Computing on rings

Genaro J. Martínez[1], Andrew Adamatzky[1], Harold V. McIntosh[2]

June 21, 2012
Chapter of the book *A computable Universe: Understanding and Exploring Nature as Computation*, edited by H. Zenil and published for World Scientific Press.
`http://www.worldscibooks.com/compsci/8306.html`.

[1] Unconventional Computing Center
University of the West of England, Bristol BS16 1QY, United Kingdom.
`{genaro.martinez,andrew.adamatzky}@uwe.ac.uk`
[2] Departamento de Aplicación de Microcomputadoras, Instituto de Ciencias, Universidad Autónoma de Puebla, Puebla, México.
`mcintosh@unam.mx`

Abstract

In this paper, we will review the features of develop computations based on rings. Particularly, we will analyse what kinds of interaction occur between gliders travelling on a cellular automaton 'cyclotron' derived from a catalog of collisions. We will demonstrate that collisions between gliders emulate the basic types of interaction that occur between localizations in non-linear media: fusion, elastic collision, and soliton-like collision. Computational outcomes of a swarm of gliders circling on a one-dimensional torus are analysed via implementation some easy computing models. Gliders in one-dimensional cellular automata are compact groups of non-quiescent patterns translating along automaton lattice. They are cellular-automaton analogous of localizations or quasi-local collective excitations travelling in a spatially extended non-linear medium. So, they can be represented as binary strings or symbols travelling along a one-dimensional ring, interacting with each other and changing their states, or symbolic values, as a result of interactions. We present a number of complex one-dimensional cellular automata with such features.

Keywords: rings, cellular automata, particles, collisions, beam routing, unconventional computing, computability

1 Introduction

Computations as effective procedures were developed from the past century for logic mathematician, such as: Kurt Gödel, Alonzo Church, Alan Turing, Emil

Post, Stephen Kleene [20]. Actually, we can speak about of unconventional computing, they are abstract or practical models in non-linear media and consequently typically massively parallel, thus we can develop computations with different physics and other complementary logic. We have the conservative logic [21], reversible computing [14, 47], reaction-diffusion computers [6], Physarum computers [5], cellular automata computers [1, 24, 34, 51, 58, 38], collision-based computing such as optical or molecular computing [3], solitons or competing patterns computing [2, 32], or hot ice computers [4].

In this paper, we consider a particular case where computations can be carried by particle collisions with complex cellular automata (CA). Here we study one-dimensional CA and glider interaction on the evolution space. But confined naturally in one dimension, on rings. This way, gliders (or particles) can be represented as set of strings which may be characterised via de de Bruijn diagrams [41, 40]. Hence each glider is coded as a regular expression and initialised on a specific initial conditions. So computations are a consequence of glider interactions cyclical on such cyclotrons. We should differentiate that this study is not related precisely as: circular Turing machines [11], circular Post machines [29] or cyclic tag systems [18, 58] because in the case, sets of strings represent gliders and they shall be transformed to other set of strings plus periodic strings that are permanent in all time. We will illustrate such devices in complex elementary CA (ECA) [57, 58] and ECA with memory (ECAM) [9, 10].

2 One-dimensional cellular automata

2.1 Elementary cellular automata (ECA)

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 Σ. Thus, a sequence of cells $\{x_i\}$ of finite length n describes a string or *global configuration* c on Σ. This way, the set of finite configurations will be expressed as Σ^n. An evolution is comprised by a sequence of configurations $\{c_i\}$ produced by the mapping $\Phi : \Sigma^n \to \Sigma^n$; thus the global relation is symbolized as:

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

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 φ as follows:

$$\varphi(x^t_{i-r}, \ldots, x^t_i, \ldots, x^t_{i+r}) \to x^{t+1}_i. \tag{2}$$

Wolfram represents one-dimensional CA with two parameters (k, r) [57], where $k = |\Sigma|$ is the number of states, and r is the neighbourhood radius, hence ECA domain is defined by parameters $(2, 1)$. There are Σ^n different neighbourhoods (where $n = 2r + 1$) and k^{k^n} distinct evolution rules. The evolutions in this paper have periodic boundary conditions.

2.2 Elementary cellular automata with memory (ECAM)

Conventional CA are ahistoric (memoryless): i.e., the new state of a cell depends on the neighbourhood 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. CA with memory have been proposed originally in [7, 8, 9, 10]. Basically memory is based on the state and history of the system, thus we design a memory function ϕ, as follows:

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

such that $\tau < t$ determines the backwards 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:

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

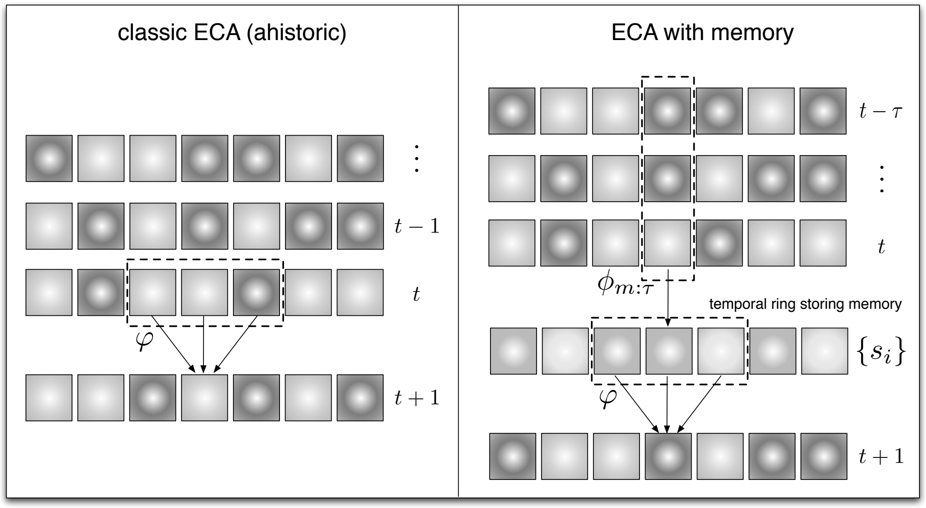


Figure 1: CA ahistoric and with memory in cells.

