Elementary Cellular Automata with Minimal Memory and Random Number Generation

Ramón Alonso-Sanz [1,2]
Larry Bull [1]

[1] *Bristol Institute of Technology*
University of the West of England, Bristol
Frenchay Campus, Bristol BS16 1QY, UK
[2] *Polytechnic University of Madrid, Spain*
ramon.alonso@upm.es
Larry.Bull@uwe.ac.uk

The effect of endowing cells with memory of their last two state values in elementary one-dimensional cellular automata is analyzed in this paper. The potential value of such elementary cellular automata with minimal memory embedded in cells as random number generators is assessed.

1. Conventional Cellular Automata

Cellular automata (CA) are discrete, spatially explicit extended dynamic systems. CA systems are composed of adjacent cells or sites arranged as a regular lattice, which evolve in discrete time steps. Each cell is characterized by an internal state whose value belongs to a finite set. The updating of these states is made simultaneously according to a common local transition rule involving a neighborhood of each cell.

Thus, if $\sigma_i^{(T)}$ is taken to denote the value of cell i at time step T, the site values evolve by iterating the mapping $\sigma_i^{(T+1)} = \phi\left(\left\{\sigma_j^{(T)}\right\}, j \in \mathcal{N}_i\right)$, with $\mathcal{N}_i$ standing for the set of cells in the neighborhood of cell i.

Here we will consider the simplest scenario, that of elementary CA [1], that is, one-dimensional CA with two possible state values ($\sigma \in \{0, 1\}$), and rules operating on nearest neighbors: $\sigma_i^{(T+1)} = \phi\left(\sigma_{i-1}^{(T)}, \sigma_i^{(T)}, \sigma_{i+1}^{(T)}\right)$. Elementary rules are characterized by a sequence of binary values β associated with each of the eight possible triplets:

$$\left(\sigma_{i-1}^{(T)}, \sigma_i^{(T)}, \sigma_{i+1}^{(T)}\right): \begin{matrix} 111 & 110 & 101 & 100 & 011 & 010 & 001 & 000 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 & \beta_8 \end{matrix}.$$

The rules are conveniently specified by their rule number, $R = \sum_{i=1}^{8} \beta_i 2^{8-i}$ varying in the [0, 255] interval.

Legal rules are reflection symmetric $(\beta_2 = \beta_5$ and $\beta_4 = \beta_7)$ and quiescent $(\beta_8 = 0)$. Complementary rules assign complementary β values, that is, $\overline{\beta}_i = 1 \oplus \beta_i$, so the rule number of two complementary rules add up to 255.

We will pay particular attention in this study to rule 30 (00011110) and to the legal rules 90 (01011010) and 150 (10010110). The spatio-temporal patterns of these rules from a single site seed are shown in Figure 1 up to $T = 60$. In totalistic rules the value of a site depends only on the sum of the values of its neighbors and not on their individual values. The rules 90 and 150 are totalistic linear (or additive) rules that employ only XOR logic, that is, sums performed modulo 2 in the two-state scenario. (A further explanation of the general meaning for a system to be *additive* is found in [1], p. 952.) Noting the sum as $\oplus$, it is: R90: $\sigma_i^{(T+1)} = \sigma_{i-1}^{(T)} \oplus \sigma_{i+1}^{(T)}$, and R150: $\sigma_i^{(T+1)} = \sigma_{i-1}^{(T)} \oplus \sigma_i^{(T)} \oplus \sigma_{i+1}^{(T)}$. Or in matricial terms, $\mathbf{C}^{(T+1)} = \mathbf{M}\,\mathbf{C}^{(T)}$, where $\mathbf{C}^{(T)}$ stands for the configuration at time step T, $\mathbf{C}^{(T)} = \left(\sigma_1^{(T)}, \sigma_2^{(T)}, \ldots, \sigma_{N-1}^{(T)}, \sigma_N^{(T)}\right)'$, and $\mathbf{M}$ is the transition matrix of the linear rule. Thus,

$$\mathbf{M}_{90} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & \ldots & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & \ldots & 0 & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 & 0 & \ldots & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & \ldots & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{M}_{150} = \mathbf{M}_{90} \oplus \mathbf{I}.$$

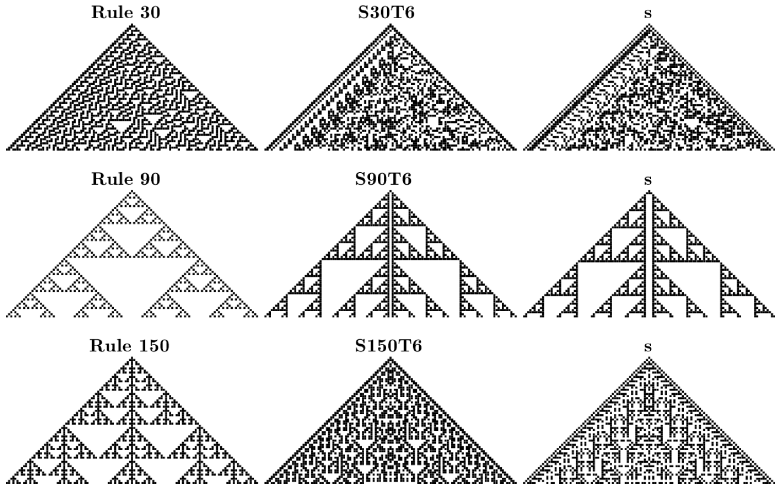


Figure 1. The ahistoric rules 30, 90, and 150 (left), and these rules with rule 6 (parity) as memory (SXT6). In the latter case, the evolving patterns of the featured (s) cells are also shown.

2. Cellular Automata with Memory

Conventional CA are ahistoric (memoryless); that is, the new state of a cell depends on the neighborhood configuration solely at the preceding time step. Historic memory can be embedded in the CA dynamics by endowing memory in cells without altering the mappings ϕ. Thus, $\sigma_i^{(T+1)} = \phi\left(\left\{\sigma_j^{(T)}\right\}, j \in \mathcal{N}_i\right)$, $s_i^{(T)}$ is a state function of the series of states of the cell i up to time step T. In the case of elementary rules, $\sigma_i^{(T+1)} = \phi\left(s_{i-1}^{(T)}, s_i^{(T)}, s_{i+1}^{(T)}\right)$.

