Complex Dynamics Emerging in Rule 30 with Majority Memory

Genaro J. Martínez* [1,2]
Andrew Adamatzky [1]
Ramon Alonso-Sanz [1]

[1] *Department of Computer Science*
University of the West of England
Bristol BS16 1QY, United Kingdom
[2] *Instituto de Ciencias Nucleares and Centro de Ciencias de la Complejidad*
Universidad Nacional Autónoma de México
**genaro.martinez@uwe.ac.uk*

Juan C. Seck-Tuoh-Mora

Centro de Investigación Avanzada en Ingeniería Industrial
Universidad Autónoma del Estado de Hidalgo Pachuca
Hidalgo, México

In cellular automata (CAs) with memory, the unchanged maps of conventional CAs are applied to cells endowed with memory of their past states in some specified interval. We implement the rule 30 automaton and show that by using the majority memory function we can transform the quasi-chaotic dynamics of classical rule 30 into domains of traveling structures with predictable behavior. We analyze morphological complexity of the automata and classify glider dynamics (particle, self-localizations) in the memory-enriched rule 30. Formal ways of encoding and classifying glider dynamics using de Bruijn diagrams, soliton reactions, and quasi-chemical representations are provided.

1. Introduction

An elementary cellular automaton (CA) is a one-dimensional array of finite automata, where each automaton takes two states and updates its state in discrete time depending on its own state and the states of its two closest neighbors. All cells update their state synchronously. The following general classification of elementary CAs was introduced in [1].

Class I. CAs evolve to a homogeneous state.

Class II. CAs that evolve periodically.

Class III. CAs that evolve chaotically.

Class IV. Include all previous cases, also known as the class of complex rules.

https://doi.org/10.25088/ComplexSystems.18.3.345

Class IV is of particular interest because the rules exhibit nontrivial behavior with rich and diverse patterns, as shown for rule 54 in [2, 3].

2. Basic Notation

2.1 One-Dimensional Cellular Automata

One-dimensional CAs are represented by an infinite array of *cells* x_i where $i \in \mathbb{Z}$ and each x takes a value from a finite alphabet Σ. Thus, a sequence of cells $\{x_i\}$ of finite length n represents a string or *global configuration* c on Σ with the set of finite configurations represented as Σ^n. An evolution is represented by a sequence of configurations $\{c_i\}$ given by the mapping $\Phi : \Sigma^n \to \Sigma^n$; thus their global relation is provided as

$$\Phi\left(c^t\right) \to c^{t+1} \tag{1}$$

where t is time and every global state of c is defined by a sequence of cell states. Also, the cell states in configuration c^t are updated at the next configuration c^{t+1} simultaneously by a local function φ:

$$\varphi\left(x_{i-r}^t, \ldots, x_i^t, \ldots, x_{i+r}^t\right) \to x_i^{t+1}. \tag{2}$$

Wolfram represents a one-dimensional CA with two parameters (k, r) where $k = |\Sigma|$ is the number of states, and r is the neighborhood radius. Elementary CAs are defined by parameters $(k = 2, r = 1)$. There are Σ^n different neighborhoods (where $n = 2r + 1$) and k^{k^n} different evolution rules.

We used automata with periodic boundary conditions in our computer experiments.

2.2 Cellular Automata with Memory

Conventional CAs are ahistoric (memoryless): that is, the new state of a cell depends on the neighborhood configuration solely at the preceding time step of φ as in equation (2).

Cellular automata with memory consider an extension to the standard CA framework by implementing memory capabilities in cells x_i from its own history.

Thus, to implement memory we incorporate a memory function ϕ,

$$\phi\left(x_i^{t-\tau}, \ldots, x_i^{t-1}, x_i^t\right) \to s_i \tag{3}$$

such that $\tau < t$ determines the degree of memory backward and each cell $s_i \in \Sigma$ is a state function of the series of states of the cell x_i with memory up to the current time step. Finally, to execute the evolution we apply the original rule as:

$$\varphi\left(\ldots, s_{i-1}^t, s_i^t, s_{i+1}^t, \ldots\right) \to x_i^{t+1}.$$

https://doi.org/10.25088/ComplexSystems.18.3.345

Thus, in CAs with memory, while the mappings φ remain unaltered, historic memory of all past iterations is retained by featuring each cell with a summary of its past states from ϕ. Therefore, cells *canalize* memory to the map φ.

As an example, we define the *majority memory* as

$$\phi_{\text{maj}} \rightarrow s_i \tag{4}$$

where, in case of a tie given by $\Sigma_1 = \Sigma_0$ from ϕ, we will take the last value x_i. So the ϕ_{maj} function represents the classic majority function [4] on the cells $\left(x_i^{t-\tau}, \ldots, x_i^{t-1}, x_i^t\right)$ and defines a temporal ring before finally getting the next global configuration c.

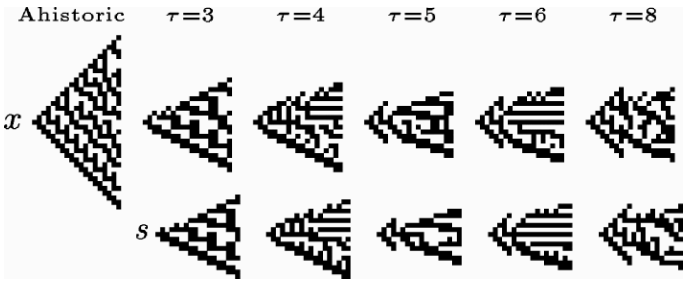


Figure 1. The effect of majority memory with increasing depths on rule 30 starting from a single site live cell.

Majority memory exerts a general inertial effect [5]. This effect, when starting from a single site live cell, notably restrains the dynamics, as illustrated using rule 30 in Figure 1. This figure shows the spatio-temporal patterns of both the current x state values and that of the underlying s values.

3. Elementary Cellular Automaton Rule 30

Rule 30 was initially studied by Wolfram in [1] because of its chaotic global behavior while looking for a random number generator. Rule30 is an elementary CA that evolves in one dimension of order (2, 1). An interesting property is that it has a surjective relation and thus does not have Garden of Eden configurations [6]. In this way, any configuration always has at least one predecessor.

