Chapter 2
Simplicity in the Universe of Cellular Automata

Because of their simplicity, rules of cellular automata can easily be understood. In a very simple version, we consider *two-state one-dimensional* (1D) *cellular automata* (CA) made of identical cells with a *periodic boundary condition*. In this case, the object of study is a ring of coupled cells with $L = I + 1$ cells, labeled consecutively from $i = 0$ to $i = I$ (Fig. 1a). Each cell i has two *states* $u_i \in \{0, 1\}$, which are coded by the colors blue and red, respectively. A clock sets the pace in discrete intervals by iterations or generations. The state u_i^{t+1} of all i at time $t + 1$ (i.e. the next generation) is determined by the states of its nearest neighbors u_{i-1}^t, u_{i+1}^t, and itself u_i^t at time t (Fig. 1c), that is, by a *Boolean function* $u_i^{t+1} = N\left(u_{i-1}^t, u_i^t, u_{i+1}^t\right)$, in accordance with a prescribed *Boolean truth table* of $8 = 2^3$ distinct 3-input patterns (Fig. 1d).

From Simple Local Rules to Global Complex Patterns

These eight 3-input patterns can nicely be mapped into the eight vertices of a toy cube (Fig. 1b), henceforth called a *Boolean cube* (Chua et al. 2002). The output of each prescribed 3-input pattern is mapped onto the corresponding colors (red for 1, blue for 0) at the vertices of the Boolean cube (in Fig. 1d yet unspecified). Because there are $2^8 = 256$ distinct combinations of 8 bits, there are exactly 256 Boolean cubes with distinct vertex color combinations. Thus we get a gallery of picturesque toy cubes.

It is convenient to associate the 8-bit patterns of each Boolean function with a decimal number N representing the corresponding 8-bit word, namely $N = \beta_7 \cdot 2^7 + \beta_6 \cdot 2^6 + \beta_5 \cdot 2^5 + \beta_4 \cdot 2^4 + \beta_3 \cdot 2^3 + \beta_2 \cdot 2^2 + \beta_1 \cdot 2^1 + \beta_0 \cdot 2^0$ with $\beta_i \in \{0, 1\}$. Notice that since $\beta_i = 0$ for each blue vertex in Fig. 1b, N is simply obtained by adding the weights (indicated next to each pattern in Fig. 1b) associated with all red vertices. For example, for the Boolean cube shown in Fig. 2b,

K. Mainzer and L. Chua, *The Universe as Automaton*, SpringerBriefs in Complexity, DOI: 10.1007/978-3-642-23477-4_2, © The Author(s) 2012

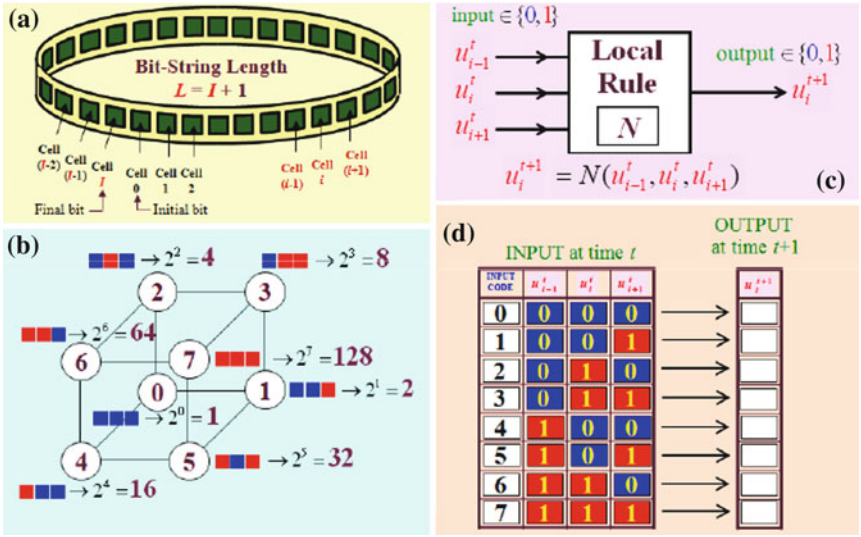

Fig. 1 Scheme of a two-state one-dimensional cellular automaton with local rule N