Cells may be featured by a weighted mean value m of *all* their previous states [2-7]: $s_i^{(T)} = m\left(\sigma_i^{(1)}, \sigma_i^{(2)}, \ldots, \sigma_i^{(T)}\right)$, or of a limited trailing memory $s_i^{(T)} = m\left(\sigma_i^{(T-\tau+1)}, \ldots, \sigma_i^{(T-1)}, \sigma_i^{(T)}\right)$. Limiting memory to the last three time steps ($\tau = 3$), cells may be featured by some elementary rule f of them, that is, $s_i^{(T)} = f\left(\sigma_i^{(T-2)}, \sigma_i^{(T-1)}, \sigma_i^{(T)}\right)$, such as its most frequent value [2], or the totalistic rule 90 or 150 [8].

We will consider in this work the lowest degree of proper ($\tau > 1$) memory conceivable, that is, featuring cells by Boolean functions of their last two states ($\tau = 2$): $s_i^{(T)} = f\left(\sigma_i^{(T-1)}, \sigma_i^{(T)}\right)$, with $s_i^{(1)} = \sigma_i^{(1)}$. These mappings are characterized by a sequence of binary values β

associated with each of the four possible pairs $\left(\sigma_i^{(T-1)}, \sigma_i^{(T)}\right)$. So as a two-bit analog of the codification in elementary CA:

$$\begin{array}{cccc} 11 & 10 & 01 & 00 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 \end{array} \equiv \sum_{i=1}^{4} \beta_s \, 2^{4-i} = \mathcal{R}.$$

The rule number of these two-input rules varies in the [0, 15] interval. Rule $\mathcal{R} = 10$ is the identity rule $s_i^{(T)} = \sigma_i^{(T)}$, and $\mathcal{R} = 6$ is now the parity rule $s_i^{(T)} = \sigma_i^{(T-1)} \oplus \sigma_i^{(T)}$. Complementary rules add up to 15 in this context.

This implementation of memory of the last two states will often be termed $\tau = 2$ memory. The spatial rules ϕ will be referred to as S-rules, and the memory rules f, actuating on time, as T-rules. Thus, SXTY will refer to the spatial rule with number X actuating on cells featured by the memory rule with number Y.

3. Elementary Rules with Minimal Memory

3.1 Spatio-Temporal Patterns

Figure 1 shows the effect of featuring cells by rule 6, on rules 30, 90, and 150, resulting in rules S30T6, S90T6, and S150T6. At the second time step, the actual configurations (■■■ , ■ ■, and ■■■ respectively) and those of featured states (■ ■ , ■■■, and ■ ■) differ. Consequently the patterns for the historic and ahistoric automata (typically) diverge as soon as at $T = 3$. Evolution in Figure 1 is shown up to $T = 60$. (A simple computer code for rule 150 with memory can be found at uncomp.uwe.ac.uk/alonso-sanz under Cellular Automata with memory.)

Appendix A shows the effect of minimal memory on some elementary rules starting from a single site seed, whereas Appendix B deals with evolving patterns starting at random with the same initial configuration. The spatio-temporal evolution of a register with 121 cells and periodic boundary conditions is shown in Appendix B up to $T = 100$.

Complementary rules have the same effect on rule 90 and on the elementary complementary to rule 90, that is, rule 165. This is reflected in Appendices A and B, and also holds when keeping memory of the last three time steps [8].

Linear rules remain linear when cells are endowed with linear memory rules. Thus, endowing the parity rule (rule 6) of the two last states in cells upon the elementary linear produces for S90T6:

$$\sigma_i^{(T+1)} = \left(\sigma_{i-1}^{(T)} \oplus \sigma_{i-1}^{(T-1)}\right) \oplus \left(\sigma_{i+1}^{(T)} \oplus \sigma_{i+1}^{(T-1)}\right),$$

and for S150T6:

$$\sigma_i^{(T+1)} = \left(\sigma_{i-1}^{(T)} \oplus \sigma_{i-1}^{(T-1)}\right) \oplus \left(\sigma_i^{(T)} \oplus \sigma_i^{(T-1)}\right) \oplus \left(\sigma_{i+1}^{(T)} \oplus \sigma_{i+1}^{(T-1)}\right).$$

In matricial terms:

$$\mathbf{C}^{(T+1)} = \mathbf{M}\left(\mathbf{C}^{(T)} \oplus \mathbf{C}^{(T-1)}\right) = \mathbf{M}\mathbf{C}^{(T)} \oplus \mathbf{M}\,\mathbf{C}^{(T-1)}.$$

3.2 Cycles

Cycles are harder to find in CA with memory than in the conventional ahistoric scenario, in which the mere repetition of a sole pattern marks the beginning of a cycle. This is not so in our CA with memory, as *two* consecutive patterns have to be repeated to start a cycle.

As a simple example, Figure 2 shows the ahistoric dynamics of rule 150 and that of S150T6 in small lattices of sizes $N = 5$ and $N = 11$, starting from a single live cell in its central site (periodic boundary conditions imposed on the edges). The ahistoric evolution generates a period-three oscillator as soon as $T = 4$ when $N = 5$. In the historic scenario, the first repetition of two consecutive patterns (again the first two) is achieved at $T = 16$, a value equal to half the total number of possible configurations $2^5 = 32$. When $N = 11$, the oscillator is of period 31 (the maximum attainable [9]) in the ahistoric formulation, whereas in S150T6 the period length is 93, notably lower than the total number of different configurations $\left(2^{11} = 2048\right)$ but notably longer than the former.

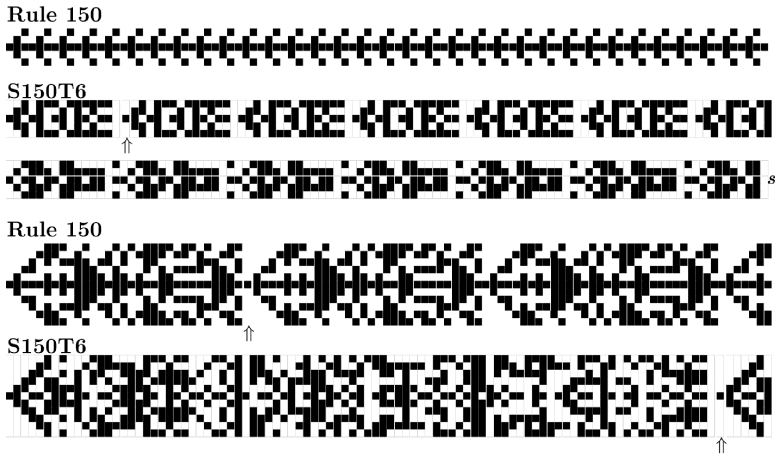


Figure 2. The ahistoric rule 150 and S150T6 in circular registers of sizes $N = 5$ (upper) and $N = 11$ (lower). Evolution up to $T = 100$.