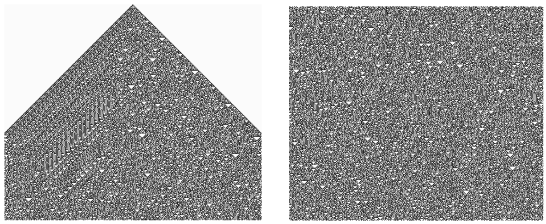
Here is the local rule φ corresponding to rule 30:

$$\varphi_{\text{R30}} = \begin{cases} 1 & \text{if } 100, 011, 010, 001 \\ 0 & \text{if } 111, 110, 101, 000 \end{cases}.$$

https://doi.org/10.25088/ComplexSystems.18.3.345

Generally speaking, rule 30 displays a typical chaotic global behavior, that is, it is in Wolfram's Class III. An interesting study on rule 30 showing a local nested structure that repeats periodically while looking for invertible properties is given in [7].

So, initially φ_{R30} has a 50% probability of states zero or one, and consequently each state appears with the same frequency.



(a) (b)

Figure 2. (a) Typical behavior of rule 30, where a single cell in state one leads to a chaotic state. (b) Shows the automaton behavior from a random initial condition with an initial density of 50% for each state. Both automata evolved on a ring of 497 cells (with a periodic boundary property) to 417 generations. White cells represent state zero and dark cells the state one.

Also, the evolution of rule 30 presents the following feature: if an initial configuration is covered all in state one, then it always evolves into one; but if this is empty or filled with state one then this always evolves to state zero. Figure 2 shows two typical cases of evolution with rule 30.

3.1 De Bruijn and Subset Diagrams in Rule 30

Given a finite sequence $w \in \Sigma^m$, such that $w = w_1, \ldots, w_m$, let $\alpha(w) = w_1$, $\beta(w) = w_2, \ldots, w_m$, and $\psi(w) = w_1, \ldots, w_{m-1}$. With these elements, we can specify a labeled directed graph known as a *de Bruijn diagram* $\mathcal{B} = \{N; E\}$ associated with the evolution rule of the CA. The nodes of $\mathcal{B}$ are defined by $N = \Sigma^{2r}$ and the set of directed edges $E \subseteq \Sigma^{2r} \times \Sigma^{2r}$ is defined as

$$E = \{(v, w) \mid v, w \in N,\ \beta(v) = \psi(w)\}. \tag{5}$$

For every directed edge $(v, w) \in E$, let $\eta(v, w) = a\, w \in \Sigma^{2r+1}$ where $a = \alpha(v)$; that is, $\eta(v, w)$ is a neighborhood of the automaton. In this way, the edge (v, w) is *labeled* by $\varphi \circ \eta(v, w)$; hence, every labeled path in $\mathcal{B}$ represents the evolution of the corresponding sequence specified by its nodes. Since each $w \in N$ can be described by a number base k of length $2r$, every node in $\mathcal{B}$ can be enumerated by a unique element in $\mathbb{Z}_{k^{2r}}$, which is useful for simplifying the diagram.

https://doi.org/10.25088/ComplexSystems.18.3.345

The de Bruijn diagram associated with rule 30 is depicted in Figure 3, where black edges indicate the neighborhoods evolving into zero and those evolving into one are shown by gray edges. The de Bruijn and subset diagrams were calculated using NXLCAU21 designed by McIntosh. The program is available from delta.cs.cinvestav.mx/~mcintosh.

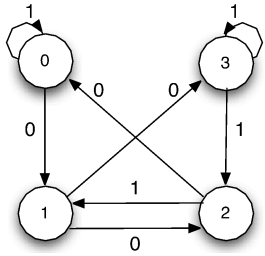


Figure 3. De Bruijn diagram for the elementary CA rule 30.

Figure 3 shows that there are four neighborhoods evolving into zero and four into one, meaning that each state has the same probability to appear during the evolution. This indicates the possibility that the automaton is surjective, that is, there are no Garden of Eden configurations. Classical analysis in graph theory has been applied over de Bruijn diagrams for studying topics such as reversibility [8]; cycles in the diagram indicate periodic elements in the evolution of the automaton if the label of the cycle corresponds to the sequence defined by its nodes, in periodic boundary conditions. The cycles in the de Bruijn diagram from Figure 3 are presented in Figure 4.

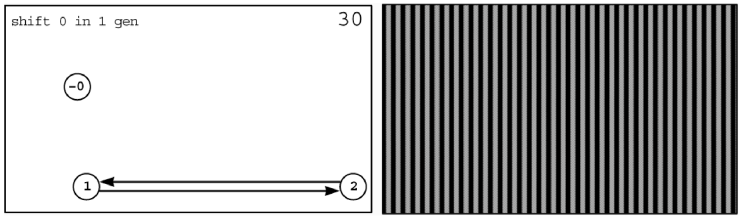


Figure 4. Cycles in the de Bruijn diagram and the corresponding periodic evolution for cycle (1, 2).

The largest cycle in Figure 4 indicates that the undefined repetition of sequence $w_b = 10$ establishes a periodic structure without displacement in one generation during the evolution of rule 30. We then say that w_b is the *filter* in rule 30. A filter is a periodic sequence that exists alone or in blocks during the evolution; thus, suppressing such a string produces a new view. In the present paper, we apply the filter to the original rule 30 and its modifications with memory. Thus, we

https://doi.org/10.25088/ComplexSystems.18.3.345

can see how a de Bruijn diagram can recognize any periodic structures in a CA [3, 9].

A de Bruijn diagram is nondeterministic in the sense that a given node may have several output edges with the same label. A classical approach to analyzing the diagram would be to construct the subset (or power) diagram in order to obtain a deterministic version for the de Bruijn diagram in the evolution rule [10, 11].

