ECAL'95: 3rd. European Conference on Artificial Life, F. Morán, A. Moreno, J. J. Merelo, and P. Chacón (eds.), pages 544-554, 1995.

Quasi-Uniform Computation-Universal Cellular Automata

Moshe Sipper

Department of Computer Science
Tel Aviv University
Tel Aviv 69978, Israel
e-mail: moshes@math.tau.ac.il

Abstract. Cellular automata (CA) are dynamical systems in which space and time are discrete, where each cell obeys the same rule and has a finite number of states. In this paper we study non-uniform CA, i.e. with non-uniform local interaction rules. Our focal point is the issue of universal computation, which has been proven for uniform automata using complicated designs embedded in cellular space. The computation-universal system presented here is simpler than previous ones, and is embedded in the minimal possible two-dimensional cellular space, namely 2-state, 5-neighbor (which is insufficient for universal computation in the uniform model). The space studied is *quasi*-uniform, meaning that a small number of rules is used (our final design consists of just two rules which is minimal), distributed such that most of the grid contains one rule except for an infinitely small region which contains the others. We maintain that such automata provide us with a *simple*, *general* model for studying Artificial Life phenomena.

1 Introduction

Cellular automata (CA) are dynamical systems in which space and time are discrete. The states of cells in a regular grid are updated synchronously according to a local interaction rule. Each cell obeys the same rule and has a finite (usually small) number of states [29]. The model was originally conceived by John von Neumann in the 1950's to provide a more realistic framework for studying the behavior of complex, extended systems [31]. Over the years it has been applied to the study of general phenomenological aspects of the world, including: communication, computation, construction, growth, reproduction, competition and evolution [4, 26, 29].

The CA model is both *general* and *simple*. Generality implies two things: (1) the model supports universal computation and (2) the basic units encode a general form of interaction rather than some specialized action. Simplicity implies that the basic units of interaction are modest in comparison to Turing machines. If we imagine a scale of complexity with Turing machines occupying the high end then simple machines are those that occupy the low end, e.g. finite state automatons. The CA model is one of the simplest, general models available.

From an Artificial Life (AL) perspective it has been noted that the main difficulty with the CA approach seems to lie with the extreme low-level representation of the interactions. CA's are programmed at the level of the local physics of the system and, therefore, higher-level cooperative structures are difficult to evolve in CA's [23]. One of our primary goals is to increase the "capacity" for AL modeling while preserving the essential features of the original CA, namely massive parallelism, locality of cellular interactions and simplicity of cells. Thus we attain a model that is close to CA with regards to generality and simplicity [24, 25]. In this paper we consider non-uniform CA, i.e. with non-uniform local interaction rules. Such automata function in the same way as uniform ones, the only difference being in the cellular rules which need not be identical for all cells.

This type of model has been investigated by others. [8] presents two generalizations of cellular automata, namely discrete neural networks and automata networks. These are compared to the

original model from a computational point of view which considers the classes of problems such models can solve. The work of [30] discusses a one-dimensional CA in which a cell probabilistically selects one of two rules, at each time step. They showed that complex patterns appear characteristic of class IV behavior[1], however universal computation was not discussed (see also [9]). In [13, 22] adaptive stochastic cellular automata are considered which are essentially non-uniform automata whose rules are drawn from the same probability distribution function. Their approach focuses on the learning aspect where an automaton is trained to solve some problem (e.g. pole balancing). As noted above, our motivation is different than these works: we wish to study Artificial Life phenomena such as evolution, emergence and multi-cellularity in a simple, general model which operates at a higher level than CA [24, 25].

The focal point of this paper is the issue of universal computation in two-dimensional CA, namely the construction of machines, embedded in cellular space, whose computational power is equivalent to that of a universal Turing machine [10]. The first such machine was described by von Neumann, who used 29 state, 5-neighbor cells [31][2]. Codd provided a detailed description of a computer embedded in an 8-state, 5-neighbor cellular space, thus reducing the complexity of von Neumann's machine [5]. Banks described a 2-state and a 3-state automaton (both 5-neighbor) which support universal computation with an infinite and finite initial configuration[3], respectively [1]. A cellular space with a minimal number of states (two) and a 9-cell neighborhood proven to support universal computation (with finite initial configuration) involves the "game of life" rule [2]. One-dimensional CA have also been shown to support universal computation [28, 27]. A recent review of theoretical results is provided in [6].

