Reproducing the cyclic tag system developed by Matthew Cook with Rule 110 using the phases f_i_1

Genaro J. Martínez[1], Harold V. McIntosh[2], Juan C. Seck Tuoh Mora[3], and Sergio V. Chapa Vergara[4]

[1] Faculty of Computing, Engineering and Mathematical Sciences, University of the West of England, Bristol, United Kingdom.
http://uncomp.uwe.ac.uk/genaro/
Email: genaro.martinez@uwe.ac.uk

[2] Departamento de Aplicación de Microcomputadoras, Instituto de Ciencias, Universidad Autónoma de Puebla, Puebla, México.
http://delta.cs.cinvestav.mx/~mcintosh/
Email: mcintosh@servidor.unam.mx

[3] Centro de Investigación Avanzada en Ingeniería Industrial, Universidad Autónoma del Estado de Hidalgo Pachuca, Hidalgo, México.
Email: jseck@uaeh.edu.mx

[4] Departamento de Computación, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, México.
Email: schapa@cs.cinvestav.mx

Abstract. This paper implements the cyclic tag system (CTS) in Rule 110 developed by Cook in [1, 2] using regular expressions denominated *phases* f_i_1 [3]. The main problem in CTS is coding the initial condition based in a system of gliders. In this way, we develop a method to control the periodic phases of the strings representing all gliders until now known in Rule 110, including glider guns. These strings form a subset of regular expressions implemented in a computational system to facilitate the construction of CTS. Thus, these phases are useful to establish distances and positions for every glider and then to delineate more sophisticated *components* or packages of gliders. In this manuscript, it is possible to find differences with the results exposed in Wolfram's book [2], inclusively some mistakes which avoid to obtain an appropriated realization of CTS in Rule 110; fortunately, these irregularities were discussed and clarified by Cook.[5]

[5] M. Cook November 2002, personal communication.

Keywords: Rule 110, cyclic tag system, gliders, phases and collisions

1 Introduction

The cellular automaton (CA) Rule 110 has received special attention by the results exposed by Cook in [4]. This work gives a list and a brief introduction about the complex activity of gliders, including the existence of a glider gun; discussing as well some similarities between the Game of Life (GoL) [5][6] and Rule 110, suggesting to call it as *LeftLife*.

Some historical facts in the research of Rule 110 are as follows: this CA evolves in one dimension with two states as alphabet, where the local function takes just three elements (left-centre-right) to determine the evolution over time. One of the first investigations about Rule 110 was described by Wolfram [6], discovering that Rule 110 displays complex behaviors by means of the existence of gliders – a glider is a periodic structure moving into the evolution space – from random initial conditions. Thus Wolfram establishes the conjecture that this rule could perform universal computation.

Lind presents the first classification of gliders in Rule 110 in [6] with 13 gliders.[7] Next the first paper dedicated to the analysis of Rule 110 is made by Li and Nordahl in [7], where a statistical study and some of the most common behaviors of Rule 110 are considered.

In 1998 a conference at the Santa Fe Institute takes place where Cook explains how Rule 110 is able to be universal; this talk is originally planned to be published in "New Constructions in Cellular Automata" [8, 38]. On the other hand, another perspective is reported by McIntosh in [9], analyzing Rule 110 as a problem of tiles and applying de Bruijn diagrams for characterizing every glider.

In this way, in March 2002 Wolfram presents his book "A New Kind of Science" [2]. The book explains in several pages the features of the gliders and the functionality of a CTS to demonstrate that Rule 110 is an elemental universal CA. But one limitation in this work is that it does not show a way to reproduce the result without the use of a proprietary software.

In this sense, on trying to reproduce the operation of the CTS, it has been found that the information proportioned in [2] is both incomplete and insufficient. Also there are mistakes in some of their elements

[6] `http://www.pentadecathlon.com/`

[7] Appendix, table 15. Also available in `http://www.stephenwolfram.com/publications/articles/ca/86-caappendix/16/text.html`

and others do not have the necessary components to produce a good functionality.

With this background, the aim of this paper is to present a general introduction to Rule 110, its system of gliders and their relation with tiles, the existence of phases expressed by regular expressions via de Bruijn diagrams and the CTS, all together to form a general context. Finally, we discuss and illustrate a detailed coding for the more relevant stages of the machine working into the evolution space of Rule 110.

2 Rule 110

Rule 110 is the binary local function of a CA in a one-dimensional order ($k = 2, r = 1$) in Wolfram's nomenclature [6], where k represent the cardinality of the set of states and r the number of neighbors to the left and right with regard of a central cell. In one dimension we have a finite array of cells with periodic boundary properties, where the first and last cells are concatenated to preserve symmetries into the evolution space. Then every configuration is an instance of each array and it is updated applying the local function simultaneously over all the neighborhoods to generate the next configuration. The local function φ is defined in Table 1.

Table 1. Local function for Rule 110 - $(01110110)_2$.

$$\begin{array}{ll} \varphi(1,1,1) \to 0 & \varphi(0,1,1) \to 1 \\ \varphi(1,1,0) \to 1 & \varphi(0,1,0) \to 1 \\ \varphi(1,0,1) \to 1 & \varphi(0,0,1) \to 1 \\ \varphi(1,0,0) \to 0 & \varphi(0,0,0) \to 0 \end{array}$$

Figure 1 shows a typical evolution of Rule 110 generated from a random initial condition. The evolution diagram starts from an initial condition of 761 cells with a density of 53% in state one, time advances up-down in 349 generations.

Figure 1 shows a typical environment produced by Rule 110. We can note a uniform compound state or periodic background marked with other colors called "ether" by Cook [4]. This ether is a difference from the two-dimensional Conway's GoL CA, which has a fixed background. Also, other periodic regions are identified by structures moving through the evolution space without changing their form, they are better known as "gliders." This word was originally adopted in GoL [5]. The chaotic regions are produced from the initial configuration appearing for some

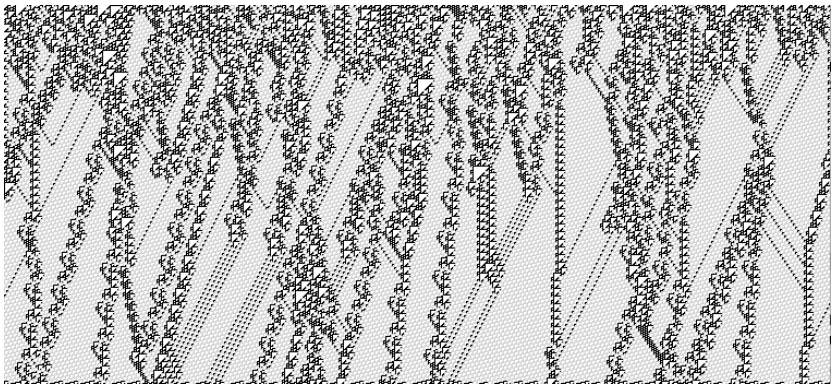

Fig. 1. Typical random evolution in Rule 110.

time, in other cases they are yielded by collisions among gliders. Thus we can look a non-trivial behavior in Rule 110.

An interesting point is that we can produce each glider from simple or multiple collisions of other gliders as it is presented in [10, 11]. Some collisions include complicated extensions or package of them. Now we shall characterize every glider or periodic pattern into the evolution space of Rule 110.

2.1 System of gliders

An extended description of gliders may be consulted in [9, 11, 12].[8] This section displays general properties and illustrative examples using the classification exposed by Cook [1].

Gliders in Rule 110 has a wide variety of kinds, extensions and combinations. We can handle groups or packages of them in one or several phases; thus notation nA means n copies of the A glider and not a package of A^n gliders. Figure 2 lists all known gliders so far both in their basic representation and in packages or extensions as well.

These gliders have important characteristic useful to define distances, slopes, speeds, periods, collisions, and phases [3, 12, 11].

Table 2 summarizes some of the most relevant properties, column *structure* gives the name of each glider including two more structures: e_r and e_l which represent the slopes of ether pattern. The next four columns

[8] The next web site has several examples where one can see large pictures of single or packages of gliders. `http://uncomp.uwe.ac.uk/genaro/rule110/glidersRule110.html`

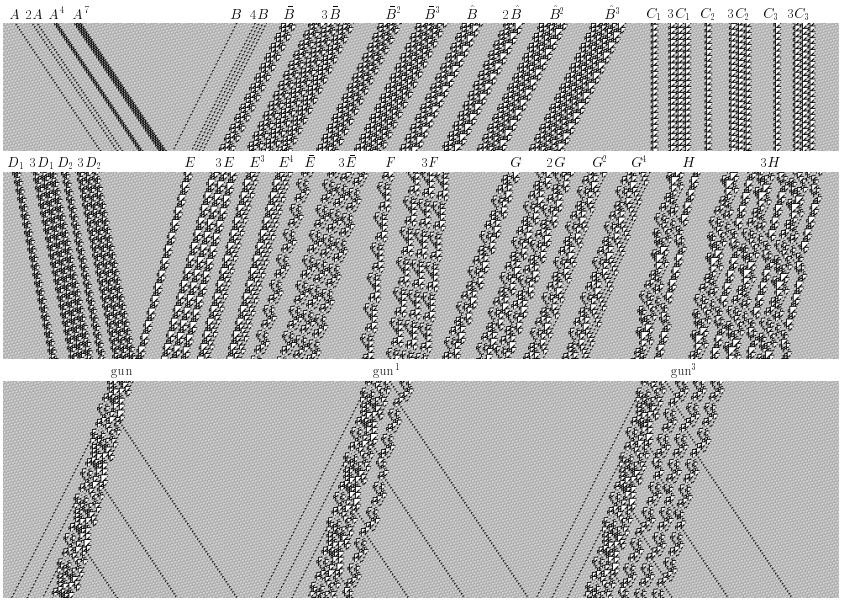

Fig. 2. System of gliders in Rule 110.

labeled *margins* indicate the number of periodic margins in each glider. The margins are partitioned in two types with even values '*ems*' and odd values '*oms*' which are distributed as well in two groups: left and right margins, because every glider has even and odd number of margins in their left or right borders (or superior and inferior ones). Particularly, the margin properties are very related to their analysis based on tiles and phases [3, 11]

Column v_g indicates the speed of each glider, where g belongs to a glider of the set of gliders $\mathcal{G}$. Speed is calculated dividing the displacement of d cells between its period p. The three types of trajectories are also indicated in this column, this way we have three different basic speeds. First there is a *positive speed* indicating a shift to the right. Second there is a *negative speed* with a shift to the left and the last one is a *zero speed* because in this particular case the differences between the first and the second speed avoid a shift.

The speed of gliders allows to control distances to get desired reactions. In general, larger gliders imply slower shifts; and any glider cannot be faster than v_{e_r} or v_{e_l} in their positive or negative speed respectively.