In CA with memory, while the mapping φ 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 φ. As an example, we can take the memory function ϕ as a *majority memory*:

$$\phi_{maj} \to s_i \tag{4}$$

where in case of a tie given by $\Sigma_1 = \Sigma_0$ in ϕ, we shall take the last value x_i. So ϕ_{maj} represents the classic majority function for three variables [46], as follows:

$$\phi_{maj} : (x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee (x_3 \wedge x_1) \rightarrow x$$

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 is allows to break it in favour of zero if $x_{\tau-1} = 0$, or to one whether $x_{\tau-1} = 1$. The representation of a ECA with memory (given previously in [31, 37, 33]) is given as follows:

$$\phi_{CARm:\tau} \tag{5}$$

where CAR represents the decimal notation of a particular ECA and m the kind of memory given with a specific value of τ. Thus the majority memory (maj) working in ECA Rule 126 checking tree cells ($\tau = 3$) of history is simply denoted as $\phi_{R126maj:3}$ [37]. Figure 1 depicts in detail the memory working on ECA. Note that memory is as simple as any CA local function and therefore preserve and increase its dynamics in new orders of complexity [31].

3 Strings in one-dimensional cellular automata

We will handle particles or gliders as sets of strings, this way each string represents a specific glider with some properties as well. The set of strings represents a subset of regular expressions. Hence we will use de Bruijn diagrams and tiling patterns to extract such strings.

3.1 Regular expressions

Several interesting problems arise in a study of formal languages; one of them is to determine the type of language derived and to which class the language belongs. This hierarchy is well-known and established by Chomsky's classification. We shall study languages determined by regular sets, since the set of expressions determined by each glider on a CA evolution rule can be associated to a particular regular expression.

The finite automaton is a mathematical model with a system of discrete inputs and outputs; the system can be placed in one of a finite set of states. This state has the information of the received inputs necessary to determine the behaviour of the system with regard of subsequent inputs. Formally, a finite automaton M consists of a finite set of states and a set of transitions among states induced by the symbols selected from some alphabet. For each symbol there is a transition from one state to other (it can return to the same one); there is an initial state where the automaton stars and some states are designated as final ones or acceptance states [25].

A directed graph called a *transition diagram* is associated with a finite automaton as follows: the vertices of the graph correspond to the states of the automaton; for a transition from state i to state j produced by an input symbol, there is an edge labeled by this symbol from i to j in the transition diagram.

The finite automaton accepts a chain w if the analogous transition sequence leads from the initial state to a final one (or acceptation).

A *language accepted* by M, represented by $L(M)$, it is the set $\{w|w$ is accepted by $M\}$. The type of languages accepted by a finite automaton is important because they complement the analysis established with regular expressions. Historically an important relation was established by Kleene demonstrating that regular expressions can be expressed by a finite automaton and vice versa, i.e., they are equivalent representations [46]. In other words, a language is a *regular set* if it is accepted by some finite automaton. The accepted languages by finite automata are described by expressions known as *regular expressions*; particularly, the accepted languages by finite automata are indeed the class of languages described by regular expressions.

The sets of *regular expressions* on an alphabet are defined recursively as [25]:

1. ϕ is the regular expression representing the empty set.
2. ϵ is the regular expression describing the set $\{\epsilon\}$.
3. For each symbol $a \in \Sigma$, a is a regular expression depicting the set $\{a\}$.
4. If a and b are regular expressions representing languages A and B respectively, then $a + b$, ab, and a^* are regular expressions representing $A \cup B$, AB and A^* respectively.

When it is necessary to distinguish between a regular expression a and the language determined by a, we shall use L_a.

The formal languages theory provides a way to study sets of chains from a finite alphabet. The languages can be seen as inputs of some classes of machines or like the final result from a typesetter substitution system i.e., a generative grammar into the Chomsky's classification [26].

language	structure
recursively enumerated	Turing machine
context sensitive	linear bounded automata
context free	pushdown automata
regular	finite automata

Table 1: Language classes.

The basic model necessary for the languages of these machines (and for all computation), is the Turing machine; the machines recognising each family of languages are described as a Turing machine with restrictions. The relevance of associating a machine or system to resolve each type of language is for establishing a classification (Table 1 of [26]).

Some languages are established by regular sets;[1] here we will take all the words recognised by the de Bruijn diagram that represent precisely a set of

[1]Examples and properties of the formal languages, grammars, finite state machines, Turing machines and equivalent systems can be consulted in [11, 25, 46, 19, 53].

regular expressions [41], and we just need those chains representing a structure on a specific complex CA evolution rule, to manipulate the evolution space with constructions of gliders.

3.2 de Bruijn diagrams

De Bruijn diagrams [40, 41] 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). We shall explain the de Bruijn diagrams illustrating their constructions for determining chains w defining a pair of gliders in $\mathcal{G}$, for complex CA.