Codd proved that there does not exist a computation-universal 2-state, 5-neighbor cellular automaton with finite initial configuration. His proof concerns the original uniform model [5]. We present a universal non-uniform system embedded in such space.

Section 2 presents the details of our construction, consisting of ten different cell rules, which are reduced to six in Section 3. Both sections involve an infinite initial configuration. Section 4 describes the implementation of a universal machine using a finite initial configuration. A quasi-uniform automaton is discussed in Section 5 where quasi-uniformity implies a small number of rules distributed such that most of the grid contains one rule except for an infinitely small region which contains the others. Our final implementation consists of just two rules which is minimal. A discussion of our results follows in Section 6 linking them with Artificial Life issues.

2 A universal 2-state, 5-neighbor non-uniform cellular automaton

In order to prove that a two-dimensional CA is computation-universal we proceed along the lines of [2, 21, 1, 5, 31] and implement the following components[4]:

1. Signals and signal pathways (wires). We must show how signals can be made to turn corners, to cross and to fan out.
2. A functionally complete set of logical gates. A set of operations is said to be *functionally complete* (or *universal*) if and only if every switching function can be expressed entirely by means of

[1] CA class numbers are those introduced by [32].

[2] The neighborhood consists of the cell itself together with its four immediate nondiagonal neighbors. This is also the neighborhood used throughout this paper.

[3] A configuration is an assignment of non-zero states to cells in the space.

[4] Another approach is one in which a row of cells simulates the squares on a Turing machine tape while at the same time simulating the head of the Turing machine. This has been applied to one-dimensional CA [28, 27, 1].

operations from this set [11]. We shall use the $NAND$ function (gate) for this purpose (this gate comprises a functionally complete set and is used extensively in VLSI since transistor switches are inherently inverters [19]).

3. A clock that generates a stream of pulses at regular intervals.
4. Memory.

In the following sections we describe the implementations of the above components (note: the terms 'gate' and 'cell' are used interchangeably).

Name	Rule	Symbol
Right propagation cell	$\begin{matrix} & * & \\ x & * & * \\ & * & \end{matrix} \mapsto x$	$\rightarrow$
Left propagation cell	$\begin{matrix} & * & \\ * & * & x \\ & * & \end{matrix} \mapsto x$	$\leftarrow$
Up propagation cell	$\begin{matrix} & * & \\ * & * & * \\ & x & \end{matrix} \mapsto x$	$\uparrow$
Down propagation cell	$\begin{matrix} & x & \\ * & * & * \\ & * & \end{matrix} \mapsto x$	$\downarrow$
$NAND$ Cell	$\begin{matrix} & y & \\ x & * & * \\ & * & \end{matrix} \mapsto x \mid y$	$\blacksquare$

Name	Rule	Symbol
Exclusive Or (XOR) cell (type a)	$\begin{matrix} & y & \\ x & * & * \\ & * & \end{matrix} \mapsto x \oplus y$	$\oplus$
Exclusive Or (XOR) cell (type b)	$\begin{matrix} & x & \\ * & * & y \\ & * & \end{matrix} \mapsto x \oplus y$	$\oplus$
Exclusive Or (XOR) cell (type c)	$\begin{matrix} & * & \\ * & * & x \\ & y & \end{matrix} \mapsto x \oplus y$	$\oplus$
Exclusive Or (XOR) cell (type d)	$\begin{matrix} & * & \\ y & * & * \\ & x & \end{matrix} \mapsto x \oplus y$	$\oplus$
No Change (NC) cell	$\begin{matrix} & * & \\ * & x & * \\ & * & \end{matrix} \mapsto x$	$\cdot$

Each rule specifies the new state of the central cell.
'$*$' represents the set of states: $\{0, 1\}$.
x,$y \in \{0, 1\}$.
'$\oplus$' is the XOR function (modulo-2 addition), '$|$' is the $NAND$ function.

Table 1. Cell types (rules).

