On complexity of chaotic elementary cellular automaton with memory: Rule 126

Genaro J. Martínez

http://uncomp.uwe.ac.uk/genaro/

International Center of Unconventional Computing
Bristol Institute of Technology
University of the West of England, Bristol, United Kingdom.

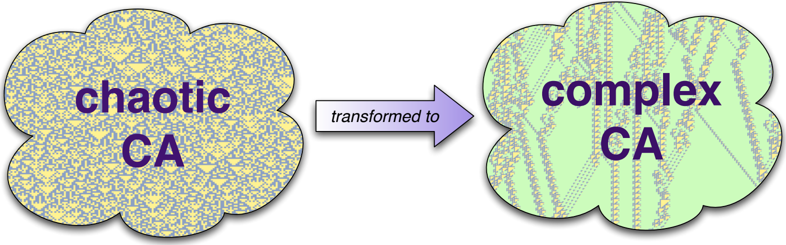
Summer Solstice 2009 International Conference on
Discrete Models of Complex Systems
Gdańsk University, Gdańsk, Poland
June 22–24, 2009

Abstract

Using Rule 126 elementary cellular automaton (ECA) we will demonstrate that a chaotic discrete system — when enriched with memory – exhibits complex dynamics. To quantify complexity of Rule 126 ECA with memory we study what types dynamics constructed in Rule 126's evolution emerge since mean field theory, basins and de Bruijn diagrams. Later we will display its complex dynamics emerging selecting a kind of memory for analyse interactions between gliders and stationary patterns implementing specific functions.

Objective and goal

In this talk we will display a simple tool to extract complex systems from a family of chaotic discrete dynamical system. We will employ a technique — memory based rule analysis of using past history of a system to construct its present state and to predict its future.



Cellular automata

Cellular automata (CA) are discrete dynamical systems evolving on an infinite regular lattice.

Definition

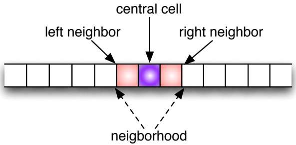
A *CA* is a 4-tuple $A =< \Sigma, u, \varphi, c_0 >$ evolving in d-dimensional latice, where $d \in \mathbb{Z}^+$. Such that:

- Σ represents the alphabet
- u the local connection, where, $u = \{x_{0,1,\ldots,n-1:d} | x \in \Sigma\}$, therefore, u is a neigborhood
- φ the local function, such that, $\varphi : \Sigma^u \rightarrow \Sigma$
- c_0 the initial condition, such that, $c_0 \in \Sigma^{\mathcal{Z}}$

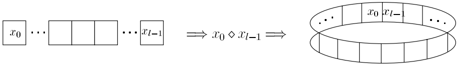
Also, the local function induces a global transition between configurations:

$$\Phi_\varphi : \Sigma^{\mathcal{Z}} \rightarrow \Sigma^{\mathcal{Z}}.$$

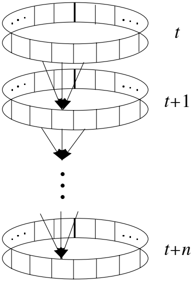
Dynamics in one dimension



boundary limit define a ring



evolution space



Elemental CA (ECA) is defined as follow:

- $\Sigma = \{0, 1\}$
- $u = \{x_1, x_0, x_{-1}\}$ such that $x \in \Sigma$
- the local function $\varphi : \Sigma^3 \rightarrow \Sigma$
- c_0 the initial condition is the first ring with $t = 0$

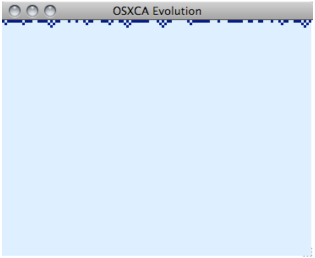
Wolfram's classification

Wolfram defines his classification in simple rules [Wolfram86], known as ECA. Also, this classification is extended to *n*-dimension.

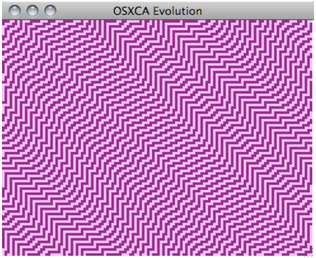
Classes

- A CA is class I, if there is a stable state $x_i \in \Sigma$, such that all finite configurations evolve to the *homogeneous configuration*.
- A CA is class II, if there is a stable state $x_i \in \Sigma$, such that any finite configuration become periodic.
- A CA is class III, if there is a stable state, such that for some pair of finite configurations c_i and c_j with the stable state, is decidable if c_i evolve to c_j, such that any configuration become chaotic.
- Class IV includes all CA also *called complex CA*.