The subset diagram is defined as $\mathcal{S} = \{\mathcal{P}, Q\}$ where $\mathcal{P} = \{P \mid P \subseteq \Sigma \bigcup \emptyset\}$ is the set of nodes of $\mathcal{S}$ and the directed edges are defined by $Q \subset \mathcal{P} \times \mathcal{P}$ where for $P_1, P_2 \in \mathcal{P}$ there is a directed edge (P_1, P_2) labeled by $a \in S$ in $\mathcal{S}$ if and only if P_2 is the maximum subset such that for every $c \in P_2$ there exists $b \in P_1$ such that $\varphi(b, c) = a$.

The inclusion of the empty set assures that every edge has a well-defined ending node. For a CA with k states, it is fulfilled that $|\mathcal{P}| = 2^{k^{2r}}$, which implies an exponential growth in the number of nodes in $\mathcal{S}$ when more states are considered. Every $P \in \mathcal{P}$ can be identified by a binary number showing the states belonging to this subset, that is, taking the states as an ordered list. The states in P can be signed by a 1 and the others by 0, making a unique binary sequence to identify the subset. The decimal value of this binary number can be taken to get a shorter representation where the empty set has a decimal number 0 and the full subset $p = \Sigma$ has the number $2^{k^{2r}} - 1$. The subset diagram corresponding to rule 30 is shown in Figure 5.

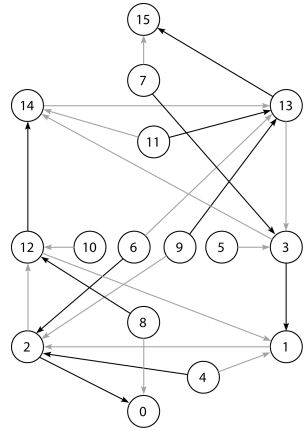


Figure 5. Subset diagram for rule 30.

In Figure 5, the subset diagram has no path starting from the full subset (node 15) going to the empty subset (node 0). This means that every sequence can be produced by the evolution of the automaton and there are no Garden of Eden sequences. Thus, the automaton is surjective. The subset diagram can also be used as a deterministic

https://doi.org/10.25088/ComplexSystems.18.3.345

automaton for calculating ancestors of any desired sequence [12] by recognizing the regular expressions that may be generated by the corresponding automaton. Some of these expressions would be able to represent interesting structures as gliders [13]; however, more effort is needed in order to get a straightforward detection of such constructions in the diagram.

Finally, such diagrams help get periodic strings that eventually represent a general filter w_b working on the original rule 30 and rule 30 with memory. Also, we will take advantage of these results to find gliders in the strings.

4. Majority Memory Helps to Discover Complex Dynamics in Rule 30

This section reports on how the majority memory ϕ helps in the discovery of complex dynamics in elementary CAs by experimentation. For an introduction to elementary CAs with memory, see [14-16].

Figure 6 displays different scenarios where the majority memory ϕ_{maj} works on rule 30 to extract the complex dynamics. The evolutions should be read from left to right and up to down. All of these evolutions use the same random initial density and filter w_b (including the original rule 30). Thus, the first evolution shown in Figure 6 is the original rule 30, that is, without majority memory. In the original evolution we can see gaps that the filter can clean. Traditionally, it was difficult to distinguish such a filter, but when ϕ_{maj} was applied to rule 30 its presence was more evident. A general technique for getting filters was developed by Wuensche in [17].

Initially, even values of τ seem to extract gliders more quickly and odd values fight to reach an order. Eventually, the majority memory will converge to one stability in Φ while τ increases.

The first snapshot calculating ϕ_{maj} with $\tau = 3$ is shown by the second evolution in Figure 6. It is not yet clear how memory induces another behavior because the global behavior is still similar to the original with only small changes.

On the other hand, the third evolution with $\tau = 4$ does extract periodic patterns. The evolution might not display impressive gliders but it already allows picking out more diversity in mobile localizations on lattices of 100×100 cells. Thus, we have enumerated and ordered values of τ from Figure 6 based on the space-time dynamics they are responsible for:

Chaotic global behavior: $\tau = 0, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21$

Periodic patterns: $\tau = 4, 6, 8, 10, 12, 14, 16, 18, 19, 20, 21$

Collision patterns: $\tau = 6, 8, 10, 12$

https://doi.org/10.25088/ComplexSystems.18.3.345

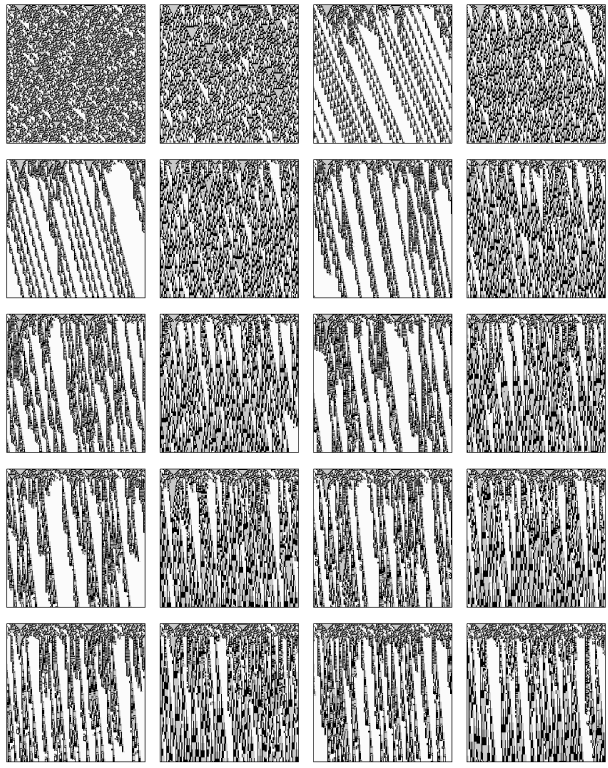


Figure 6. Complex dynamics emerging in rule 30 with majority memory ϕ_{maj} from a range of values from $\tau = 3$ to $\tau = 21$. The first evolution shows the original function. Evolutions were calculated on a ring of 104 cells in 104 generations with a random initial density of 50% and the same initial conditions were used in all cases. Also, the filter w_b was applied to clearly show the structures.