2.1 Signals and wires

A wire in our system consists of a path of connected *propagation cells* each containing one of the propagation rules. A signal consists of a state, or stream of states being propagated along the wire. There are four propagation cell types (i.e. four different rules), one for each direction: right, left, up, down (Table 1). Figure 1a shows a wire constructed from propagation cells. Figures 1b - 1d demonstrate signal propagation along the wire. Note that all cells which are not part of the machine (i.e. its components) contain the NC (No Change) rule (Table 1) which simply preserves its initial state indefinitely.

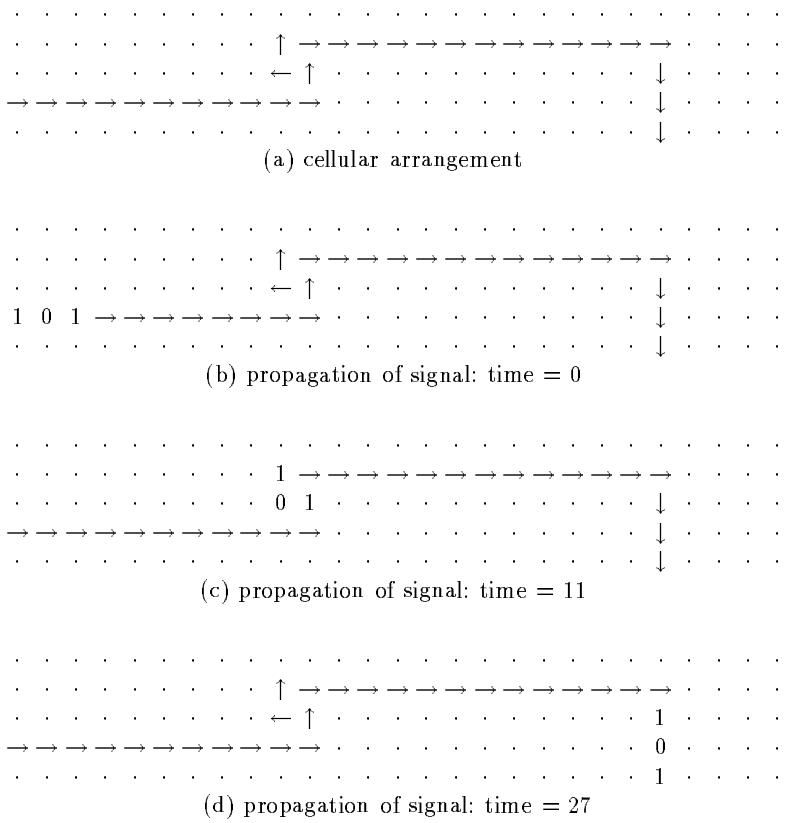
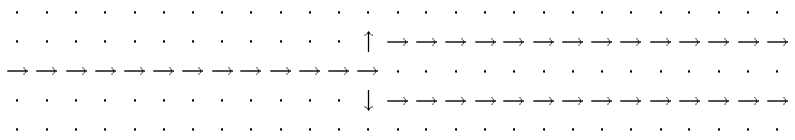


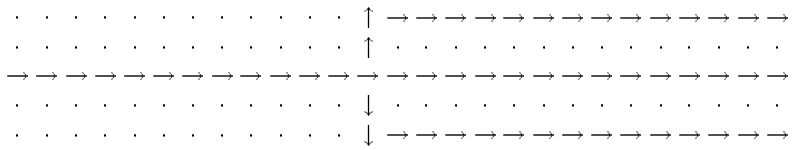
Fig. 1. Signal propagation along a wire.

A wire in our system possesses a distinct direction, a characteristic which is highly desirable as it simplifies signal propagation [5]. In most cases signals must propagate in one direction only, and should bi-directional propagation be required then two parallel wires in opposite directions may be used. We note in Figure 1 that wires support signal propagation across corners. Fan out of signals is also straightforward as evident in Figure 2.

The last problem we must address concerning signals is wire crossing (there are four possible crossings since wires may run in four directions). We first demonstrate that at least three gates



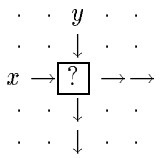
(a) two-way fan out



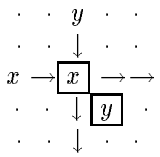
(b) three-way fan out

Fig. 2. Signal fan out.