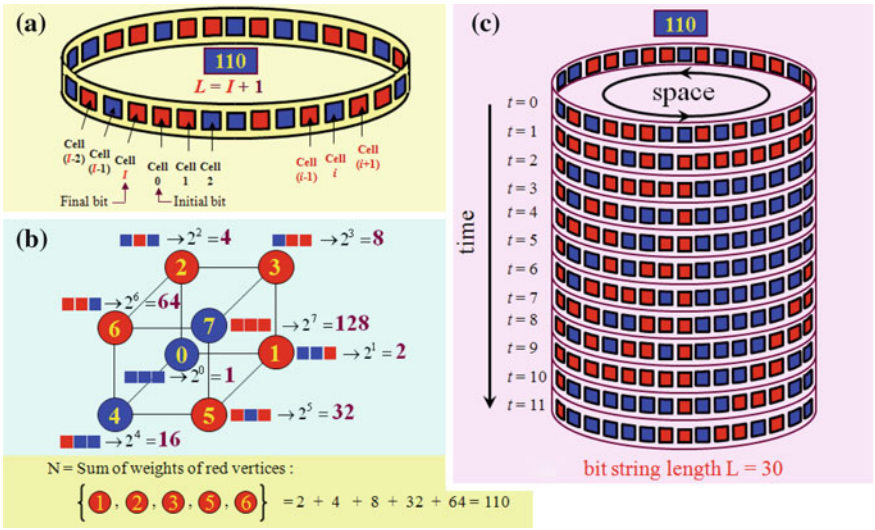

Fig. 2 Example of local rule 110

we have $N = 0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 2^6 + 2^5 + 2^3 + 2^2 + 2^1 = 110$.

For the example of local rule 110, the ring and the colored vertices of the Boolean cube are shown in Fig. 2a, b. Given any initial binary bit-configuration at

time $t = 0$, the local rule N is used to update the state of each cell i at time $t + 1$, using the states of the three neighboring cells $i - 1$, i, and $i + 1$, centered at location i, respectively. The *space–time pattern* for the initial state is shown in Fig. 2c for $t = 0, 1, 2, \ldots, 11$.

In principle, one can draw and paint the patterns of CA following these rules step by step. Modern computers with high speed and capacity allow extensive computer experiments to study pattern formations of these automata. Stephen Wolfram discovered remarkable analogies with patterns in physics and biology (Wolfram 2002). In the world of CA many phenomena of the physical world seem to evolve. Some automata generate symmetric patterns reminding us of the coloring in sea shells, skins or feathers. Other automata reproduce rhythms like oscillating waves. Some of these automata stop their development after a finite number of steps, independent of their initial state, and remain in a constant color state like a system reaching an equilibrium state for all future steps. Some automata develop complex patterns reminding us of the growth of corals or plants, depending sensitively on tiny changes to the initial states. This phenomenon is well-known as the butterfly-effect, when local events lead to global effects in chaotic and unstable situations (such as weather and climate). Even these chaotic patterns can be generated by CA.

One can try to classify these patterns with respect to their outward appearance just as zoologists and botanists distinguish birds and plants in taxonomies. This observational method is used by Wolfram for patterns experimentally generated by computers. But sometimes outward features are misleading. The fundamental question arises: Are there laws of complex pattern formation for CA like those in nature? Can the development of complex patterns be predicted in a mathematically rigorous way as in physics? We argue for a mathematically precise explanation of the dynamics in CA. Therefore, they must also be characterized by complex dynamical systems determined with differential equations like those in physics. This is, of course, beyond the scope of elementary rules of toy worlds. But we should keep this perspective in mind.

Cellular Automata as Dynamical Systems

For maximum generality, each cell i is assumed to be a *dynamical system* with an intrinsic state x_i, an output y_i, and *three inputs* u_{i-1}, u_i, and u_{i+1} where u_{i-1} denotes the input coming from the left neighboring cell $i - 1$, u_i denotes the self input of cell i, and $i + 1$ denotes the input coming from the right neighboring cell $i + 1$ in the ring of Fig. 1a. Each cell evolves with its prescribed dynamics and its own time scale. When coupled together, the system evolves consistently with its own rule as well as the rule of interaction imposed by the coupling laws.