4.1 Morphological Complexity in Rule 30 with Memory

In this section we explore some techniques for finding global complex dynamics in rule 30 with and without majority memory.

We evaluate the morphological complexity of a CA using the morphological richness approach in [18]. We calculate the statistical morphological richness μ as follows. Given the space-time configuration of a one-dimensional CA, we extract the 3×3 cell neighborhood state for each site of the configuration and build a distribution of the neighborhood states over an extended period of the automaton's development time.

Examples of morphological richness μ are shown in Figure 7. A control case, where the next state of a cell is calculated at random from the distribution of space-time neighborhood states, is uniform (Figure 7(a)). Two-dimensional random configurations are morpho-

https://doi.org/10.25088/ComplexSystems.18.3.345

logically rich. The morphology of memoryless, classical rule 30 is characterized by few peaks in the local domain distributions, where several space-time templates dominate in the global space-time configuration (Figure 7(b)). The statistical morphological richness μ decreases.

Incorporating memory in the cell-state transition rules leads to an erosion of the distribution (Figure 7(c)) and thus slight increases in μ. With an increase in the memory depth, the shape of the morphological distribution changes just slightly, up to minor height variations in the major peaks (Figures 7(d) through (f)).

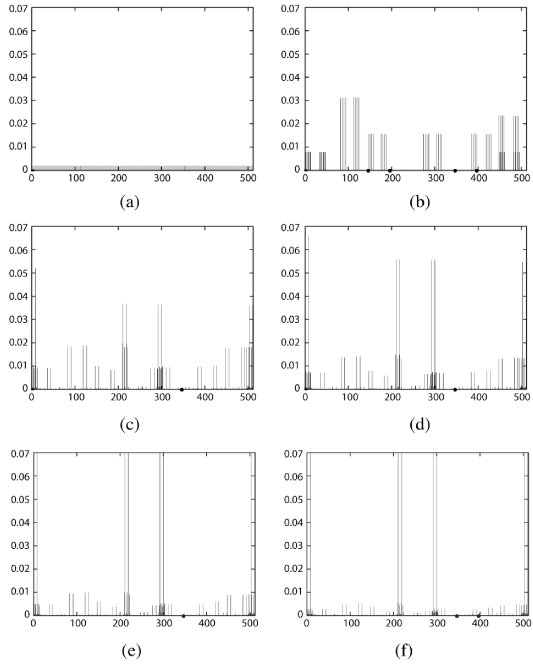


Figure 7. Morphological richness. Cellular automaton length of 1500 cells with a running time of 5000 steps. (a) Random update of cell states. (b) Rule 30 without memory. Rule 30 with memory: (c) $\tau = 3$, (d) $\tau = 5$, (e) $\tau = 10$, and (f) $\tau = 21$.

The number ρ of 3×3 blocks (of states 0 and 1) that never appear in the space-time configuration of a CA can be used to express an estimate of the nominal morphological richness; smaller ρ indicate a richer nominal configuration.

The difference between statistical μ and nominal ρ measures of morphological richness is that μ allows picking most common configurations of local domains, while ρ just shows how many blocks of 3×3 states appeared in the automaton evolution at least once.

https://doi.org/10.25088/ComplexSystems.18.3.345

For the case of randomly updating cell states, all blocks are present in the space-time configuration and $\rho = 0$.

Memoryless automata governed by rule 30 have $\rho = 434$ so the total number of possible blocks is 512. When memory is first incorporated into the cell-state transition function, richness decreases, for example, with $\tau = 1$ we have $\rho = 448$. Then we observe a consistent increase in complexity. Thus, rule 30 with small-depth memory ($\tau = 2$) $\rho = 140$, drastically decreases to $\rho = 68$ for $\tau = 3$. The richness is stabilized, or rather oscillates around ρ values of 20 to 40 with a further increase of memory.

In summary, we found that majority memory increases the nominal complexity of a CA but decreases its statistical complexity.

4.2 Gliders in Rule 30 with Memory $\tau = 8$

Most frequently the complex dynamics of an elementary CA is related to gliders, glider guns, and nontrivial reactions between localizations, for example, rules 110 or 54 [2, 19]. The phenomena, and their regular expressions [3, 9], may lead to the discovery of novel systems with computational universality [20, 21].

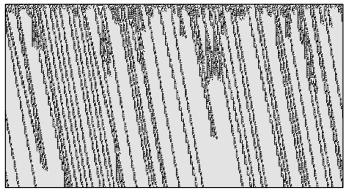


Figure 8. Gliders emerging in rule 30 with ϕ_{maj} and $\tau = 8$. This evolution shows how some kinds of gliders arise and still interact from random initial conditions. The evolution was calculated on a ring of 590 cells to 320 generations, with an initial density of 50%.

Among the sets of complex dynamics in rule 30 determined by τ (shown in Figure 6), we have chosen the memory ϕ_{maj} with $\tau = 8$. In this way, Figure 8 illustrates an ample evolution space of its global dynamics.

Of course, these gliders may not be as impressive as others from such well-known complex rules as 110, 54, or some other one-dimensional rules [1, 2, 19, 22, 23]. However, it is interesting that ϕ_{maj} is able to open complex patterns from chaotic rules.

Nevertheless, even though rule 30 does not offer an ample range of complex dynamics, it is useful for describing gliders and collisions. So, we shall illustrate how a chaotic CA can be decomposed as a complex system.

https://doi.org/10.25088/ComplexSystems.18.3.345

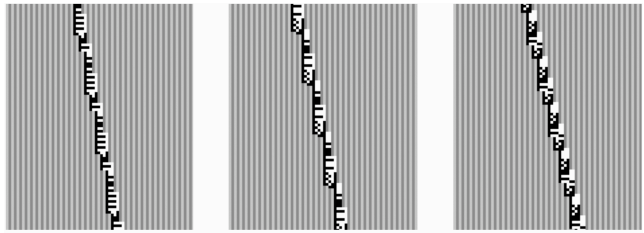


Figure 9. Set of gliders G_{R30m} with memory ϕ_{maj} and $\tau = 8$.