The last component of a cycle in the $N = 5$ simulation of S150T6 in Figure 2 is that of the empty configuration (as a result of two consecutive identical configurations). But just after this, the dynamics restart. Such a "cataleptic" episode is infeasible in the ahistoric context.

In general, the maximum period conceivably attainable in an elementary CA of size N is 2^N, whereas in CA with a memory of two time steps the upper bound of period length becomes $(2^N)^2$. That would lead to the conjecture that CA with memory in cells explore the configuration space better, so that the number of unreachable configurations is smaller compared to conventional CA, that is, the Gardens of Eden are less populated [10]. In any case, it takes into account that the main features of the map ϕ prevail; so that, as an example, configurations containing an odd number of sites with value 1 can never be generated by the evolution of rule 90; this is so also with memory.

In order to circumvent the difficult analytical study of the cycles in the CA dynamics, the so-called return map helps to visually detect the mere existence of cycles by plotting the points representing successive configurations. Thus, (x_T, x_{T+1}), where x_T is a real number representing the configuration at time step T. Usually the binary configurations are mapped in the $[0, 2]$ interval as follows: $x_T = \sigma_1^{(T)} + \sum_{i=2}^{N} \sigma_i^{(T)}(0.5)^{i-1}$. But here the configurations will be mapped in the [0, 1] interval by dividing the integer representing the binary configuration $x_T = \sum_{i=1}^{N} \sigma_i^{(T)} 2^{N-i}$, by the maximum integer attainable in a register of size N, that is, $x_{\max} = \sum_{i=1}^{N} 2^{N-i}$.

Figures 3 and 4 show the return maps of rules 30, 90, and 150 and rules S30T6, S90T6, and S150T6, respectively. The same initial random configuration over an $N = 50$ register is set in both figures. The well-known characteristic signatures of the ahistoric rules are completely changed with parity memory into other ones with a random aspect.

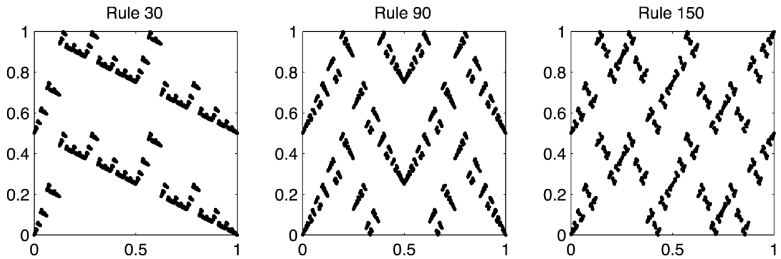


Figure 3. Pairs of successive numbers in a simulation up to 10 000 time steps using rules 30, 90, and 150.

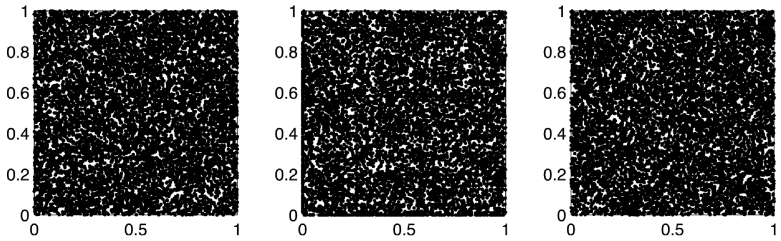


Figure 4. Pairs of successive numbers in a simulation up to 10 000 time steps using the rules with parity memory S30T6, S90T6, and S150T6.

3.3 Random Sequences

Random number sequences can be found in a large number of applications ranging from technological (e.g., cryptography, communications, computer-based gaming, VLSI, and hardware built-in self-test or BIST) to scientific large-scale simulations on supercomputers, which consume huge quantities of random numbers. In some applications, the quality of the random numbers is not that important. However, in many applications for which random number generators (RNGs) are most heavily used, such as Monte Carlo simulations in statistical physics, the quality of the RNG is crucial, as an inadequate election can produce incorrect results.

Yet finding good RNGs is a nontrivial task [11, 12]. Fairly recent studies have shown that CA are a promising technique for generating *pseudorandom* numbers. (Random sequences generated on a digital computer are usually referred to as pseudorandom, as distinguished from true random numbers, resulting from some natural physical process.) For examples see the coevolved CA combinations designed in [13] or the self-programmable CA in [14]. This kind of generator has the advantage of being highly parallel and thus is easily scalable with relatively little hardware cost. Moreover, due to their Boolean nature, CA are free of numerical errors derived from the finite precision of floating-point representation of real numbers in computers. This facilitates "portability", so when a random number sequence has been generated on some particular machine, it is easy to generate the same sequence on other machines. Hybrid CA using both rules 90 and 150 have also been implemented as RNGs [15-17]. The CA with memory here may also be called "hybrid", but in space and time. In both scenarios (memory and hybrid), a synergic effect emerges, so that rules that separately cannot be used as randomizers, when combined, have very good statistical properties.

As rules 90 and 150 operate à la congruential form, it is expected that they perform well with respect to the features of a RNG. The intriguing properties of rule 30 regarding randomness have been largely studied, among others, by Wolfram [1, 18]. So the radical transformation from Figure 3 to Figure 4 is not surprising.

The random aspect of the return map, the already mentioned increase in period length, and the fact that the correlation in state values induced by the local transition rule in conventional CA turns out to be weakened by the action of the temporal rule (causing a sort of random restart at every time step), are important characteristics of rules S30T6, S90T6, and S150T6. These features justify the potential value of the three rules as good RNGs, and accomplish, at least qualitatively, Knuth's comment that, "…random numbers should not be generated with a method chosen at random. Some theory should be used." [19, p. 6]

But these three positive features are not sufficient to qualify the mentioned rules as good RNGs. The rules still must past the tests of randomness specifically designed to decide on that qualification.

To cope with this issue, the rules 30, 90, and 150 were run in a register of 150 cells up to $T = 10\,000$. One hundred windows of size 50, from sites [1, 50] up to [100, 150], were sampled. (This mechanism differs from the one followed in the general study on proper CA as RNGs made in [20]. There, the 100 simulations were obtained starting from 100 different, and notably wider, initial random configurations. Savic [21] has tested two-neighbor CA for randomness.) Thus, extracting a real number in [0, 1] from every window as described previously, 100 series of 10 000 numbers each are obtained. The whole process is readily parallelizable, ideally by assigning every window to a processor, with the communication being just the states of the border cells.