(cells) are required for this operation. To see this note that one gate is insufficient since there are two bits of information, denoted x and y, whereas the intersection cell can only contain one bit:



Thus, either x crosses the y (vertical) line, in which case the y signal is lost, or conversely the x signal is lost. In case there are two gates then they must eventually contain x and y (otherwise information is lost). The situation is therefore as follows:

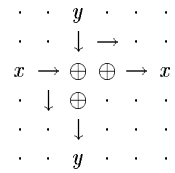


The x signal gets transferred, however there is no way for the y signal to get to its gate (note that the y gate must be below the x line), since this is exactly the crossing problem we are trying to solve, and we have already shown that this cannot be done using one cell (the remaining one). A similar argument holds for the reverse situation, i.e. the intersection cell contains y. We therefore conclude that at least three gates are required for the crossing operation.

Towards this end we have selected the XOR cells (Table 1) since wire crossing can be implemented using a minimal number of such gates (three). An implementation of one of the four possible crossings is given in Figure 3 (the other three are derived analogously).

Note that in uniform cellular automata the implementation of wires and signals is highly complicated [31, 5, 1]. The wire itself and the operations of propagation, crossing and fan out are attained using complex structures composed of several cells. The large number of possible interactions between these structures makes the design task arduous.

It is important to note the direct relationship between path length and time: if two paths branch out from a common point A to points B, C respectively, and if path length AB is strictly greater



The implementation is based on the equivalences:
$x \equiv (x \oplus y) \oplus y$
$y \equiv (x \oplus y) \oplus x$

The XOR gates used are type a (see Table 1).

Fig. 3. Implementation of wire crossing.

than path length AC, then a signal which fans out at A will arrive at B strictly later than at C (this issue was emphasized by [5]).

2.2 Logical gates

Table 1 includes a 2-input, 1-output $NAND$ gate (cell), which forms a functionally complete set, thereby providing us with the second component discussed above. Two neighboring cells act as inputs while the central cell acts as the gate's output. Since $NAND$ is functionally complete other gates such as NOT, AND and OR can be constructed from it.

The XOR gate is not required for completeness purposes, however we have included it since wire crossing can be implemented using a minimal number of gates (Section 2.1). Four XOR cell types (rules) are needed to implement the four possible crossings. Once all crossings are possible we only need one $NAND$ gate since the two wire inputs can always be made to arrive at the two input cells of the gate's neighborhood[5].

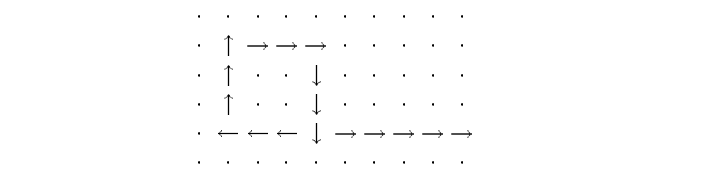
2.3 Clock

The third component of our system is a clock that generates a stream of pulses at regular intervals. We implement this using a wire loop, i.e. a loop of propagation cells. Figure 4 presents the implementation. Note that any desired waveform can be produced by adjusting the size and content of the loop. The implementation of the clock is made simple due to the manner in which wires are constructed, i.e. as cellular arrangements. Thus it is possible to obtain such a closed loop, which proves highly useful in our case.

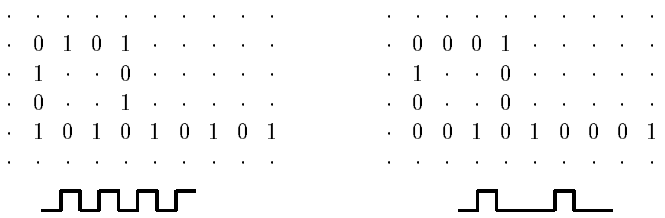
3 Reducing the number of rules

In Section 2 we presented the components of a universal machine employing ten different cell rules (Table 1). This number may be reduced to six, by using a more complex wiring scheme, involving the implementation of the propagation cells using XOR gates (implementation is not shown due to lack of space).