Column *lineal volume* describes the minimum and maximum number of necessary cells for determining the corresponding glider or another

Table 2. Properties of each glider in Rule 110.

structure	margins left - right				v_g	lineal volume
	ems	*oms*	*ems*	*oms*		
e_r	.	1	.	1	$2/3 \approx 0.666666$	14
e_l	1	.	1	.	$-1/2 = -0.5$	14
A	.	1	.	1	$2/3 \approx 0.666666$	6
B	1	.	1	.	$-2/4 = -0.5$	8
$\bar{B}^n$	3	.	3	.	$-6/12 = -0.5$	22
$\hat{B}^n$	3	.	3	.	$-6/12 = -0.5$	39
C_1	1	1	1	1	$0/7 = 0$	9-23
C_2	1	1	1	1	$0/7 = 0$	17
C_3	1	1	1	1	$0/7 = 0$	11
D_1	1	2	1	2	$2/10 = 0.2$	11-25
D_2	1	2	1	2	$2/10 = 0.2$	19
E^n	3	1	3	1	$-4/15 \approx -0.266666$	19
$\bar{E}$	6	2	6	2	$-8/30 \approx -0.266666$	21
F	6	4	6	4	$-4/36 \approx -0.111111$	15-29
G^n	9	2	9	2	$-14/42 \approx -0.333333$	24-38
H	17	8	17	8	$-18/92 \approx -0.195652$	39-53
glider gun	15	5	15	5	$-20/77 \approx -0.259740$	27-55

periodic structure. For example, C_1 has two values expressing that nine or twenty-three cells are needed to represent this glider.

2.2 Subset of tiles

A notorious property in Rule 110 is given by the local function φ which covers the evolution space in a two-dimensional representation with tiles or triangles of different sizes [13, 9]. A tile is described by T_n where $n \in \mathbb{N}$ indicates the size of each triangle. State 0 defines the interior of the triangles and state 1 determines the perimeter. These triangles are classified in two sets α and β for $n > 1$. Figure 3 shows some of them in ascendent numeration, forming two subsets of countable families, such that $|\{T_n^\alpha\}| = |\{T_n^\beta\}|$.

The T_0 tile is just a cell in state 0; thus when the initial configuration is assigned by the expressions: 0*, 1* and (10)*, the evolution space is established by an homogenous evolution with state 0 (or tile T_0). Nevertheless, the behavior is not the same for tiles T_1, T_2^α, T_2^β, T_3^α, T_3^β, ..., T_n^α, T_n^β, ... The evolution space can be covered by any T_n tiles for $0 \leq n \leq 4$ and for $n \geq 5$, it is covered by at least two different tiles. Let

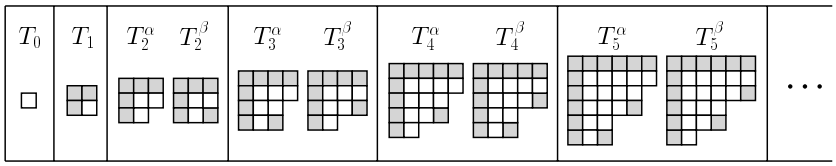

Fig. 3. Two sets of tiles in Rule 110: α and β.

T_i and $T_j \in \mathcal{T}$ where $i \neq j$, then both sets cannot operate in the plane under the function of Rule 110 if they cover partially the space (there are gaps) or overlap with another tiles.

Thus, the tile families $\{T_n^\alpha\}$ and $\{T_n^\beta\}$ give a detailed description of the evolution space in Rule 110 through their sets: α and β. A second important point is that the tiles establish properties by the periodic margins in their recurrent structures (gliders and ether). Their interpretation is also important to derive phases, including non-periodic structures.

2.3 Regular language based on gliders and de Bruijn diagrams

This section is devoted to give a brief introduction in phases and de Bruijn diagrams; a more extended explanation can be found in [3] and the complete set of regular expressions may be consulted from a digital file.[9]

The regular language L_{R110} is based on a set of regular expressions Ψ_{R110} determining each glider of $\mathcal{G}$. L_{R110} is established via de Bruijn diagrams and with the characterization of tiles, where both have been analyzed for defining useful features called "phases." They indicate with precision the position and the exact moment where each glider must be positioned into a given initial condition.

The set of regular expressions and their basic operations are able to construct desired initial conditions to yield evolutions with important features; the main interest is to control and produce collisions among gliders. In this way L_{R110} becomes a powerful tool to codify initial conditions; immediate applications with relevant results have been performed over hundreds, thousands, millions and thousands of million of cells[10] [10–12, 14].

De Bruijn diagrams [15–17] are very adequate for describing evolution rules in one-dimensional cellular automata, although originally they were

[9] `http://uncomp.uwe.ac.uk/genaro/rule110/listPhasesR110.txt`
[10] `http://uncomp.uwe.ac.uk/genaro/rule110/ctsRule110.html`

used in shift-register theory (the treatment of sequences where their elements overlap each other). The de Bruijn diagrams can extract any periodic string of Rule 110 or another CA; particularity we employ the connected cycles from extended de Bruijn diagrams to calculate any string and its shifts over a number of generations.

A glider in Rule 110 can be seen as a periodic construction preserving a defined cyclic border with ether in the evolution space. Essentially a glider has the following characteristics: *volume* (number of cells representing its form), *period* (number of evolutions to recover the original sequence), *displacement* (change of horizontal position measured in cells on finishing its period) and speed (velocity produced by the period between the displacement). Thus a set of gliders with different volume and speed can be represented.

In order to explain how the sequences of each glider are determined, a de Bruijn diagram for an A glider is firstly calculated and it is decribed how the periodic sequences are extracted from it or representing this glider and specifying as well its set of regular expressions.

A glider moves two cells to the right in three times (see Table 2), the corresponding extended de Bruijn diagram (2-shift, 3-gen) is depicted in Figure 4. The cycles in the diagram are periodic sequences describing each phase in the glider; however these sequences are not ordered yet, hence they must be classified.

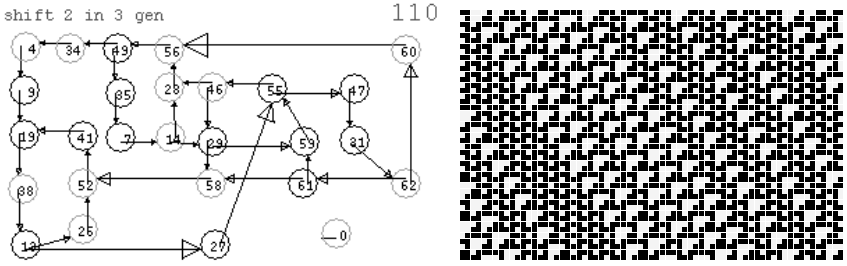

Fig. 4. De Bruijn diagram for A gliders within ether configurations.

Diagram in Figure 4 has two cycles: a cycle formed by just a vertex 0 and another large cycle of 26 vertices composed by other nine internal cycles. The evolution of the right illustrates the location of the different periodic sequences producing the A glider in distinct numbers.

The sequences or regular expressions determining the phases of the A glider are obtained following paths through the edges of the diagram. For instance, the following cycles specify different formations:

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 among 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 specific phase (in the following subsection we will see that this phase is called $e(\mathrm{f}_1\_1)$).

Cycle with period 1 represented by vertex 0 produces an homogeneous evolution in state 0. The evolution space in Figure 4 shows different packages of A gliders, the initial condition is constructed following some of the seven possible cycles of the de Bruijn diagram or a combination of them. In this way the number of A gliders or the number of intermediate tiles T_3^{β} can be selected changing from one cycle to another.

Table 3. Four sets of phases Ph_i in Rule 110.

phases level one (Ph_1)	$\rightarrow$ {f_1_1, f_2_1, f_3_1, f_4_1}
phases level two (Ph_2)	$\rightarrow$ {f_1_2, f_2_2, f_3_2, f_4_2}
phases level three (Ph_3)	$\rightarrow$ {f_1_3, f_2_3, f_3_3, f_4_3}
phases level four (Ph_4)	$\rightarrow$ {f_1_4, f_2_4, f_3_4, f_4_4}

The alignment of the f_i_1 phases is analyzed to determine the whole set of strings for every glider. First it is necessary to describe the form and limits of each glider by tiles, then a phase is fixed (in our case the phase f_i_1) and a horizontal line is placed in the evolution space bounded by two tiles T_3. Thus, the sequence between both tiles aligned in each of the four levels determines a periodic sequence representing a particular structure in the evolution space of Rule 110. Thus all periodic sequences in a specific phase are calculated, enumerating the phases for each glider or non-periodic structure.

Table 3 represents every disjoint subset of phases, each level contains four phases. Variable f_i indicates the phase currently used where the second subscript j (completing notation f_i_j) indicates the selected set Ph_j of regular expressions. Finally, this notation codifies initial conditions by phases as follows:

$$\#_1(\#_2, \mathrm{f}_i\_1) \tag{1}$$

where $\#_1$ represents a glider according to Cook's classification (Table 2) and $\#_2$ the phase of the glider with period greater than four.[11]

Therefore, the full list of regular expressions Ψ_{R110} for each glider, including a glider gun, is given in [3] and it also serves as a simple finite-machine for the OSXLCAU21 system[12] developed to code gliders in Rule 110.

3 Universal CA

Universality in CA is developed and solved by von Neumann in [18] as a previous step in the specification of the universal constructor for his 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.

There are several significant simplifications in universal-computing CA with fewer states and dimensions: Codd in 1968 [19], Banks in 1971 [20], Smith in 1971 [21], Conway in 1982 [22], Lindgren and Nordahl in 1990 [23], and finally Cook in 1998 [1].

Analogous to the search of a minimal universal Turing machine [24], Cook details the minimal universal CA with Rule 110; showing how a "simple" elemental CA is able to simulate an equivalent Turing machine within a novel CTS, performing a computation [1, 25] with collision of gliders in millions of cells.

3.1 Tag systems

The concept of "tag system" has an interesting antecedent in the work developed by Emil Post looking for a unique effective procedure for proving if for any formula could be formally derivable in its own logic of propositions. This issue is referred as the "finiteness problem" consisting on finding a canonical form to solve a given decision problem (for more details see "Emil L. Post: His Life and Work" in [26]); inducing the search and development for some auxiliary procedure.

The tag system is a variation from the Turing machine [29] for implementing computable processes; Minsky demonstrates that tag systems are universal [30] and unsolvable [34]. Another relevant result is obtained

[11] The arrangement by capital letters for the $\#_2$ parameter into the OSXLCAU21 system [36] does not have a particular meaning; it is only used to represent the different levels of phases in periods module four.

[12] `http://uncomp.uwe.ac.uk/genaro/OSXCASystems.html`

by Cocke and Minsky [31] proving which tag systems are universal just with the deletion number $\nu = 2$.

Therefore, it is worthwhile to remember the original concept to understand tag systems. Thus a literal transcription from [27] is given[13]:

> Given, a positive integer ν, and μ symbols which may be taken to be $0, 1, \ldots, \mu - 1$. With each of these μ symbols a finite sequence of these symbols is associated as follow.
>
> $$\begin{aligned} 0 &\to a_{0,1}a_{0,2}\ldots a_{0,\nu_0} \\ 1 &\to a_{0,1}a_{0,2}\ldots a_{0,\nu_1} \\ &\vdots\ \vdots\ \vdots \\ \mu - 1 &\to a_{\mu-1,1}a_{\mu-1,2}\ldots a_{\mu-1,\nu_{\mu-1}}. \end{aligned}$$
>
> It is understood that in each sequence the same symbol may occur several times, and that a particular associated sequence may be null. In terms of this basis, we set up the following operation for obtaining from any given non-null sequence:
>
> $$B = b_1 b_1 \ldots b_l$$
>
> on the symbols $0, 1, \ldots, \mu - 1$, a unique derived sequence B' on those symbols. To the right end of basis, and from the left end of this augmented sequence remove the first ν elements — all if there be less than ν elements. As long as B' is not a null sequence, this operation can then be applied to B' to yield a sequence B'', to B'', if not null, to yield B''', and so on. The problem of "tag" for the given basis is then to obtain a finite process for determining for any initial sequence B whether the resulting iterative process does or does not terminate.

