AND task. Notice that to calculate the unbiased performance, IC's are randomly produced with independent values for each cell with probability 0.5. This means that the density of "1's" in the IC's used to calculate performance of evolved rules tends to be around 0.5, where we find greater conflict between the two tasks. (performance details in Rocha ,1998, 2000, Rocha and Hordijk, 2004).

3.3 The Evolved CA Rules

The significance of having rules that can perform well on more than one task was discussed in (Rocha ,2000). What we want to highlight here is the manner in which evolved CA particle systems dealt with the different requirements for information integration across the lattice demanded by the AND task. Because the logical tasks divide the lattice into two halves, we expected evolved CA rules to create additional domains and particles which would behave more like static, local memory stores, whose information could be accessed at a latter time as needed.

Table II: Catalog of regular domains, particles and particle interactions for rule ϕ_{AND}

Regular Domains	$\Lambda^0 = \{0+\}$, $\Lambda^1 = \{1+\}$, $\Lambda^2 = \{(01)+\}$, $\Lambda^3 = \{(110)+\} \vee \{(001)+\}$	
Particles (velocities)	$\alpha \sim \Lambda^1\ \Lambda^0$ (–) , $\beta \sim \Lambda^0\ \Lambda^1$ (0), $\beta` \sim \Lambda^0\ \Lambda^3$ (0), $\beta`` \sim \Lambda^3\ \Lambda^1$ (0), $\gamma \sim \Lambda^1\ \Lambda^2$ (-1), $\delta \sim \Lambda^2\ \Lambda^3$ (-3), $\epsilon \sim \Lambda^1\ \Lambda^3$ (3), $\eta \sim \Lambda^0\ \Lambda^2$ (3), $\mu \sim \Lambda^2\ \Lambda^0$ (1), $\nu \sim \Lambda^3\ \Lambda^0$ (-3), **Note:** The domain combinations $\Lambda^2\ \Lambda^1$, and $\Lambda^3\ \Lambda^2$ were not observed as stable boundaries or particles.	
Observed Interactions	**decay**	$\alpha \rightarrow \gamma + \mu$
	react	$\beta + \gamma \rightarrow \eta$, $\beta`` + \gamma \rightarrow \nu + \eta$, $\mu + \beta \rightarrow \delta + \beta``$, $\mu + \beta` \rightarrow \delta$ , $\eta + \delta \rightarrow \beta`$, $\gamma + \delta \rightarrow \epsilon$, $\epsilon + \nu \rightarrow \gamma + \mu$
	annihilate	$\beta` + \nu \rightarrow \Lambda^0$, $\eta + \mu \rightarrow \Lambda^0$, $\epsilon + \beta`` \rightarrow \Lambda^1$

Indeed this is what we observed in the best CA rule for the AND task, ϕ_{AND} (005F1053405F045F005FFD5F005DFF5F) – performance details in (Rocha and Hordijk, 2004). The strategy of this rule builds on rule ϕ_{DMC} by creating an additional intermediate domain, which keeps local lattice information without expanding. The particle catalog of ϕ_{AND} is detailed in Table II. Figures 4 shows a space-time diagram for this rule, with particle interaction schematics.

The most striking feature of the particle catalog of rule ϕ_{AND} is the existence of several particles with zero velocity. These are particles which remain in the same position in the lattice until other particles collide with them. Whereas rule ϕ_{DMC} had only one particle with zero velocity (β), rule ϕ_{AND} produces three such particles (β, $\beta`$, and $\beta``$). We named all these particles β, to highlight the similarity of their behavior with particle β of rule ϕ_{DMC}.

Both particles $\beta`$ and $\beta``$ exist due to the fourth domain Λ^3 introduced by rule ϕ_{AND}. This domain does not expand into

final domains Λ^0 and Λ^1 , so the respective particles with these domains have zero velocity. It only expands into intermediate domain Λ^2 with particle δ. We note that Λ^3 typically exists as {(110)+} but it can also exist as{(001)+}. We consider these patterns to be the same domain because they behave in exactly the same manner in terms of particle interactions, and are in effect interchangeable.