For a one-dimensional CA of order (k, r), the de Bruijn diagram is defined as a directed graph with k^{2r} vertices and 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 neighbourhood of $2r + 1$ states represented by $i \diamond j$. In this case, the edge connecting i to j is labeled with $\varphi(i \diamond j)$ (the value of the neighbourhood defined by the local function) [54, 55].

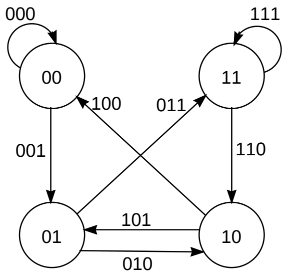


Figure 2: Generic de Bruijn diagram for a ECA (2,1).

The connection matrix M corresponding with the de Bruijn diagram is as follows:

$$M_{i,j} = \begin{cases} 1 & \text{if } j = ki, ki+1, \ldots, ki+k-1 \text{ (mod } k^{2r}) \\ 0 & \text{in other case} \end{cases} \qquad (6)$$

Module $k^{2r} = 2^2 = 4$ represent the number of vertices in the de Bruijn diagram and j must take values from $k * i = 2i$ to $(k * i) + k - 1 = (2 * i) + 2 - 1 = 2i + 1$. Vertices are labeled by fractions of neighbourhoods originated by 00, 01, 10, and 11, the overlap determines each connection. In Table 2 the intersections derived from the elements of each vertex are showed; they are the edges of the de Bruijn diagram as we can see in Figure 2.

The de Bruijn diagram has four vertices which can be renamed as $\{0, 1, 2, 3\}$ corresponding with the four partial neighbourhoods of two cells $\{00, 01, 10, 11\}$, and eight edges representing neighbourhoods of size $2r + 1$.

(0,**0**) ◇ (**0**,0)	0**0**0
(0,**0**) ◇ (**0**,1)	0**0**1
(0,**1**) ◇ (**1**,0)	0**1**0
(0,**1**) ◇ (**1**,1)	0**1**1
(1,**0**) ◇ (**0**,0)	1**0**0
(1,**0**) ◇ (**0**,1)	1**0**1
(1,**1**) ◇ (**1**,0)	1**1**0
(1,**1**) ◇ (**1**,1)	1**1**1

Table 2: Intersections determining the edges of the de Brujin diagram.

Paths in the de Bruijn diagram may represent chains, configurations or classes of configurations in the evolution space.

Vertices at the de Bruijn diagram are sequences of symbols in the set of states and the symbols are sequences of vertices in the diagram. The edges describe how such sequences can be overlapped; consequently, different intersection degrees yield distinct de Bruijn diagrams. Thus, the connection takes place between an initial symbol, the overlapping symbols and a terminal one (Table 2). Sequences derived from a de Bruijn diagram are the set of regular expressions that a CA can generate since its evolution rule [41].

Also, we have the extended de Bruijn diagrams[2] that calculate all the periodic sequences by the cycles defined in the diagram. These ones also calculate the shift of a periodic sequence for a certain number of steps; thus we can get de Bruijn diagrams describing all the periodic sequences characteriding a glider in any complex CA.

In order to illustrate how the sequences of each glider are determined, we calculate the de Bruijn diagram composing an *A* glider in Rule 110, and discussing how the periodic sequences are extracted for representing this glider and specifying as well the set of regular expressions for such glider.

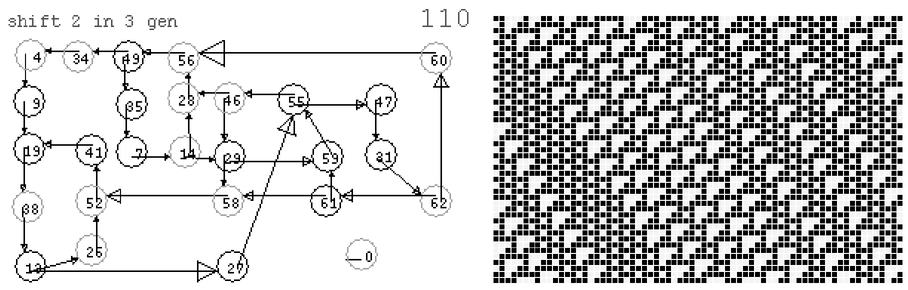


Figure 3: De Bruijn diagram calculating *A* gliders and ether configurations.

The *A* glider moves two cells to the right in three times (for details see

[2]The de Bruijn diagrams were calculated with the NXLCAU21 system developed by McIntosh for NextStep (OpenStep and LCAU21 to MsDos). Application and code source are available from: `http://delta.cs.cinvestav.mx/~mcintosh/cellularautomata/SOFTWARE.html`

[44]). We compute the extended de Bruijn diagram (2-shift, 3-gen) depicted in Figure 3. The cycles of the diagram have the periodic sequences describing the A glider; however, these sequences are not ordered yet. Therefore, we must determine and classify them.

In the figure we have two cycles: a cycle formed by vertex 0 and a large cycle of 26 vertices which is composed as well by 9 internal cycles. The evolution of the right illustrates the location of the different periodic sequences producing the A glider in distinct numbers.

Following the paths through the edges we obtain the sequences or regular expressions determining the phases of the A glider. For example, we have cycles formed by:

I. The expression (1110), vertices 29, 59, 55, 46 determining A^n gliders.

II. The expression (111110), vertices 61, 59, 55, 47, 31, 62 defining nA gliders with a T_3 tile[3] between each glider.

III. The expression (11111000100110), vertices 13, 27, 55, 47, 31, 62, 60, 56, 49, 34, 4, 9, 19, 38 describing ether configurations in a phase (in the following subsection we will see that it corresponds to the phase $e(\text{f}_1\_1)$).