Let us see some simple samples with parameters $\mu = 2$, $\nu = 2$ and the next production rules: $0 \to 0$ and $1 \to 10$. The symbol $\vdash$ means an effectively computable relation [33]; let us made some operations with different initial conditions looking its "halt" condition.

(a) $0000 \vdash 000 \vdash 0$ (*halt* disappear)
(b) $1111 \vdash 1110 \vdash 1010 \vdash 1010$ (*halt* periodic)
(c) $0101 \vdash 010 \vdash 0$ (*halt* disappear)
(d) $1010 \vdash 1010$ (*halt* periodic)
(e) $00001111 \vdash 0011110 \vdash 111100 \vdash 110010 \vdash 001010 \vdash 10100 \vdash 10010 \vdash 01010 \vdash 0100 \vdash 000 \vdash 0$ (*halt* disappear)

[13] This work is finished by Post in 1941 but it is published up to 1965. It may be consulted in [28, 26]. M. Davis June 2007, personal communication.

Post determines two relevant conjectures in tag systems; the first showing that a tag system can be recursively unsolvable [26] and the second one finding intractable problems [32]. In particular for $\mu = 2$ and $\nu > 2$, it is established that not all the terminal processes are known, a special case is illustrated for the pair of productions $0 \rightarrow 00$ and $1 \rightarrow 1101$.

3.2 Cyclic tag systems

CTS are new machines proposed by Cook in [1] as a tool to implement computations in Rule 110. CTS are a variant of tag systems with some changes; they have the same action of reading a tape in the front and adding characters at its end, nevertheless there are some new restrictions and characteristics:

1. CTS need at least two letters in their alphabet ($\mu > 1$).
2. Only the first character is deleted ($\nu = 1$) and its respective sequence is added.
3. In all cases if the machine reads a character zero then the production rule is always null ($0 \rightarrow \epsilon$, where ϵ represents the empty word).
4. There are k sequences from μ^* which are periodically accessed to specify the current production rule when a nonzero character is taken by the system. Therefore the period of each cycle is determinate by k.

This way a cycle determines a partial computation over the tape. Cook does not specify any particular halt condition because perhaps it is a direct consequence of tag systems. Let us see some samples of a CTS working with $\mu = 2$, $k = 3$ and the next production rules: $1 \rightarrow 11$, $1 \rightarrow 10$ and $1 \rightarrow \epsilon$. To avoid writing every chain where there is not necessity of adding characters, the $\vdash_k$ relation is just indicated to simplify space and redundant information.

For example, the succession $00001 \vdash_1\vdash_2\vdash_3\vdash_1\vdash_2 10$ represent the next particular relations $00001 \vdash_1 0001 \vdash_2 001 \vdash_3 01 \vdash_1 1 \vdash_2 10$. Also, each relation indicates which μ sequence is selected.

(a) $00001 \vdash_1\vdash_2\vdash_3\vdash_1\vdash_2 10 \vdash_3\vdash_1$ (*halt* disappear)
(b) $10101 \vdash_1 010111 \vdash_2\vdash_3\vdash_1\vdash_2 1110 \vdash_3\vdash_1 1011 \vdash_2 01110 \vdash_3\vdash_1 11011 \vdash_2 101110 \vdash_3\vdash_1\vdash_2 11010 \vdash_3\vdash_1 01011 \vdash_2\vdash_3\vdash_1\vdash_2 110 \vdash_3\vdash_1 011 \vdash_2\vdash_3\vdash_1 11 \vdash_2 110 \vdash_3\vdash_1 011$ (*halt* periodic)
(c) $01100 \vdash_1\vdash_2 10010 \vdash_3\vdash_1\vdash_2\vdash_3\vdash_1$ (*halt* disappear)
(d) $11111 \vdash_1 111111 \vdash_2 1111110 \vdash_3\vdash_1 1111011 \vdash_2 11101110 \vdash_3\vdash_1 1011101 1 \vdash_2 011101110 \vdash_3\vdash_1 110111011 \vdash_2 1011101110 \vdash_3\vdash_1\vdash_2 1101110$

$\vdash_3\vdash_1 0111011 \vdash_2\vdash_3\vdash_1 101111 \vdash_2 0111110 \vdash_3\vdash_1 1111011 \vdash_2 11101110$
$\vdash_3\vdash_1 10111011 \vdash_2 011101110 \vdash_3\vdash_1 110111011 \vdash_2 1011101110 \vdash_3\vdash_1\vdash_2$
$110111010 \vdash_3\vdash_1 011101011 \vdash_2\vdash_3\vdash_1 10101111 \vdash_2 010111110 \vdash_3\vdash_1$
$0111110 \vdash_2\vdash_3\vdash_1 111011 \vdash_2 1101110 \vdash_3\vdash_1 0111011 \vdash_2\vdash_3\vdash_1 1011 \vdash_2$
$01110 \vdash_3\vdash_1 11011 \vdash_2 101110 \vdash_3\vdash_1\vdash_2 11010 \vdash_3\vdash_1 01011 \vdash_2\vdash_3\vdash_1\vdash_2$
$110 \vdash_3\vdash_1 011 \vdash_2\vdash_3\vdash_1 11 \vdash_2 110 \vdash_3\vdash_1 011$ (*halt* periodic)

This representation offers two facilities; avoiding long chains in every step for the null relation and finding the periodic behavior between chains and relations where both need to be repeated. In general in this case one must hope that any chain eventually reaches a terminal condition.

The intention of the previous example is to show that the halt condition in CTS behaves in the same way that in tag systems, consequently. Particularity CTS tend to growth quickly and it is more complicated to see this behavior in this way. In this sense, Morita in [35] has demonstrated how to implement a particular halt condition in CTS given an output string when the system is halting, and how a partitioned CA can simulate any CTS, consequently computing all the recursive functions.[14]

3.3 Comparing behaviors of tag systems and CTS

Cook explains how a CTS must simulate a tag system [1]. This section displays a very simple case where the "behavior" of a CTS emulates a terminal tag system.

Let us take a tag system with $\mu = 2$, $v = 1$ and the next productions: $0 \to 1$ and $1 \to \epsilon$. Its behavior is showed with some chains having in all cases the disappearing halt condition.

(a) $0000 \vdash 000 \vdash 000 \vdash 00 \vdash 0$
(b) $1111 \vdash 111 \vdash 11 \vdash 1$
(c) $010101 \vdash 101011 \vdash 01011 \vdash 10111 \vdash 0111 \vdash 1111 \vdash 111 \vdash 11 \vdash 1$
(d) $000111 \vdash 001111 \vdash 011111 \vdash 111111 \vdash 11111 \vdash 1111 \vdash 111 \vdash 11 \vdash 1$

Thus we need to specify a CTS with $\mu = 2$, $k = 2$ and the next sequences: $1 \to 1$ and $1 \to \epsilon$. Let us take the same previous chains.

(a) $0000 \vdash_1\vdash_2\vdash_1\vdash_2\vdash_1$
(b) $1111 \vdash_1 1111 \vdash_2\vdash_1 111 \vdash_2\vdash_1 11 \vdash_2\vdash_1 1 \vdash_2$
(c) $010101 \vdash_1\vdash_2\vdash_1\vdash_2\vdash_1\vdash_2 1 \vdash_1 1 \vdash_2$
(d) $000111 \vdash_1\vdash_2\vdash_1\vdash_2\vdash_1 11 \vdash_2\vdash_1 1 \vdash_2$

[14] K. Morita December 2006, personal communication.

It is clear that both machines have the same behavior yielding analogous terminal states; although frequently a CTS uses more relations than a traditional tag system. CTS are relevant as equivalent Turing machines and they were studied with certain dedication in the last years. Open questions are:

- How CTS could find intractable and undecidable problems from more basic sequences?
- What is the number of terminal subsets with minimum values in μ and k?

Similar to Post with his tag system, Cook determines that for a CTS with $\mu = 2$, $k = 2$, the next relations $1 \rightarrow 11$ and $1 \rightarrow 10$, and also beginning with a 1 in the tape; it is unknown whether the process is terminal or not. Another particular problem is to establish a halt condition in Rule 110 with a CTS; this details are discussed in the next section.

4 Reproducing a CTS in Rule 110

This section presents several aspects for reconstructing the operation of a CTS in Rule 110 as the one presented in [2]. We must use a CTS with $\mu = 2$, $k = 2$ and the next relations $1 \rightarrow 11$ and $1 \rightarrow 10$, begging with a 1 into the tape. A segment of its behavior is given as follow:

$1 \vdash_1 11 \vdash_2 110 \vdash_1 1011 \vdash_2 01110 \vdash_1\vdash_2 11010 \vdash_1 101011 \vdash_2 0101110 \vdash_1\vdash_2 0111010 \vdash_1\vdash_2 1101010 \vdash_1 10101011 \vdash_2 010101110 \vdash_1\vdash_2 010111010 \vdash_1\vdash_2 011101010 \vdash_1\vdash_2 110101010 \vdash_1 1010101011 \vdash_2 01010101110 \vdash_1\vdash_2 01010111010 \vdash_1\vdash_2 01011101010 \vdash_1\vdash_2 01110101010 \vdash_1\vdash_2 11010101010 \vdash_1 101010101011 \vdash_2 0101010101110 \vdash_1\vdash_2 0101010111010 \vdash_1\vdash_2 01010111010 10 \vdash_1\vdash_2 0101110101010 \vdash_1\vdash_2 0111010101010 \vdash_1\vdash_2 1101010101010 \vdash_1 10101010101011 \vdash_2 010101010101110 \vdash_1\vdash_2 010101010111010 \vdash_1\vdash_2 01010 1011101010 \vdash_1\vdash_2 010101110101010 \vdash_1\vdash_2 010111010101010 \ldots.$

This example and CTS in general have an apparent inspiration in the mathematical puzzle proposed by Kolakoski [37]. But, particularity Cook was looking for a simple procedure to construct sequences avoiding a quick repetition.[15]

Some characteristics of the previous CTS are as follows: A chain tends to growth in certain order by its length or package of bits and intuitively it seems that these relations do not repeat. For example, taking as basis

[15] M. Cook June 2007, personal communication.

the expression 1(10)*, the CTS moves (from right to left) and adds a new pair of bits; when 1(10)* is reached again, the number of relations selected in each interval growths lineally in order of $f_1 = 2(n+1)$. Nevertheless, the number of bits into the chain growths parabolically as we can see in Fig. 5. Function f_2 represents the interpolated polynomial from the first ten initial values and function f_3 the extrapolation.

Therefore, if we take consecutive copies of 1(10)* with their respective intervals determined by the number of j relations (represented as $\vdash_i^j$), we obtain the next chains: $1 \vdash_i^2 110 \vdash_i^4 11010 \vdash_i^6 1101010 \vdash_i^8 1101010 \vdash_i^{10} 110101010 \vdash_i^{12} 11010101010 \vdash_i^{14} 1101010101010 \vdash_i^{16} \ldots$. So, there is not a succession of two contiguous zeros as it was previously generated by the CTS. The parabolic growth is given in every chain and its linear growth in the interval of relations.