[5] Note that both signals must arrive synchronously. This is possible since delays can always be introduced by using, e.g., loops which are feasible once crossings are implemented.



(a) cellular arrangement



(b) examples of pulse signals

Fig. 4. Implementation of clock.

4 Implementing a universal machine using finite configurations

The components presented in the previous sections are sufficient in order to build a universal machine using an infinite initial configuration. Codd conjectured that an unbounded but boundable propagation is a necessary condition for computation universality and used this conjecture in a proof that there does not exist a 2-state, 5-neighbor universal cellular automaton with finite initial configuration [5]. Following his work universality was implemented by using more states or larger neighborhoods [1, 2, 21].

The problem with finite configurations involving the above components is that a computation may require an arbitrary amount of space and therefore some method must exist for increasing the information storage (memory) by arbitrarily large amounts. In order to prove universality we implement Minsky's two-register universal machine, which consists of [20]:

1. A programming unit (finite control).
2. Two potentially infinite registers.
3. The following set of instructions:
   - Increase the contents of a register by one.
   - Decrease the contents of a register by one.
   - Test whether the contents of a register equal zero.

The finite control unit may be realized using the components described in Sections 2 and 3. The major difficulty is the implementation of the infinite registers and the three operations associated with them. According to Codd's proof a single rule in 2-state, 5-neighbor cellular space is insufficient since, as noted above, unbounded but boundable propagations cannot be attained [5].

While other researchers have turned to cellular spaces with more states or larger neighborhoods, our approach is based on non-uniformity. As noted above the minimal number of rules needed to implement a register is two. Indeed, we have found two such rules: one which we denote the *background* rule, the other being Banks' rule [1] (rules are not shown due to lack of space). The

implementation of a universal computer consists of a finite control unit, which occupies a finite part of the grid. All other cells contain the *background* rule except for two cellular columns, infinite in one direction, containing Banks' rule. These *register columns* originate at the upper part of the control unit and each one represents one register (Figure 5a depicts the basic machine configuration after initialization has taken place, see ahead). The cells in these columns are denoted register cells.

The three register instructions are implemented as follows: at any given moment a register column consists of an infinite[6] number of cells in state 1, and a finite number in state 0, occupying the bottom part of the column. The number of 0s represents the register's value. Initially, both register columns (i.e. all register cells) are transformed (from state 0) to state 1, thus setting the register's value to zero. For each column this is accomplished by setting the bottom register cell along with its left and right neighbors to 1. The two 1s on both sides act as signals which travel upward along the column, setting all its cells to 1. Figure 5a demonstrates this process at $time = 4$: three cells have already been transformed to 1, the fourth is currently being transformed after which the (dual) signal will continue its upward movement. The overall effect of this process is that the value of both registers is initialized to zero.

The zero test is straightforward since it involves testing only the bottom register cell: if its state is 1 the register's value is zero, otherwise it is not. Adding one to a register is achieved by setting to 1 the cell which is at distance two to the right of the bottom register cell. Figure 5b demonstrates this operation: the left grid depicts the configuration before the operation, where the register's value is 3 and the appropriate cell is set to 1 (i.e. two cells to the right of the bottom cell). The right grid depicts the effect of the operation (i.e. the configuration after several time steps): the column's number of zeros has increased by one which means the register's value is now 4. Subtracting one from a register is done by setting to 1 both neighboring cells of the bottom register cell (Figure 5c demonstrates this operation). Thus, the registers along with their instructions have been implemented.

The initial configuration of the machine is finite, since all but a finite number of cells are set to zero. The total number of rules needed to implement a universal computer equals the number of rules necessary for implementation of the finite control unit plus the two additional memory rules (background and Banks). Thus we need a total of 12 rules using our implementation of Section 2 and 8 rules using the implementation of Section 3.