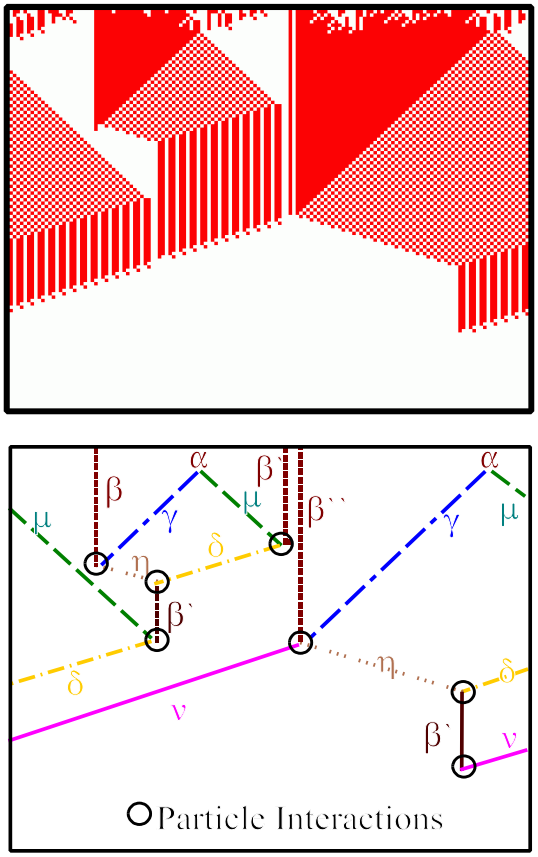

Figure 3: Space-time diagram and respective particle interactions for ϕ_{AND} given a random IC, leading to an all "0" lattice.

DomainΛ^3 functions as a static intermediate memory store. In rule $\phi_{DMC,}$ without Λ^3, when the particles involving domain Λ^2 collide with others, the result is always one of the final domains Λ^0 or Λ^1, while in rule ϕ_{AND} some collisions result in the additional intermediate domain Λ^3. This way, domain Λ^3, contained by static particles β` or β``, preserves an intermediate result without spreading it into neighbor domains. The intermediate result can later be integrated with particles from other lattice regions: a collision with particle ν results in the all "0" domain Λ^0, and a collision with particle ϵ results in the all "1" domain Λ^1.

The existence of the fourth domain and its static particles is particularly useful for the logical tasks ("AND" in this case). Because two arbitrary halves are defined, the task encourages the evolution of rules that can "hold" intermediate results in one part of the lattice to be integrated with those from another part. Indeed, the logical task can be better executed when a more *static type of memory* is produced to hold intermediate results, which in this case is implemented by domain Λ^3 and its static particles β` and β``.

4. The Nature of Evolved Memory

We can think of the set of particle interaction rules that emerge in the evolving CA experiments, as a process that maps between the random initial state of the CA lattice (IC), into a final desired state for the task. Crutchfield et al (2002) regard this process as a computation that produces a final outcome from the IC input. As we detail below, we do not see this process as a computation, but the individual particles are certainly the elements in the space-time behavior of the CA which communicate information across the lattice: the loci of information processing (Ibid). Therefore, the collection of particle interactions in space-time, is a dynamic process of integration of the information carried by each of the individual particles into a final domain.

The type of memory the CA domains implement is quite different from the concept of random-access memory (RAM) we are familiar with in universal computers, and also quite distinct from the inert type of memory that DNA grants living organisms (Rocha and Hordijk, 2004).

RAM is a type of memory that can be accessed at any time, and whose value is independent of dynamics. This is the same as saying that the value of the memory is the same independently of the rate of access to it. When a computer stores the value of a variable in a memory store (e.g. the tape in a universal Turing machine), that value remains unchanged when accessed, and the speed of the computer to access the memory and perform computations also does not change it. Similarly, a computation is a process of integrating memory in store, with algebraic and logical operations. But speed of the computer does not change the value of the computation: 2+2=4 in any computer.

Clearly this does not happen with the domains and particles of the evolved CA. Particles have a velocity, and the resulting domains of all particle interactions in space-time depend on when the particles meet each other. If the particles start from different locations in the lattice, even preserving lattice density, they may collide differently and produce a different outcome for the tasks we studied here. It is as if 2+2=4, only when 2 and 2 meet at the right time. This is why we do not see the process of particle interactions as a full-blown computation, but rather a process of information integration based on dynamic memory, rather than RAM.

It was because of this issue that we created the logical tasks. In this case, the evolved CA came as close as possible to creating static (RAM-like) memory stores. Indeed, the fourth domain Λ^3 created by rule ϕ_{AND}, is a domain that preserves its memory without spreading it into the final outcome domains Λ^0 and Λ^1. In a sense, it keeps its memory until it is accessed. The several particles β created by this rule

have zero velocity, therefore they preserve the same information until a particle of non-zero velocity collides with them. In this sense, the domain functions more like a traditional memory store.

However, the information stored is still not separated from the dynamics. The domains are not rate-independent like RAM nor inert in the sense that DNA is (Rocha and Hordijk, 2004). It is by virtue of their dynamics, the way their particles collide, that information is expressed. Conversely, in DNA or RAM, information is read out by "third-party" machinery, without destroying or reacting with the memory. So while the β particles of the evolved CA were able to create static memory stores, these are still reactive with and destroyed by the embedding dynamics.