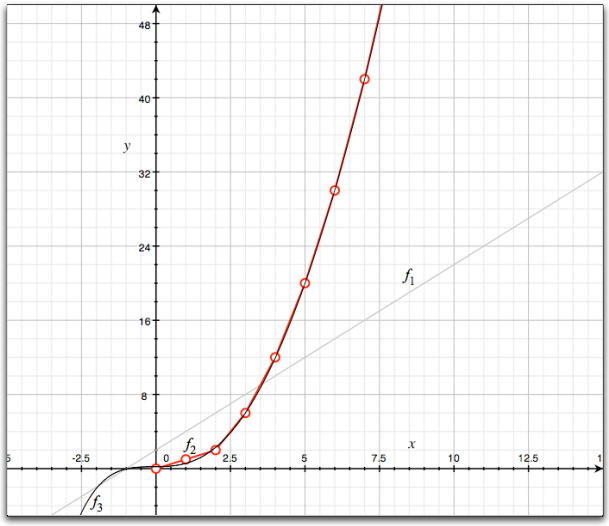

Fig. 5. Growth function for the CTS.

Now we expose how to interpret gliders and their collisions to emulate a CTS in Rule 110, basically we must use packages of gliders to represent data and operators; their reactions read, transform and delete data in the tape. Several details are delicate reason why its reconstruction is laborious and not an easy task. In essence we can see the representation in Figure 6 and the features and labels of this diagram are explained in the next paragraphs.

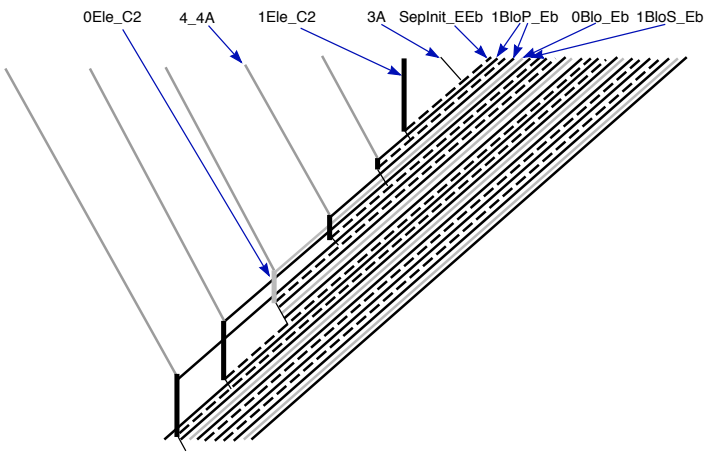

Fig. 6. Interpretation of a CTS working in Rule 110.

4.1 Components based on packages of gliders

The construction of the CTS in Rule 110 can be partitioned in three parts. First is the left periodic part controlled by packages of 4_A^4 gliders, this part is static and controls the production of 0's and 1's. The second part is the center determining the initial value in the tape, and the third one is the right cyclic part which has the data to process, adding a leader component specifying added or erased data from the tape in the evolution space.

In the left part, the four packages of A^4 gliders must be carefully explained because although they are static, their phases change periodically. The important point to implement these components is defining both distances and phases, because a distinct phase or distance induces a non-wished reaction; in fact all components follow this restriction because every glider of each component must be correctly aligned.

Fig. 7. Packages of 4_A^4 gliders.

Packages defined by A^4 gliders have three different phases: f_1_1, f_2_1 and f_3_1. In order to construct the first 4_A^4 we need to establish the

phase of each A^4; let us assign phases in the following way: A^4(f$_3$_1)-27e-A^4(f$_2$_1)-23e-A^4(f$_1$_1)-25e-A^4(f$_3$_1) as it is illustrated in Figure 7. Spaces between each 4_A^4 are static but the phases rotate to obtain collisions like a soliton with the $\bar{E}$ gliders generated in the system, this rotation in each A^4 is also projected in the whole package 4_A^4. Eventually this becomes periodic as follows:

{649e-A^4(f$_2$_1)-27e-A^4(f$_1$_1)-23e-A^4(f$_3$_1)-25e-A^4(f$_2$_1)-649e-A^4(f$_1$_1)-27e-A^4(f$_3$_1)-23e-A^4(f$_2$_1)-25e-A^4(f$_1$_1)]-649e-A^4(f$_3$_1)-27e-A^4(f$_2$_1)-23e-A^4(f$_1$_1)-25e-A^4(f$_3$_1)}*

if for every 4_A^4 we take a phase representing the complete package, we can rename it as:

$$\{649e\text{-}4\_A^4(\mathrm{F}_2)\text{-}649e\text{-}4\_A^4(\mathrm{F}_1)\text{-}649e\text{-}4\_A^4(\mathrm{F}_3)\}^*$$

this rotation is important to preserve the left part of the system.

Distances between each 4_A^4 must be carefully specified since short spaces do not allow a good construction because A gliders are faster than E and $\bar{E}$ gliders, and they would reach the data formed by C_2 gliders before new packages of data arrive. Cook has considered this problem determining a necessary minimum distance when the twelfth 1 is produced into the tape of its diagram in [1]. This is very close to the fourth 0 but the next 4_A^4 package does not disturb this element; other distances for the rest of collisions are sufficiently spaced taking finally an interval of at least 649 ether configurations.

Elements 1Ele_C_2 and 0Ele_C_2

The central part is constituted by an initial 1 into the tape represented by a package of four C_2 gliders. This way, an element 1Ele_C_2 represents a 1 and the element 0Ele_C_2 represents a 0 in the tape.

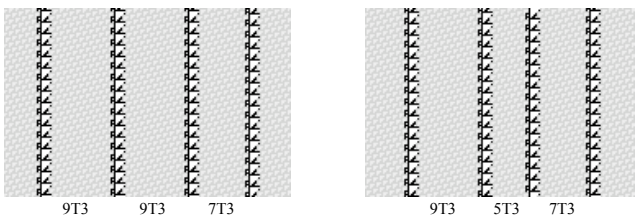

Fig. 8. Elements 1Ele_C_2 and 0Ele_C_2.

Particularity, the number of elements listed in Wolfram's book [2] to construct a CTS is incomplete and with mistakes, this situation made impossible to reconstruct the system.[16] In fact a discussion about the external factors involved in the publishing of the original work in Rule 110 can be consulted in [38, 39].

The first picture in Figure 8 shows the element 1Ele_C_2, since the same idea is used in the others components, first we should reproduce each element by the phases f_i_1. Thus the code in phases is: C_2(A,f_1_1)-2e-C_2(A,f_1_1)-2e-C_2(A,f_1_1)-e-C_2(B,f_2_1). The first three C_2 gliders are in phase (A,f_1_1) and the fourth C_2 is in phase (B,f_2_1), hence the distances among them are: $9T_3$-$9T_3$-$7T_3$. In order to take the distance, we count the number of T_3 tiles between gliders. The phase of the last C_2 cannot be other because it produces another reaction. The same analysis for the element 0Ele_C_2 yields the distances $9T_3$-$5T_3$-$7T_3$.

Thus differences in both blocks are relevant, the first difference is that the distinct central space between both packages and the second one is that their phases are dissimilar as well. Their phases f_i_1 are projected at each element, for instance the element 1Ele_C_2 has 7 phases in which it can be represented in the initial configuration, hence we take the first glider and the rest are aligned together. Each sequence of bits for every element is presented in the appendix.

Element 0Blo_$\bar{E}$

The left part stores blocks of data without transformation in packages of E and $\bar{E}$ gliders.

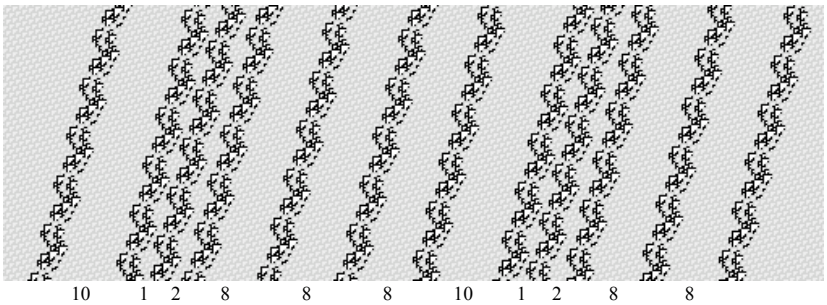

Fig. 9. Element 0Blo_$\bar{E}$.

[16] These problems were clarified by M. Cook. in november 2002 (personal communication).

The element 0Blo_$\bar{E}$ is formed by $12\bar{E}$ gliders as we can see in figure 9. There must be an accurate phase and distance between each one of them because in otherwise the whole system will be disturbed. This element represents a 0 into the tape in the CTS if a leader permit this conversion, when it is pre-transformed then finally we have an element 0Ele_C_2. Essentially this block of gliders do not need important changes like we shall see in the next element; it is only necessary to preserve distances and phases.

Elements 1BloP_$\bar{E}$ (primary) and 1BloS_$\bar{E}$ (standard)

Cook establishes the existence of two components to represent 1's: *primary* and *standard*; if one of them is not used the system does not work suitably. Let us note that the primary element was completely ignored in Wolfram's book [2].

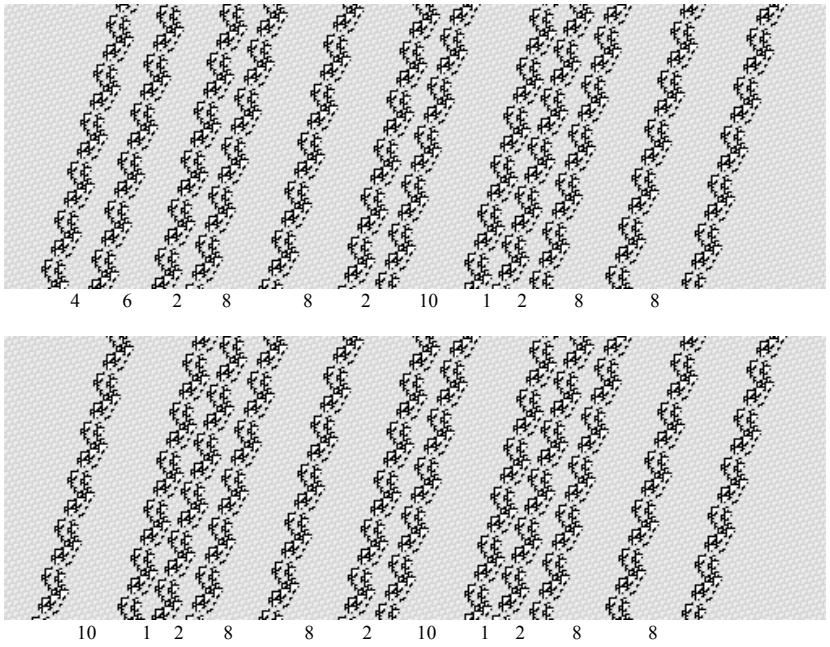

Fig. 10. Elements 1BloP_$\bar{E}$ (primary) and 1BloS_$\bar{E}$ (standard) respectively.

They are differences in distance of their first two $\bar{E}$ gliders as figure 10 displays. In essence both blocks produce the same element 1Add_$\bar{E}$ although coming from different intervals. The reason to use both blocks

is that Rule 110 is not symmetric; therefore if we use only one element then surely we would obtain a good production in the first collision generating an element 1Add_$\bar{E}$; moreover when the second package of 4_A^4 gliders finds the second block, the system is completely destroyed.

