How to Make Dull Cellular Automata Complex by Adding Memory: Rule 126 Case Study

GENARO J. MARTÍNEZ,1,2 ANDREW ADAMATZKY,2 JUAN C. SECK-TUOH-MORA,3 AND RAMON ALONSO-SANZ4
1Departamento de Gravitación, Instituto de Ciencias Nucleares, Universidad Nacional Autónoma de México, México; 2Department of Computer Science, Bristol Institute of Technology, University of the West of England, Bristol, United Kingdom; 3TecnoLogías Avanzadas en Ingeniería, Centro de Investigación Avanzada en Ingeniería Industrial, Universidad Autónoma del Estado de Hidalgo, México; and 4Departamento de Estadística y Métodos de Gestión en Agricultura, ETSI Agrónomos, Polytechnic University of Madrid, Madrid, Spain

Received July 14, 2009; revised December 2, 2009; accepted December 16, 2009

Using Rule 126 elementary cellular automaton (ECA), we demonstrate that a chaotic discrete system — when enriched with memory — hence exhibits complex dynamics where such space exploits on an ample universe of periodic patterns induced from original information of the ahistorical system. First, we analyze classic ECA Rule 126 to identify basic characteristics with mean field theory, basins, and de Bruijn diagrams. To derive this complex dynamics, we use a kind of memory on Rule 126; from here interactions between gliders are studied for detecting stationary patterns, glider guns, and simulating specific simple computable functions produced by glider collisions. © 2010 Wiley Periodicals, Inc. Complexity 15: 34–49, 2010

Key Words: elementary cellular automata; memory; Rule 126; gliders; glider guns; filters; chaos; complex dynamics

1. INTRODUCTION

In this article, we are making use of the memory tool to get a complex system from a chaotic function in discrete dynamical environments. Such technique takes the past history of the system for constructing its present and future: the memory [1–6]. It was previously reported in [7] how the chaotic elementary cellular automaton (ECA) Rule 30 is decomposed into a complex system applying memory on this function. Recent results show that other chaotic functions (Rule 86 and Rule 101) yield complex dynamics selecting a kind of memory, including a controller to obtain self-organization by structure reactions and simple computations implemented by soliton reactions [8].

We focus this work on cellular automata (CA) evolving in one dimension, in particular taking the well-known ECA where each function evaluates a central cell and its two nearest neighbors (from left and right), and every cell takes a value from a binary alphabet. Such ECA were introduced by Wolfram and have been widely studied in several directions [9].
Among such ECA, there is a set of functions evolving in chaotic global behavior where a number of cells remain unordered and their transitions have a large number of ancestors [10]. In this sense, special attention is given on a chaotic one: the ECA Rule 126 introduced by Wolfram in [11]. Particularly in [12],1 it is commented that Rule 126 generates a regular language with average growing faster than any polynomial time. This property can be analyzed as well by de Bruijn diagrams with additional features; in particular, another partial analysis with de Bruijn and subset diagrams was done by McIntosh [13] showing that string 010 in Rule 126 is the minimal Garden of Eden configuration (having no ancestors).

Based on the idea developed in previous results for obtaining complex dynamics from chaotic functions selecting memory and working systematically, it was suspected that a complex dynamics may emerge in Rule 126 given its relation to regular languages; making use of gliders coded by regular expressions, as it was studied in Rule 110 [14] and Rule 54 [15].

In this way, Rule 126 provides a special case of how a chaotic behavior can be decomposed selecting a kind of memory into an extraordinary activity of gliders, glider guns, still-life structures, and a huge number of reactions. Such features can be compared to Brain Brian's rule behavior or Conway's Life but in one dimension; actually none traditional ECA could have a glider dynamics comparable to the one revealed in this ECA with memory denoted as φ_{R126maj} (following notation described in [7, 8]).

1.1. One-Dimensional Cellular Automata

One-dimensional CA is represented by an array of cells \( x_i \) where \( i \in \mathbb{Z} \) (integer set) and each \( x \) takes a value from a finite alphabet \( \Sigma \). Thus, a sequence of cells \( \{x_i\} \) of finite length \( n \) describes a string or global configuration \( c \) on \( \Sigma \). This way, the set of finite configurations will be expressed as \( \Sigma^n \). An evolution is comprised by a sequence of configurations \( \{c_i\} \) produced by the mapping \( \Phi : \Sigma^n \rightarrow \Sigma^n ; \) thus, the global relation is symbolized as:

\[
\Phi(c^t) \rightarrow c^{t+1},
\]

where \( t \) represents time and every global state of \( c \) is defined by a sequence of cell states. The global relation is determined over the cell states in configuration \( c^t \) updated at the next configuration \( c^{t+1} \) simultaneously by a local function \( \varphi \) as follows:

\[
\varphi(x_i^{t-\tau}, \ldots, x_i^{t-1}, x_i^t) \rightarrow x_i^{t+1}.
\]