We now classify the family of gliders and enumerate some of their properties. Figure 9 displays the family of gliders $G_{\mathrm{R30m}} = \{g_1, g_2, g_3\}$. As was hoped, an immediate consequence is that gliders in CAs with memory have longer periods.

Structure	v_g	Lineal Volume
w_b	$0 / c = 0$	2
g_1	$2 / 11 \approx 0.1818$	5
g_2	$4 / 19 \approx 0.2105$	7
g_3	$4 / 17 \approx 0.2352$	6

Table 1. Properties of gliders G_{R30m} with memory ϕ_{maj} and $\tau = 8$.

Table 1 summarizes the basic properties of each glider. Practically, all gliders in this domain have a constant displacement of four cells to the right and no glider with speed zero was found, and yet finding a glider gun in this domain is more complicated. Nevertheless, some interesting reactions did originate from G_{R30m}.

Structure w_b does not have a displacement and it is also not a glider. This pattern is the periodic background in rule 30 and represents the filter. It was really hard to detect the existence of a periodic background evolving from the original rule. But when ϕ_{maj} was applied, a periodic pattern began to emerge that was inherited from φ_{R30}. Finally, this filter was confirmed with its respective de Bruijn and cycle diagrams (see Section 3.1).

4.3 Reactions between Gliders from G_{R30m}

We now demonstrate some simple examples of collisions between gliders. Codes for all gliders, necessary to generate the whole set of binary collisions, are presented in the Appendix.

https://doi.org/10.25088/ComplexSystems.18.3.345

Figure 10 shows how a stream of g_1 gliders is deleted from a reaction cycle:

$$g_3 + g_1 \rightarrow g_2 \text{ and } g_2 + g_1 \rightarrow g_3.$$

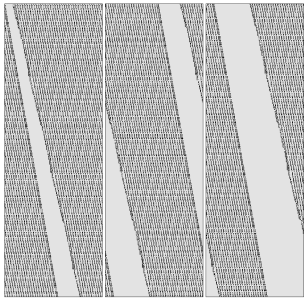


Figure 10. Deleting streams of g_1 gliders initialized with a single g_3 glider.

To obtain such a cycle by glider reactions, we can code an unlimited initial condition as $\ldots g_3 \ldots g_1 \ldots . g_1 \ldots . g_1 \ldots .$, that can be reduced as $\ldots g_3 \ldots g_1 (\ldots . g_1)^*$ (where a dot represents a copy of w_b). Finally, the evolution produces the given cycle, with nine periods of g_3 and eight periods of g_2. Thus, each column presents 1135, 2270, and 3405 generations, respectively. In such a representation, codes of gliders will be different from codes used in our previous papers [3, 9, 19]. The glider reactions were produced using the OSXLCAU21 system available at uncomp.uwe.ac.uk/ genaro/OSXCASystems.html.

Since the better way to preserve φ_{R30} and ϕ_{maj} is to code the gliders as "natural", we consider the codification from its original initial condition. See Appendix B where some strings are defined to get gliders with memory from their original functions.

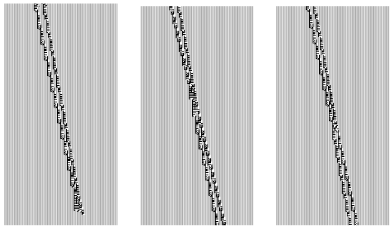


Figure 11. Some simple reactions display how to (a) *delete*, (b) *read*, and (c) *preserve* information with gliders using rule 30 with memory.

https://doi.org/10.25088/ComplexSystems.18.3.345

Some simple but interesting reactions from G_{R30m} are illustrated in Figure 11. The first reaction shows the annihilation of gliders g_2 and g_1. The second reaction shows how a transformation g_3 glider transforms a g_1 glider into a g_2. The third reaction shows a soliton-like collision between gliders g_2 and g_1. The soliton reaction between gliders is particularly promising because it can be used to implement computation, for example, as in the carry-ripple adder embedded by phase coding solitons in parity CAs [24, 25].

4.4 Quasi-Chemistry of Gliders

Assuming gliders g_1, g_2, and g_3 are chemical species a, b, and c in a well-stirred chemical reactor, we can derive the following set of quasi-chemical reactions from the interactions between them:

$$\begin{aligned}
a + b &\overset{0.6}{\rightarrow} \epsilon \\
a + b &\overset{0.2}{\rightarrow} 2\,a \\
a + b &\overset{0.2}{\rightarrow} c \\
a + c &\overset{0.44}{\rightarrow} b \\
a + c &\overset{0.12}{\rightarrow} b + c \\
a + c &\overset{0.22}{\rightarrow} 3\,a \\
a + c &\overset{0.22}{\rightarrow} a \\
b + c &\overset{0.5}{\rightarrow} b \\
b + c &\overset{0.5}{\rightarrow} 2\,a + b
\end{aligned} \tag{6}$$

where reaction rates are evaluated from the frequencies of the interactions.

We evaluated global dynamics of the quasi-chemical system (6) with constant volume (reflecting the finite size of an automaton lattice), constant temperature, and variable pressure using *Chemical Kinetics Simulator* (available at www.almaden.ibm.com/st/computation_science/ck/?cks). Figure 12 shows the temporal dynamics of species concentrations in the system with 10^7 molecules.

When all three species are present in equal concentrations at the beginning (Figure 12(a)), we observe an exponential decay of species b and c and a stabilization of the concentration of species a. When only species b and c are initially present in a well-stirred reaction, species c is produced by their reactions. This leads to an outburst in species a concentration (Figure 12(b)) on the background of an expo-

https://doi.org/10.25088/ComplexSystems.18.3.345

nential decline of species c and b, until species b and c disappear and the concentration of species a becomes constant.