Element SepInit_$E\bar{E}$

A leader element renamed as SepInit_$E\bar{E}$ (see Figure 11) is important to separate packages of data and determine their incorporation into of the tape. Particularity its has a small but detailed code determining which data without transformation would be added or erased from the tape, depending on the value that it finds. This element is very delicate because we can define two different phases for the same index but they eventually are different, because the E glider has 4 phases and the $\bar{E}$ glider has 8 phases. Therefore, we could have two equals phases apparently from the initial gliders but $\bar{E}$ is in different phases.

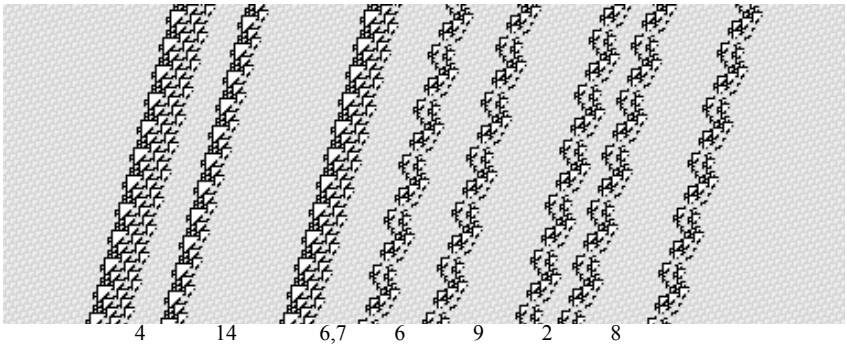

Fig. 11. Element SepInit_$E\bar{E}$.

For this reason we have two distances at the same interval, thus if a phase works suitably the next will not because their $\bar{E}$ gliders would be out of phase. Then we should omit the second version of this element presented in Wolfram's book [2] because it is a consequence of SepInit_$E\bar{E}$ after colliding with three A gliders and it is not necessary to code it from the initial configuration since it is reached after several reactions on the evolution.

Elements 1Add_$\bar{E}$ and 0Add_$\bar{E}$

Figure 12 illustrates the elements 1Add_$\bar{E}$ and 0Add_$\bar{E}$ produced by two previous different packages of data. An element 1Add_$\bar{E}$ must be

generated by a block 1BloP_$\bar{E}$ or by 1BloS_$\bar{E}$. This way, both elements can produce the same element.

On the other hand, an element 0Add_$\bar{E}$ is generated by a block 0Blo_$\bar{E}$ although in this case it is not necessary a primary element. Nevertheless, we could produce $\bar{E}$ gliders modifying their first two distances and preserving them without changing others gliders to get a reliable reaction. This is possible if we desire to experiment with other combinations of blocks of data although this could complicate the implementation of the system.

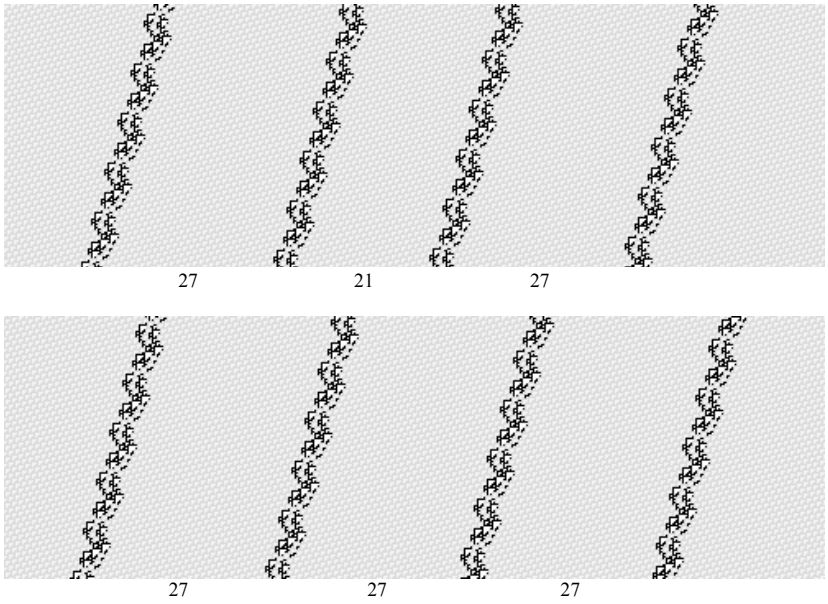

Fig. 12. Elements 1Add_Eb and 0Add_$\bar{E}$ respectively.

Again, here we have another mistake in Wolfram's book [2] (page 681). The element 1Add_$\bar{E}$ has distances 27e-21e-27e in the $\bar{E}$ gliders but in [2] this element has distances 20e-27e-21e. It is easy to deduce that this was an error on selecting the figure. The first $\bar{E}$ of such figure in [2] is an invisible glider for the CTS because it will cross as soliton all data and operators; thus the next three $\bar{E}$ gliders correspond to the element 1Add_$\bar{E}$.

So, if a leader element SepInit_$E\bar{E}$ reaches an element 1Ele_$\bar{E}$, it erases this value from the tape and adds a new data that shall be trans-

formed. In other case, if it finds an element 0Ele_$\bar{E}$, then it erases this element from the tape and also erases a set of unchanged data which come from the right until finding a new leader element. Essentially this operation represents the addition of new values from periodic packages of gliders coming from the right. Thus an element 1Add_$\bar{E}$ is transformed into 1Ele_$\bar{E}$ colliding against a package of 4_A^4 gliders representing a value 1 in the tape, and the element 0Add_$\bar{E}$ is transformed into 0Ele_$\bar{E}$ colliding against a package of 4_A^4 gliders representing a value 0 in the tape.

Table 4. Distances between the gliders of each element.

element	distance
1Ele_C_2	9-9-7
0Ele_C_2	9-5-7
1BloP_$\bar{E}$	4-6-2-8-8-2-10-1-2-8-8
1BloS_$\bar{E}$	10-1-2-8-8-2-10-1-2-8-8
0Blo_$\bar{E}$	10-1-2-8-8-8-10-1-2-8-8
SepInit_$E\bar{E}$	4-14-(6 or 7)-6-9-2-8
1Add_$\bar{E}$	27-21-27
0Add_$\bar{E}$	27-27-27

Finally, Table 4 enumerates all distances (number of T_3 tiles) for every necessary element in the construction; this is a practical way to identify any element in the evolution space. Finally, we can code the construction of this CTS with phases representation in three main big blocks as follows:

left: ...-217e-4_A^4(F2)-649e-4_A^4(F1)-649e-4_A^4(F3)-649e-4_A^4(F2)-649e-4_A^4(F1)-649e-4_A^4(F3)-216e-

center: 1Ele_C_2(A,f$_1$_1)-e-A^3(f$_1$_1)-

right: SepInit_$E\bar{E}$(C,f$_3$_1)-1BloP_$\bar{E}$(C,f$_4$_1)-SepInit_$E\bar{E}$(C,f$_3$_1)-1BloP_$\bar{E}$(C,f$_4$_1)-0Blo_$\bar{E}$(C,f$_4$_1)-1BloS_$\bar{E}$(A,f$_4$_1)-SepInit_$E\bar{E}$(A,f$_2$_1)(2)-1BloP_$\bar{E}$(F,f$_1$_1)-SepInit_$E\bar{E}$(A,f$_3$_1)(2)-1BloP_$\bar{E}$(F,f$_1$_1)-0Blo_$\bar{E}$(E,f$_4$_1)-1BloS_$\bar{E}$(C,f$_4$_1)-e-SepInit_$E\bar{E}$(B,f$_1$_1)(2)-1BloP_$\bar{E}$(F,f$_3$_1)-e-SepInit_$E\bar{E}$(B,f$_1$_1)(2)-217e-...

this initial conditions in Rule 110 are able to generate the serial sequence of bits 1110111 and a separator at the end also with two solitons;

achieving a satisfactory construction in 57,400 generations with an initial configuration of 56,240 cells. This square implies an evolution space of 3,228,176,000 cells; thus the simulation of the CTS in Rule 110 is highly inefficient and it could be hardly used in practical applications. Moreover, the relevance here is to show its universality produced by a simply impressive construction and global synchronization.

In the next section we explain and display the (more relevant) stages in the evolution space of Rule 110 with the CTS working. First we must see two possible cases.

4.2 Beginning from 0

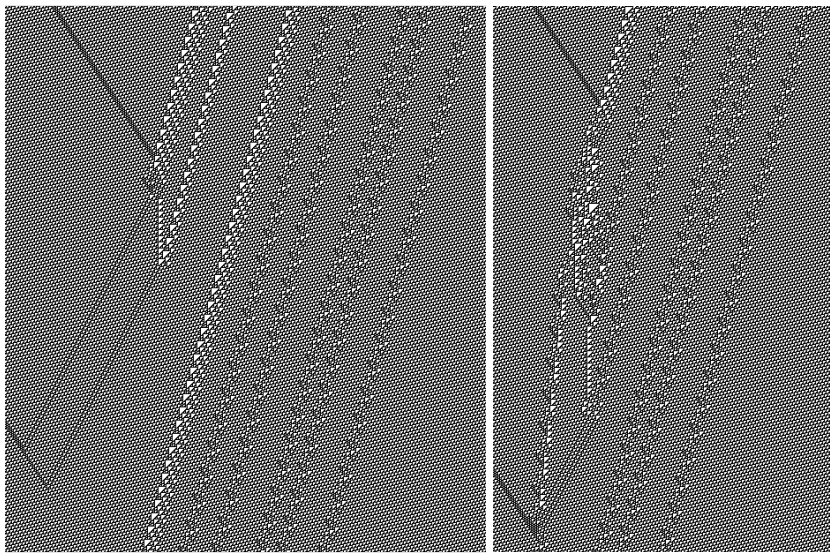

Fig. 13. CTS beginning from a 0 in Rule 110.

If the system begins from zero in the tape then the machine must stop in one step because this element is erased and there is not any added element to the tape (remember that $0 \rightarrow \epsilon$), corresponding to the pair in the original productions. This operation is fulfilled with the packages of gliders previously discussed because independently of how the blocks of data are structured, they do not produce a new element in the tape. The structure of data is gradually destroyed as we can see in figure 13, after the second package of A^4 gliders reaches the a second leader component.

4.3 Beginning from 1

The CTS presented in [2] begin from 1 in the tape like the illustration in figure 6. The next snapshots show the most relevant stages of the machine working in Rule 110. In every picture we indicate with colored labels every different element or component.

Figure 14 displays the initial stage of the CTS with a 1 in the tape. This data is represented by the element 1Ele_C_2 and depicts as well the central part of the machine from its initial condition also with a package of A^3 gliders, such that, this package would prepare the first leader element SepInit_$E\bar{E}$ coming from the right periodic part. So this package of A^3 gliders arrives joined or separated producing the same reaction.

The first reaction in figure 14 deletes its respective 1 in the tape (element 1Ele_C_2) and as consequence an element separator prepares the next data that would be aggregated. If it finds an element 0Ele_C_2 then does not allow to add data on the tape until finding another separator. Both $\bar{E}$ gliders which remain to the first production are invisible in all the system and do not affect in any operation because they cross as solitons the subsequent packages of 4_A^4 gliders.

Figure 15 shows the continuation of the previous evolution. This snapshot describes a production of the element 1Add_$\bar{E}$ from the primary component 1BloP_$\bar{E}$. Its construction uses a basic periodic reactions identified vertically by the columns of C_1 and pairs of C_3 gliders. A C_1 stops an $\bar{E}$ producing a new pair of C_3 which also receive a pair of $\bar{E}$ gliders yielding a C_1 again. This operation is repeated until find a leader element receiving three A gliders.

