Chapter 1

An Introduction To Cellular Automata

Moshe Sipper and Marco Tomassini

1.1 What are cellular automata?

Cellular automata (CA) were originally conceived by Ulam and von Neumann in the 1940s to provide a formal framework for investigating the behavior of complex, extended systems [32]. CAs are dynamical systems in which space and time are discrete. A cellular automaton consists of an array of cells, each of which can be in one of a finite number of possible states, updated synchronously in discrete time steps, according to a local, identical interaction rule. The state of a cell at the next time step is determined by the current states of a surrounding neighborhood of cells [20, 29, 33].

The cellular array (grid) is n-dimensional, where $n = 1, 2, 3$ is used in practice; in this volume we shall concentrate on $n = 1, 2$, i.e., one- and two-dimensional grids. The *identical* rule contained in each cell is essentially a finite state machine, usually specified in the form of a rule table (also known as the transition function), with an entry for every possible neighborhood configuration of states. The *cellular neighborhood* of a cell consists of the surrounding (adjacent) cells. For one-dimensional CAs, a cell is connected

[1]The authors are with the Logic Systems Laboratory, Swiss Federal Institute of Technology, IN-Ecublens, CH-1015 Lausanne, Switzerland. E-mail: {name.surname}@di.epfl.ch. M. Tomassini is also with the Computer Science Institute, University of Lausanne.

to r local neighbors (cells) on either side, as well as to itself, where r is a parameter referred to as the radius (thus, each cell has $2r + 1$ neighbors). For two-dimensional CAs, two types of cellular neighborhoods are usually considered: 5 cells, consisting of the cell along with its four immediate non-diagonal neighbors, and 9 cells, consisting of the cell along with its eight surrounding neighbors. When considering a finite-sized grid, spatially periodic boundary conditions are frequently applied, resulting in a circular grid for the one-dimensional case, and a toroidal one for the two-dimensional case.

As an example, let us consider the parity rule (also known as the XOR rule) for a 2-state, 5-neighbor, two-dimensional CA [20]. Each cell is assigned a state of 1 at the next time step if the parity of its current state and the states of its four neighbors is odd, and is assigned a state of 0 if the parity is even (alternatively, this may be considered a modulo-2 addition). The rule table consists of entries of the form:

$$\begin{array}{ccc} & \boxed{0} & \\ \boxed{1} & \boxed{1} & \boxed{0} \\ & \boxed{1} & \end{array} \mapsto 1$$

This means that if the current state of the cell is 1 and the states of the north, east, south, and west cells are $0, 0, 1, 1$, respectively, then the state of the cell at the next time step will be 1 (odd parity). The rule is completely specified by the rule table given in Table 1.1. Figure 1.1 demonstrates patterns that are produced by the parity CA.

1.2 Formal definitions

A d-dimensional CA consists of a finite or infinite d-dimensional grid of cells, each of which can take on a value from a finite, typically small, set of integers. The value of each cell at time step t is a function of the values of a small local neighborhood of cells at time $t - 1$. The cells update their states simultaneously according to a given local rule.

Formally, a *cellular automaton* A is a quadruple

$$A = (S, G, d, f),$$

where S is a finite set of states, G is the cellular neighborhood, $d \in Z^+$ is the dimension of A, and f is the local cellular interaction rule, also referred to as the transition function.