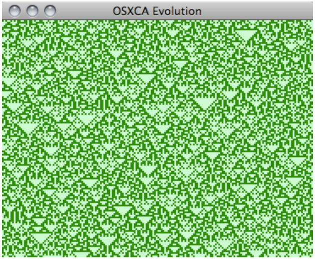
Wolfram's classes



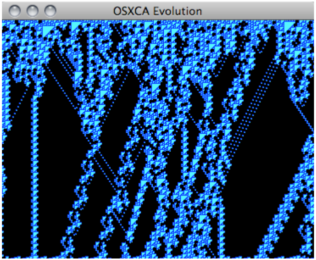
Rule 32



Rule 15



Rule 90



Rule 110

Figure: Behavior classes in ECA: *uniform*, *periodic*, *chaotic* and *complex* respectively.

The case of study: ECA Rule 126

$$\varphi_{R126} = \begin{cases} 1 & \text{if} \quad 110, 101, 100, 011, 010, 001 \\ 0 & \text{if} \quad 111, 000 \end{cases}$$

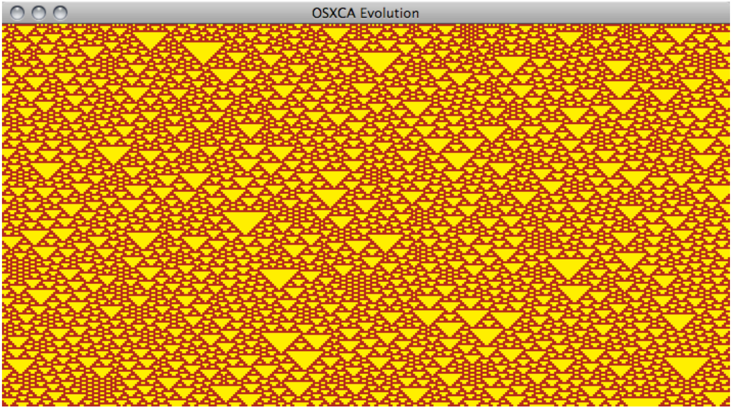


Figure: Chaotic ECA evolution rule 126. Initial density start with a 66% on a ring of 356 cells to 187 generations.

Mean field analysis

Mean field theory is a proven technique for discovering statistical properties of CA without analyzing evolution spaces of individual rules. In this way, it was proposed to explain Wolfram's classes by probability theory, resulting in 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.

Thus for one dimension we have:

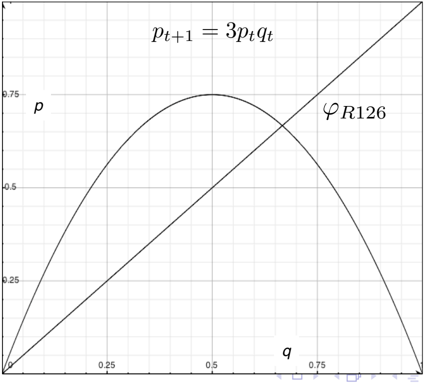
$$p_{t+1} = \sum_{j=0}^{k^{2r+1}-1} \varphi_j(X) p_t^v (1-p_t)^{n-v} \quad (1)$$

such that j is a number of relations from their neighborhoods and X the combination of cells $x_{i-r}, \dots, x_i, \dots, x_{i+r}$. n represents the number of cells in neighborhood, v indicates how often state one occurs in Moore's neighborhood, $n-v$ shows how often state zero occurs in the neighborhood, p_t is a probability of cell being in state one, q_t is a probability of cell being in state zero (therefore $q = 1 - p$).

Mean field polynomial for φ_{R126}

Mean field curve confirms that probability of state '1' in space-time configurations of ECA Rule 126 is 0.75 this probability of high densities of 1's with its maximum point in 0.5.

Rule 126 is chaotic because the curve cross the identity. The first unstable fixed point at the origin $f = 0$ show that given very small number of cells, all they in state '1' will spread quickly on the lattice. The stable fixed point is $f = 0.6683$, which represent 'concentration' of '1's that diminish during automaton development. Such stable fixed point hints on existence of non-trivial periodic structures emerging on ECA Rule 126, as was confirmed using filters.



Attractors analysis

Generally a basin could classifier CA with chaotic or complex behavior following also previous results on attractors [Wuensche92].