Figure 16 illustrates two aspects, the production of an element 1Ele_C_2 and the existence of invisible $\bar{E}$ gliders crossing as solitons the successive packages of 4_A^4 gliders. Each block of A^4 gliders transforms each $\bar{E}$ in 1Add_$\bar{E}$ into a C_2 of 1Ele_C_2. Analogous, in order to construct an element 0Ele_C_2 we have a similar structure to the previous example and their distances determine this element with their internal gliders.

Figure 17 displays the evolution following figure 16. In this case, we can see an element 1Ele_C_2 constructed from a package of 4_A^4 gliders, However, it has a very short life because quickly a separator element arrives erasing and preparing new data that would be aggregated to the tape. An important point is that $\bar{E}$ gliders which cross as solitons preserve distances and phases.

Figure 18 shows how data are pre-transformed and eventually aggregated onto the tape and where a standard component must be correctly placed. Top evolution describes the transformation from a primary component 1BloP_$\bar{E}$ in an element 1Add_$\bar{E}$ with a glider soliton. There is as

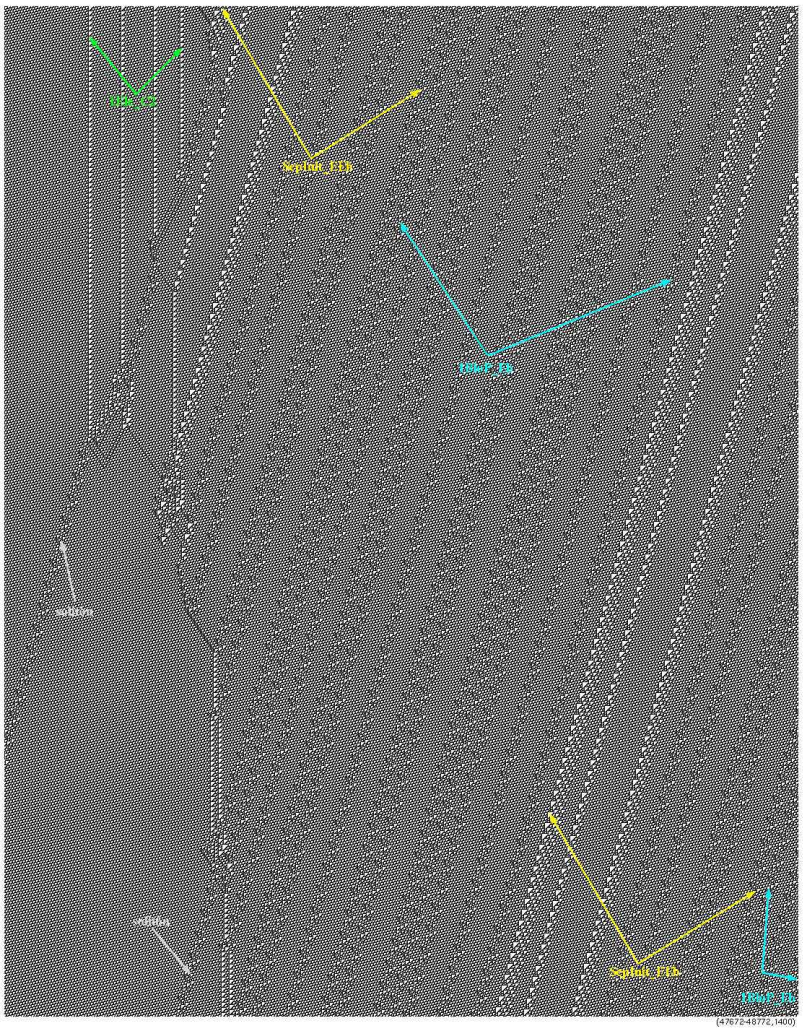

Fig. 14. Initial stage of CTS in Rule 110.

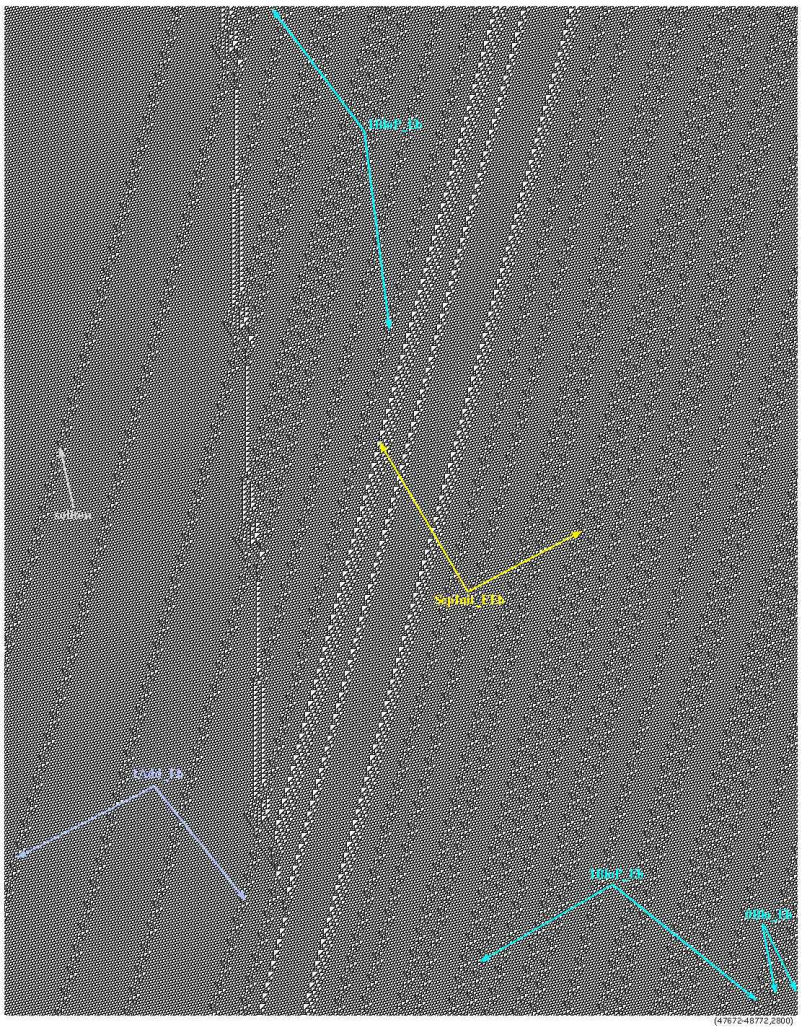

Fig. 15. Construction of the element 1Add_$\bar{E}$ from a primary component

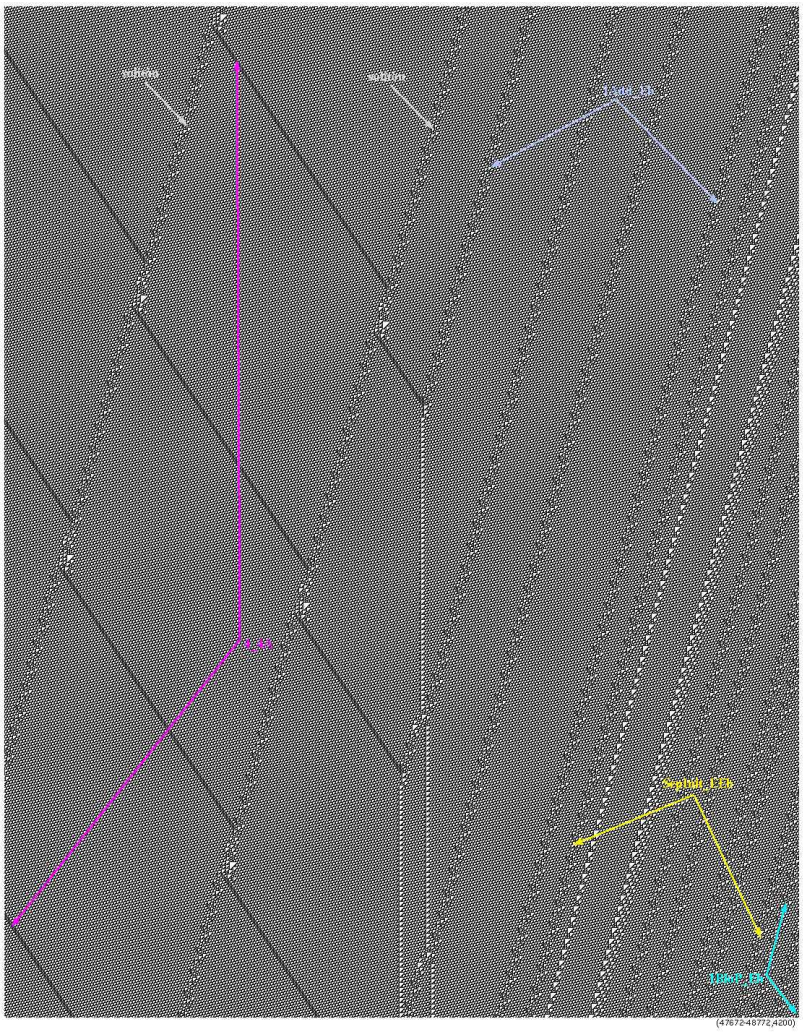

Fig. 16. Operators based in packages of 4_A^4 gliders.

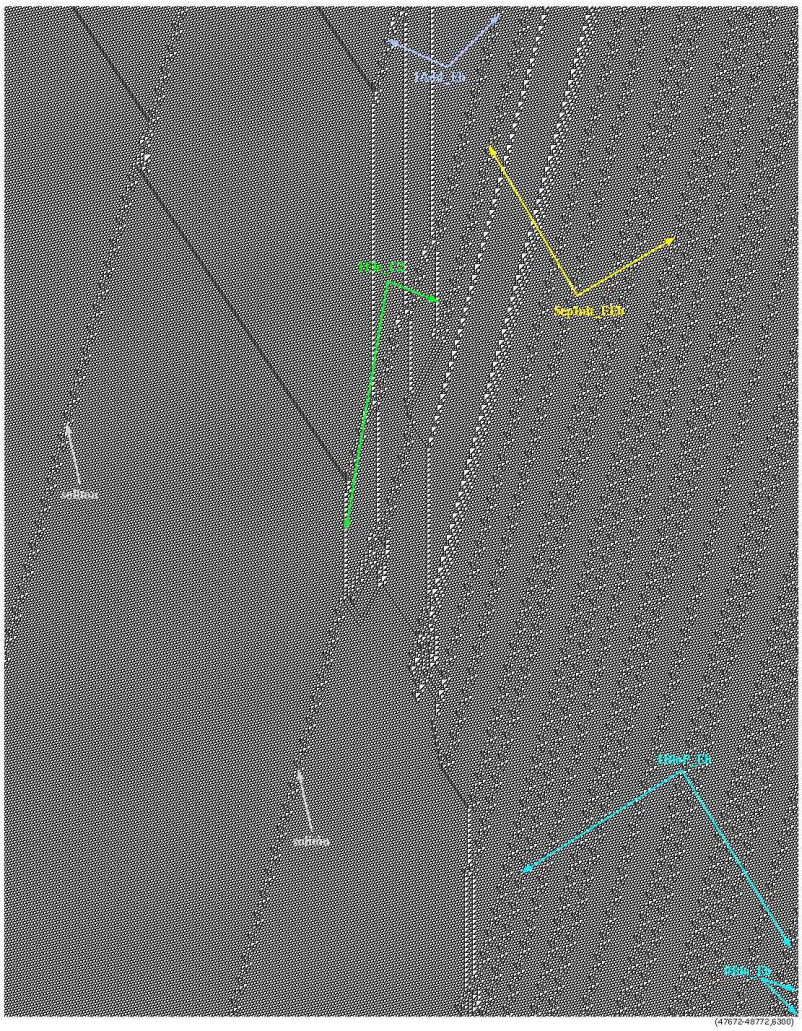

Fig. 17. Constructing an element 1Ele_C_2.