Wolfram represents one-dimensional CA with two parameters \((k, r)\), where \( k = |\Sigma| \) is the number of states, and \( r \)

is the neighborhood radius, hence ECA domain is defined by parameters \((2, 1)\). There are \( \Sigma^d \) different neighborhoods (where \( n = 2r + 1 \) and \( k^n \)) distinct evolution rules. The evolutions in this article have periodic boundary conditions.

1.2. Cellular Automata with Memory

Conventional CA are ahistoric (memoryless): i.e., the new state of a cell depends on the neighborhood configuration solely at the preceding time step of \( \varphi \). CA with memory can be considered as an extension of the standard framework of CA where every cell \( x_i \) is allowed to remember some period of its previous evolution. Basically, memory is based on the state and history of the system, thus we design a memory function \( \phi \), as follows:

\[
\phi(x_i^{t-\tau}, \ldots, x_i^{t-1}, x_i^t) \rightarrow s_i
\]

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

\[
\varphi(\ldots, x_i^{t-1}, s_i, x_i^{t+1}, \ldots) \rightarrow x_i^{t+1}.
\]

In CA with memory, while the mapping \( \varphi \) remains unaltered, a historic memory of past iterations is retained by featuring each cell as a summary of its previous states; therefore, cells canalize memory to the map \( \varphi \). As an example, we can take the memory function \( \phi \) as a majority memory:

\[
\phi_{maj} \rightarrow s_i
\]

So, \( \phi_{maj} \) represents the classic majority function for three variables [16], as follows:

\[
\Phi_{maj} : (x_1 \land x_2) \lor (x_2 \land x_3) \lor (x_3 \land x_1) \rightarrow x
\]

on cells \((x_i^{t-\tau}, \ldots, x_i^{t-1}, x_i^t)\) and defines a temporal ring before calculating the next global configuration \( c \). In case of a tie, it allows to break it in favor of zero if \( x_{t-1} = 0 \), or to one whether \( x_{t-1} = 1 \). The representation of a ECA with memory (given previously in [7, 8]) is given as follows:

\[
\Phi_{CAR_{maj}}
\]

where \( CAR \) represents the decimal notation of a particular ECA and \( m \) the kind of memory given with a specific value of \( \tau \). Thus, the majority memory (maj) working in ECA Rule 126 checking tree cells \( (\tau = 3) \) of history is simply denoted as \( \Phi_{R126maj} \). Figure 1 depicts in detail the memory working on ECA.

Note that memory is as simple as any CA local function but sometimes the global behavior produced by the local rule is totally unpredictable.
1.2.1. Disclaimer

The memory mechanism considered here differs from other CA with memory previously reported, often referred as higher order (in-time) CA. These ones, in most cases, explicitly alter the function \( \phi \) and incorporate memory by directly determining the new configuration in terms of the configurations at previous time-steps. Thus, in second order in time (memory of capacity two) rules, the transition rule operates as:

\[ x_t^{t+1} = \phi(x_t, x_{t-1}) \]

Double memory (in transition rule and in cells) can be implemented as:

\[ x_t^{t+1} = \phi(s_t, \{s_{t-1}\}) \]

Particularly interesting is the reversible formulation:

\[ x_t^{t+1} = \phi^{-1}(x_t^{t-1}) \]

reversible CA with memory are studied in [6].

Some authors define rules with memory as those with dependence in \( \phi \) on the state of the cell to be updated [20, 21]. So, one-dimensional rules with no memory adopt the form:

\[ x_t^{t+1} = \psi(x_{t-1}, x_t, x_{t+1}) \]

The use of the locution associative memory usually refers, when used in the CA context, to the study of configuration attractors [25, 26], which are argued by Wuensche [27] to constitute the network’s global states contents addressable memory in the sense of Hopfield [28].

Finally, it is not intended here to emulate human memory, i.e., the associative, pattern matching, highly parallel function of human memory. The aim is just to store the past, or just a part of it, to make it work in dynamics. Thus, working storage might replace here the use of the term memory, avoiding the anthropomorphic, and rather unavoidable, connotations of the word memory [6].