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

Basins in φ_{R126} with DDLab

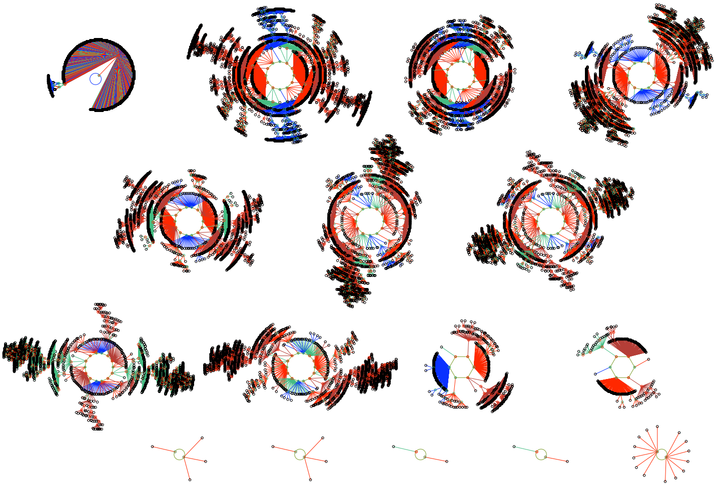


Figure: 16 non-equivalent basins in ECA Rule 126 for $l = 18$.

ECA 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 φ. 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.
Thus to implement a memory we design a memory function ϕ, as follow:

$$\phi(x_i^{t-\tau}, \ldots, x_i^{t-1}, x_i^t) \to s_i \quad (2)$$

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

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

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

ECA with memory

Firstly we should consider a kind of memory, in this case the majority memory ϕ_{maj} and then a value for τ. This value represent the number of cells backward to consider in the memory. Therefore a way to represent functions with memory and one ECA associated is proposed as follow:

$$\phi_{CAm:\tau} \tag{3}$$

such that CA represents the decimal notation of an specific ECA and m a kind of memory given. This way the majority memory working in ECA rule 126 checking tree cells on its history is denoted simply as $\phi_{R126maj:3}$.
Implementing the majority memory ϕ_{maj} we can select some ECA and experimentally look what is the effect.

ECA with memory

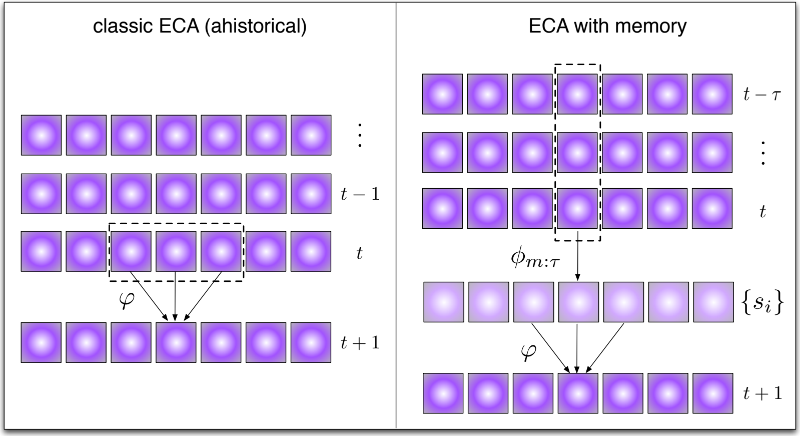


Figure: Memory working on ECA (preserving discrete domain).

Complex dynamics emerging in $\phi_{R126maj}$

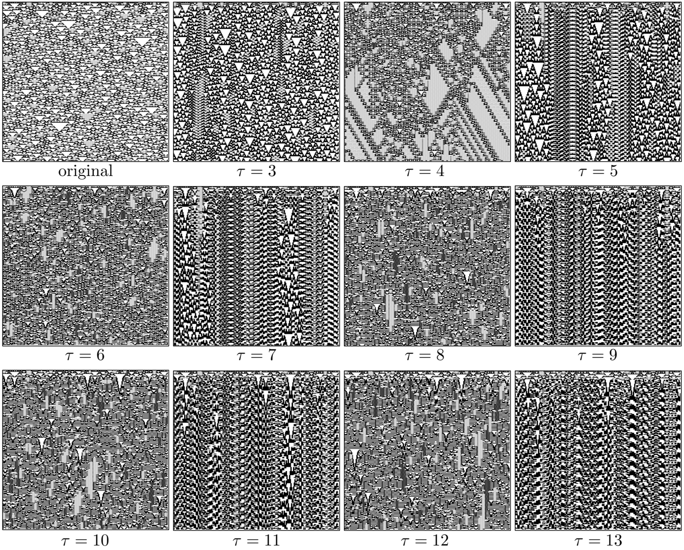


Figure: ECA Rule 126 with majority memory $\phi_{R126maj:\tau}$ since 13 values of τ are tested. All they were calculated on a ring of 246 cells for 236 generations also filtered.