Each *input* is assumed to be a *constant integer* $u_i \in \{-1, 1\}$, and the *output* y_i *converges* to a *constant* either –1 or 1 from a zero initial condition $x_i(0) = 0$. Actually, it takes a finite amount of time for any dynamical system to converge to

an *attractor*. But, for the purpose of idealized CA, each attractor is assumed to be reached instantaneously. Under this assumption and with respect to the *binary input* and *output*, our *dynamical system* can be defined by a *nonlinear map* which is uniquely described by a *truth table* of three input variables $(u_{i-1},\ u_i,\ u_{i+1})$. The choice of {–1, 1} and not {0, 1} as binary signals is crucial, because the state x_i and output y_i evolves in *real time* via a carefully designed *scalar ordinary differential equation*. According to this differential equation, the output y_i which is defined via an output equation $y_i = y(x_i)$ tends to either 1 or –1 after the solution x_i (with zero initial state $x_i(0) = 0$), reaches a *steady state*. In this way, the *attractors* of the *dynamical system* can be used to encode a *binary truth table*.

Aside from the *cell's intrinsic time scale* (which is of no concern in CA), an external clocking mechanism is introduced to reset the input u_i of each cell i at the end of each clock cycle by feeding back the steady state *output* $y_i \in \{-1,\ 1\}$, as an updated *input* $u_i \in \{-1,\ 1\}$, for the *next iteration*. This mechanism corresponds to the *periodic boundary condition* of the 1D cellular automaton in Fig. 1a.

Although CA are concerned only with the ring's evolutions over *discrete times*, any computer used to simulate CA is always a *continuous time system* with very small but non-zero time scale. Computers use transistors as devices, and each CA iteration involves the physical evolution of millions of transistors with its own $u_i \in \{-1,\ 1\}$, intrinsic dynamics. These transistors evolve in accordance with a large system of nonlinear differential equations governing the entire internal computer circuit and return the desired output after converging to their attractors in a non-zero amount of time.

These considerations lead us to the important result that, even in *discrete systems* like CA, there are *two different time scales* involved. The first one applies to the rule N while the second applies to the global patterns of evolution. To understand the complex dynamics of global patterns, it is necessary to analyze both time scales. By unfolding the truth tables of CA into an appropriate nonlinear dynamical system, we can exploit the theory of *nonlinear differential equations* to arrive at phenomena based on a precise mathematical theory, and not only on empirical observations.

For this purpose, we substituted the binary symbol 0 by the *real number* –1, and the input and output values 0 and 1 in the truth table of Fig. 1d by the real numbers –1 and 1, respectively. An advantage of working with the numeric rather than the symbolic truth table is the remarkable insights provided by the equivalent Boolean cube representation. Here, the eight vertices of the cube (–1,–1,–1), (–1,–1,1), (–1,1,–1), (–1,1,1), (1,–1,–1), (1,–1,1), (1,1,–1) and (1,1,1) are located exactly at the coordinates $(u_{i-1},\ u_i,\ u_{i+1})$ of a *coordinate system* with the origin located at the center of the cube. The vertex $n = 0, 1, 2, \ldots, 7$ corresponding to row n of the truth table is coded blue if the output is –1, and red if the output is 1.

The choice of {–1, 1} instead of {0, 1} as binary signals is necessary, when the truth table is mapped onto a dynamical system where the states evolve in real time via an ordinary differential equation which is always based on the real number system. Each cell i is coupled only to its left neighbor cell $i - 1$ and right neighbor

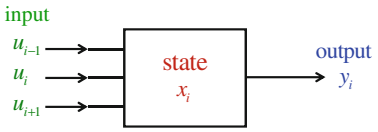

Fig. 3 Cell as dynamical system with state variable x_i, an output variable y_i, and three constant binary inputs u_{i-1}, u_i, and u_{i+1}

cell $i + 1$. As a dynamical system, each cell i has a *state variable* x_i, an *output variable* y_i, and three constant *binary inputs* u_{i-1}, u_i, and u_{i+1} (Fig. 3).