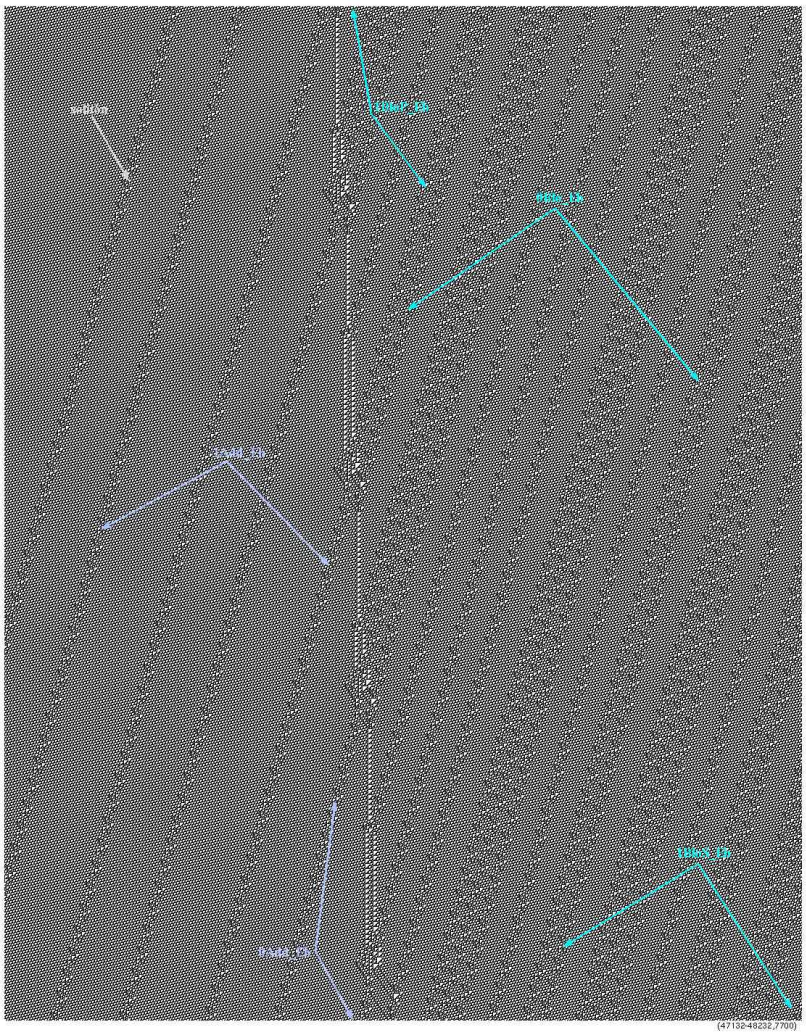

Fig. 18. Constructing an element 1Add_$\bar{E}$.

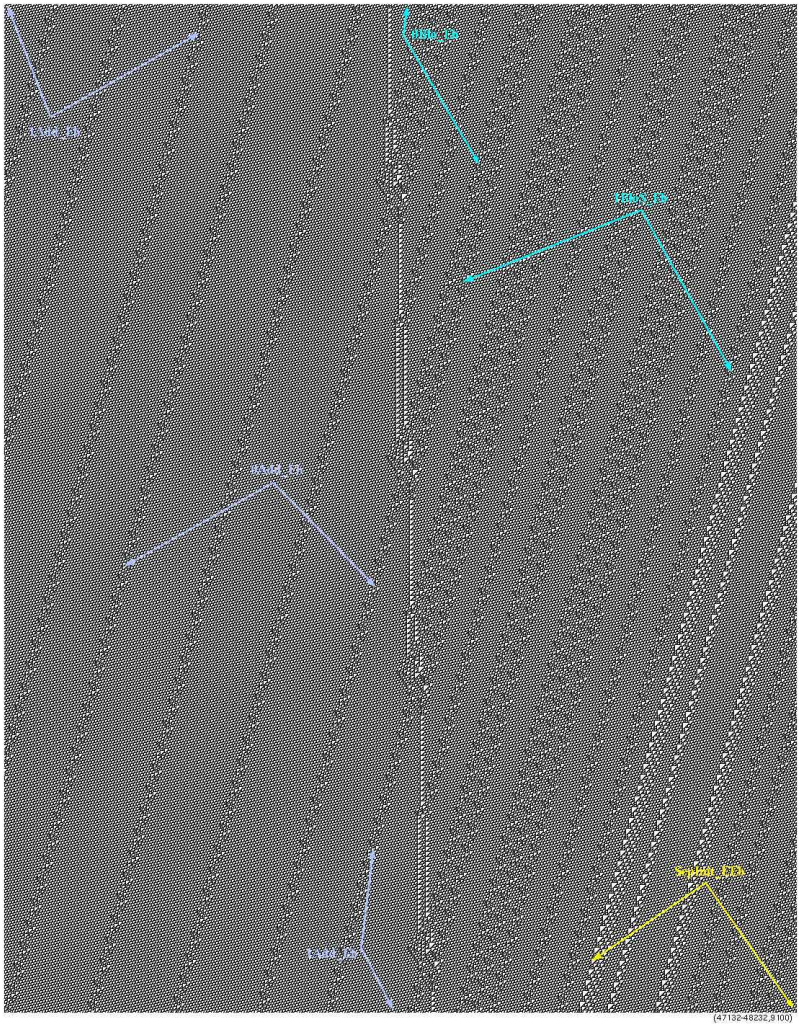

Fig. 19. Constructing an element 0Add_$\bar{E}$.

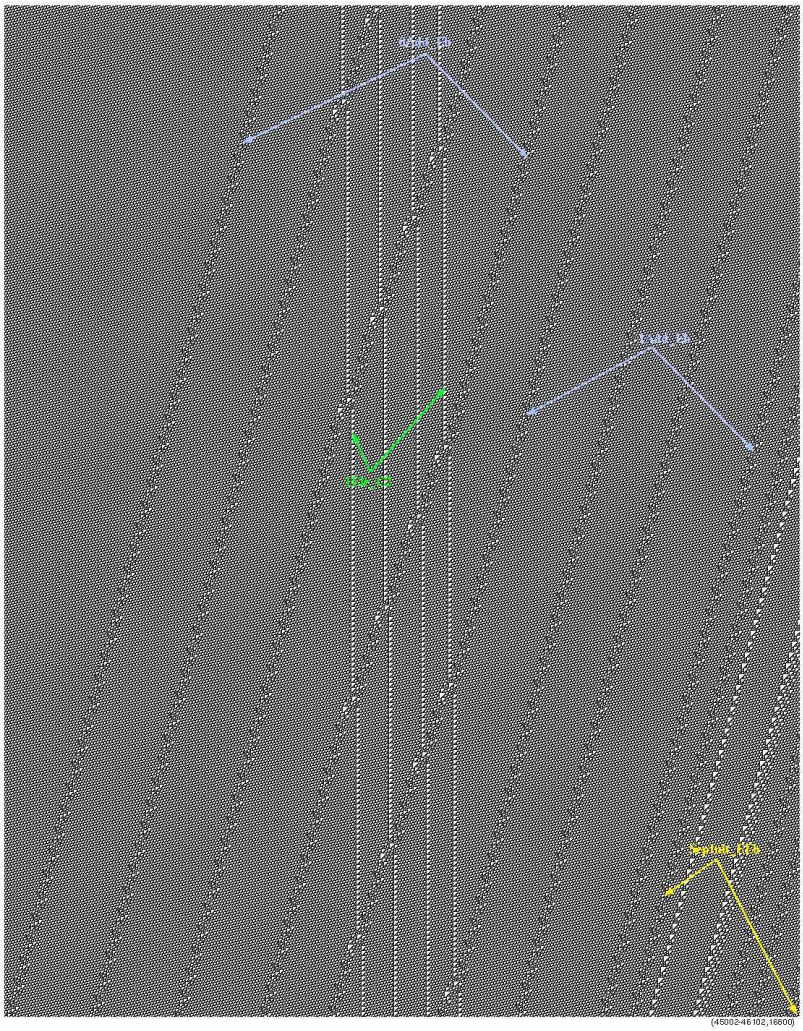

Fig. 20. Transformed data crossing the tape of values.

well an element 0Blo_$\bar{E}$ which changes in an element 0Add_$\bar{E}$ and the standard component 1BloS_$\bar{E}$ is transformed in an element 1Add_$\bar{E}$.

Figure 19 describes the formation of a 0Add_$\bar{E}$ element. Note that when there is not a leader element between these blocks, the system does not produce solitons. In this snapshot, all $\bar{E}$ gliders are processed as data and they will be transformed for a package of 4_A^4 gliders. Thus, soliton patterns are only originated from leader components. We can see completely as well in this snapshot a standard component 1BloS_$\bar{E}$ needed for the suitable behavior of the system (adjusting of the non-symmetric evolution space of Rule 110).

Finally in Fig. 19 a leader element SepInit_$E\bar{E}$ arrives to close a periodic block coming from the right. This periodic block begins with: a separator, a primary black block, a separator, another primary black block, a white block and a standard black block. Therefore, the real period by components to the right can be written as follows:

{SepInit_$E\bar{E}$-1BloP_$\bar{E}$-SepInit_$E\bar{E}$-1BloP_$\bar{E}$-0Blo_$\bar{E}$-1BloS_$\bar{E}$}*.

Figure 20 presents the construction of a 1Ele_C_2 element. In this stage of the evolution, we can see how data is aggregated before they cross the tape with their respective values. Also, these reactions are the same with the element 0Ele_C_2. Also, there is a unique option between C_2 and $\bar{E}$ gliders to cross as solitons because any another reaction does not preserve this property [12]. It is really a laborious task to obtain the global synchronization.

Figure 21 shows a constructed 0Ele_C_2 element and its function in the system. At the top, an element 1Add_$\bar{E}$ is previously produced by a standard component 1BloS_$\bar{E}$ crossing an element 0Ele_C_2. Hence a leader element comes to delete this 0 from the tape and all the subsequent incoming data. In this sequence, there are 1BloP_$\bar{E}$, 0Blo_$\bar{E}$ and 1BloS_$\bar{E}$ elements.

Similar when a 1 is deleted from the tape, when a 0 is deleted, distances are smaller in the first collisions and a T_{14} tile is generated in the process. This difference of distances determines a change of phases which will erase $\bar{E}$ gliders instead of produce C gliders. In order to delete these packages of gliders, a reaction $A^3 \rightarrow \bar{E}$ is used. Figure 22 illustrates how must be erased sequentially packages of data and they arriving order does not affect if they are deleted or pre-transformed. Finally, Figure 23 shows the final stage deleting data and a new period would begin again.