The cycle with period 1 represented by vertex 0 produces a homogenous evolution with state 0. The evolution of the right (Figure 3) shows different packages of A gliders, the initial condition is constructed following some of the seven possible cycles of the de Bruijn diagram or several of them. We can select the number of A gliders or the number of intermediate ether configurations changing from one cycle to another.

Following each phase initiated by every T_3 tile, the phases f_i_1 for the A glider are as follows:

- $A(\text{f}_1\_1) = 111110$
- $A(\text{f}_2\_1) = 11111000111000100110$
- $A(\text{f}_3\_1) = 11111000100110100110$

The sequence is defined taking the first value from the first cell of T_3 tile on the left until reaching a second cell representing the first value of the second T_3 tile on the right. Finally, such set of strings correspond precisely to the sequences derived from the de Bruijn diagram concatenated with fragments of ether configuration. This way, the set of strings $e(\text{f}_1\_1)$, $A(\text{f}_i\_1)\ \forall\ i = \{1, 2, 3\}$ under the operations $+, \cdot, *$, they are regular expressions. The full set of stings to code gliders in Rule 110 $\mathcal{L}_{R110}$ is represented from the de Bruijn diagrams and tiles [44].[4] Tiles were necessary to represent strings in bigger gliders where the de Bruijn diagram is very hard to calculate.

[3]A tile T_3 is formed for the ether configuration in Rule 110, for details please see [44].

[4]The full set of regular expressions to code each glider in Rule 110, including the glider gun, is available from: `http://uncomp.uwe.ac.uk/genaro/rule110/listPhasesR110.txt`.

4 Universal CA and some CA computers

Universality in CA was conceptualised, developed and solved by von Neumann in [56] as a previous step in the specification of his universal constructor for a two-dimensional automaton of 29-states. The element of universality into this constructor is a necessary ingredient to handle non-reliable pieces (atomic elements) for assembling reliable components with the capacity of executing computations by collisions of signals. Actually such ideas speaking about of complex CA, as Life or Rule 110, is controlling gliders to make they reliable components to implement complex and very large CA engineering.

There are some significant simplifications in universal-computing CA with less states and dimensions: Codd in 1968 [17], Banks in 1971 [12], Smith in 1971 [50], Conway in 1982 [13], Lindgren and Nordahl in 1990 [27], and finally Cook in 1998 [18, 58].

Analogous to the search of a minimal universal Turing machine [48], Cook details the minimal universal CA with ECA Rule 110; showing how a "simple" elemental CA is able to simulate an equivalent Turing machine deriving a novel *cyclic tag system* (CTS), performing computations [18, 2] across of collisions of gliders in millions of cells [45].[5]

4.1 Toffoli's symbology

In the late 1970s Fredkin and Toffoli developed a concept of a general-purpose computation based on ballistic interactions between quanta of information that are represented by abstract particles [52]. The Boolean states of logical variables are represented by balls or atoms, which preserve their identity when they collide with each other. They came up with the idea of a billiard-ball model of computation, with underpinning mechanics of elastically colliding balls and mirrors reflecting the balls' trajectories. Later Margolus developed a special class of CA which implement the billiard-ball model. Margolus' partitioned CA exhibited computational universality because they simulated Fredkin gate via collision of soft spheres [36].

The following basic functions with two input arguments u and v can be expressed via collision between two localizations:

1. $f(u, v) = c$, fusion
2. $f(u, v) = u + v$, interaction and subsequent change of state
3. $f_i(u, v) \mapsto (u, v)$ identity, solitonic collision
4. $f_r(u, v) \mapsto (v, u)$ reflection, elastic collision

To map Toffoli's supercollider [52] onto a one-dimensional CA we use the notion of an idealized particle $p \in \Sigma^+$ (without energy and potential energy).

[5] A brief list of CA computers is available from: `http://uncomp.uwe.ac.uk/genaro/otherRules.html`.

For our study, the particle p is represented by a binary string of cell states derived from a de Bruijn diagram.

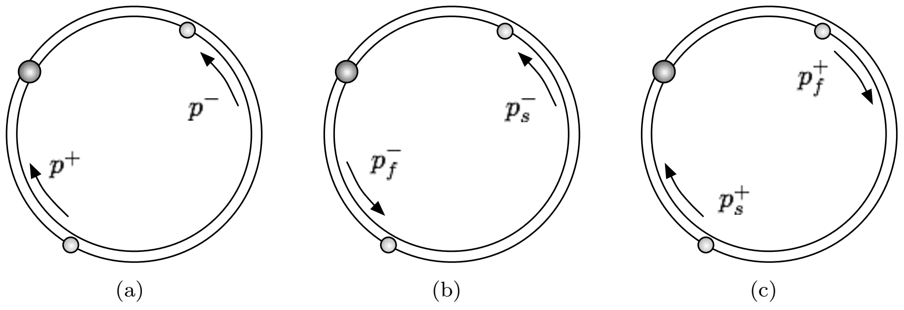


Figure 4: Representation of abstract particles in a one-dimensional CA ring beam routing.

Figure 4 shows two typical scenarios where particles p_f and p_S travel in a CA cyclotron. The first scenario (Fig. 4a) shows two particles travelling in opposite directions which then collide. Their collision site is shown by a dark circle as a contact point in (Fig. 4). The second scenario demonstrates a typical beam routing where a fast particle p_f eventually catches up with a slow particle p_s at a collision site (Fig. 4b). If the particles collide like solitons [28], then the faster particle p_f simply overtakes the slower particle p_s and continues its motion (Fig. 4c).