Thus, the *dynamical system* is determined by a

$$\begin{aligned}\text{state equation}: \dot{x}_i &= f(x_i; u_{i-1}, u_i, u_{i+1}) \\ x(0) &= 0 \,(\text{initital condition}) \\ \text{output equation}: y_i &= y(x_i).\end{aligned}$$

Every CA can be mapped into a nonlinear dynamical system whose attractors encode precisely the associated truth table $N = 0, 1, 2, 3, \ldots, 255$. Function f models the *time-dependent change of states* and is defined by a scalar, ordinary differential equation of the form

$$\dot{x} = \mathrm{g}(x_i) + w(u_{i-1}, u_i, u_{i+1}) \text{ with } \mathrm{g}(x_i) \triangleq -x_i + |x_i + 1| - |x_i - 1|.$$

There are many possible choices of *nonlinear basis functions* for $g(x_i)$ and $w(u_{i-1}, u_i, u_{i+1})$. We have chosen the absolute value function $|x| = x$ for positive numbers x and $|x| = -x$ for negative numbers x as a nonlinear basis function, because the resulting equation can be expressed in an optimally compact form, and it allows us to derive the solution of the state equation in an explicit form. The scalar function $w(u_{i-1}, u_i, u_{i+1})$ can be chosen to be a composite function $w(\sigma)$ of a single variable $\sigma \triangleq b_1 u_{i-1} + b_2 u_i + b_3 u_{i+1}$ with $w(\sigma) \triangleq \{z_2 \pm ||z_1 \pm |z_o + \sigma||\}$. This function is used to define the appropriate differential equation for generating the truth table of all 256 Boolean cubes. Thus, each rule of a cellular automaton corresponds to a particular set of *six real numbers* $\{z_o, z_1, z_2; b_1, b_2, b_3\}$, and *two integers* ± 1. Only 8 bits are needed to uniquely specify the differential equation associated with each rule N of a cellular automaton.

It can be proven that once the parameters defining a particular rule N are specified, then for any one of the eight inputs u_{i-1}, u_i, and u_{i+1} listed in the corresponding truth table of N, the solution x_i of the scalar differential equation will either increase monotonically from the initial state $x_i = 0$ towards a *positive equilibrium value* $\overline{x_i}(n) \geq 1$, henceforth denoted by attractor $Q_+(n)$, or decrease monotonically towards a *negative equilibrium state* $\overline{x_i}(n) \leq -1$, henceforth denoted by attractor $Q_-(n)$, when the input (u_{i-1}, u_i, u_{i+1}) is chosen from the coordinates of vertex n of the associated Boolean cube, or equivalently, from row n of the corresponding truth table, for $n = 0, 1, 2, \ldots, 7$ (Chua et al. 2002). Vertex n is painted red whenever its equilibrium value $\overline{x_i}(n) \geq 1$, and blue whenever $\overline{x_i}(n) \leq -1$. The color of all eight vertices for the associated *Boolean cube* will

then be uniquely specified by the *equilibrium solutions* of the eight associated *differential equations*.

In general, we can summarize: once the parameters associated with a particular rule of a cellular automaton are specified, the corresponding *truth table* or *Boolean cube*, will be uniquely generated by the *scalar differential equation* alone. If the output equation of the dynamical system is $y_i = y(x_i) \triangleq \frac{1}{2}(|x_i + 1| - |x_i - 1|)$, then $y_i = +1$ when $x_i \geq 1$, and $y_i = -1$ when $x_i \leq -1$. The steady-state output at equilibrium is given explicitly by the formula $y_i = \text{sgn}\{\{\text{w}(\sigma)\}$ for any function $\text{w}(\sigma) \triangleq \text{w}(u_{i-1}, u_i, u_{i+1})$ with signum function sgn $(x) = +1$ for positive numbers x, sgn $(x) = -1$ for negative numbers x and sgn $(0) = 0$. For the particular $\text{w}(\sigma)$ in Fig. 4 the output (color) at equilibrium is given explicitly by the

$$\textit{attractor color code}: y_i = \text{sgn}\{z_2 \pm |[z_1 \pm |z_o + \sigma]|\}.$$

Figure 4 contains four examples of dynamical systems and the rules they encode, each one identified by its rule number $N = 0, 1, 2, \ldots, 255$. The truth table for each rule N is generated by the associated dynamical system defined in the upper portion of each quadrant, and not from the truth table, thereby proving that each dynamical system and the rule of the cellular automaton that it encodes are one and the same. The truth table for each rule in Fig. 4 is cast in a format with only $2^{2^3} = 256$ distinct 1×3 neighborhood patterns. Each color picture consists of 30×61 pixels, generated by a 1D cellular automaton with 61 cells and a boundary condition with a specific rule N.