2. THE BASIC FUNCTION: ECA RULE 126

The local-state transition function \( \psi \) corresponding to Rule 126 is represented as follows:

\[
\psi_{126} = \begin{cases} 
1 & \text{if } 110, 101, 100, 011, 010, 001 \\
0 & \text{if } 111, 000 
\end{cases}
\]

Rule 126 has a chaotic global behavior typical from Class III in Wolfram’s classification [9]. In \( \psi_{126} \), we can easily recognize an initial high probability of alive cells, i.e., cells in state “1”; with a 75% to appear in the next time and complement of only 25% to get state 0. It will be always a new alive cell iff \( \psi_{126} \) has one or two alive cells such that the equilibrium comes when there is an overpopulation condition. Figure 2 shows these cases in typical evolutions of Rule 126, both evolving from a single cell in state “1” [Figure 2(a)] and from a random initial configuration [Figure 2(b)] where a high density of 1’s is evidently in the evolution.

While looking on chaotic space-time configuration in Figure 2, we understand the difficulty for analyzing the rule’s behavior and selecting any coherent activity among periodic structures without special tools.
2.1. Mean Field Approximation in ECA Rule 126

This section presents a probabilistic analysis with mean field theory to search basic properties about $\varphi_{R126}$ evolution space and its related chaotic behavior. Such analysis will offer a better spectrum where we can start to explore the evolution space from more useful and specific initial conditions where some interesting behaviors may emerge.

Mean field theory is a well-known technique for discovering statistical properties of CA without analyzing evolution spaces of individual rules [13]. The method assumes that states in $\Sigma$ are independent and do not correlate each other in the local function $\varphi_{R126}$. Thus, we can study probabilities of states in a neighborhood in terms of the probability of a single state (the state in which the neighborhood evolves), and probability of the neighborhood is product of the probabilities of each cell in it.

In this way, [29] presents an explanation of Wolfram’s classes by a mixture of probability theory and de Bruijn diagrams, resulting a classification based on mean field theory curve:

- class I: monotonic, entirely on one side of diagonal;
- class II: horizontal tangency, never reaches diagonal;
- class IV: horizontal plus diagonal tangency, no crossing;
- class III: no tangencies, curve crosses diagonal.

For the one-dimensional case, all neighborhoods are considered as follows:

$$p_{t+1} = \sum_{j=0}^{2r+1-1} \varphi_j(X) p_j^t (1 - p_j)^{n-v}$$

such that $j$ is an index number relating each neighborhood and $X$ are cells $x_{i-r}, \ldots, x_i, \ldots, x_{i+r}$. Thus, $n$ is the number of cells into every neighborhood, $v$ indicates how often state “1” occurs in $X$, $n - v$ shows how often state “0” occurs in the neighborhood $X$, $p_t$ is the probability of cell being in state “1,” while $q_t$ is the probability of cell being in state “0,” i.e., $q = 1 - p$. The polynomial for Rule 126 is defined as follows:

$$p_{t+1} = 3p_t q_t.$$  \hspace{1cm} (7)

Because $\varphi_{R126}$ is classified as a chaotic rule, we expect no tangencies and its curve must cross the identity; remembering that $\varphi_{R126}$ has a 75% of probability to produce a state one.

Mean field curve (Figure 3) confirms that probability of state “1” in space-time configurations of $\varphi_{R126}$ is 0.75 for high densities related to big populations of 1’s. The curve also
demonstrates that $\varphi_{126}$ is chaotic because the curve cross the identity with a first fixed point at the origin $f = 0$ and the nonexistence of any unstable fixed point inducing non-stable regions in the evolution. Nevertheless, the stable fixed point is $f = 0.6683$, which represents a “concentration” of 1s diminishing during the automaton evolution.

Therefore, the initial inspection indicates no evidence of complex behavior emerging in $\varphi_{126}$. Of course, a deeper analysis is necessary for obtaining more features from a chaotic rule, so the next sections explain other techniques to study in particular periodic structures.

## 2.2. Basins of Attraction

A basin (of attraction) field of a finite CA is the set of basins of attraction into which all possible states and trajectories will be organized by the local function $\varphi$. The topology of a single basin of attraction may be represented by a diagram, the state transition graph. Thus, the set of graphs composing the field specifies the global behavior of the system [30].