4.2 CA as cyclotrons

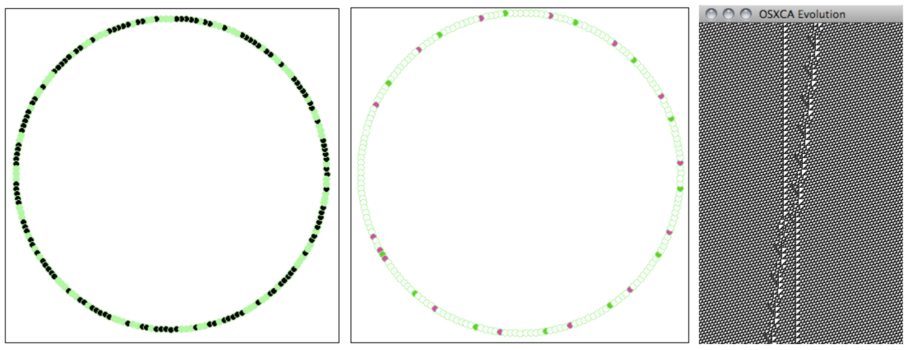
Typically, we can find all types of particles manifest in CA gliders, including positive p^+, negative p^-, and patterns with neutral p^0[6] displacements [42], and also composite particles assembled from elementary localizations. Let us consider the case where a quiescent state is substituted by cells synchronized together as an ether (periodic background). This phenomenon is associated with ECA Rule 110 φ_{R110}.[7] Its evolution space is dominated by a number of particles emerging in various different orders, some of which are really quite complex constructions. Consequently, the number of collisions between particles is increased. Each particle has a period, displacement, velocity, mass, volume, and phase [42, 44].[8]

Figure 5 displays a one-dimensional configuration where two particles collide repeatedly and interact as solitons so that the identities of the particles are preserved in the collisions. A negative particle p_F^- collides and overtakes a

[6]Neutral displacement of complex structures in one dimension is related directly to *still life* configurations [13] in two-dimensions.

[7]Rule 110 repository `http://uncomp.uwe.ac.uk/genaro/Rule110.html`

[8]A full description of particles in Rule 110 is available at `http://uncomp.uwe.ac.uk/genaro/rule110/glidersRule110.html`



(a) (b) (c)

Figure 5: Example of a soliton-type interaction between particles in ECA Rule 110: (a)–(b) two steps of beam routing, (c) exact configuration at the time of collision.

neutral particle $p_{C_1}^{-}$. Figure 5a presents a whole set of cells in state 1 (dark points) where the ether configuration makes it impossible to distinguish the particles: p_F^{-} and $p_{C_1}^{-}$. However, we can apply a filter and thereby select particles from their background ether (Fig. 5b).[9] Space-time configurations of a cellular automaton exhibiting a collision between particles p_F^{-} and $p_{C_1}^{-}$ are shown in Fig. 5c.

Filters selected in CA are a useful tool for understand "hidden" properties of CA. This tool was developed by Wuensche in the context of automatic classification of CA [59]. The filters were derived from mechanical computation techniques [23], pattern recognition [49], and analysis of cell-state frequencies [59]. Thus, a filter is a sequence of cells that have a high frequency in the evolution space. Such d-dimensional string repeat periodically, coexisting with any complex structure without altering or disturbing the global dynamics.

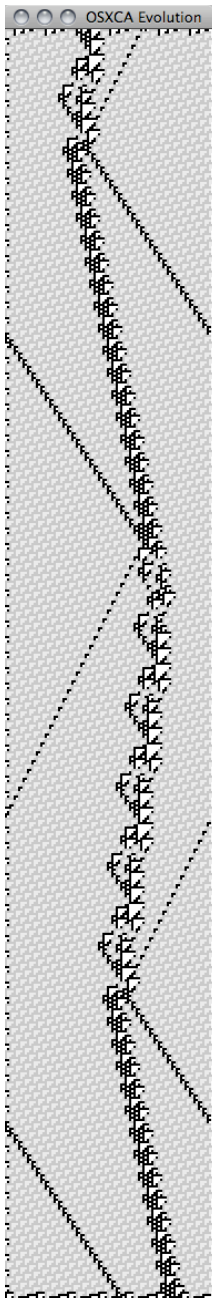
4.3 Extending Toffoli's symbology

While Toffoli considers a particle travelling to collide with other particle [52], here we will consider packages of gliders colliding simultaneously or in series. Hence, we can develop finite machines where the vertex are not just states, they are sets of strings that represent particles turning inside cyclotrons. Consequently, the transition on these meta-vertices mean a change of such cyclotrons given for the sum of their collisions [38].