Complex dynamics emerging in $\phi_{R126maj:4}$

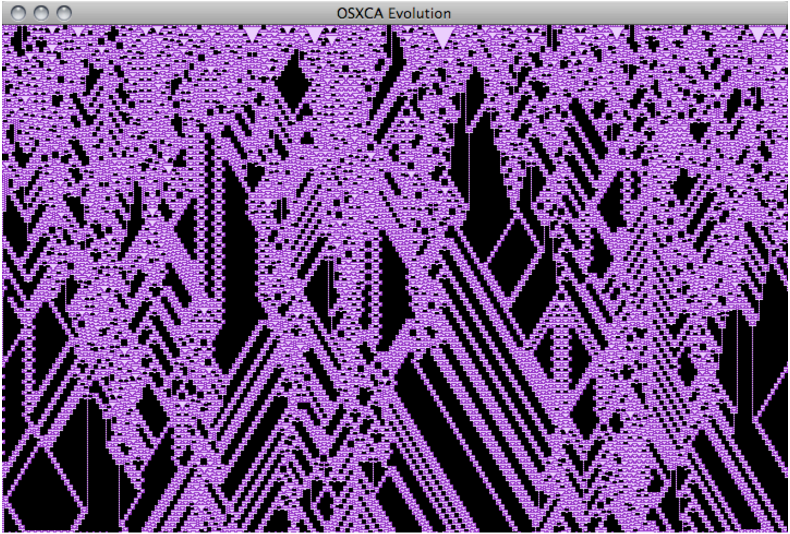
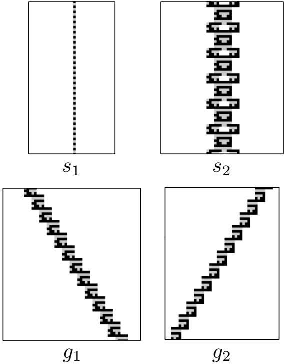


Figure: A new complex ECA with majority memory $\phi_{R126maj:4}$. Evolving on a ring of 246 cells for 236 generations also filtered.

Complex dynamics emerging in $\phi_{R126maj:4}$



structure	v_g	lineal volume	mass
s_1	$0/2 = 0$	1	1
s_2	$0/10 = 0$	12	28
g_1	$3/5 \approx 0.6$	8	17
g_2	$-3/5 \approx -0.6$	8	17
gun_1	$0/19 = 0$	6	-
gun_2	$0/27 = 0$	6	-
gun_3	$0/110 = 0$	10	-
gun_4	$0/84 = 0$	15	-

Figure: Basic gliders in $\phi_{R126maj:4}$. Two stationary configurations s_1 and s_2 respectively, and two gliders g_1 and g_2..

Self-organization by structure formation

Coding glider positions to get a reaction desired hence we can think about of solutions of some related problems on complexity behavior. One of them is precisely the problem of self-organization (by structures). In this way, we present how each basic glider can be produced in collisions between other different gliders.

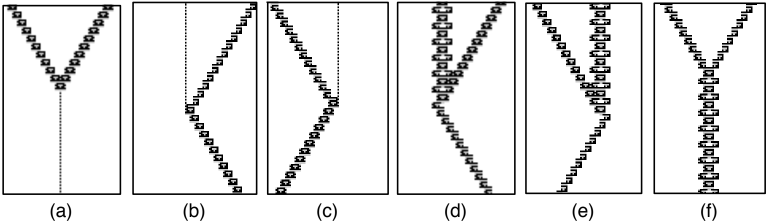


Figure: Generating of basic localizations since collisions between other localizations. The following reactions are illustrated, as follow: (a) $g_1 + g_2 = s_1$, (b) $s_1 + g_2 = g_1$, (c) $g_1 + s_1 = g_2$, (d) $s_2 + g_2 = g_1$, (e) $g_2 + s_1 = g_2$, and (f) $g_1 + g_2 = s_2$.