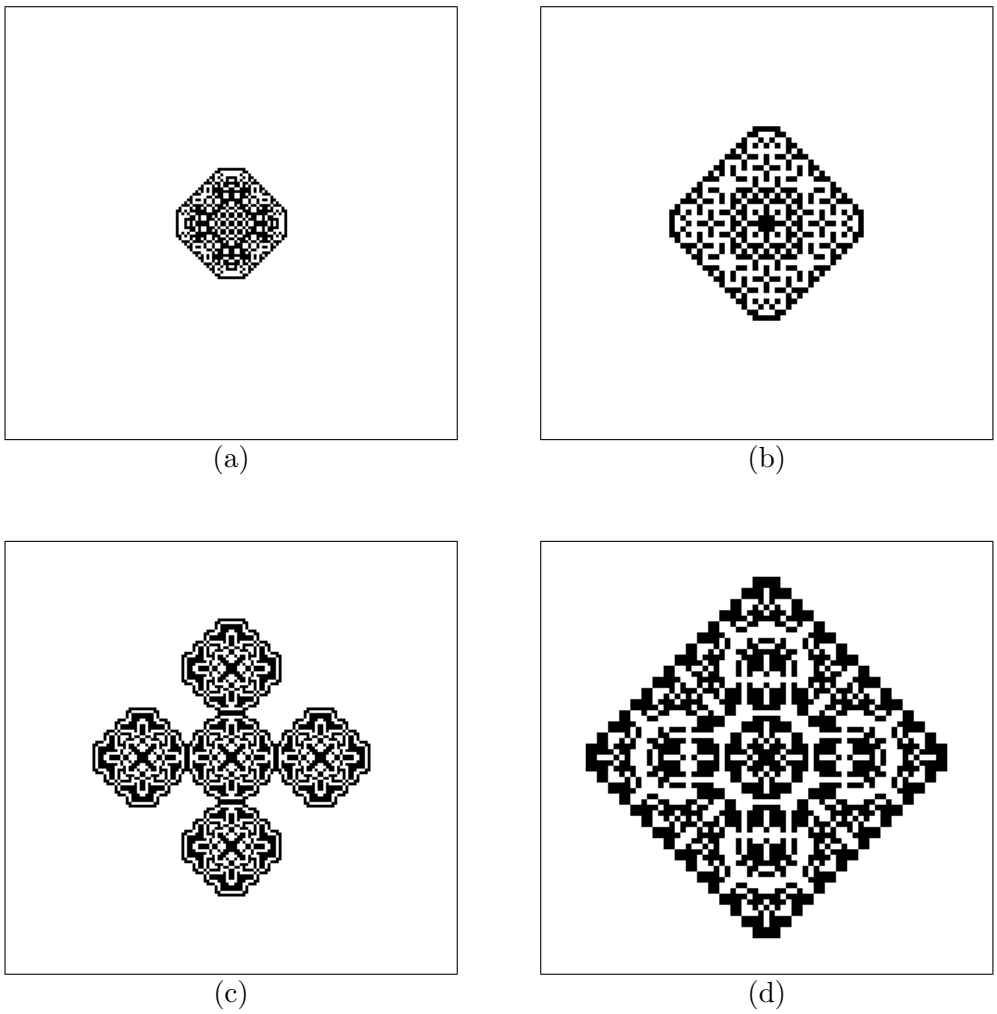

Figure 1.1: Patterns produced by the parity rule, starting from a 20×20 rectangular pattern. White squares represent cells in state 0, black squares represent cells in state 1. (a) after 30 time steps ($t = 30$), (b) $t = 60$, (c) $t = 90$, (d) $t = 120$.

CNESW	S_{next}	CNESW	S_{next}	CNESW	S_{next}	CNESW	S_{next}
00000	0	01000	1	10000	1	11000	0
00001	1	01001	0	10001	0	11001	1
00010	1	01010	0	10010	0	11010	1
00011	0	01011	1	10011	1	11011	0
00100	1	01100	0	10100	0	11100	1
00101	0	01101	1	10101	1	11101	0
00110	0	01110	1	10110	1	11110	0
00111	1	01111	0	10111	0	11111	1

Table 1.1: Parity rule table. CNESW denotes the current states of the center, north, east, south, and west cells, respectively. S_{next} is the cell's state at the next time step.

Given the position of a cell, $\mathbf{i}$, $\mathbf{i} \in Z^d$, in a regular d-dimensional uniform lattice, or *grid* (i.e., $\mathbf{i}$ is an integer vector in a d-dimensional space), its *neighborhood* G is defined by:

$$G_{\mathbf{i}} = \{\mathbf{i}, \mathbf{i} + \mathbf{r}_1, \mathbf{i} + \mathbf{r}_2, \ldots, \mathbf{i} + \mathbf{r}_n\},$$

where n is a fixed parameter that determines the neighborhood size, and $\mathbf{r}_j$ is a fixed vector in the d-dimensional space.