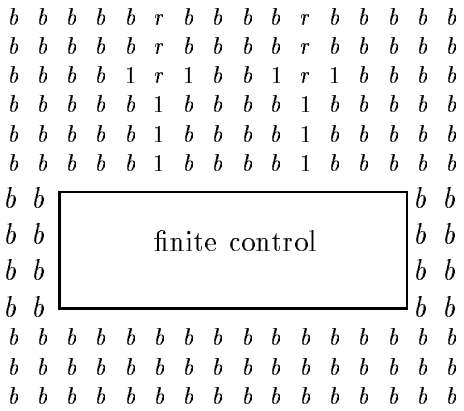
5 A quasi-uniform cellular space

As noted in Section 3 by increasing the complexity of our machine a reduced set of rules (six) may be used to construct the finite control. We can go one step further and construct the finite control with only one rule, e.g. the Banks rule [1] (note that this increases the complexity of the basic operations). As noted in Section 4 the complicating issue is not the finite control but rather the infinite memory which cannot be implemented in uniform 2-state, 5-neighbor cellular space.

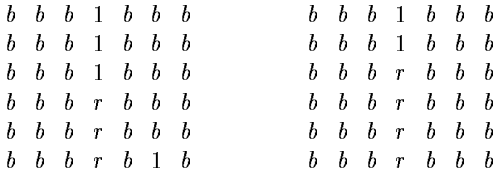
Our conclusion from the previous paragraph is that a universal computer can be implemented using only two rules: background and Banks. Thus we have implemented universal computation using the minimal number of rules.

The two rules necessary to implement universal computation are distributed unevenly. Most of the grid contains the background rule, except for an *infinitely small* region which contains the other one. By this we mean that although there is an infinite number of both rules in the grid, each (infinite) row contains an *infinite* number of background rules with only a *finite* number of the other. In fact, except for a finite region of the grid each row contains only two Banks rules and an infinite number of background rules. Hence we say that our cellular space is *quasi*-uniform.

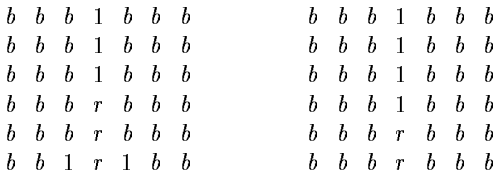
[6] More precisely the number of cells in state 1 tends to infinity as time progresses, see ahead.



(a) initial configuration: both registers set to zero.



(b) adding one to a register.



(c) subtracting one from a register

b denotes a cell in state 0 containing the background rule, r denotes a cell in state 0 containing Banks' rule. In figures (b) and (c) the left grid shows the configuration before the operation, the right grid shows the configuration upon its completion (after several time steps). The bottom line represents the bottom register cell and its neighbors.

Fig. 5. Register operation.

6 Discussion

We presented a quasi-uniform 2-state, 5-neighbor cellular automaton capable of universal computation. Quasi-uniformity implies a small number of rules distributed such that most of the grid contains one rule except for an infinitely small region which contains the others. Three implementations were described in which the number of rules used is: 12, 8 and finally 2.

The following paragraphs provide a discussion of a speculative nature linking our result with that

of Langton [12] (see also [14]). He addressed the following question: under what conditions can we expect a dynamics of information to emerge spontaneously and come to dominate the behavior of a physical system? This was studied in the context of CA where the question becomes: under what conditions can we expect a complex dynamics of information to emerge spontaneously and come to dominate the behavior of a CA? [12].

Langton showed that the rule space of CA consists of two primary regimes of rules, periodic and chaotic, separated by a transition regime. His main conclusion was that information processing can emerge spontaneously and come to dominate the dynamics of a physical system in the vicinity of a critical phase transition.

According to Codd's proof a uniform (single rule) 2-state, 5-neighbor cellular space is insufficient for universal computation since unbounded but boundable propagations cannot be attained [5]. Either every configuration yields an unboundable propagation or every configuration yields a bounded propagation. In the context of Langton's work bounded propagations correspond to fixed point rules (class I) and unboundable propagations correspond either to periodic rules (class II) or chaotic ones (class III). Complex behavior (class IV) cannot be attained.

By using a quasi-uniform, two rule, cellular space we have been able to achieve unbounded but boundable propagations, thus attaining class IV behavior. Langton suggested that the information dynamics which gave rise to life came into existence when global or local conditions brought some medium through a critical phase transition.