Generally, a basin can also recognize CA with chaotic or complex behavior following previous results on attractors [30]. Thus, we have that Wolfram’s classes can be represented as a basin classification:

- class I: very short transients, mainly point attractors (but possibly also periodic attractors) very high in-degree, and very high leaf density (very ordered dynamics);
- class II: very short transients, mainly short periodic attractors (but also point attractors), high in-degree, and very high leaf density;
- class IV: moderate transients, moderate-length periodic attractors, moderate in-degree, and very moderate leaf density (possibly complex dynamics);
- class III: very long transients, very long periodic attractors, low in-degree, and low leaf density (chaotic dynamics).

The basins depicted in Figure 5 show the whole set of non-equivalent basins in Rule 126 from $l = 2$ to $l = 18$ ($l$ means length of array) attractors, all they display not high densities from an attractor of mass one and attractors of mass 14.3

This way Rule 126 displays some nonsymmetric basins and some of them have long transients that induce a relation with chaotic rules.

Particularly we can see specific cycles in Figure 4 where it is possible to find:

- a. static configurations as still life patterns ($l = 8$);
- b. traveling configurations as gliders ($l = 15$);
- c. meshes ($l = 12$);
- d. or empty universes ($l = 14$).

The cycle diagrams expose only displacements to the left, and this empty universe evolving to the stable state 0 is constructed all times on the first basin for each cycle, see Figure 5.

This way some cycles could induce some nontrivial activity in Rule 126, and this empty universe evolving to the stable state 0 is constructed all times on the first basin for each cycle, see Figure 5.

This way some cycles could induce some nontrivial activity in Rule 126, and this empty universe evolving to the stable state 0 is constructed all times on the first basin for each cycle, see Figure 5.

### 2.3. De Bruijn Diagrams

De Bruijn diagrams [13, 31] are very adequate for describing evolution rules in one-dimensional CA, although originally they were used in shift-register theory (the treatment of sequences where their elements overlap each other). Paths in a de Bruijn diagram may represent chains, configurations, or classes of configurations in the evolution space.

For a one-dimensional CA of order $(k, r)$, the de Bruijn diagram is defined as a directed graph with $k^{2r}$ vertices and

---

3Basins and attractors were calculated with Discrete Dynamical System DDLab available from http://www.ddlab.org/.
The whole set of nonequivalent basins in ECA Rule 126 from $l = 2$ to $l = 18$.

$k^{2r+1}$ edges. The vertices are labeled with the elements of the alphabet of length $2r$. An edge is directed from vertex $i$ to vertex $j$, if and only if, the $2r - 1$ final symbols of $i$ are the same that the $2r - 1$ initial ones in $j$ forming a neighborhood of $2r+1$ states represented by $i \circ j$. In this case, the edge connecting $i$ to $j$ is labeled with $\phi(i \circ j)$ (the value of the neighborhood defined by the local function) [32].

The de Bruijn diagram associated to Rule 126 is depicted in Figure 6.

Figure 6 exposes that there are two neighborhoods evolving into 0 and six neighborhoods into 1, so the higher frequency is for state 1, indicating the possibility of having an injective automaton, that is, the existence of Garden of Eden configurations [13, 31]. Classical analysis in graph theory has been applied over de Bruijn diagrams for studying topics such as reversibility [33]; in other sense, cycles in the diagram indicate periodic constructions in the evolution of the automaton if the label of the cycle agrees with the sequence defined by its nodes, taking periodic boundary conditions. Let us take the equivalent construction of a de Bruijn diagram to describe the evolution in two steps of Rule 126 (having now nodes composed by sequences of four symbols); the cycles of this new diagram are presented in Figure 7.

The extended de Bruijn diagrams [13] are useful for calculating all periodic sequences by the cycles defined in the diagram. These ones also show the shift of a sequence for a

De Bruijn diagrams were calculated using NXLCU21 designed by Harold V. McIntosh; available from http://delta.cs.cinvestav.mx/~mcintosh.
FIGURE 7

Patterns calculated from extended de Bruijn diagrams, in particular from cycles of order \((x, 2)\) (that means a \(x\)-shift in \(y\)-generations).

Cycles inside de Bruijn diagrams can be used for obtaining regular expressions representing a periodic pattern. Figure 7 displays three patterns calculated as: (a) shift \(-3\) in \(2\) generations representing a pattern with displacement to the left, (b) shift \(0\) in \(2\) generations describing a static pattern traveling without displacement, and (c) shift \(+3\) in \(2\) generations is exactly the symmetric pattern given in the first evolution.