The *local transition rule* f

$$f : S^n \to S$$

maps the state $s_{\mathbf{i}} \in S$ of a given cell $\mathbf{i}$ into another state from the set S, as a function of the states of the cells in the neighborhood $G_{\mathbf{i}}$. In uniform CAs f is identical for all cells, whereas in non-uniform ones f may differ between different cells, i.e., f depends on $\mathbf{i}$, $f_{\mathbf{i}}$.

For a finite-size CA of size N (such as those treated in this book) a *configuration* of the grid at time t is defined as

$$C(t) = (s_{\mathbf{0}}(t), s_{\mathbf{1}}(t), \ldots, s_{\mathbf{N-1}}(t)),$$

where $s_{\mathbf{i}}(t) \in S$ is the state of cell $\mathbf{i}$ at time t. The progression of the CA in time is then given by the iteration of the *global mapping* F

$$F : C(t) \to C(t+1), \qquad t = 0, 1, \ldots$$

through the simultaneous application in each cell of the local transition rule f. The global dynamics of the CA can be described as a directed graph, referred to as the CA's *phase space* [33].

An oft-explored system is that of one-dimensional CAs with two possible states per cell, i.e., $S = \{0, 1\}$. In this case f is a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and the neighborhood size n is usually taken to be $n = 2r + 1$ such that:

$$s_i(t + 1) = f(s_{i-r}(t), ..., s_i(t), ..., s_{i+r}(t)),$$

where $r \in Z^+$ is a parameter, known as the *radius*, representing the standard one-dimensional cellular neighborhood. Considering the $r = 1$ case one obtains so-called *elementary* CAs, for which the neighborhood size is $n = 3$:

$$f : \{0, 1\}^3 \rightarrow \{0, 1\}, \qquad s_i(t + 1) = f(s_{i-1}(t), s_i(t), s_{i+1}(t)).$$

The domain of f is the set of all 2^3 3-tuples, which gives rise to $2^8 = 256$ distinct elementary rules. It is common to use Wolfram's decimal numbering convention for describing these rules [33].[2] For two-state CAs a configuration of a size N grid at time t is a binary sequence $C(t)$, $C(t) \in \{0, 1\}^N$. For finite-size grids, spatially periodic boundary conditions are frequently assumed, resulting in a circular grid; formally, this implies that cellular indices are computed modulus N.

1.3 Cellular automata as complex and computational systems

As noted above, the CA model was originally introduced in the late 1940s by Ulam and von Neumann and used extensively by the latter to study issues related with the logic of life [32]. In particular, von Neumann asked whether we can use purely mathematical-logical considerations to discover the specific features of biological automata that make them self-replicating [19, 21].

Von Neumann used two-dimensional CAs with 29 states per cell and a 5-cell neighborhood. He showed that a universal computer can be embedded in such cellular space, namely, a device whose computational power is equivalent to that of a universal Turing machine [13]. He also described how

[2] For example, $f(111) = 1$, $f(110) = 0$, $f(101) = 1$, $f(100) = 1$, $f(011) = 1$, $f(010) = 0$, $f(001) = 0$, $f(000) = 0$, is denoted rule 184 (the decimal equivalent of 10111000).

a universal constructor may be built, namely, a machine capable of constructing, through the use of a "constructing arm," any configuration whose description can be stored on its input tape. This universal constructor is therefore capable, given its own description, of constructing a copy of itself, i.e., of self replicating (Figure 1.2). The terms 'machine' and 'tape' refer here to configurations, i.e., patterns of states (as defined in Section 1.2). The mechanisms von Neumann proposed for achieving self-replicating structures within a cellular automaton bear strong resemblance to those employed by biological life, discovered during the following decade [21]. Von Neumann's universal computer-constructor was simplified by [6] who used an 8-state, 5-neighbor cellular space (for more on these issues see Chapter **??**).