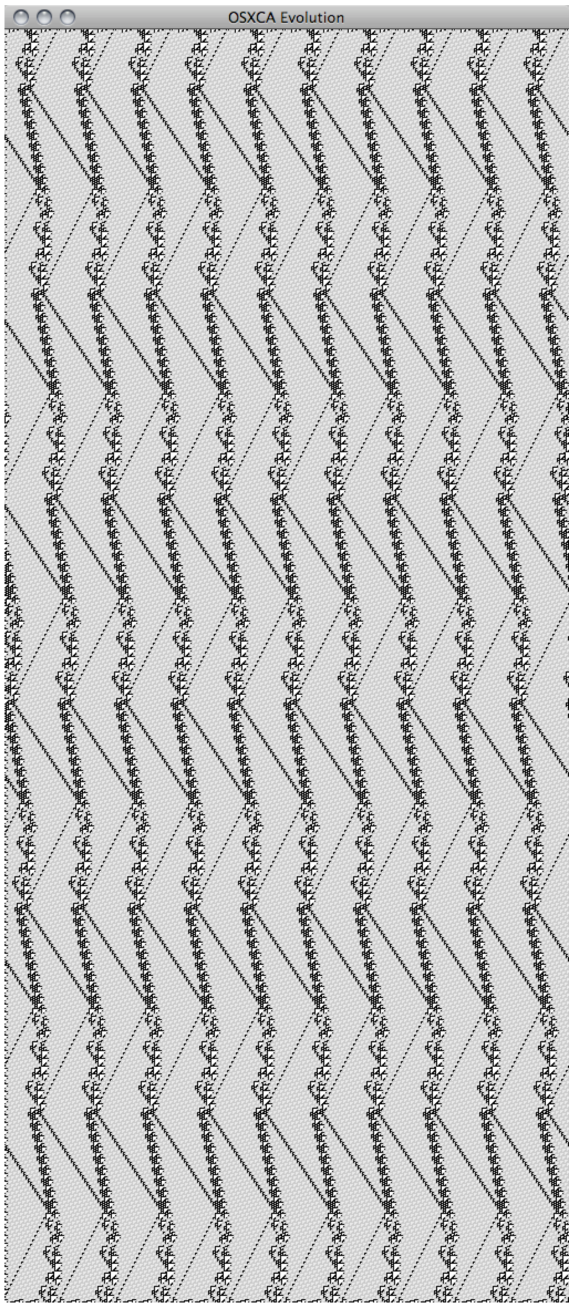
Let see an easy sample with multiple collisions. We have collisions that can be represented as cycles of collisions. In this case, we have the next relations:

1. $p_F \leftarrow p_B = p_{D_1} + p_{A^2}$

[9]Ring evolution is simulated with DDLab [60] available in `http://www.ddlab.org`.



(a)



(b)

Figure 6: Collisions between particles $p_F \leftarrow p_B$ (a) evolving with periodic boundaries to 93 cells in 573 generations, (b) internal structure of collisions without periodic boundaries evolving in 510 cells in 1,150 generations.

2. $p_{A^2} \leftarrow p_{D_1} = p_B + p_F$

given the cycle, hence we can construct this one across a cyclotron representation.

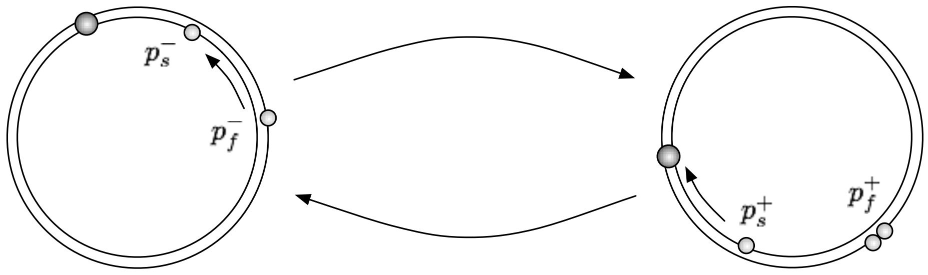


Figure 7: Beam routing as cyclotrons working as a finite machine. In both cases fast particles reach to slow particles.

The cycle relates two cyclotrons, which when connected represent a simple state machine. This machine has two meta-vertices (states) and the transition is determined by a contact point where gliders collide, as we can see in Fig. 7.

Figure 6 illustrate two kinds of evolutions where gliders in Rule 110 have periodic reactions. First evolution (Fig. 6a) has an initial condition with 93 cells with the next regular expression (here the symbol $-$ means the concatenation operation):

$$e - F(\mathrm{H,f_1\_1}) - e - B(\mathrm{f_1\_1}) - e$$

the evolution has boundaries conditions and one exact distance preserve periodic collisions between such particles evolving in 573 generations. However, without boundaries conditions this periodic reaction can be preserved as the evolution (Fig. 6b). The multiple collisions are synchronised to construct a meta glider [43]. The regular expression to reproduce such global behaviour (as an infinite string) is following:

$$(F(\mathrm{H,f_1\_1}) - e - B(\mathrm{f_1\_1}) - e)^*,$$

the evolution coded in an initial condition with 510 cells evolving in 1,150 generations. Therefore, the cyclotron representation helps to synthesise such dynamics and construct specific finite machine to represent operations between them.

4.4 Implementing simple functions in rings

We can employ the particles codification to represent solutions of some basic computing functions. Let us consider a ECAM rule $\phi_{R30maj:8}$ [31, 33]. We want to implement a simple substitution function addToHead working on two strings $w_1 = A_1, \ldots, A_n$ and $w_2 = B_1, \ldots, B_m$, where $n, m \geq 1$. For example,

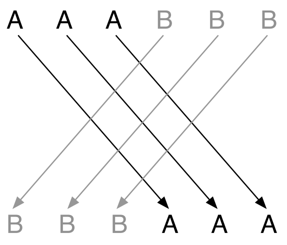


Figure 8: Schematic diagram adding the string w_2 to head of the list w_3.

if $w_1 = AAA$, $w_2 = BBB$ and $w_3 = w_1 w_2$ then the addToHead($|w_2|$) will yield: $w_3 = w_2 w_1$ or $w_3 = BBBAAA$ (see schematic diagram of Fig. 8).