So, we can also see in Figure 7 that it is possible to find patterns traveling in both directions, as gliders or mobile structures. However, generally, these constructions (strings) cannot live in combination with others structures, and therefore it is really hard to have this kind of objects with such characteristics. Although moreover Rule 126 has at least one glider! This will be explained in the next sections.

2.4. Filters for Recognizing Dynamics in Rule 126

Filters are a useful tool for discovering hidden order in chaotic or complex rules. Filters were introduced in CA studies by Wuensche who employed them to automatically classify cell-state transition functions, see [34]. Also filters related to tiles were successfully applied and deduced in analyzing space-time behavior of ECA governed by Rules 110 and 54 [15, 35, 36].

This way, we have found that Rule 126 has two types of two-dimensional tiles (which together work as filters over \(\varphi_{R126}\)):

- the tile \(t_1 = \begin{bmatrix} 1111 \\ 000000 \\ 011110 \\ 1100011 \\ 0110110 \\ 1111111 \end{bmatrix} \)
- the tile \(t_2 = \begin{bmatrix} 111111 \\ 1001 \end{bmatrix} \)

Filter \(t_1\) works more significantly on configurations generated by \(\varphi_{R126}\), the second one is not frequently found although it is exploited when Rule 126 is altered with memory (as we can see in following sections).

The application of the first filter is effective to discover gaps with little patterns traveling on triangles of ‘1’ states in the evolution space. Although even in this case it may be unclear how a dynamics would be interpreted, a careful inspection on the evolution brings to light very small gliders (as still life), as shown in Figure 8.

This glider emerging in Rule 126 and localized by a filter is precisely the periodic pattern calculated with the basin (Figure 4(a)) and the de Bruijn diagram (Figure 7(b)); in particular, the last one offers more information because such cycles allow to classify the whole phases when this glider is coded in the initial condition. The next sections demonstrate the effect of filters for recognizing an amazing universe evolving in this CA with memory.

3. CA \(\varphi_{R126maj:4}\) AND COMPLEX DYNAMICS

This section discusses both relevant aspects of classic Rule 126 (\(\varphi_{R126}\)) and Rule 126 with memory (\(\varphi_{R126maj:4}\)).

3.1. Dynamics Emerging with Majority Memory

As it was explained in [7, 37] a new family of evolution rules derived from classic ECA can be found selecting a kind of memory.

Figure 9 illustrates dynamics for some values of \(\tau\) in \(\varphi_{R126maj:4}\). The space-time configurations are also filtered to show a raw dynamics. We found that large odd values of \(\tau\) tend

\[5\text{Evolutions of } \varphi_{R126maj:4} \text{ were calculated with OSXL-CAU21 system available in http://uncomp.uwe.ac.uk/genaro/OSXCASystems.html.}\]
CA with majority memory $\phi^{R_{126}\text{maj}} \tau$, where 13 values of $\tau$ are evolved and filtered over a ring of 246 cells for 236 generations.
to define macrocells-like patterns [9, 13]. Even values of $\tau$ are responsible of a mixture of periodic and chaotic dynamics.

On exploring systematically distinct values of $\tau$, we found that $\phi_{R126maj:4}$ produces an impressive and nontrivial emergence of patterns traveling and colliding; so yet when the memory is working as a new function, it is possible to recognize some fragments inherited of the original rule $\phi_{R126}$.

We start our simulations with a single nonquiescent cell, an example of this space-time configuration is provided in Figure 10 showing the first 1156 steps, where in this case the automaton needed other 12,000 steps to reach a stationary configuration. Filter is convenient to eliminate the nonrelevant information about gliders. In Figure 10 we can see a number of gliders, glider guns, still-life configurations, and a wide number of combinations of such patterns colliding and traveling with different velocities and densities. Consequently, we can classify a number of periodic structures, objects, and interesting reactions.

Basic primitive gliders are displayed in Figure 11, there are still-life patterns $s_1$ and $s_2$, and gliders $g_1$ and $g_2$, respectively. These structures can be ordered in a set $G_{\phi_{R126maj:4}} = \{s_1, s_2, g_1, g_2\}$.

Figure 12 shows the three more frequent glider guns (gun1, gun2, and gun3) emerging in $\phi_{R126maj:4}$. The next three guns [Figure 12(a–c)] are combined in basic guns synchronized by multiple reactions, which preserve emission of gliders although some of them can get other frequencies. We can include them increasing the set of periodic structures $G_{\phi_{R126maj:4}} = \{s_1, s_2, g_1, g_2, \text{gun}_1, \text{gun}_2, \text{gun}_3, \text{and gun}_4\}$. The last gun is presented in Figure 14 as a collision of gliders.