Self-organization by structure formation

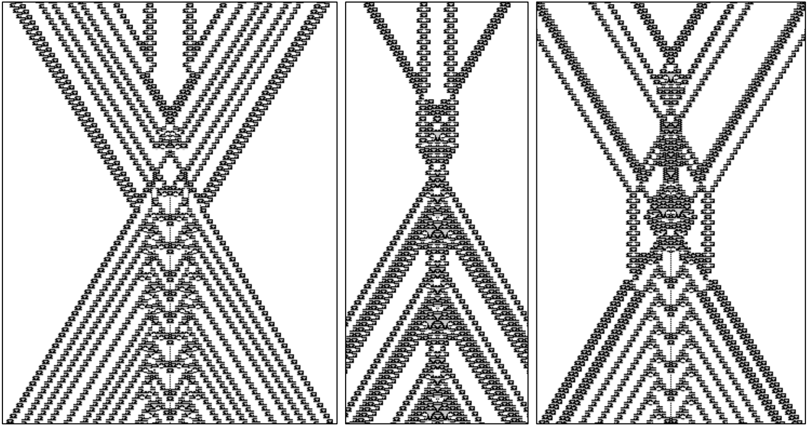


Figure: Generating gliders guns by multiple colliding gliders. Unlimited grown in $\phi_{R126maj:4}$.

Coding gliders by basin representation

Basically we will represent the s_1 gliders because this evolve in both ECA φ_{R126} and $\phi_{R126maj:4}$.

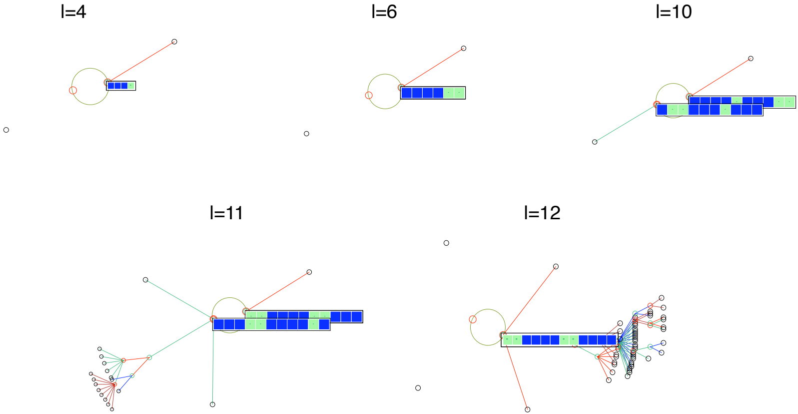


Figure: Generating gliders guns by multiple colliding gliders. Unlimited grown in $\phi_{R126maj:4}$.

Coding gliders by basin representation

Basins display attractors for $l = 4$, 6, 10, 11 and 12 respectively, where a glider s_1 evolve on each string. Basically we have that:

- for $l = 4$ the string $w = 1110$ produce s_1 gliders without intervals (second basin).
- for $l = 6$ the string $w = 111100$ produce the same s_1 gliders (second basin).
- for $l = 10$ the strings $w = 1110111101$ and $w = 0011100111$ produce two s_1 gliders and with two spaces between each glider (fifth basin). Also the strings $w = 1110111101$ and $w = 0011100111$ produce a s_1 glider with one space (fourth basin).
- for $l = 11$ the strings $w = 11100111101$ and $w = 00111100111$ produce a s_1 glider but with three spaces between them (third basin).
- for $l = 12$ the strings $w = 001111001111$ produce s_1 gliders without space (fourth basin), and the string $w = 111011101110$ produce the same s_1 glider (seventh basin).

Coding gliders since de Bruijn diagrams

Basically we will represent the s_1 gliders because this evolve in both ECA φ_{R126} and $\phi_{R126maj:4}$.

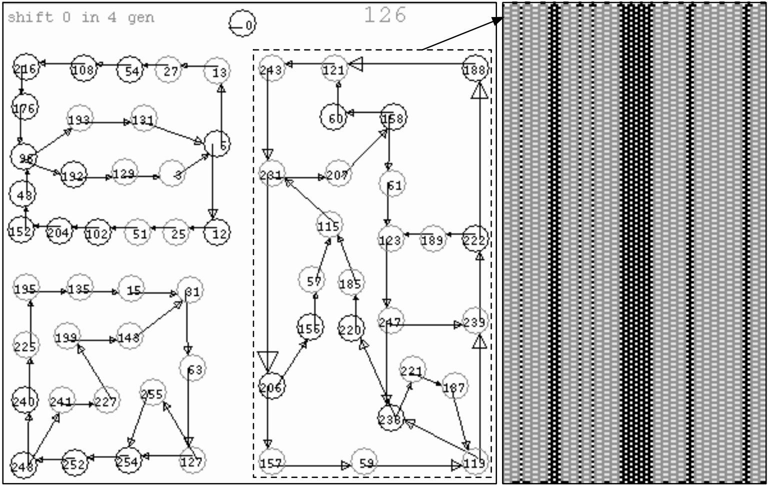


Figure: Cycles in the de Bruijn diagram and the corresponding periodic evolution for cycle $(0, 4)$.