To implement such function in $\phi_{R30maj:8}$ we must represent every data 'quantum' as a particle. Gliders g_1 and g_2 are coded to reproduce a soliton reaction. Another problem is to synchronise several gliders and obtain the same result with multiple collisions.

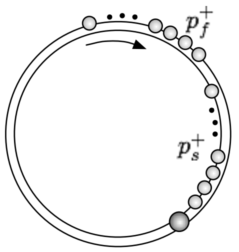


Figure 9: Beam routing performing identity reactions in $\phi_{R30maj:8}$. A cycle realises its operation, two cycles re-initialise the beam state and the operations can then be repeated.

The codification is not sophisticated however a systematic analysis of reactions is required. It is known [31, 33] than a periodic gap and one fixed phase between particles is sufficient to reproduce the addToHead function for any string $A^n B^m$.

4.5 Full computable systems in complex ECA

Late of a detailed study about of the universality in Rule 110 [45], we can see that this cyclic tag system (CTS) [18, 58] working in Rule 110 can be expressed based on cycles of collisions self. Thus we can do cyclotrons or rings of particles that represent the periodic packages of gliders to implement a CTS in Rule 110 evolution space. This results is explained in details at the paper *Cellular Automaton Supercolliders*, please see [38].

5 Final remarks

This paper display a compact review of previous results about of some complex ECA and study of gliders in ECA through the de Bruijn diagrams and regular languages. Particularly, focused in ECA Rule 110 and some ECAM. Hence following the Toffoli's symbology to idealise supercolliders in CA, we proof how these cyclotrons can be designed to implement computations as rings of strings.

Acknowledgement

Genaro J. Martínez thanks to support given by DGAPA-UNAM and EPSRC grant EP/F054343/1.

References

[1] Adamatzky, A. (2001) *Computing in Nonlinear Media and Automata Collectives*, Institute of Physics Publishing, Bristol and Philadelphia.

[2] Adamatzky, A. (Ed.) (2002) *Collision-Based Computing*, Springer.

[3] Adamatzky, A. (2002) "New media for collision-based computing," In [2], 411–442.

[4] Adamatzky, A. (2009) "Hot Ice Computers," *Physics Letters A* **374** 264–271.

[5] Adamatzky, A. (2010) *Physarum Machines: Computers from Slime Mould*, World Scientific Series on Nonlinear Science, Series A.

[6] Adamatzky, A., Costello, B. L., and Asai, T. (2005) *Reaction-Diffusion Computers*, Elsevier.

[7] Alonso-Sanz, R. & Martin, M. (2003) "Elementary CA with memory," *Complex Systems* **14** 99–126.

[8] Alonso-Sanz, R. (2006) "Elementary rules with elementary memory rules: the case of linear rules," *Journal of Cellular Automata* **1** 71–87.

[9] Alonso-Sanz, R. (2009) *Cellular Automata with Memory*, Old City Publishing.

[10] Alonso-Sanz, R. (2011) *Discrete Systems with Memory*, World Scientific Series on Nonlinear Science, Series A.

[11] Arbib, M. A. (1969) *Theories of Abstract Automata*, Prentice-Hall Series in Automatic Computation.

[12] Banks, E. R. (1971) Information and transmission in cellular automata, *PhD Dissertion*, Cambridge, MA, MIT.

[13] Berlekamp, E. R., Conway, J. H., & Guy, R. K. (1982) *Winning Ways for your Mathematical Plays*, Academic Press, vol. 2, chapter 25.

[14] Bennett, C. H. (1973) "Logical reversibility of computation," *IBM Journal of Research and Development* **17(6)** 525–532.

[15] Boccara, N., Nasser, J., & Roger, M. (1991) "Particle like structures and their interactions in spatio-temporal patterns generated by one-dimensional deterministic cellular automaton rules," *Physical Review A* **44(2)** 866–875.

[16] Chopard, B. & Droz, M. (1998) *Cellular Automata Modeling of Physical Systems*, Collection Aléa Saclay, Cambridge University Press.

[17] Codd, E. F. (1968) *Cellular Automata*, Academic Press, Inc. New York and London.

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

[19] Davis, M. (1982). *Computability and Unsolvability*, Dover Publications, Inc. New York.

[20] Davis, M. (2000) *The Universal Computing: the road from Leibniz to Turing*, W. W. Norton & Company, Inc.

[21] Fredkin, E. & Toffoli, T. (1982) Conservative logic, *Int. J. Theoret. Phys.* **21** 219–253.

[22] Fredkin, E. & Toffoli, T. (2001) "Design Principles for Achieving High-Performance Submicron Digital Technologies," In [2], 27–46.

[23] Hanson, J. E. & Crutchfield, J. P. (1997) "Computacional Mechanics of Cellular Automata: An Example," *Physics D* **103** 169–189.

[24] Hey, A. J. G. (1998) *Feynman and computation: exploring the limits of computers*, Perseus Books.

[25] Hopcroft, J. E. & Ullman, J. D. (1987) *Introduction to Automata Theory Languages, and Computation*, Addison-Wesley Publishing Company.

[26] Hurd, L. P. (1987). Formal Language Characterizations of Cellular Automaton Limit Sets, *Complex Systems* **1** 69–80.

[27] Lindgren, K. & Nordahl, M. G. (1990) Universal Computation in Simple One-Dimensional Cellular Automata, *Complex Systems* **4** 229–318.

[28] Jakubowski, M. H., Steiglitz, K., & Squier, R. (2001) "Computing with Solitons: A Review and Prospectus," *Multiple-Valued Logic* **6(5-6)** 439–462.