Basic properties of $G_{\phi_{R126maj:4}}$ are given in Table 1, where also glider guns can be classified by frequencies of glider emission as follows:

- small gun1 fires a gliders $g_1$ and $g_2$ (in turn) every five steps;
- medium gun2 fires $g_1$ and $g_2$ (in turn) every 26 steps;
- large gun3 fires five $g_1$ and $g_2$ gliders every 100 steps.

**FIGURE 10**

Filtered space-time configuration of $\phi_{R126maj:4}$ with 602 cells, periodic boundaries, starting from one nonquiescent cell and running for 1156 steps.

**FIGURE 11**

Basic gliders in $\phi_{R126maj:4}$. Two stationary configurations (as still life patterns) $s_1$ and $s_2$, respectively, and two gliders $g_1$ and $g_2$.
Frequencies are related to the number of emitted gliders (in intervals) by a glider gun. Hence, the gun_2 generates one \( g_1 \) and another \( g_2 \) glider eight steps before to produce the next ones, and a gun_3 yields five \( g_1 \) and five \( g_2 \) gliders 100 steps before to produce the other ones.

3.2. Collisions between Gliders

Coding glider positions to get a desired reaction is a well-known solution for some related problems about complex behaviors. One of them is precisely the problem of self-organization (by structures Figure 13) [38]. In this way, we present how each basic glider can be produced by collisions between other different gliders.

A little bit more complicated is to obtain glider guns by collisions. Figure 14(a,b) depicts the production of a gun_2 from a multiple collision of gliders. Later on, we shall present two new combinations of guns [Figure 14(c,e)] and a new gun_4 [Figure 14(d)].

3.3. Other Collisions

A large variety of objects can be generated by collisions between gliders in \( \phi_{R126maj:4} \). Some of these objects are useful for designing complex dynamical structures. For instance, Figure 15(a–c) shows how to delete (as an eater configuration) a single glider or a stream of gliders \( g_1 \) and \( g_2 \). Two kinds of annihilations are depicted as well [see Figure 15(d,e)].

Figure 16 describes a number of black hole patterns emerging in \( \phi_{R126maj:4} \). Traditionally a black hole is a Life object\(^6\) absorbing any glider that comes close to the main body (in this case, still life patterns). Its relevance consists of knowing how many patterns are able of attracting gliders and consuming all of them forever, in these cases the \( g_1 \) and \( g_2 \) gliders.

Figure 17 displays soliton reactions, where colliding gliders preserve their original forms. As known, a soliton has a small change in their phase and displacement; also soliton reactions are obtained with single gliders or by a stream of them.

The last set of examples (Figure 18) presents other binary reactions and some multiple collisions between \( g_1 \) and \( g_2 \) gliders. Some of them are conservative and others produce a new number of these structures. Finally, we can take some collisions to exploit their dynamics and controlling glider reactions (see an extended relations of collisions in Table 2). Thus, it can be obtained a full number of reactions for generating desired gliders. The set of collisions is useful as “raw material” in the implementation of computations on \( \phi_{R126maj:4} \).

4. Computing in \( \phi_{R126maj:4} \)

Given the large number of reactions in \( \phi_{R126maj:4} \) (see Table 2) the rule could be useful for implementing collision-based computing schemes [36, 39].

Figure 19 illustrates the interaction of gliders traveling, colliding one another, and implementing a Boolean conjunction as result. Initially from previous collisions we can embed logical constructions of AND and NOT gates from this figure, as follows:

- for the gate \( (\neg a \land b) \) the implementation with \( \phi_{R126maj:4} \) is in Figure 18(b); but only with one \( g_2 \) glider (see Table 2).
- for the gate \( (a \land b) \) the implementation corresponds to Figure 13(a).