Rules 30, 90, and 150 are unbiased in their production of 0s and 1s: they produce 1 when receiving any four of the eight possible inputs and 0 in the contrary case. As a consequence, the distribution of 0s and 1s in any of the windows sampled is, let us say, uniform, so that the real numbers generated exhibit the mean and variance of a U(0, 1) distribution, that is, 0.5 and 1 / 12 respectively.

3.3.1 Testing Randomness

In order to demonstrate the efficacy of a proposed RNG, it is usually subject to a battery of empirical and theoretical tests, among which the most well known are those described by Knuth in [19].

Although there are compiled batteries of tests to deal with the issue of qualifying randomness (e.g., the suites ENT [www.fourmilab.ch/random], NIST [csrc.nist.gov], or maybe the most currently applied DIEHARD [www.stat.fsu.edu/pub/diehard]), we opted for the robust set of four tests implemented by the Numerical Algorithms Group (NAG®, www.nag.co.uk). We embedded calls to the NAG subprograms in the Fortran source code used to implement the CA with memory in cells, so the whole process is expedited. To make sure that the process is well implemented, we reproduced the examples provided by NAG (the documentation of the NAG routines referring to the nonparametric tests of randomness [G08EAF,

G08EBF, G08ECF, and G08EDF] is straightforward and available on the NAG website). These examples analyze the randomness of a sequence of 10 000 numbers obtained by means of the NAG subprogram G05KAF, which generates numbers uniformly taken from a uniform distribution between 0 and 1, by a multiplicative congruential algorithm working modulo 2^{59}.

The four randomness tests implemented by NAG are the runs (concerned with the lengths of monotonically increasing or decreasing series), gaps (between numbers in a certain range), pairs, and triplets. A detailed description of these tests is beyond the scope of this paper. Suffice it to say that they belong to the class of Chi-square tests, in which the final operative parameter to decide on rejection of the null hypothesis of randomness is that of the tail probability associated with the chi-square statistic (with the corresponding degree of freedom), that is, the significance level. Qualified good results are in the (0.1, 0.9) interval, ideally close to 0.5, with extremities on both sides representing unsatisfactory random sequences.

3.3.2 Results

The mean and standard deviation of the probability parameter obtained are shown in Table 1. This table also reports (under the column headed NAG) the probability parameters obtained by applying the tests to 100 sequences of 10 000 random numbers generated by the program provided by NAG to generate pseudorandom numbers from a uniform distribution U(0, 1). These parameters act as a reference, as it is expected that numbers obtained with good randomizers should be close to them.

	NAG				**S30T6**			
	Run	Pair	Trip	Gap	Run	Pair	Trip	Gap
$\overline{P}$	.481	.527	.519	.508	.044	.560	.000	.110
sd	.308	.273	.300	.274	.104	.281	.000	.195
	S90T6				**S150T6**			
	Run	Pair	Trip	Gap	Run	Pair	Trip	Gap
$\overline{P}$	.000	.130	.000	.331	.464	.534	.000	.490
sd	.000	.181	.000	.301	.308	.299	.000	.278

Table 1. Randomness tests: mean ($\overline{P}$) and standard deviation (sd) of probability across 100 simulations of 10 000 observations each, obtained using rules with memory of the parity of the last two state values. Results of the simulations with the NAG package serve as a reference to the results with CA rules. Good simulations have probabilities in the (0.1, 0.9) interval, ideally close to 0.5.

Rules without memory fail to pass the tests. Their low P-values are given in [20]. It seems that the *regularities*, the triangular features appreciated in their spatio-temporal patterns, are translated into some kind of tendencies that the tests for randomness detect. These poor results obtained by the conventional ahistoric rules are due to the correlation induced by extracting the values of a wide sequence of adjacent sites. To try to remove correlation, it is customary to sample only a rather limited number of sites (either adjacent or spaced), or solely one as considered by Wolfram regarding rule 30.

Rules with parity memory (with no obvious patterns in their space-time diagrams) increase their performance, showing some acceptable probability parameters in Table 1 under some of the tests, but not under every test and dramatically failing in respect to the triplets test.

There is a known weakness common to all the conventional (non-CA) linear congruential generators, as pointed out by Marsaglia in [22]. If groups of successive values are used as the Cartesian coordinates of points in an n-dimensional space, they do not uniformly fill up the volume. Instead, they lie on a relatively small number of parallel hyperplanes producing a lattice structure. (The maximal distance between adjacent hyperplanes is a convenient measure of the quality of the generator and its determination is the goal of the so-called spectral test [19]. When the distance between hyperplanes is small, the illusion that points are uniformly distributed in the hypercube is reinforced. This criterion is thus frequently employed to find the best multiplier and modulus for a conventional multiplicative linear congruential operator.) The clusterization phenomenon turns out to be apparent in CA rules without memory, as shown in Figure 5, and it is detected to some extent when memory is present, as shown in Figure 6. This may be the origin of the weakness of the studied rules regarding randomness.

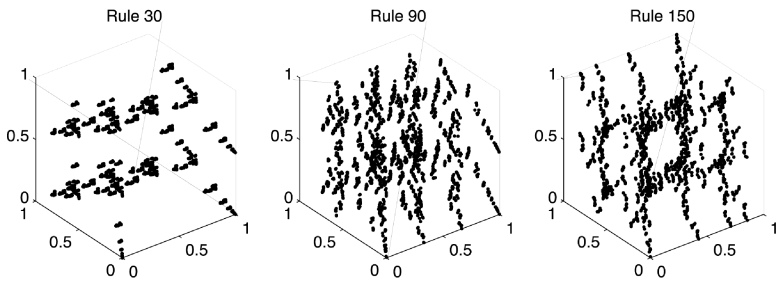


Figure 5. Grids of triplets of successive numbers in the simulation of Figure 3.