[29] Kudlek, M. & Rogozhin, Y. (2001) Small Universal Circular Post Machine, *Computer Science Journal of Moldova* **9(25)** 34–52.

[30] Lindgren, K. & Nordahl M. (1990) Universal Computation in Simple One-Dimensional Cellular Automata, *Complex Systems* **4** 229–318.

[31] Martínez, G. J., Adamatzky, A., Alonso-Sanz, R., & Seck-Tuoh-Mora, J. C. (2010) "Complex dynamic emerging in Rule 30 with majority memory," *Complex Systems* **18(3)** 345–365.

[32] Martínez, G. J., Adamatzky, A., Morita, K., & Margenstern, M. (2010) "Computation with competing patterns in life-like automaton." In: *Game of Life Cellular Automata*, A. Adamatzky (Ed.) Springer, United Kingdom, pp. 547–572, chapter 27.

[33] Martínez, Genaro J., Adamatzky, Andrew, and Alonso-Sanz, Ramon, "Complex dynamics of elementary cellular automata emerging in chaotic rule," *International Journal of Bifurcation and Chaos*, by publish.

[34] Margolus, N. (1984) "Physics-like models of computation," *Physica D* **10(1-2)** 81–95.

[35] Margolus, N. (1999) "Crystalline computation," In [24], 267–305.

[36] Margolus, N. (2003) "Universal Cellular Automata Based on the Collisions of Soft Spheres," In [1], 231–260,.

[37] Martínez, G. J., Adamatzky, A., Seck-Tuoh-Mora, J. C., & Alonso-Sanz, R. (2010) "How to make dull cellular automata complex by adding memory: Rule 126 case study," *Complexity* **15(6)** 34–49.

[38] Martínez, G. J., Adamatzky, A., & Stephens, C. R. (2011) "Cellular automaton supercolliders," *International Journal of Modern Physics C* **22(4)**, 419–439.

[39] Martínez, G. J. & McIntosh, H. V. (2001) "ATLAS: Collisions of gliders like phases of ether in rule 110," `http://uncomp.uwe.ac.uk/genaro/Papers/Papers_on_CA.html`.

[40] McIntosh, H. V. (1991) Linear cellular automata via de Bruijn diagrams, `http://delta.cs.cinvestav.mx/~mcintosh/cellularautomata/Papers.html`.

[41] McIntosh, H. V. (2009) *One Dimensional Cellular Automata*, Luniver Press.

[42] Martínez, G. J., McIntosh, H. V., & Seck-Tuoh-Mora, J. C. (2006) "Gliders in Rule 110," *Int. J. of Unconventional Computing* **2(1)** 1–49.

[43] Martínez, G. J., McIntosh, H. V., Seck-Tuoh-Mora, J. C., & Chapa-Vergara, S. V. (2007) "Rule 110 objects and other constructions based-collisions," *Journal of Cellular Automata* **2(3)** 219–242.

[44] Martínez, G. J., McIntosh, H. V., Seck-Tuoh-Mora, J. C., & Chapa-Vergara, S. V. (2008) "Determining a regular language by glider-based structures called *phases* f_i*_1* in Rule 110," *Journal of Cellular Automata* **3(3)** 231–270.

[45] Martínez, G. J., McIntosh, H. V., Seck-Tuoh-Mora, J. C., & Chapa-Vergara, S. V. (2011) "Reproducing the cyclic tag system developed by Matthew Cook with Rule 110 using the phases f_1_1," *Journal of Cellular Automata* **6(2-3)** 121–161.

[46] Minsky, M. (1967) *Computation: Finite and Infinite Machines*, Prentice Hall.

[47] Morita, K. (2008) "Reversible computing and cellular automata—A survey", *Theoretical Computer Science* **395** 101–131.

[48] Shannon, C. E. (1956) A Universal Turing Machine with Two Internal States, *Automata Studies*, Princeton University Press, pp. 157–165.

[49] Shalizi, C. R., Haslinger, R., Rouquier, J-B. K., Kristina L., & Moore, C. (2005) "Automatic filters for the detection of coherent structure in spatiotemporal systems", *Physical Review E* **73(3)** 036104.

[50] Smith III, A. R. (1971) Simple computation-universal cellular spaces, *J. of the Assoc. for Computing Machinery* **18** 339–353.

[51] Toffoli, T. (1998) "Non-Conventional Computers", In *Encyclopedia of Electrical and Electronics Engineering* **14** (John Webster Ed.), Wiley & Sons, 455–471.

[52] Toffoli, T. (2002) "Symbol Super Colliders," In [2], 1–22.

[53] Turing, A. (1936) On Computable numbers, with an application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society* **42(2)** 230–265. 1937 Corrections, Ibid **43** 544–546.

[54] Voorhees, B. H. (1996) *Computational analysis of one-dimensional cellular automata*, World Scientific Series on Nonlinear Science, Series A, Vol. 15.

[55] Voorhees, B. H. (2008) Remarks on Applications of De Bruijn Diagrams and Their Fragments, *Journal of Cellular Automata* **3(3)** 187–204.

[56] von Neumann, J. (1966) *Theory of Self-reproducing Automata* (edited and completed by A. W. Burks), University of Illinois Press, Urbana and London.

[57] Wolfram, S. (1994) *Cellular Automata and Complexity*, Addison-Wesley Publishing Company.

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

[59] Wuensche, A. (1999) “Classifying Cellular Automata Automatically,” *Complexity* **4(3)** 47–66.

[60] Wuensche, A. (2011) *Exploring Discrete Dynamics*, Luniver Press.