As an example, let us examine one of the rules from Fig. 4, rule 110, which will later on be identified as the simplest universal Turing machine known to date. With its differential equation, one can identify $\sigma = b_1 u_{i-1} + b_2 u_i + b_3 u_{i+1}$ with $b_1 = 1$, $b_2 = 2$, and $b_3 = -3$, and $w(\sigma) \triangleq \{z_2 \pm |[z_1 \pm |z_o + \sigma]|\}$ with $z_2 = -2$, $z_1 = 0$, and $z_o = -1$. Thus, the attractor color code is explicitly given by $y_i = \text{sgn}[-2 + |u_{i-1} + 2u_i - 3u_{i+1} - 1)|]$.

Digital Dynamics with Difference Equations

The dynamics of dynamical systems are modeled with continuous differential equations. For computing the dynamics for digital CA, a program must use a "do loop" instruction which feeds back the output y_i^t of each cell at iteration t back to its inputs, to obtain the output y_i^{t+1} at the next iteration $t + 1$. Using the superscripts t and $t + 1$ as iteration number from one to the next generation, we can express each rule N explicitly in the form of a *nonlinear difference equation* with

$$u_i^{t+1} = \text{sgn}\{z_2 + c_2|[z_1 + c_1|(z_o + b_1 u_{i-1}^t + b_2 u_i^t + b_3 u_{i+1}^t|]|\},$$

where the *eight parameters* $\{z_o, z_1, z_2; b_1, b_2, b_3; c_1, c_2\}$ are given for each rule. Thus, the first main result of this chapter is that each of 256 1D CA that were studied by Steven Wolfram experimentally can be generated from a single scalar