This point is obvious when we notice that while processes such as the transcription of mRNA from DNA and RNA Editing work on genetic memory without access to its content (the encoded proteins), our evolved CA cannot manipulate their particles without access to their content. Particle reactions are simply domain interactions. In this sense, information carriers and content are inseparable. This way, we can say that domains and particles do not function as inert memory stores to be manipulated without access to content.

Does this mean that we cannot witness the emergence of a type of memory more like RAM (and genotypes) computationally? Our stumbling block was in obtaining a means to manipulate memory without recourse to its content. This has been a recurrent stumbling block in Artificial Life. For instance, Langton (1986) proposed a self-reproduction scheme in CA in which the separation between genotype (memory) and phenotype (content) was blurred. This lack of separation was actually seen as a worthwhile model for studying Artificial Life, with a generalized concept of genotype/phenotype mappings (Langton ,1989). But as it was clear for theoretical biologists looking at Artificial Life, a strict separation between genotype and phenotype is the key feature of life-as-we-know-it and a necessary condition for open-ended evolution (Pattee, 1995) (Rocha, 2001). Thus, the study of the emergence of a strict separation between genotype and phenotype, between memory and content, from a purely dynamic milieu should still be the number one goal of Artificial Life.

We submit that the dynamics produced by one-dimensional CA may be too simple to achieve what we desire to model. Indeed, homogeneous CA as a model of material dynamics, our artificial chemistry, is rather poor. In Biology, the genotype/phenotype mapping is based on the existence of two basic, distinct types of material (chemical) structures: DNA/RNA and aminoacid chains. Both are quite different: DNA is remarkably unreactive, or biochemically inert, whereas aminoacid chains are incredibly rich biochemical machines. In contrast, our one-dimensional homogenous CA compute the same exact update rule in each cell.

It seems reasonable that in order to evolve a system in which more reactive structures use non-reactive structures as information stores, we need to work with more heterogeneous dynamical systems where different populations of artificial "chemistry" structures interact.

References

Crutchfield,J.P., Mitchell,M., (1995). "The evolution of emergent computation". *PNAS*. **92**, 10742-10746.

Crutchfield,J.P., Mitchell,M., Das,R., (2002). "The Evolutionary Design of Collective Computation in Cellular Automata". In: *Evolutionary Dynamics: Exploring the Interplay of Selection, Neutrality, Accident, and Function.* Crutchfield,J.P., Schuster,P.K. (Eds.). Oxford University Press, pp. 361-412.

Das,R., Mitchell,M., Crutchfield,J.P., (1994). "A genetic algorithm discovers particle-based computation in cellular automata". In: *Parallel Problem Solving from Nature - PPSN III.* Davidor,Y., Schwefel,H.-P., Manner,R. (Eds.), Springer-Verlag, pp. 344-353.

Hanson,J.E., Crutchfield,J.P., (1992). "The attractor-basin portrait of a cellular automaton". *Journal of Statistical Physics*. **66** (5/6), 1415-1462.

Juillé,H., Pollack,J.B., (1998). "Coevolving the "ideal" trainer: application to the discovery of cellular automata rules.". In: *Genetic Programming Conference (GP-98), .* Koza,J.R., et al (Eds.), Morgan Kaufmann Publishers.

Langton,C.G., (1986). "Studying artificial life with cellular automata". *Physica D*. **22** (1-3), 120-149.

Langton,C.G., (1989). "Artificial Life". In: *Artificial Life.* Langton,C. (Ed.). Addison-Wesley, pp. 1-47.

Mitchell,M., (1998). "A complex-systems perspective on the "computation vs. dynamics" debate in cognitive science". In: *Proc. 20th Conf. of the Cog. Sci. Society.* Gernsbacher,M.A., Derry,S.J. (Eds.), pp. 710-715.

Pattee,H.H., (1995). "Artificial Life needs a real Epistemology". *Lecture Notes in Artificial Intelligence.* **929**, pp. 23-38.

Pinker,S., (2002). *The Blank Slate: The Modern Denial of Human Nature*. Penguin.

Rocha,L.M., (1998). "Syntactic autonomy". In: *Joint Conference on the Science and Technology of Intelligent Systems ISIC/CIRA/ISAS* IEEE Press, pp. 706-711.

Rocha,L.M., (2000). "Syntactic autonomy : Why there is no autonomy without symbols and how self-organizing systems might evolve them". *Annals of the New York Academy of Sciences.* **901**, 207-223.

Rocha,L.M., (2001). "Evolution with material symbol systems". *Biosystems*. **60** (1-3), 95-121.

Rocha,L.M., Hordijk,W., (2004). "Material Representations: From the Genetic Code to the Evolution of Cellular Automata". *Artificial Life*. In Press.

Van Gelder,T., Port,R., (1995). "It's about time: an overview of the dynamical approach to cognition". In: *Mind as Motion: Explorations in the Dynamics of Cognition.* Port,R., Van Gelder,T. (Eds.). MIT Press, pp. 1-43.