Coding gliders since de Bruijn diagrams

- $w_1 = 1000$ - produce s_1 glider
- $w_2 = 10000$ - produce s_1 glider with one interval
- $w_3 = 000011$ - filter
- $w_4 = 011000$ - filter
- $w_5 = 010001100$ - produce $3s_1$ gliders with one interval
- $w_6 = 0010001100$ - produce $2s_1$ gliders with two intervals
- $w_7 = 0001000011$ - produce $2s_1$ gliders with two intervals
- $w_8 = 10000110000$ - produce s_1 glider with three intervals
- $w_9 = 00001000011$ - produce s_1 glider with three intervals

Thus we can construct any initial condition controlling s_1 gliders and intervals between them. For example, the expression $((w_3 w_7)^* + w_9)$ will code two spaces of b_1 with two s_1 gliders together finishing always with one s_1 glider. This way we can control and code easily gliders to solve problems based-collisions.

Summary of reactions in $\phi_{R126maj:4}$

Summary of collisions in ϕ_{R126}			
binary	multiple	soliton	guns
$s_1 \leftarrow g_2 = s_1$	$s_2 \leftarrow g_2 = 2g_1$	$g_1 \rightarrow s_1 = s_1 + g_1$	$s_1 \leftarrow 2g_2 = \text{gun}_1$
$g_1 \rightarrow s_1 = s_1$	$g_1 \rightarrow s_2 = 2g_2$	$s_1 \leftarrow g_2 = g_2 + s_1$	$\text{gun}_2 \leftarrow g_2 = \text{gun}_1$
$s_1 \leftarrow g_2 = g_1$	$s_2 \leftarrow 2g_2 = 2g_1$	$2g_1 \rightarrow s_2 = s_2 + 2g_1$	$g_1 \leftrightarrow g_2 \leftarrow 2g_2 = \text{gun}_2$
$g_1 \rightarrow s_1 = g_2$	$2g_1 \rightarrow s_2 = 2g_2$	$s_2 \leftarrow 2g_2 = 2g_2 + s_2$	$g_1 \leftrightarrow g_2 = \text{gun}_3$
$s_2 \leftarrow g_2 = g_1$	$s_2 \leftarrow 2g_2 = 2g_1 + g_2$	$2g_1 \rightarrow 2s_2 = 2g_1 + 2s_2$	$2g_1 \leftrightarrow 2g_2 = \text{gun}_2$ *
$g_1 \rightarrow s_2 = g_2$	$2g_1 \rightarrow s_2 = g_1 + 2g_2$	$2s_2 \leftarrow 2g_2 = 2g_2 + 2s_2$	$3g_1 \leftrightarrow 3g_2 = \text{gun}_1$ *
$g_1 \leftrightarrow g_2 = \emptyset$	$2g_1 \leftrightarrow 2g_2 = \emptyset$	$2g_1 \leftrightarrow 2g_2 = 2g_2 + 2g_1$	(* means gun composed)
$g_1 \leftrightarrow g_2 = s_1$	$2g_1 \leftrightarrow 2g_2 = g_1$		
$g_1 \leftrightarrow g_2 = s_2$	$2g_1 \leftrightarrow 2g_2 = g_2$		
$g_1 \leftrightarrow g_2 = g_1$	$2g_1 \leftrightarrow 2g_2 = 2g_1$		
$g_1 \leftrightarrow g_2 = g_2$	$2g_1 \leftrightarrow 2g_2 = 2g_2$		
$g_1 \leftrightarrow g_2 = 2g_1$	$g_1 \leftrightarrow 2g_2 = g_1$		
$g_1 \leftrightarrow g_2 = 2g_2$	$2g_1 \leftrightarrow g_2 = g_2$		
$g_1 \leftrightarrow g_2 = g_2 + 2g_1$	$g_1 \leftrightarrow 2g_2 = g_2$		
$g_1 \leftrightarrow g_2 = g_1 + 2g_2$	$2g_1 \leftrightarrow g_2 = g_1$		
	$g_1 \leftrightarrow 2g_2 = \emptyset$		
	$2g_1 \leftrightarrow g_2 = \emptyset$		
	$g_1 \leftrightarrow 2g_2 = 2g_2 + g_1$		
	$2g_1 \leftrightarrow g_2 = g_2 + 2g_1$		
	$g_1 \leftrightarrow 2g_2 = 2g_2 + 2g_1$		
	$g_1 \leftrightarrow 2g_2 = 2g_2 + g_2 + 2g_1$		
	$2g_1 \leftrightarrow g_2 = 2g_2 + 2g_1$		
	$2g_1 \leftrightarrow g_2 = 2g_2 + 2g_1 + g_1$		
	$3g_1 \leftrightarrow 3g_2 = 2g_1$		
	$g_1 \rightarrow s_1 \leftarrow g_2 = \emptyset$		
	$g_1 \rightarrow s_1 \leftarrow g_2 = s_1$		
	$g_1 \rightarrow 2s_2 \leftarrow g_2 = g_1 + g_2$		
	$g_1 \rightarrow 2s_2 \leftarrow g_2 = g_2 + g_1$		
	$2g_1 \rightarrow s_1 \leftarrow 2g_2 = 2g_2 + 2s_2 + 2g_1$		
	$3g_1 \leftrightarrow 3g_2 = g_2 + 2g_2 + 2g_1$		
	$4g_1 \leftrightarrow g_2 = 2g_1 + g_1$		
	$g_1 \leftrightarrow 4g_2 = g_2 + 2g_2$		