Imagine an information-based world consisting of a uniform cellular automaton, which is not within the class IV region. If we wanted to attain class IV behavior the *entire* space would have to "jump", i.e. go through a phase transition, to a class IV rule (assuming this is at all possible, e.g. in our case of a 2-state, 5-neighbor space it is not). However, as a conclusion of the work presented above we offer an alternative: a small perturbation may be enough to cause some infinitely small part of the world to change. This would be sufficient to induce a (possible) transition from class II or class III behavior to class IV behavior. Furthermore, such a change could be effected upon a very simple world (in our case 2-state, 5-neighbor). As noted by [3] "frozen accidents" play an important role in the evolutionary process. These accidents are mainly caused by external conditions, i.e. external relative to a given system's laws of functioning.

A (highly) tentative comparison may be drawn to the famous experiment performed by [17] (see also [18]) in which methane, ammonia, water and hydrogen, representing a possible atmosphere of the primitive Earth, were subjected to an electric spark for a week. After this period simple amino acids were found in the system. The analogy to our CA world is as follows: we start with a simple uniform world, consisting of a single rule, which does not support complex (class IV) behavior[7]. At some point, a "spark" causes a perturbation in which a small number of cells change their rule. This is all that is needed. Our central conclusion is that such an infinitely small change in our world can suffice to generate a phase transition such that class IV behavior becomes possible. Note that this can happen independently in other regions of the world as well.

While the above discussion has been of a speculative nature we may also draw some practical conclusions from our work. As noted in Section 1 the main difficulty with the CA approach seems to lie with the extreme low-level representation of the interactions. Essentially, we construct world models at the level of physics. By slightly changing the rules of the game (no pun intended) we can increase the "capacity" for AL modeling while preserving the main features of CA, namely massive parallelism, locality of cellular interactions and simplicity of cells. Thus, we obtain a simple, general model (Section 1) which allows us to evolve complex behavior with the ability to explore, in-depth,

[7] either due to inability of the cellular space to support such behavior at all or due to the rule being in the non class IV regions of rule space.

the inner workings of the evolutionary process.

We have already begun venturing along this path [24, 25]. In these works a CA derived model is explored in which the basic units are slightly more complex than those of CA. Furthermore, evolution proceeds not only in state space as in CA but also in rule space, i.e. rules evolve over time. We have observed several phenomena of interest involving emergence, evolution and multi-cellularity. Another example is that of *embryonics*, standing for embryological electronics [15, 16, 7]. This is a CA based approach in which three principles of natural organization are employed: multi-cellular organization, cellular differentiation and cellular division. Their intent is to create an architecture which is complex enough for (quasi) universal computation yet simple enough for physical implementation. The approach represents another attempt at confronting the aforementioned problem of CA, namely the low level of operation.

It is hoped that the development of such AL models will serve the two-fold goal of: (1) increasing our understanding of biology and (2) enhancing our understanding of artificial models, thereby providing us with the ability to improve their performance. AL research opens new doors providing us with novel opportunities to explore issues such as adaptation, evolution and emergence which are central both in natural environments as well as man-made ones.

Acknowledgment

I am grateful to Daniel Mange for his suggestions which helped improve this paper.

References

1. E. R. Banks. Universality in cellular automata. In *IEEE 11th Annual Symposium on Switching and Automata Theory*, pages 194–215, Santa Monica, California, October 1970.
2. E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning ways for your mathematical plays*, volume 2, chapter 25, pages 817–850. Academic Press, New York, 1982.
3. E. W. Bonabeau and G. Theraulaz. Why do we need artificial life? *Artificial Life Journal*, 1(3):303–325, 1994. The MIT Press, Cambridge, MA.
4. A. Burks, editor. *Essays on cellular automata.* University of Illinois Press, Urbana, Illinois, 1970.
5. E. F. Codd. *Cellular Automata.* Academic Press, New York, 1968.
6. K. Culik II, L. P. Hurd, and S. Yu. Computation theoretic aspects of cellular automata. *Physica D*, 45:357–378, 1990.
7. S. Durand, A. Stauffer, and D. Mange. Biodule: an introduction to digital biology. Technical report, LSL, Swiss Federal Institute of Technology, Lausanne, Switzerland, September 1994.
8. M. Garzon. Cellular automata and discrete neural networks. *Physica D*, 45:431–440, 1990.
9. H. Hartman and G. Y. Vichniac. Inhomogeneous cellular automata. In E. Bienenstock, F. Fogelman, and G. Weisbuch, editors, *Disordered Systems and Biological Organization*, pages 53–57. Springer-Verlag, Berlin, 1986.
10. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory Languages and Computation.* Addison-Wesley, 1979.
11. Z. Kohavi. *Switching and Finite Automata Theory.* McGraw-Hill Book Company, 1970.
12. C. G. Langton. Life at the edge of chaos. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 41–91, Redwood City, CA, 1992. Addison-Wesley.
13. Y. C. Lee, S. Qian, R. D. Jones, C. W. Barnes, G. W. Flake, M. K. O'Rourke, K. Lee, H. H. Chen, G. Z. Sun, Y. Q. Zhang, D. Chen, and C. L. Giles. Adaptive stochastic cellular automata: theory. *Physica D*, 45:159–180, 1990.