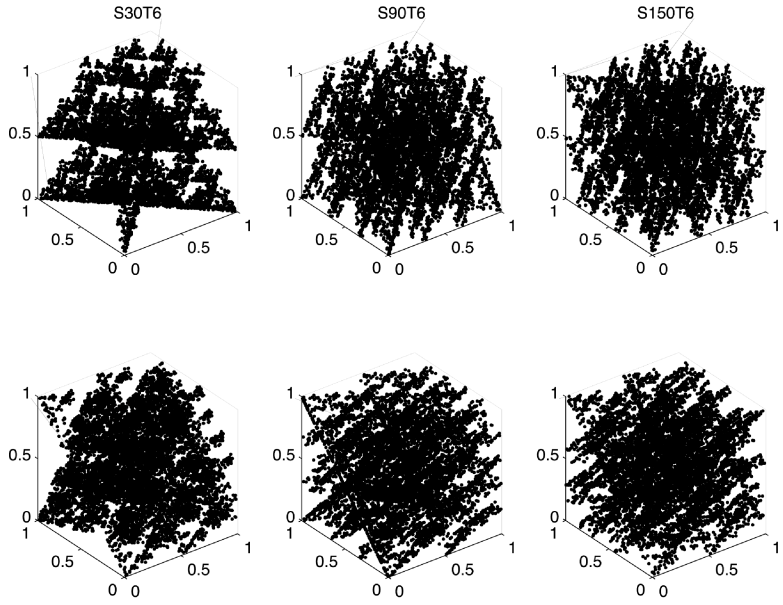


Figure 6. Grids of triplets of successive numbers in the simulation of Figure 4. Two different perspectives of every dataset are shown. $N = 50$.

This does not happen when keeping parity memory of the three last states as reported in [20], or moreover of the last four states as shown in Figure 7, in which case the space seems conveniently filled with scattered points. Table 2 shows how the probability parameters in this parity of the last four states' memory scenario turn out to be close to 0.5, that is, a genuine random sequence.

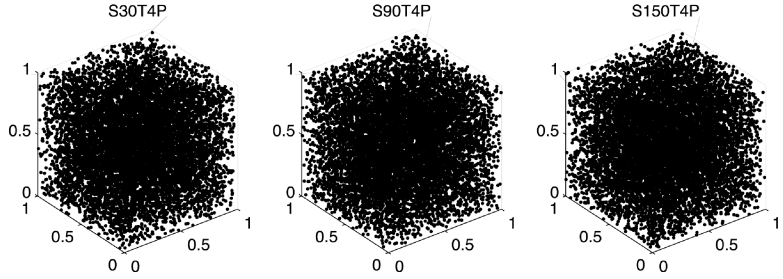


Figure 7. Grids of triplets of successive numbers in a simulation up to $T = 10\,000$, using rules with memory of the parity of the last four state values. $N = 50$.

	S30T4P				S150T4P				S150T4P			
	Run	Pair	Trip	Gap	Run	Pair	Trip	Gap	Run	Pair	Trip	Gap
$\overline{P}$	.493	.552	.553	.452	.433	.424	.468	.465	.518	.507	.514	.512
sd	.290	.268	.294	.307	.320	.284	.280	.281	.294	.269	.279	.304

Table 2. Probability parameters in the scenario of Table 1, but using rules with memory of the parity of the last four state values.

3.4 An Alternative Minimal Memory Mechanism

An alternative mechanism that only demands an additional bit of memory per cell is that of keeping unlimited track of the sum of previous state values, $s_i^{(T)} = \sigma_i^{(1)} \oplus \cdots \oplus \sigma_i^{(T-1)} \oplus \sigma_i^{(T)}$, as $s_i^{(T)} = s_i^{(T-1)} \oplus \sigma_i^{(T)}$.

Appendix C shows the effect of such a minimal memory mechanism on some elementary rules, in a register of size 150 and up to $T = 60$. Unlimited trailing parity memory has been coded as UP in the rule name codification.

In the case of linear rules it holds that,

$$\begin{aligned} \mathbf{C}^{(T+1)} &= \mathbf{M}\left(\mathbf{C}^{(T)} \oplus \mathbf{C}^{(T-1)} \oplus \cdots \oplus C^{(1)}\right) = \\ &\mathbf{M}\,\mathbf{C}^{(T)} \oplus \mathbf{C}^{(T)} = (\mathbf{M} \oplus \mathbf{I})\,\mathbf{C}^{(T)}. \end{aligned} \tag{1}$$

Thus, after $T = 2$, rule S150TUP evolves as rule 90 and rule S90TUP evolves as rule 150. This can be checked in Figure 8, in which the evolution from $T = 3$ is that of rule 90.

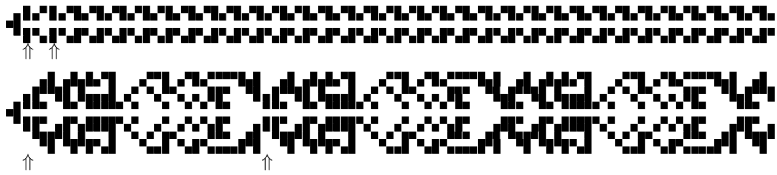


Figure 8. The rule S150TUP in circular registers of sizes $N = 5$ and $N = 11$.

4. Conclusion

The dynamics of elementary rules is dramatically altered when endowing cells with memory of the last two time steps, compared to the conventional cellular automata (CA) paradigm that merely takes into account the last configuration. Particularly interesting is the effect of the parity rule acting as memory on rule 30 and on the linear rules 90 and 150, as it generates a seemingly random dynamic, albeit failing in most of the randomness tests.

CA with memory in cells can be considered as a natural and promising extension of the basic paradigm. A major impediment to modeling with CA stems from the difficulty of utilizing their complex behavior to exhibit a particular behavior or perform a particular function: embedding memory in cells broadens the spectrum of CA as a tool for modeling. It is likely that in some contexts, a transition rule with memory could match the "correct" behavior of the CA system of a given complex system.

Apart from their potential applications, CA with memory are of aesthetic and mathematical interest. The study of the effect of memory on CA has been rather neglected. Nevertheless, it seems plausible that further study of the effect of memory on CA (and in lattice gas automata and in other generalized CA scenarios such as structurally dynamic CA, in which case memory may also be embedded in links, as explored in [23]) should turn out to be profitable.