Figure: Table of binary, multiple and other collisions.

Implementing basic functions

Given an ample number of reactions in $\phi_{R126maj:4}$ the rule could be useful in implementing collision-based computing schemes. This figure illustrates the interaction of gliders traveling, colliding one with another and implementing a Boolean conjunction in the result of collision. Initially since previous collisions we can embed logical constructions of AND and NOT gates.

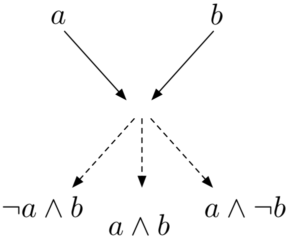


Figure: Colliding interactions deriving in logic gates.

Implementing basic functions

Considering than a glider g_1 represents value 0, two g_1 gliders together represent a value 1. Two gliders $2g_2$ traveling in positive direction will represent the operator and one the register. Thus the register will reads FALSE or TRUE if them become be produced successfully.

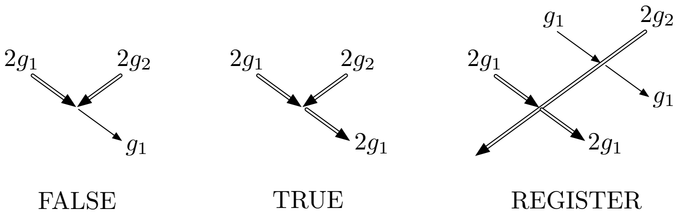


Figure: Colliding interactions deriving in logic gates.

Implementing basic functions

The basic reactions required to produce a primitive computational scheme in $\phi_{R126maj:4}$. The following set of relations is used (see table reactions):

$$2g_1 \leftrightarrow 2g_2 = \epsilon \quad g_1 \leftrightarrow 2g_2 = g_1 \quad g_1 \leftrightarrow 2g_2 = 2g_2 + g_1$$
$$2g_1 \leftrightarrow 2g_2 = g_1 \quad 2g_1 \leftrightarrow 2g_2 = 2g_2 + 2g_1$$

so we can represent serial reactions as:

$$\begin{array}{ll} 2g_1 + 2g_2 = \epsilon & \text{empty word} \\ 2g_1 + 2g_2 = g_1 & \text{FALSE} \\ 2g_1 + 2g_2 = 2g_1 & \text{TRUE.} \end{array}$$

A NOT gate can be represented as:

- FALSE $+ 2g_2 =$ TRUE $+ 2g_2$, and
- TRUE $+ 2g_2 =$ FALSE $+ 2g_2$.