14. W. Li, N. H. Packard, and C. G. Langton. Transition phenomena in cellular automata rule space. *Physica D*, 45:77–94, 1990.
15. D. Mange and A. Stauffer. Introduction to embryonics: Towards new self-repairing and self-reproducing hardware based on biological-like properties. In N. M. Thalmann and D. Thalmann, editors, *Artificial Life and Virtual Reality*, pages 61–72, Chichester, England, 1994. John Wiley.
16. P. Marchal, C. Piguet, D. Mange, A. Stauffer, and S. Durand. Embryological development on silicon. In R. A. Brooks and P. Maes, editors, *Artificial Life IV*, pages 365–370, Cambridge, Massachusetts, 1994. The MIT Press.
17. S. L. Miller. A production of amino acids under possible primitive earth conditions. *Science*, 117:528–529, May 1953.
18. S. L. Miller and H. C. Urey. Organic compound synthesis on the primitive earth. *Science*, 130(3370):245–251, July 1959.
19. J. Millman and A. Grabel. *Microelectronics*. McGraw-Hill Book Company, second edition, 1987.
20. M. L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, New Jersey, 1967.
21. F. Nourai and R. S. Kashef. A universal four-state cellular computer. *IEEE Transactions on Computers*, c-24(8):766–776, August 1975.
22. S. Qian, Y. C. Lee, R. D. Jones, C. W. Barnes, G. W. Flake, M. K. O'Rourke, K. Lee, H. H. Chen, G. Z. Sun, Y. Q. Zhang, D. Chen, and C. L. Giles. Adaptive stochastic cellular automata: applications. *Physica D*, 45:181–188, 1990.
23. S. Rasmussen, C. Knudsen, and R. Feldberg. Dynamics of programmable matter. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 211–254, Redwood City, CA, 1992. Addison-Wesley.
24. M. Sipper. Non-uniform cellular automata: Evolution in rule space and formation of complex structures. In R. A. Brooks and P. Maes, editors, *Artificial Life IV*, pages 394–399, Cambridge, Massachusetts, 1994. The MIT Press.
25. M. Sipper. Studying artificial life using a simple, general cellular model. *Artificial Life Journal*, 2(1):443–477, 1995. The MIT Press, Cambridge, MA.
26. A. Smith. Cellular automata theory. Technical Report 2, Stanford Electronic Lab., Stanford University, 1969.
27. A. R. Smith. Simple computation-universal cellular spaces. *Journal of ACM*, 18:339–353, 1971.
28. A. R. Smith. Simple nontrivial self-reproducing machines. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 709–725, Redwood City, CA, 1992. Addison-Wesley.
29. T. Toffoli and N. Margolus. *Cellular Automata Machines*. The MIT Press, Cambridge, Massachusetts, 1987.
30. G. Y. Vichniac, P. Tamayo, and H. Hartman. Annealed and quenched inhomogeneous cellular automata. *Journal of Statistical Physics*, 45:875–883, 1986.
31. J. von Neumann. *The Theory of Self-Reproducing Automata*. University of Illinois Press, Illinois, 1966. Edited and completed by A.W. Burks.
32. S. Wolfram. Universality and complexity in cellular automata. *Physica D*, 10:1–35, 1984.

This article was processed using the LaTeX macro package with LLNCS style