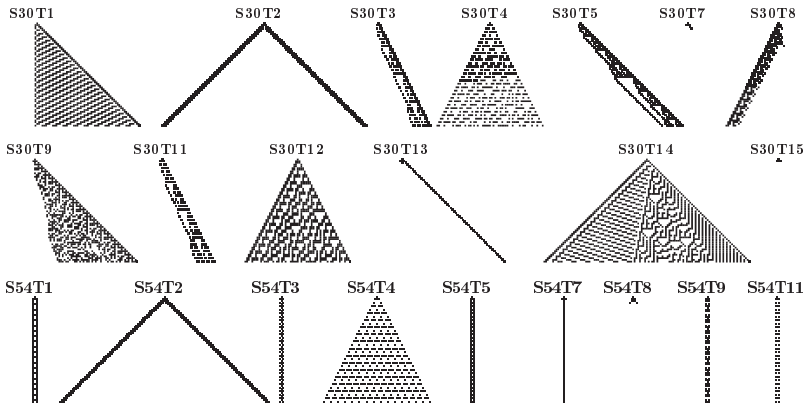
Perhaps, as a result of a further full rigorous study of CA with memory, it will be possible to paraphrase Toffoli [24] in presenting CA with memory, as an alternative to (rather than an approximation of) integral equations in modeling, in particular, to Volterra integral equations that appear in the study of many phenomena incorporating memory, which are important in applied sciences, such as population dynamics, diffusion, neural networks, and so on.

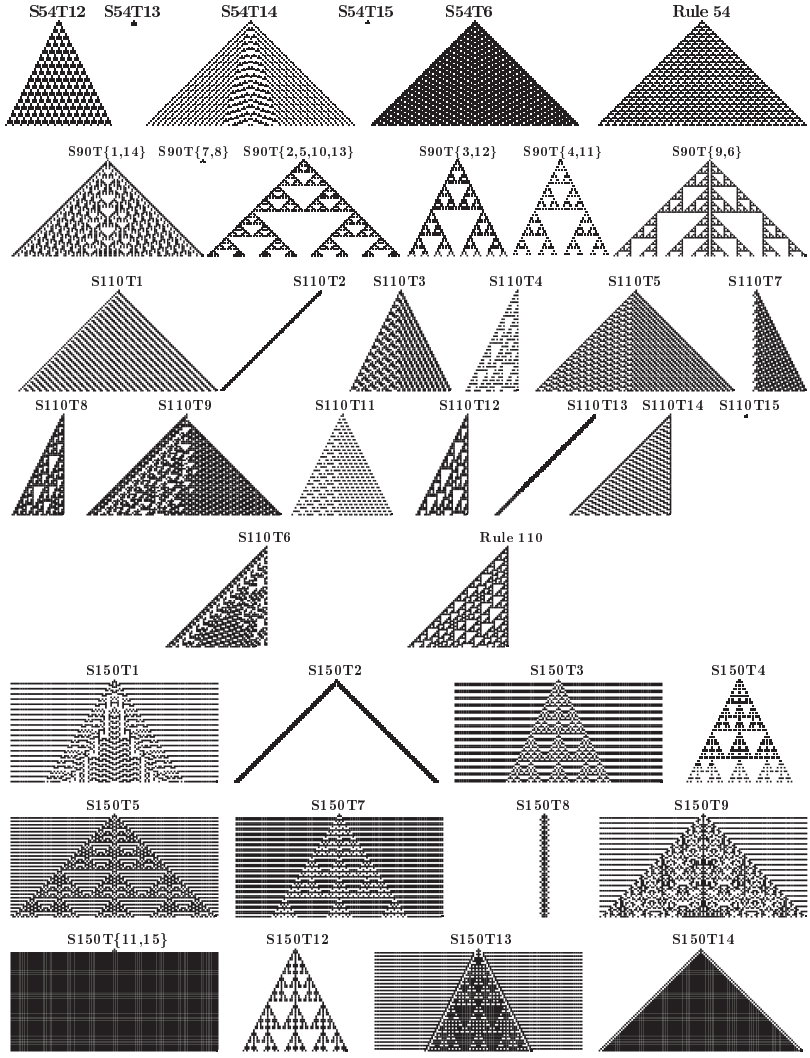
Acknowledgment

This work was supported by EPRSC Project EP/E049281/1.