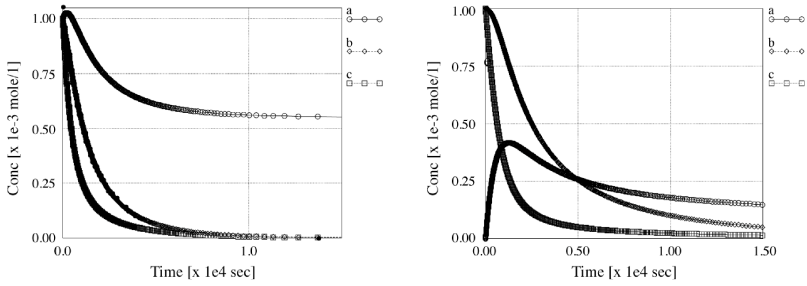


Figure 12. Dynamics of concentrations of species a (circle), b (diamond), and c (square) governed by the reactions in equation (6). (a) Initial concentrations of all species are 0.001 mole/l. (b) Initial concentrations of species b and c are 0.001 mole/l, species a is nil.

4.5 Glider Machines

Table 2 shows the interactions found between gliders, depending on distance σ between the interacting gliders.

$\sigma = 3$	$\sigma = 4$	$\sigma = 5$	$\sigma = 6$
$b + a \to \{a, b\}$	$b + a \to \{\emptyset\}$	$b + a \to \{\emptyset\}$	$b + a \to a$
$c + a \to b$	$c + a \to b$	$c + a \to \{b, c\}$	$c + a \to \{b, c\}$
$c + b \to b$	$c + b \to b$	$c + b \to \{a, b\}$	$c + b \to \{a, b\}$

Table 2. Glider interactions.

Taking into account the gliders' velocities from Table 1, we can construct the following finite state indeterministic machine with an internal state h and input state p, h, $p \in \{a, b, c, \emptyset\}$. The machine can be characterized by an input-output transition matrix $M = \left(m_{ij}\right)_{i,j\in\{a,b,c,\emptyset\}}$, where for $j = h^t$, $i = p^t$, $m_{ij} = h^{t+1}$. The matrix has the following form:

$$M = \begin{array}{c|cccc} h^{t+1} & a & b & c & \emptyset \\ \hline a & a & \{a, \emptyset\} & b & a \\ b & b & b & \{a, b\} & b \\ c & c & c & c & c \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \end{array}$$

https://doi.org/10.25088/ComplexSystems.18.3.345

Starting at a randomly chosen initial state and subjected to random uniformly distributed input strings, the machine will end in the state c with probability $\frac{1}{4}$ and in the state set with probability $\frac{3}{4}$. The machine starting in the initial h generates the string $l(h)$ as follows: $l(\emptyset) = \emptyset^*$, $l(a) = (a\, b^*\, a^*)^*\, \emptyset^*$, $l(b) = (b^*\, a^*)^*\, \emptyset^*$, $l(c) = c^*$.

5. Discussion

We enriched elementary CA rule 30 with majority memory and demonstrated that by applying certain filtering procedures we can extract rich dynamics of traveling localizations, or gliders. We inferred a sophisticated system of quasi-chemical reactions between the gliders. It was shown that the majority memory increases nominal complexity but decreases statistical complexity of patterns generated by the CA. By applying methods of de Brujin diagrams and graph theory, we proved the surjectivity of rule 30 CA with memory and provided blue prints for future detailed analysis of glider dynamics.

Recalling previous results on the classification of one-dimensional CA [17, 23, 26], we envisage that introducing majority memory ϕ_{maj} into elementary CA will open a new field of research in the selection of nontrivial rules of cell-state transitions and precise mechanics of relationships between chaotic and complex systems.

This is because rule 30 was grouped into a cluster of rules with similar behavior, that can be transformed one to another using combinations of reflection, negation, and complement (as done by Wuensche in [26]). Figure 13 shows a diagram that explains how the original cluster for rule 30 is presented in [26]. Obviously, the cluster can be arbitrarily enriched using not only ϕ_{maj} but any type of memory and τ. Thus, the dynamical complexity of automata with ϕ_{maj} is the same as for the set of functions $\{\varphi_{\text{R30}}, \varphi_{\text{R86}}, \varphi_{\text{R135}}, \varphi_{\text{R149}}\}$, particularly because the local functions φ_{R86} and φ_{R149} are responsible for the leftward motion of gliders.

Therefore, memory in elementary and other CA families offers a new approach for discovering complex dynamics based on gliders and nontrivial interactions between gliders. This can be substantiated by a number of different techniques, for example, number-conservation [27, 28], exhaustive search [29], tiling [9, 30], de Bruijn diagrams [3], Z-parameter [17], genetic algorithms [31], mean field theory [32], or from a differential equations viewpoint [33].

https://doi.org/10.25088/ComplexSystems.18.3.345

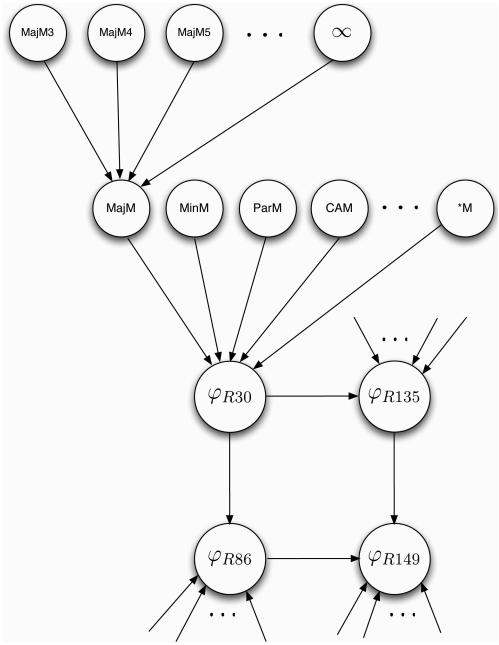


Figure 13. A new family of elementary CA that can be composed.

Acknowledgments

Genaro J. Martínez and Ramon Alonso-Sanz are supported by EPSRC (grants EP/F054343/1 and EP/E049281/1). Juan C. Seck-Tuoh-Mora is supported by CONACYT (project CB-2007/083554).