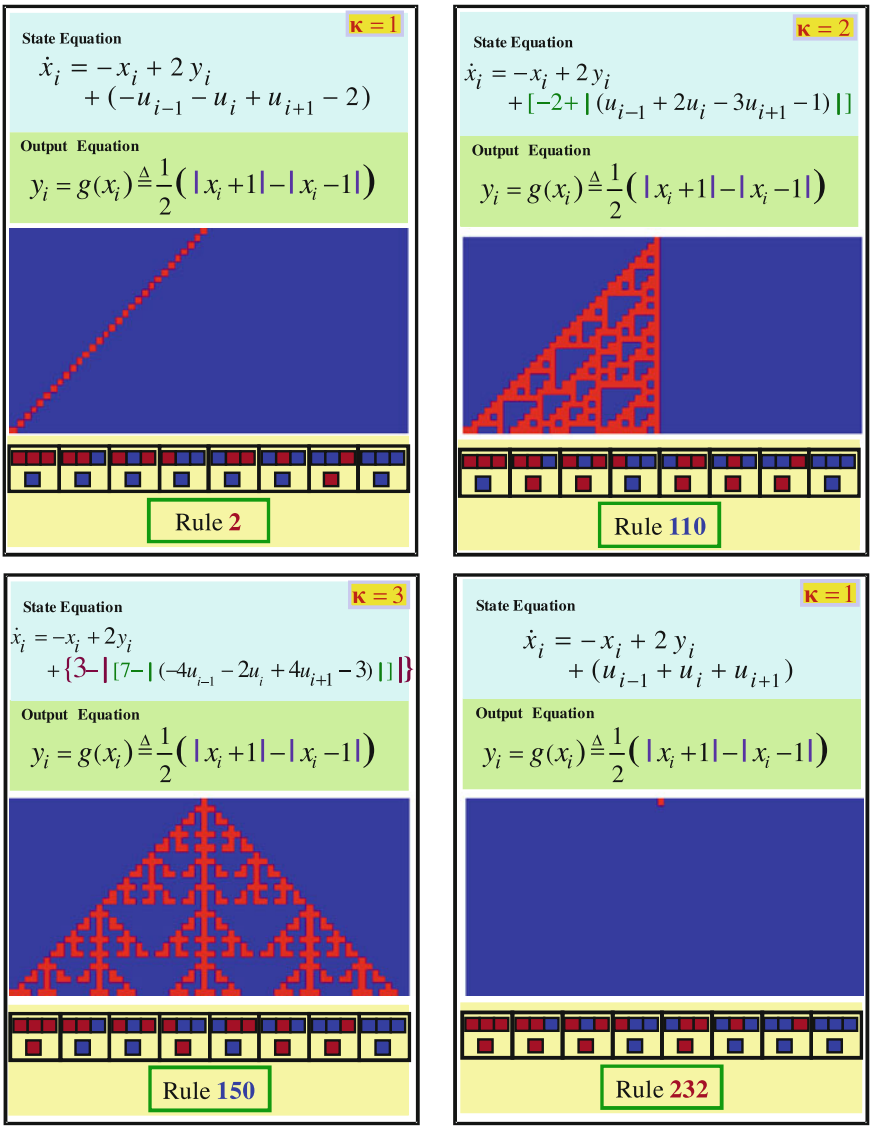

Fig. 4 Cellular automata with rules 2, 110, 150, 232 as dynamical systems. The initial condition is $x(0) = 0$

nonlinear differential equation or a corresponding nonlinear difference equation with at most eight parameters. These equation are also universal in the sense of a universal Turing machine (UTM), because we will later on see that at least one of the 256 rules (for example, rule 110) is capable of universal computation (Chua et al. 2003). For rule 110 (Fig. 5), we get $u_i^{t+1} = \text{sgn}\,(-2 + |u_{i-1}^t + 2u_i^t -$

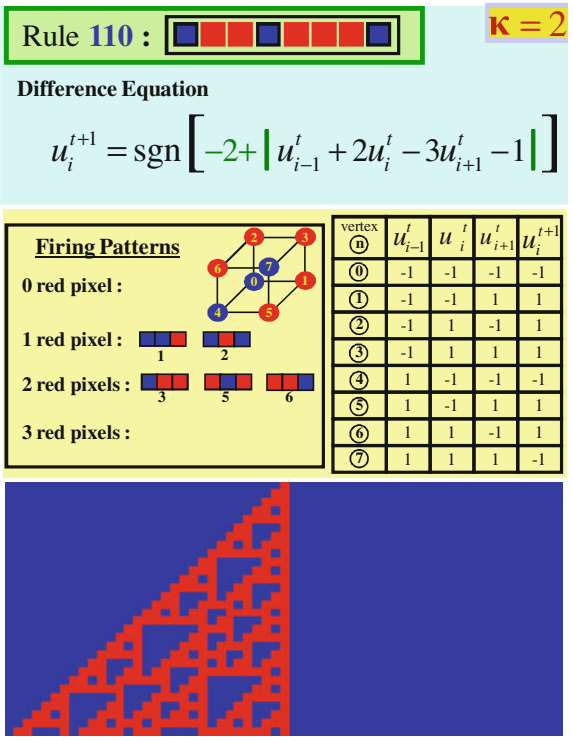

Fig. 5 Cellular automaton as dynamical system with difference equation

$3u_{i+1}^t - 1|$). This kind of difference equation can be understood with elementary knowledge of basic mathematics, although it demonstrates important features of nonlinear dynamics.

References

L.O. Chua, S. Yoon, R. Dogaru, A nonlinear dynamics perspective of Wolfram's new kind of science Part I: Threshold of complexity. Int. J. Bifurcation Chaos (IJBC) **12**(12), 2655–2766 (2002)

L.O. Chua, V.I. Sbitnev, S. Yoon, A nonlinear dynamics perspective of Wolfram's new kind of science Part II: Universal neuron. Int. J. Bifurcation Chaos (IJBC) **13**(9), 2377–2491 (2003)

S. Wolfram, *A New Kind of Science* (Wolfram Media Inc, Champaign, 2002)

http://www.springer.com/978-3-642-23476-7

The Universe as Automaton
From Simplicity and Symmetry to Complexity
Mainzer, K.; Chua, L.
2012, VIII, 108 p. 30 illus. in color., Softcover
ISBN: 978-3-642-23476-7