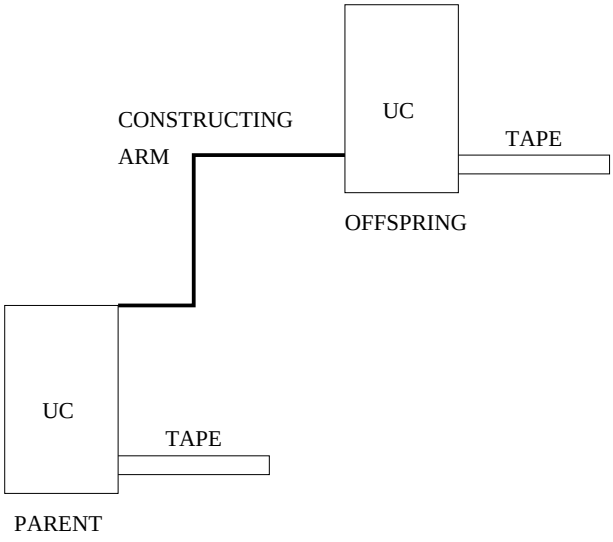

Figure 1.2: A schematic diagram of von Neumann's self-replicating cellular automaton. The system is a universal constructor (UC), namely, a machine capable of constructing, through the use of a "constructing arm," any configuration whose description can be stored on its input tape. This universal constructor is therefore capable, given its own description, of constructing a copy of itself, i.e., of self-replicating. (The machine is not drawn to scale.)

Over the years CAs have been applied to the study of general phenomenological aspects of the world, including communication, computation, construction, growth, reproduction, competition, and evolution (see, e.g.,

[4, 18, 20, 29]). One of the most well-known CA rules, the "game of life," was conceived by Conway in the late 1960s and was shown by him to be computation-universal [2]. For a review of computation-theoretic CA results refer to [7].

The question of whether cellular automata can model not only general phenomenological aspects of our world, but also directly model the laws of physics themselves was raised by [10, 26]. A primary theme of this research is the formulation of computational models of physics that are *information-preserving*, and thus retain one of the most fundamental features of microscopic physics, namely, reversibility [10, 16, 27]. This approach has been used to provide extremely simple models of common differential equations of physics, such as the heat and wave equations [28] and the Navier-Stokes equation [11]. CAs also provide a useful model for a branch of dynamical systems theory which studies the emergence of well-characterized collective phenomena, such as ordering, turbulence, chaos, symmetry-breaking, and fractality, in discrete systems [5, 30].

The systematic study of CAs in this latter context was pioneered by Wolfram and studied extensively by him [33]. He investigated CAs and their relationships to dynamical systems, identifying the following four qualitative classes of CA behavior, with analogs in the field of dynamical systems (the latter are shown in parenthesis; see also [15]):

1. Class I relaxes to a homogeneous state (limit points).

2. Class II converges to simple separated periodic structures (limit cycles).

3. Class III yields chaotic aperiodic patterns (chaotic behavior of the kind associated with strange attractors).

4. Class IV yields complex patterns of localized structures, including propagating structures (very long transients with no apparent analog in continuous dynamical systems).

Figure 1.3 demonstrates these four classes using one-dimensional CAs (as studied by Wolfram). Finally, biological modeling has also been carried out using CAs [8].