Table 1: Properties of Gliders \( \sigma_{\phi_{R126maj:4}} \)

<table>
<thead>
<tr>
<th>Structure</th>
<th>( v_g )</th>
<th>Linear volume</th>
<th>Mass</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1 )</td>
<td>0/2 = 0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>0/10 = 0</td>
<td>12</td>
<td>28</td>
</tr>
<tr>
<td>( g_1 )</td>
<td>3/5 = 0.6</td>
<td>8</td>
<td>17</td>
</tr>
<tr>
<td>( g_2 )</td>
<td>−3/5 = −0.6</td>
<td>8</td>
<td>17</td>
</tr>
<tr>
<td>gun_1</td>
<td>0/19 = 0</td>
<td>6</td>
<td>—</td>
</tr>
<tr>
<td>gun_2</td>
<td>0/27 = 0</td>
<td>6</td>
<td>—</td>
</tr>
<tr>
<td>gun_3</td>
<td>0/110 = 0</td>
<td>10</td>
<td>—</td>
</tr>
<tr>
<td>gun_4</td>
<td>0/84 = 0</td>
<td>15</td>
<td>—</td>
</tr>
</tbody>
</table>

\( v_g \) is calculated as displacement between period. The linear volume is the maximum distance between two cells in state ‘1’ (diameter of the set of cells in state ‘1’), finally mass is the number of cells in state ‘1’.

\( \sigma_{\phi_{R126maj:4}} \) is a large classification of Life objects in http://www.conwaylife.com/wiki/index.php.

---

\( \phi_{R126maj:4} \)
FIGURE 13

Generating basic gliders from collisions between other ones, this is self-organization by structures. The following reactions are illustrated as follows: (a) \( g_1 + g_2 = s_1 \), (b) \( s_1 + g_2 = g_1 \), (c) \( g_1 + s_1 = g_2 \), (d) \( s_2 + g_2 = g_1 \), (e) \( g_2 + s_1 = g_2 \), and (f) \( g_1 + g_2 = s_2 \).

- for the gate \((a \land \neg b)\) the implementation is presented in Figure 18(a); but only with one \(g_1\) glider (see Table 2).
- for one fanout gate \((a \leftrightarrow b = a + a + b)\) the implementation is shown in Figure 18(d).

Indeed, here we also adopt ideas developed by Rennard in his design of Life computing architectures [40]. Glider \(g_1\) represents value 0, two \(g_1\) gliders together represent a value 1. Two gliders \(2g_2\) traveling in positive direction describe the operator and one the register. Thus, the register will read \texttt{false} or \texttt{true} if they are produced successfully.

Figure 20 illustrates the basic reactions required to produce a primitive computational scheme in \(\phi_{R126maj:4}\). The following set of relations is applied (see Table 2):

\[
\begin{align*}
2g_1 \leftrightarrow 2g_2 &= \epsilon \\
g_1 \leftrightarrow 2g_2 &= g_1 \\
g_1 \leftrightarrow 2g_2 &= 2g_2 + g_1 \\
2g_1 \leftrightarrow 2g_2 &= g_1 \\
2g_1 \leftrightarrow 2g_2 &= 2g_2 + 2g_1
\end{align*}
\]

FIGURE 14

Generating gliders guns by collisions.
FIGURE 15

Eater reactions between gliders and still life (a, b, and c) and annihilation of gliders by collisions (d and e).

FIGURE 16

Black hole patterns consuming $g_1$ and $g_2$ gliders.

FIGURE 17

Examples of soliton reactions.

FIGURE 18

Examples of other collisions.
so we can represent serial reactions as:

\[
\begin{align*}
2g_1 + 2g_2 &= \epsilon \quad &\text{empty word} \\
2g_1 + 2g_2 &= g_1 \quad &\text{FALSE} \\
2g_1 + 2g_2 &= 2g_1 \quad &\text{TRUE}.
\end{align*}
\]

and a NOT gate can be represented as:

- \text{FALSE} + 2g_2 = \text{TRUE} + 2g_2, or
- \text{TRUE} + 2g_2 = \text{FALSE} + 2g_2.

### 4.1. Constructing Formal Languages Since Gliders Collisions

in \( \varphi_{R126\text{maj}:4} \)

To be considered as a mathematical machine, \( \varphi_{R126\text{maj}:4} \) should compute sets of formal languages [41]. We consider such implementation as an easy way to illustrate how implement some collision-based processes in \( \varphi_{R126\text{maj}:4} \) (Figure 21).

#### Table 2

Table of binary, multiple, and other collisions.