Implementing basic functions

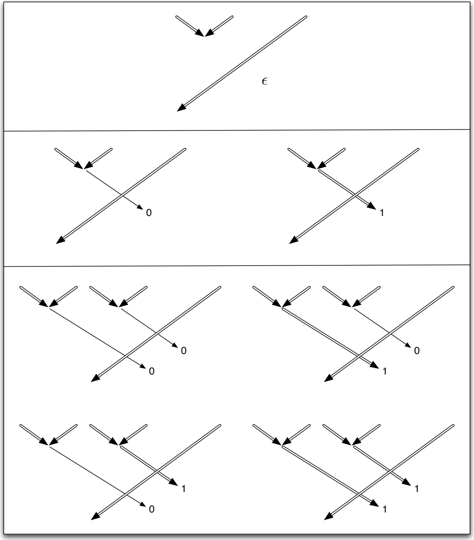


Figure: Constructing the formal languages Σ^0 (top), Σ^1 (middle), and Σ^2 (bottom) by glider reactions in $\phi_{majR126:4}$.

Final remarks

1. We have demonstrated that memory in ECA offers a new approach to discover complex dynamics based on particles and non-trivial reactions across them.
2. We have enriched some chaotic ECA rules with majority memory and demonstrated that by applying certain filtering procedures we can extract rich dynamics of travelling localizations, or particles.
3. Complex ECA with memory display promising applications to solve a diversity of problems.
4. Finally, the memory ϕ can be applied to any CA or dynamical system.

Final remarks

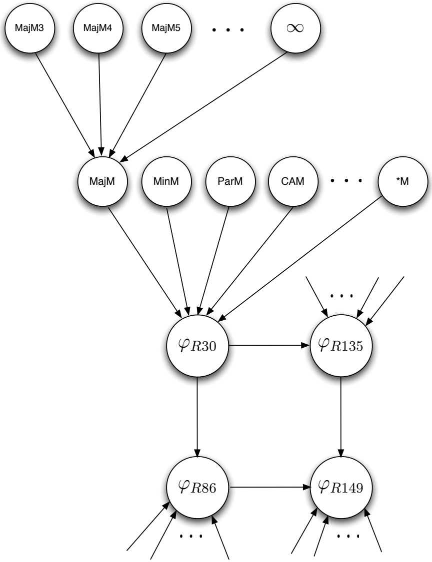


Figure: A new class of ECA with memory arising since classic ECA.

Final remarks

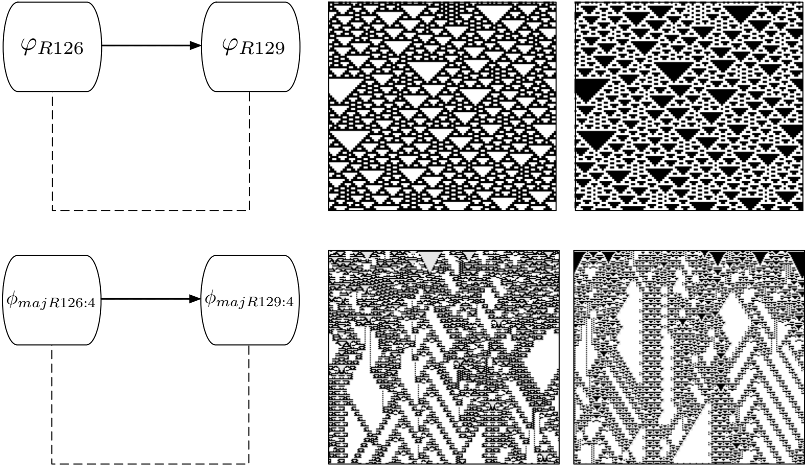


Figure: Inheritance by cluster classification [Wuensche92] but now with memory.

Thank you!

Andrew Wuensche and Mike Lesser
The Global Dynamics of Cellular Automata
Santa Fe Institute Studies in the Sciences of Complexity, 1992.

Stephen Wolfram
Cellular Automata and Complexity
Addison-Wesley Publishing Company, 1994.

Harold V. McIntosh
One Dimensional Cellular Automata
Luniver Press, 2009.

Ramon Alonso-Sanz
Elementary rules with elementary memory rules: the case of linear rules
Journal of Cellular Automata **1** 71–87, 2006.

Genaro J. Martínez, Andrew Adamatzky, Ramon Alonso-Sanz and J.C. Seck-Tuoh-Mora
Complex dynamic emerging in Rule 30 with majority memory
Complex Systems **18(3), 345-365**, 2010.

Genaro J. Martínez, Andrew Adamatzky, J.C. Seck-Tuoh-Mora, and Ramon Alonso-Sanz
How to make dull cellular automata complex by adding memory: Rule 126 case study
Complexity **15(6), 34-49**, 2010.