We have seen above that CAs have been used as a formal model for studying phenomena of interest in several scientific fields, including physics, biology, and computer science. In recent years there is a growing interest in the utilization of CAs as actual computing devices. CAs exhibit three

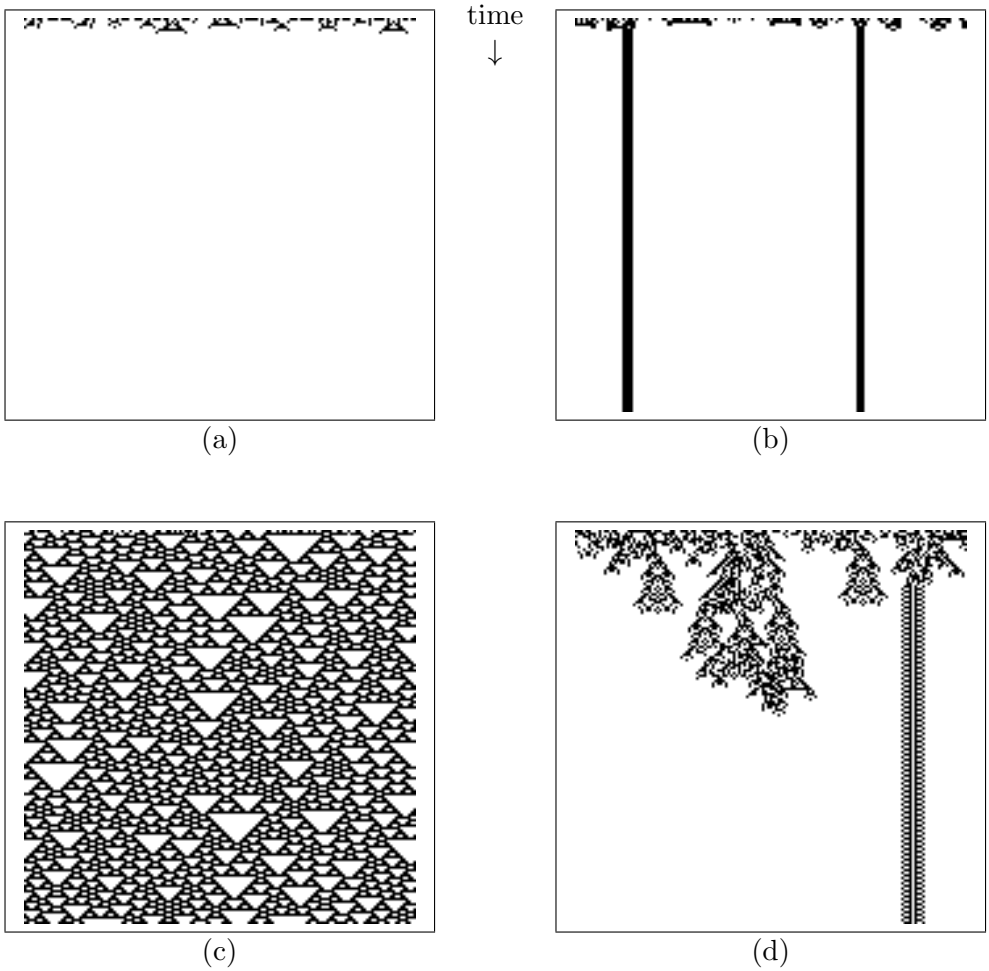

Figure 1.3: Wolfram classes. One dimensional CAs are shown, where the horizontal axis depicts the configuration at a certain time t and the vertical axis depicts successive time steps (increasing down the page). CAs are binary (2 states per cell) with radius $r = 2$ (two neighbors on both sides of the cell). (a) Class I. (b) Class II. (c) Class III. (d) Class IV.

notable features: massive parallelism, locality of cellular interactions, and simplicity of basic components (cells). They perform computations in a distributed fashion on a spatially extended grid. As such they differ from the standard approach to parallel computation in which a problem is split into independent sub-problems, each solved by a different processor, later to be combined in order to yield the final solution. CAs suggest a new approach in which complex behavior arises in a bottom-up manner from non-linear, spatially extended, local interactions [20]. This is often referred to as *emergent computation*, meaning the appearance of global information processing capabilities that are not explicitly represented in the system's elementary components or in their local interconnections [9]. The CA's properties greatly facilitate its implementation as electronic hardware [20, 23] (see Chapter **??**). CAs also suggest a possible approach to attaining novel computational architectures at the nanometer scale [1].