<table>
<thead>
<tr>
<th>Binary</th>
<th>Multiple</th>
<th>Soliton</th>
<th>Guns</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1 \leftarrow g_2 = s_1 )</td>
<td>( s_2 \leftarrow g_2 = 2g_1 )</td>
<td>( g_1 \rightarrow s_1 = s_1 + g_1 )</td>
<td>( s_1 \leftarrow 2g_2 = \text{gun}_1 )</td>
</tr>
<tr>
<td>( g_1 \rightarrow s_1 = s_1 )</td>
<td>( g_1 \rightarrow s_2 = 2g_2 )</td>
<td>( s_1 \leftarrow g_2 = g_2 + s_1 )</td>
<td>( \text{gun}_2 \leftarrow g_2 = \text{gun}_1 )</td>
</tr>
<tr>
<td>( s_1 \leftarrow g_2 = g_1 )</td>
<td>( s_2 \leftarrow 2g_2 = g_1 )</td>
<td>( 2g_1 \rightarrow s_2 = s_2 + 2g_1 )</td>
<td>( g_1 \leftarrow g_2 - 2g_2 = \text{gun}_2 )</td>
</tr>
<tr>
<td>( g_1 \rightarrow s_1 = g_2 )</td>
<td>( 2g_1 \rightarrow s_2 = 2g_2 )</td>
<td>( s_2 \leftarrow 2g_2 = 2g_2 + s_2 )</td>
<td>( g_1 \leftarrow g_2 - 2g_2 = \text{gun}_3 )</td>
</tr>
<tr>
<td>( s_2 \leftarrow g_2 = g_1 )</td>
<td>( s_2 \leftarrow 2g_2 = 2g_1 + g_2 )</td>
<td>( 2g_1 \rightarrow 2s_2 = 2g_1 + 2s_2 )</td>
<td>( 2g_1 \rightarrow 2g_2 = \text{gun}_2 )</td>
</tr>
<tr>
<td>( g_1 \rightarrow g_2 = s_1 )</td>
<td>( 2g_1 \rightarrow 2g_2 = g_1 + 2g_2 )</td>
<td>( 2s_2 \leftarrow 2g_2 = 2g_2 + 2s_2 )</td>
<td>( 2g_1 \leftarrow 2g_2 = 2g_2 + 2g_1 )</td>
</tr>
<tr>
<td>( g_1 \rightarrow g_2 = g_1 )</td>
<td>( 2g_1 \rightarrow g_2 = g_2 )</td>
<td>( 3g_1 \leftarrow 3g_2 = \text{gun}_1 )</td>
<td>( ^\star ) means gun composed</td>
</tr>
</tbody>
</table>

Let \( \Sigma \) be a nonempty finite alphabet. Thus, a string over \( \Sigma \) is a finite sequence of symbols from \( \Sigma \). A set of all strings over \( \Sigma \) of length \( n \) is denoted by \( \Sigma^n \). For example, if \( \Sigma = \{0, 1\} \), then \( \Sigma^2 = \{00, 01, 10, 11\} \).
$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$, and so on. This way $\Sigma^0 = \{\epsilon\}$ for any alphabet $\Sigma$. The set of all strings over $\Sigma$ of any length is the Kleene closure of $\Sigma$ and is denoted as $\Sigma^*$. However, we can write such expression in terms of $\Sigma^n$, as follows:

$$\sum^* = \bigcup_{n \in \mathbb{Z}_0} \Sigma^n. \quad (8)$$

This way for a binary alphabet $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$. Finally, a set of strings on $\Sigma$ is called a formal language.

**FIGURE 21**

Constructing the formal collision-based languages in $\phi_{126maj4}$: They are $\Sigma^0$ (top), $\Sigma^1$ (middle), and $\Sigma^2$ (bottom).
Making use of gliders in $\phi_{126maj}^4$, we can get any set of strings of $\Sigma^*$. For example, for the formal language $\Sigma^*$ we need to code all the initial conditions with reactions $\text{TRUE}$. For the formal language $(00 + 11)^*$ the initial condition will be coded as $(g_{1g1} + g_{2g1})^*$. To yield an arbitrary length of strings, we increase the number of cells in the CA. All distances between gliders must be preserved and the left part become periodic.

5. CONCLUSIONS
We have enriched ECA Rule 126 with majority memory and have demonstrated that by applying certain filtering procedures we can extract rich dynamics of gliders, guns and infer a sophisticated system of reactions between gliders. We have discovered how a complex dynamics emerges from a chaotic system selecting an adequate memory. We have shown that the majority memory increases nominal complexity but decreases statistical complexity of patterns generated by the CA. By applying methods such as de Bruijn diagrams, cycles, and graph theory, we have proved that Rule 126 with memory opens a new spectrum of complex rules, in this case a new CA with memory: $\phi_{126maj}^4$. Finally, we have demonstrated some capacities for computing specific logical and memory functions, and a future work will be constructing a universal device.

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-2007/83554).

REFERENCES