Appendix

A. Binary Reactions and Beyond

We represent binary collisions in rule 30, ϕ_{maj} and $\tau = 8$ in the form $g_j \xrightarrow{space} g_i$, where $j > i$, $g \in G_{\text{R30}}$ and *space* is the interval between gliders given by the number of strings w_b. Collisions that produce ϵ mean the annihilation of gliders. Reactions are developed by increasing the distance between gliders before collision.

Collisions of type $g_2 \rightarrow g_1$

1. $g_2 \xrightarrow{3} g_1 = g_1 + g_2$ (soliton)
2. $g_2 \xrightarrow{4} g_1 = \epsilon$

https://doi.org/10.25088/ComplexSystems.18.3.345

3. $g_2 \xrightarrow{5} g_1 = \epsilon$

4. $g_2 \xrightarrow{6} g_1 = 2\,g_1$

5. $g_2 \xrightarrow{7} g_1 = \epsilon$

6. $g_2 \xrightarrow{8} g_1 = g_3$

Collisions of type $g_3 \to g_1$

1. $g_3 \xrightarrow{3} g_1 = g_2$

2. $g_3 \xrightarrow{4} g_1 = g_2$

3. $g_3 \xrightarrow{5} g_1 = g_2 + g_3$

4. $g_3 \xrightarrow{6} g_1 = g_1^3$

5. $g_3 \xrightarrow{7} g_1 = g_1 + g_1^2$

6. $g_3 \xrightarrow{8} g_1 = g_2$

7. $g_3 \xrightarrow{9} g_1 = g_1$

8. $g_3 \xrightarrow{10} g_1 = g_2$

9. $g_3 \xrightarrow{11} g_1 = g_1$

Collisions of type $g_3 \to g_2$

1. $g_3 \xrightarrow{3} g_2 = g_2$

2. $g_3 \xrightarrow{4} g_2 = g_2$

3. $g_3 \xrightarrow{5} g_2 = 2\,g_1 + g_2$

4. $g_3 \xrightarrow{6} g_2 = 2\,g_1 + g_2$

Some other reactions with packages of gliders

1. $g_2 \xrightarrow{5} 2\,g_1 = g_1$

2. $g_2 \xrightarrow{8} 2\,g_1 = 3\,g_1$

https://doi.org/10.25088/ComplexSystems.18.3.345

3. $g_2 \xrightarrow{10} 2\,g_1 = g_2$ (sequence g_2, g_3, g_2)
4. $2\,g_2 \xrightarrow{6} g_1 = 3\,g_1$
5. $2\,g_2 \xrightarrow{7} g_1 = g_2$
6. $2\,g_2 \xrightarrow{8} g_1 = g_1 + g_3$
7. $g_3 \xrightarrow{5} 2\,g_1 = \epsilon$
8. $g_3 \xrightarrow{6} 2\,g_1 = g_3$
9. $g_3 \xrightarrow{7} 2\,g_1 = g_1 + g_2$
10. $g_3 \xrightarrow{8} 2\,g_1 = g_1^3 + g_1$
11. $g_3 \xrightarrow{9} 2\,g_1 = g_1^4$
12. $g_3 \xrightarrow{10} 2\,g_1 = 2\,g_1$
13. $2\,g_3 \xrightarrow{5} g_2 = g_2$ (wall)

B. Coding Gliders G_{R30m}

We can enumerate strings conforming gliders in rule 30 with ϕ_{maj} and $\tau = 8$, in given initial conditions and using "phases" (we omit strings that do not produce gliders).

Note that such strings evolve initially with φ_{R30} and a value of τ given, then ϕ_{maj} will open these strings when memory works. Thus we can code initial conditions with gliders in CA with memory, that also was implemented in OSXLCAU21 system to get our simulations.

Table 2 enumerates strings for each glider represented as a tiling so we know their "phases" [9, 19]. In this case, however, it was difficult to classify such strings as regular expressions because not all strings from the tiling representation evolve into gliders.

https://doi.org/10.25088/ComplexSystems.18.3.345

g_1 Glider	g_2 Glider	g_3 Glider
1 - g_1	100 - g_2	110110 - 2 g_1 join
111 - g_1	111 - g_1	101110 - g_3
10000 - g_1	10000 - g_1	111100 - 2 g_1
11110 - g_1	11110 - g_1	100011 - 2 g_1 join
100 - g_2	1100110 - $g_2 + g_3$	1011 - g_3
11110 - g_1	1010110 - g_1	
10000 - g_1	1101100 - g_2	
11001 - $g_2 + g_3$		

Table 3. Strings evolving in gliders of G_{R30m}.

References

[1] S. Wolfram, *Cellular Automata and Complexity: Collected Papers*, Reading, MA: Addison-Wesley Publishing Company, 1994.

[2] G. J. Martínez, A. Adamatzky, and H. V. McIntosh, "Phenomenology of Glider Collisions in Cellular Automaton Rule 54 and Associated Logical Gates," *Chaos, Solitons and Fractals*, **28**(1), 2006 pp. 100-111.

[3] G. J. Martínez, A. Adamatzky, and H. V. McIntosh, "On the Representation of Gliders in Rule 54 by de Bruijn and Cycle Diagrams," in *Proceedings of the Eighth International Conference on Cellular Automata for Research and Industry (Part 1)*, Yokohama, Japan, *Lecture Notes in Computer Science*, **5191**, Berlin: Springer, 2008 pp. 83-91. doi.10.1007/978-3-540-79992-4_ 11.

[4] M. L. Minsky, *Computation: Finite and Infinite Machines*, Englewood Cliffs, NJ: Prentice-Hall, 1967.

[5] R. Alonso-Sanz, *Cellular Automata with Memory*, Philadelphia: Old City Publishing, Inc., 2008.

[6] S. Amoroso and G. Cooper, "The Garden-of-Eden Theorem for Finite Configurations," *Proceedings of the American Mathematical Society*, **26**, 1970 pp. 158-164.

[7] E. S. Rowland, "Local Nested Structure in Rule 30," *Complex Systems*, **16**(3), 2006 pp. 239-258.