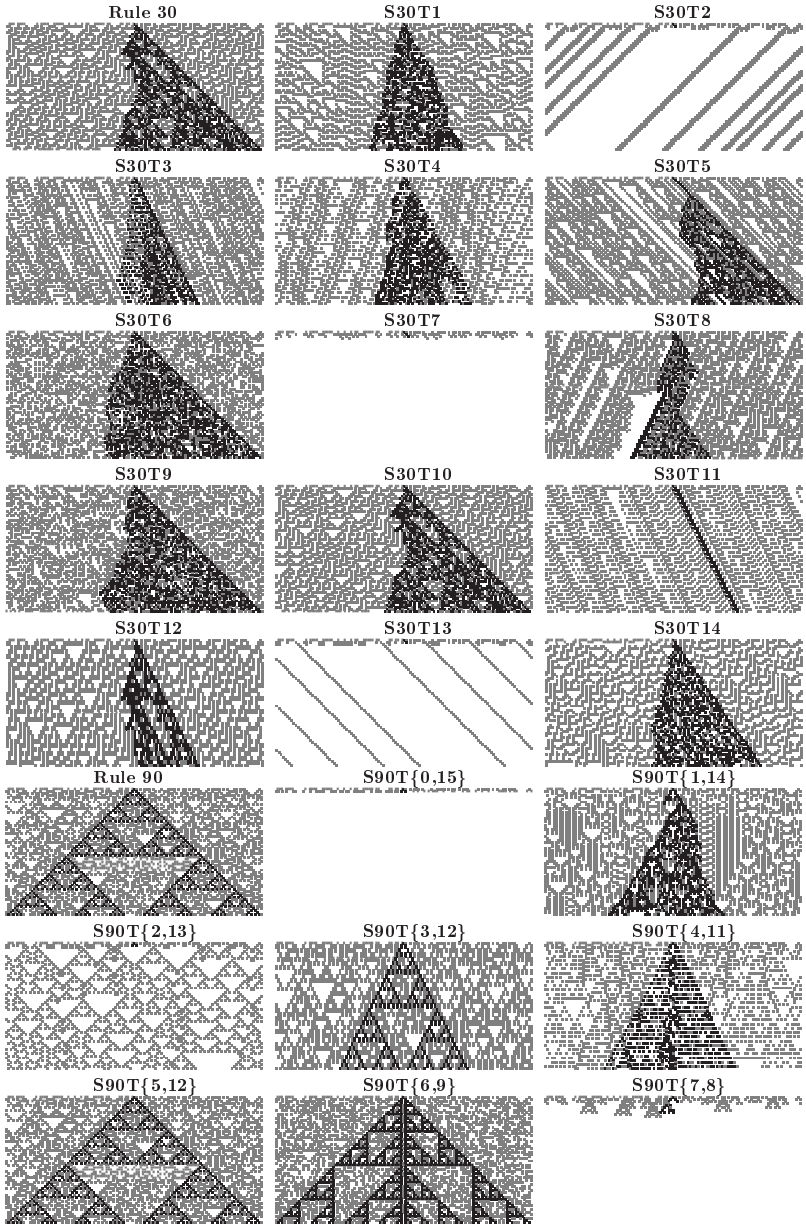
Appendix

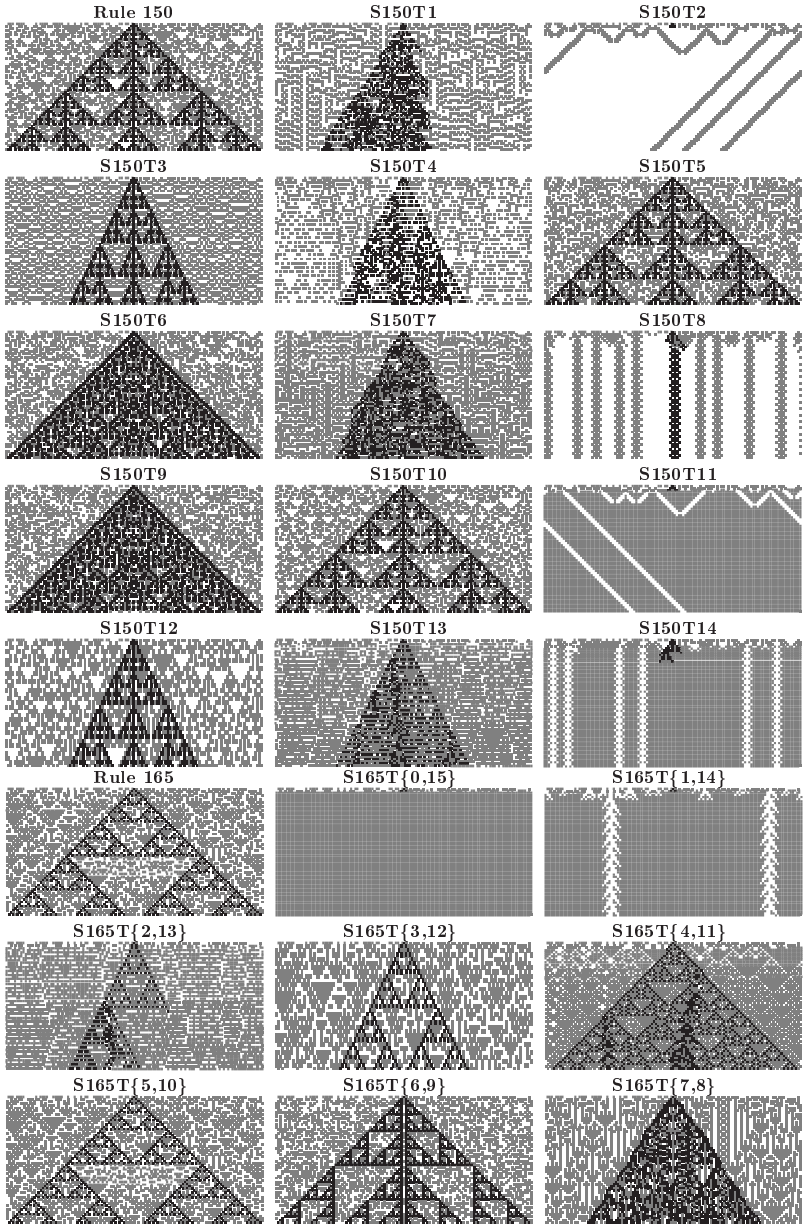
A. Elementary Rules from a Single Active Cell



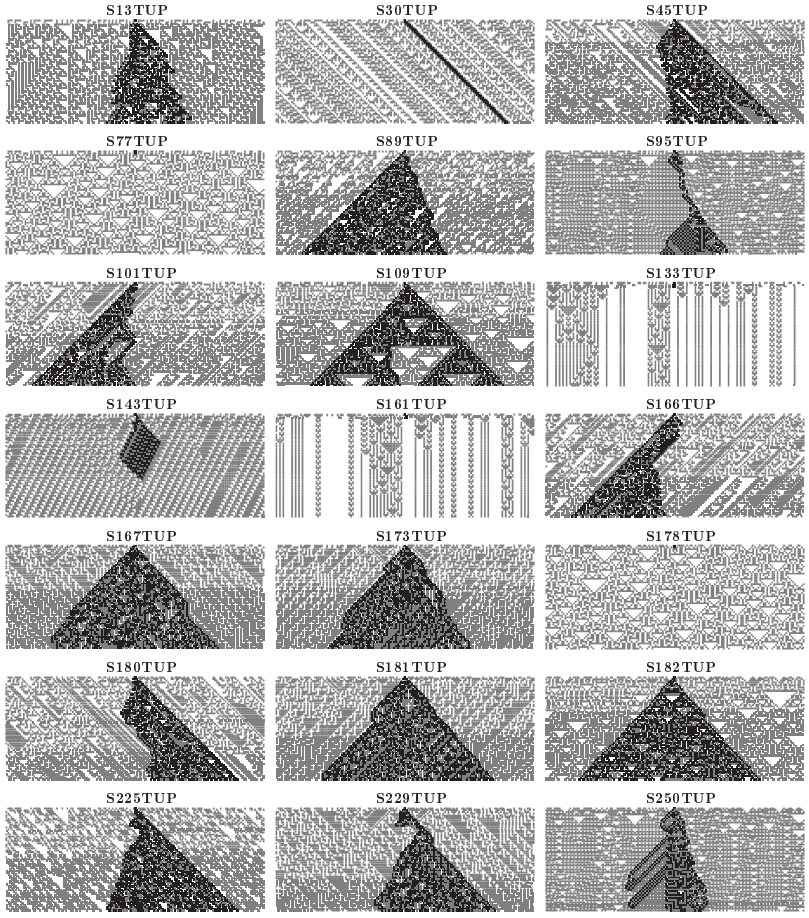

S54T12
S54T13
S54T14
S54T15
S54T6
Rule 54
S90T{1,14}
S90T{7,8}
S90T{2,5,10,13}
S90T{3,12}
S90T{4,11}
S90T{9,6}
S110T1
S110T2
S110T3
S110T4
S110T5
S110T7
S110T8
S110T9
S110T11
S110T12
S110T13
S110T14
S110T15
S110T6
Rule 110
S150T1
S150T2
S150T3
S150T4
S150T5
S150T7
S150T8
S150T9
S150T{11,15}
S150T12
S150T13
S150T14