When considering CAs that perform computations two possibilities manifest themselves: (1) Embedding a universal Turing machine within the CA (see Chapter **??**), or (2) using the CA in a direct, parallel manner: the input to the computation is encoded as an initial configuration, the output is the configuration after a certain number of time steps, and the intermediate steps that transform the input to the output are considered to be the steps in the computation (Chapter **??**). In this latter case, the "program" emerges through "execution" of the CA rule in each cell.

1.4 Variations on the original model

In this section we briefly outline a number of variations of the original, classic CA model, presented above. These variations concern the cellular rules, the connectivity architectures, temporal considerations, and determinism. These issues are further discussed in Chapter **??**.

1.4.1 Non-uniform CAs

Non-uniform cellular automata function in the same way as uniform ones, the only difference being in the cellular rules that need not be identical for all cells. Note that non-uniform CAs share the basic "attractive" properties of uniform ones (simplicity, parallelism, locality). From a hardware point of view we observe that the resources required by non-uniform CAs are identical to those of uniform ones since a cell in both cases contains a rule. Although simulations of uniform CAs on serial computers may optimize

memory requirements by retaining a single copy of the rule, rather than have each cell hold one, this in no way detracts from our argument. Indeed, one of the primary motivations for studying CAs stems from the observation that they are naturally suited for hardware implementation with the potential of exhibiting extremely fast and reliable computation that is robust to noisy input data and component failure [20].

Non-uniform CAs have been investigated by [31] who discuss 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. Garzon [12] presented two generalizations of cellular automata, namely, discrete neural networks and automata networks. These were compared to the original model from a computational point of view which considers the classes of problems such models can solve. Chapter **??** examines the non-uniform CA model from a computational aspect as well as an evolutionary one [20].

1.4.2 Non-standard architectures

Another possible variation concerns the connectivity pattern of the cells, the architecture, which is standard and homogeneous in the original CA. One can consider so-called *non-standard* connectivity architectures, where each cell has a small, identical number of connections, yet not necessarily from its most immediate neighboring cells [20, 22]. It can be shown that such architectures are computationally more efficient than standard architectures in solving certain computational tasks. Furthermore, one can successfully *evolve* non-standard architectures using evolutionary computation techniques. These issues are treated in Chapter **??**.

1.4.3 Asynchronous CAs

One of the prominent features of the CA model is its synchronous mode of operation, meaning that all cells are updated simultaneously. A preliminary study of asynchronous CAs, where one cell is updated at each time step, was carried out by [14], where the different dynamical behavior of synchronous and asynchronous CAs was compared; the authors argued that some of the apparent self-organization of CAs is an artifact of the synchronization of the clocks. Wolfram [33] noted that asynchronous updating makes it more difficult for information to propagate through the CA and that, furthermore, such CAs may be harder to analyze. Asynchronous CAs have also been

discussed in [17, 3, 20]. This issue is treated within an evolutionary context in Chapter **??**, where asynchronous CAs are evolved to solve computational tasks [25].

1.4.4 Probabilistic CAs

In a deterministic cellular automaton, for any given input, the system always goes through the same trajectory of states, ending with the same output. For a nondeterministic, or probabilistic CA the same input may result in different trajectories, and possibly different outputs. Nondeterminism may be inherent to the system's functional definition or it may result due to faults. As an example, consider a two-state CA, where a cell updates its state in a non-deterministic manner, setting it at the next time step to that specified in the rule table, with probability $1 - p_f$, or the complementary state, with probability p_f. The value p_f can be regarded as the probability that a cell will malfunction (this type of fault was studied, e.g., by [24]).

Bibliography

[1] S. C. Benjamin and N. F. Johnson. A possible nanometer-scale computing device based on an adding cellular automaton. *Applied Physics Letters*, 70(17):2321–2323, April 1997.

[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] H. Bersini and V. Detour. Asynchrony induces stability in cellular automata based models. In R. A. Brooks and P. Maes, editors, *Artificial Life IV*, pages 382–387, Cambridge, Massachusetts, 1994. The MIT Press.