[8] J. C. Seck-Tuoh-Mora, S. V. Chapa-Vergara, G. J. Martínez, and H. V. McIntosh, "Procedures for Calculating Reversible One-Dimensional Cellular Automata," *Physica D: Nonlinear Phenomena*, **202**(1-2), 2005 pp. 134-141. cat.inist.fr/?aModele=afficheN&cpsidt=16660256.

[9] G. J. Martínez, H. V. McIntosh, J. C. Seck-Tuoh-Mora, and S. V. Chapa-Vergara, "Determining a Regular Language by Glider-Based Structures Called *Phases f_i_1* in Rule 110," *Journal of Cellular Automata*, **3**(3), 2008 pp. 231-270.

[10] H. V. McIntosh, *One-Dimensional Cellular Automata*, Beckington, UK: Luniver Press, 2009.

https://doi.org/10.25088/ComplexSystems.18.3.345

[11] B. H. Voorhees, *Computational Analysis of One-Dimensional Cellular Automata*, Series A, Vol. 15, River Edge, NJ: World Scientific Series on Nonlinear Science, 1996.

[12] J. C. Seck-Tuoh-Mora, G. J. Martínez, and H. V. McIntosh, "Calculating Ancestors in One-Dimensional Cellular Automata," *International Journal of Modern Physics C (IJMPC)*, **15**(8), 2004 pp. 1151-1169. doi.10.1142/S0129183104006625.

[13] G. J. Martínez, H. V. McIntosh, J. C. Seck-Tuoh-Mora, and S. V. Chapa-Vergara, "A Note About the Regular Language of Rule 110 and Its General Machine: The Scalar Subset Diagram," in *Proceedings of the Third International Workshop on Natural Computing* (*Japan Society for Artificial Intelligence)*, **C3004**, 2008 pp. 39-49.

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

[15] R. Alonso-Sanz and M. Martin, "One-Dimensional Cellular Automata with Memory in Cells of the Most Recent Value," *Complex Systems*, **15**(3), 2005 pp. 203-236.

[16] R. Alonso-Sanz, "Elementary Rules with Elementary Memory Rules: The Case of Linear Rules," *Journal of Cellular Automata*, **1**(1), 2006 pp. 71-87.

[17] A. Wuensche, "Classifying Cellular Automata Automatically: Finding Gliders, Filtering, and Relating Space-Time Patterns, Attractor Basins, and the Z Parameter," *Complexity*, **4**(3), 1999 pp. 47-66. doi.10.1002/(SICI)1099-0526(199901/02)4:3<47::AID-CPLX9>3.3.-CO;2-M.

[18] A. Adamatzky and O. Holland, "Phenomenology of Excitation in 2-D Cellular Automata and Swarm Systems," *Chaos, Solitons & Fractals*, **9**(7), 1998 pp. 1233-1265. doi.10.1016/S0960-0779(97)00123-9.

[19] G. J. Martínez, H. V. McIntosh, and J. C. Seck-Tuoh-Mora, "Gliders in Rule 110," *International Journal of Unconventional Computing*, **2**(1), 2006 pp. 1-50.

[20] M. Cook, "Universality in Elementary Cellular Automata," *Complex Systems*, **15**(1), 2004 pp. 1-40.

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

[22] N. Boccara, J. Nasser, and M. Roger, "Particlelike Structures and Their Interactions in Spatiotemporal Patterns Generated by One-Dimensional Deterministic Cellular Automaton Rules," *Physical Review A*, **44**(2), 1991 pp. 866-875. doi.10.1103/PhysRevA.44.866.

[23] A. Wuensche, "Complexity in One-D Cellular Automata: Gliders, Basins of Attraction and the Z Parameter," *Santa Fe Institute* working paper 94-04-025, 1994.

[24] J. K. Park, K. Steiglitz, and W. P. Thurston, "Soliton-Like Behavior in Automata," *Physica D:Nonlinear Phenomena*, **19**(3), 1986 pp. 423-432. doi.10.1016/-167-2789(86)90068-0.

[25] M. H. Jakubowski, K. Steiglitz, and R. K. Squier, "Computing with Solitons: A Review and Prospectus," *Multiple-Valued Logic*, **6**(5-6), 2001. Also republished in A. Adamatzky, ed., *Collision-Based Computing*, New York: Springer, 2002 pp. 277-299.

https://doi.org/10.25088/ComplexSystems.18.3.345

[26] A. Wuensche and M. Lesser, *The Global Dynamics of Cellular Automata: An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata*, Santa Fe Institute Studies in the Sciences of Complexity, Reading, MA: Addison-Wesley Publishing Company, 1992.

[27] N. Boccara and H. Fuks, "Number-Conserving Cellular Automaton Rules," *Fundamenta Informaticae*, **52**(1-3), 2002 pp. 1-13.

[28] K. Imai, A. Ikazaki, C. Iwamoto, and K. Morita, "A Logically Universal Number-Conserving Cellular Automaton with a Unary Table-Lookup Function," *IEICE Transactions on Information and Systems*, **E87-D**(3), 2004 pp. 694-699.

[29] D. Eppstein, "Searching for Spaceships," *MSRI Publications*, **42**, 2002 pp. 433-453.

[30] M. Margenstern, *Cellular Automata in Hyperbolic Spaces*, Vol. 1: Theory, Philadelphia: Old City Publishing, Inc., 2007.

[31] R. Das, M. Mitchell, and J. P. Crutchfield, "A Genetic Algorithm Discovers Particle-Based Computation in Cellular Automata," in *Proceedings of the International Conference on Evolutionary Computation (The Third Conference on Parallel Problem Solving from Nature(PPSN III)*, Jerusalem, *Lecture Notes in Computer Science*, **866**, London: Springer-Verlag, 1994 pp. 344-353.

[32] H. V. McIntosh, "Wolfram's Class IV and a Good Life," *Physica D*, **45**(1-3), 1990 pp. 105-121. doi.10.1016/0167-2789(90)90177-Q.

[33] L. O. Chua, *A Nonlinear Dynamics Perspective of Wolfram's New Kind of Science*, Hackensack, NJ: World Scientific Publishing Company, 2007.

https://doi.org/10.25088/ComplexSystems.18.3.345