B. Elementary Rules Starting at Random




Rule 150
S150T1
S150T2
S150T3
S150T4
S150T5
S150T6
S150T7
S150T8
S150T9
S150T10
S150T11
S150T12
S150T13
S150T14
Rule 165
S165T{0,15}
S165T{1,14}
S165T{2,13}
S165T{3,12}
S165T{4,11}
S165T{5,10}
S165T{6,9}
S165T{7,8}

C. Unlimited Parity Memory



References

[1] S. Wolfram, *A New Kind of Science*, Champaign, IL: Wolfram Media, Inc., 2002.

[2] R. Alonso-Sanz and M. Martin, "One-Dimensional Cellular Automata with Memory in Cells of the Most Frequent Recent Value," *Complex Systems*, **15**(3), 2006 pp. 203-236.

[3] R. Alonso-Sanz, "One-Dimensional, $r = 2$ Cellular Automata with Memory," *International Journal of Bifurcation and Chaos (IJBC)*, **14**(9), 2004 pp. 3217-3248. doi.10.1142/S0218127404011338.

[4] R. Alonso-Sanz and M. Martin, "Three-State One-Dimensional Cellular Automata with Memory," *Chaos, Solitons and Fractals*, **21**(4), 2004 pp. 809-834. doi.10.1016/j.chaos.2003.12.083.

[5] R. Alonso-Sanz, "Reversible Cellular Automata with Memory: Two-Dimensional Patterns from a Single Site Seed," *Physica D: Nonlinear Phenomena*, **175**(1-2), 2003 pp. 1-30. doi.10.1016/S0167-2789(02)00693-0.

[6] R. Alonso-Sanz and M. Martin, "Elementary Cellular Automata with Memory," *Complex Systems*, **14**(2), 2003 pp. 99-126.

[7] R. Alonso-Sanz and M. Martin, "Two-Dimensional Cellular Automata with Memory: Patterns Starting with a Single Site Seed," *International Journal of Modern Physics C (IJMPC)*, **13**(1), 2002 pp. 49-65 and the references therein. doi.10.1142/S0129183102002973.

[8] R. Alonso-Sanz and M. Martin, "Elementary Cellular Automata with Elementary Memory Rules in Cells: The Case of Linear Rules," *Journal of Cellular Automata*, **1**, 2006 pp. 71-87.

[9] O. Martin, A. M. Odlyzko, and S. Wolfram, "Algebraic Properties of Cellular Automata," *Communications in M athematical Physics*, **93**(2), 1984 pp. 219-258. www.projecteuclid.org/euclid.cmp/1103941055.

[10] T. Checherine-Silberstein and M. Coornaert, "The Garden of Eden Theorem for Linear Cellular Automata," *Ergodic Theory and Dynamical Systems*, **26**, 2006 pp. 53-68. doi.10.1017/S0143385705000520.

[11] S. K. Park and K. W. Miller, "Random Number Generators: Good Ones Are Hard to Find," *Communications of the ACM*, **31**(10), 1988 pp. 1192-1201. doi.acm.org/10.1145/63039.63042.

[12] M. Tomassini, M. Sipper, and M. Perrenoud, "On the Generation of High-Quality Random Numbers by Two-Dimensional Cellular Automata," *IEEE Transactions on Computers*, **49**(10), 2000 pp. 1146-1151. doi.10.1109/12.888056.

[13] M. Sipper and M. Tomassini, "Generating Parallel Random Number Generators by Cellular Programming," *International Journal of Modern Physics C (IJMPC)*, **7**(2), 1996 pp. 181-190.

[14] S. U. Guan and S. K. Tan, "Pseudorandom Number Generation with Self-Programmable Cellular Automata," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **23**(7), 2003 pp. 1095-1101.

[15] P. D. Hortensius, R. D. McLeod, and H. C. Card, "Parallel Random Number Generation for VLSI Systems using Cellular Automata," *IEEE Transactions on Computers*, **38**(10), 1989 pp. 1466-1472. dx.doi.10.1109/12.35843.

[16] P. D. Hortensius, R. D. McLeod, W. Pries, D. M. Miller, and H. C. Card, "Cellular Automata-Based Pseudorandom Number Generators for Built-In Self-Test," *IEEE Transactions on Computer-Aided Design*, **8**(8), 1989 pp. 842-859.

[17] P. Tsalides, T. A. York, and A. Thanailakis, "Pseudorandom Number Generators for VLSI Systems Based on Linear Cellular Automata," *IEE Proceedings on Computers and Digital Technologies*, **138**(4), 1991 pp. 241-249.

[18] S. Wolfram, "Random Sequence Generation by Cellular Automata," *Advances in Applied Mathematics*, **7**(2), 1986 pp. 123-169. dx.doi.10.1016/0196-8858(86)90028-X.

[19] D. E. Knuth, *The Art of Computer Programming (TAOCP),* Vol. 2 (Seminumerical Algorithms), 2nd ed., Reading, MA: Addison-Wesley, 1998.

[20] R. Alonso-Sanz and L. Bull, "Random Number Generation by Cellular Automata with Memory," *International Journal of Modern Physics C (IJMPC)*, **19**(2), 2008 pp. 351-367. doi .10.1142/S012918310801211X.

[21] S. Savic, "Testing Two-Neighbor Cellular Automata for Randomness," NKS Summer School, 2004.

[22] G. Marsaglia, "Random Numbers Fall Mainly in the Planes," *Proceedings of the National Academy of Sciences*, **61**(1), 1968 pp. 25-28.

[23] R. Alonso-Sanz, "A Structurally Dynamic Cellular Automaton with Memory," *Chaos, Solitons & Fractals*, **32**(4), 2006 pp. 1285-1295. doi.10.1016/j.chaos.2005.12.047.

[24] T. Toffoli, "Cellular Automata As an Alternative to (Rather than an Approximation of) Differential Equations in Modeling Physics," *Physica D: Nonlinear Phenomena*, **10**(1-2), 1984 pp. 117-127. doi.10.1016/0167-2789(84)90254-9.