[4] A. Burks, editor. *Essays on Cellular Automata*. University of Illinois Press, Urbana, Illinois, 1970.

[5] B. Chopard and M. Droz. *Cellular Automata Modeling of Physical Systems*. Cambridge University Press, Cambridge, UK, 1998.

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

[7] K. Culik II, L. P. Hurd, and S. Yu. Computation theoretic aspects of cellular automata. *Physica D*, 45:357–378, 1990.

[8] G. B. Ermentrout and L. Edelstein-Keshet. Cellular automata approaches to biological modeling. *Journal of Theoretical Biology*, 160:97–133, 1993.

[9] S. Forrest, editor. *Emergent Computation: Self-organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks*. The MIT Press, Cambridge, MA, 1991.

[10] E. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21:219–253, 1982.

[11] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the Navier-Stokes equation. *Physical Review Letters*, 56:1505–1508, 1986.

[12] M. Garzon. *Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks*. Springer-Verlag, Berlin, 1995.

[13] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory Languages and Computation*. Addison-Wesley, Redwood City, CA, 1979.

[14] T. E. Ingerson and R. L. Buvel. Structure in asynchronous cellular automata. *Physica D*, 10:59–68, 1984.

[15] 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.

[16] N. Margolus. Physics-like models of computation. *Physica D*, 10:81–95, 1984.

[17] M. A. Nowak, S. Bonhoeffer, and R. M. May. Spatial games and the maintenance of cooperation. *Proceedings of the National Academy of Sciences USA*, 91:4877–4881, May 1994.

[18] J.-Y. Perrier, M. Sipper, and J. Zahnd. Toward a viable, self-reproducing universal computer. *Physica D*, 97:335–352, 1996.

[19] M. Sipper. An introduction to artificial life. *Explorations in Artificial Life (special issue of AI Expert)*, pages 4–8, September 1995. Miller Freeman, San Francisco, CA.

[20] M. Sipper. *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*. Springer-Verlag, Heidelberg, 1997.

[21] M. Sipper. If the milieu is reasonable: Lessons from nature on creating life. *Journal of Transfigural Mathematics*, 3(1):7–22, 1997.

[22] M. Sipper and E. Ruppin. Co-evolving architectures for cellular machines. *Physica D*, 99:428–441, 1997.

[23] M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Pérez-Uribe, and A. Stauffer. A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems. *IEEE Transactions on Evolutionary Computation*, 1(1):83–97, April 1997.

[24] M. Sipper, M. Tomassini, and O. Beuret. Studying probabilistic faults in evolved non-uniform cellular automata. *International Journal of Modern Physics C*, 7(6):923–939, 1996.

[25] M. Sipper, M. Tomassini, and M. S. Capcarrère. Evolving asynchronous and scalable non-uniform cellular automata. In G. D. Smith, N. C. Steele, and R. F. Albrecht, editors, *Proceedings of International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA97)*, pages 66–70. Springer-Verlag, Vienna, 1997.

[26] T. Toffoli. Cellular automata mechanics. Technical Report 208, Comp. Comm. Sci. Dept., The University of Michigan, 1977.

[27] T. Toffoli. Reversible computing. In J. W. De Bakker and J. Van Leeuwen, editors, *Automata, Languages and Programming*, pages 632–644. Springer-Verlag, 1980.

[28] T. Toffoli. Cellular automata as an alternative to (rather than an approximation of) differential equations in modeling physics. *Physica D*, 10:117–127, 1984.

[29] T. Toffoli and N. Margolus. *Cellular Automata Machines*. The MIT Press, Cambridge, Massachusetts, 1987.

[30] G. Vichniac. Simulating physics with cellular automata. *Physica D*, 10:96–115, 1984.

[31] G. Y. Vichniac, P. Tamayo, and H. Hartman. Annealed and quenched inhomogeneous cellular automata. *Journal of Statistical Physics*, 45:875–883, 1986.

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

[33] S. Wolfram. *Cellular Automata and Complexity*. Addison-Wesley, Reading, MA, 1994.