Production rules in CTS establishes for 0 that the first element of the chain must be erased and the other elements are conserved without additional data. If the first value is 1 the first element of the chain is deleted and 10 or 11 are aggregated depending of the k value. This

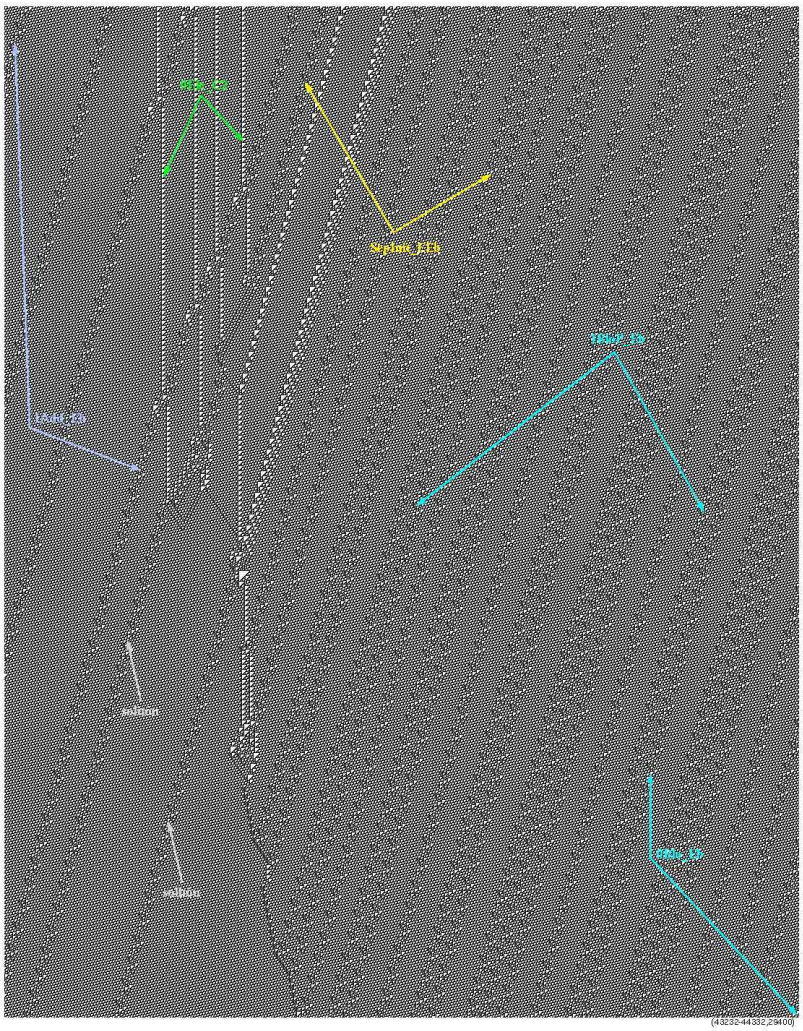

Fig. 21. Deleting a 0Ele_C2 element.

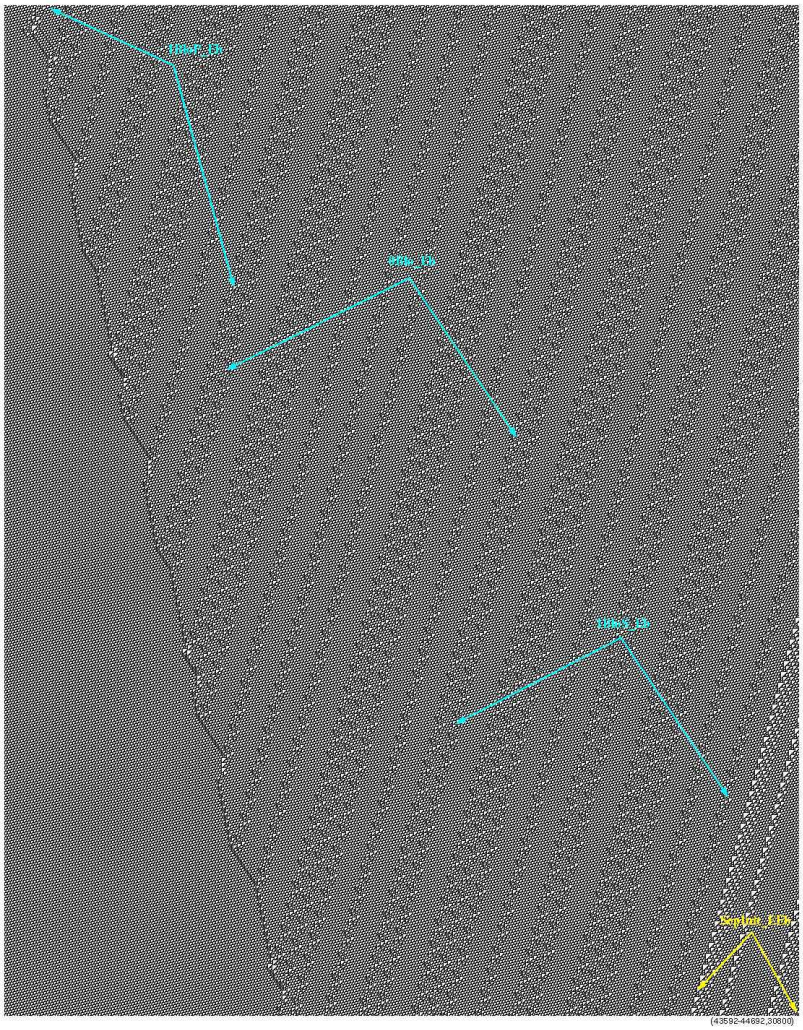

Fig. 22. Deleting unchanged data blocks with the reaction $A^3 \rightarrow \bar{E}$.

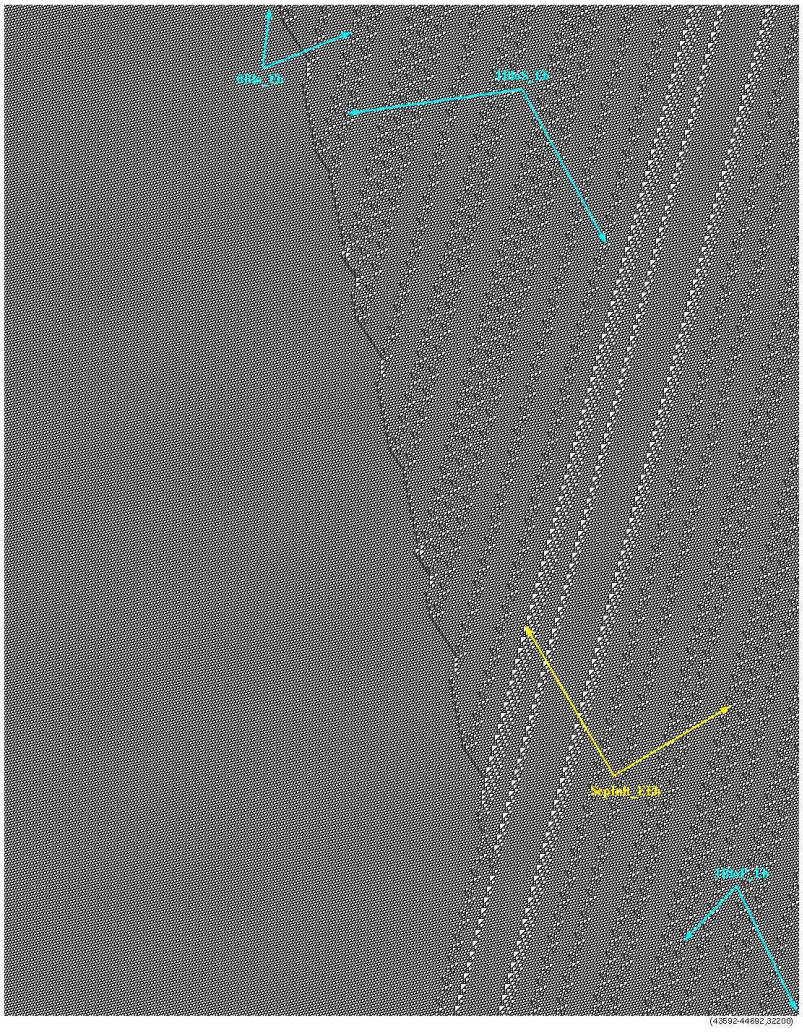

Fig. 23. Final stage deleting data.

behavior is well represented when a separator finds 0 or 1 and deletes it from the tape. If the deleted data was 0, a separator does not allow the production of new data until it finds a new one; on the other case whether the deleted data is 1, a separator aggregates the new elements 11 or 10 that will be transformed later.

This construction is repeated indefinitely. Thus, using this procedure, we can calculate up to the sixth 1 of the sequence 011<1>0 produced by the CTS; verifying that phases f_i_1 are useful to reproduce this system in a different way. Finally, in terms of periodic phases, this CTS can be written as follow:

left: {649e-4_A^4(F_i)}*, for $1 \leq i \leq 3$ in sequential order

center: 246e-1Ele_C2(A,f_1_1)-e-A^3(f_1_1)

right: {SepInit_$E\bar{E}$(#,f_i_1)-1BloP_$\bar{E}$(#,f_i_1)-SepInit_$E\bar{E}$(#,f_i_1)-1BloP_$\bar{E}$(#,f_i_1)-0Blo_$\bar{E}$(#,f_i_1)-1BloS_$\bar{E}$(#,f_i_1)}* (where $1 \leq i \leq 4$ and # represents a particular phase).

5 Final notes

The use of phases f_i_1 allows to identify, control and code gliders from initial conditions in Rule 110. Some questions arise over the universality of Rule 110; one of them is determining if the periodic right side will become periodic into the tape of the CTS. In this case, could we find a periodicity on the tape? How can be connected equivalent Turing machines to CTS? Some variants may be Turing machines [40] and circular Post's machines [41, 42].

Eventually, futures problem to solve in Rule 110 are the implementation of an specific Turing machine using the glider system and looking for possibilities to implement a universal and self-reproducing constructor.

Nowadays, some investigations have found important and related works around of the universality in Rule 110 [43].[17] In the case of the CTS, there are some solved problems like a Fibonnaci sequence done by Chapman (January 2003). On the other hand, Neary and Woods in [44] show that Cook's machine can run in a polynomial time. Morita in [35] proofs how to implement an explicit halt condition in the CTS and how a partitioned CA may simulate any CTS.

Finally, in this paper we have reproduced the necessary components and their initial conditions for a CTS working in Rule 110; explaining as

[17] A list of papers related in Rule 110 and its universality is presented in `http://uncomp.uwe.ac.uk/genaro/Rule110.html`

well each stage of the evolution space with several details, obtaining a successful partial computation; writing the sequence 1110111 on the tape and a leader component at the end with two solitons. This reconstruction is performed in an evolution space of 56,240 cells in 57,400 generations (3,228,176,000 cells); running on a computer Pentium II to 233 mhz, operating system OpenStep and 256MB of RAM (February 2003).[18]

Acknowledgements

In special to Matthew Cook for his kind help four years ago with useful commentaries and for the last interview in Zurich (June 2007). To Kenichi Morita and Martin Davis by their commentaries and close papers. Also, the first author thanks a previous support by CONACyT with registry number 139509 and an actual one by EPSRC grant reference EP/D066174; and the third author thanks the support of CONACyT with reference number CB-2006-1/59422.

References

1. Cook, M. (2004) Universality in Elementary Cellular Automata, *Complex Systems* **15 (1)** 1–40.
2. Wolfram, S. (2002) *A New Kind of Science*, Wolfram Media, Inc., Champaign, Illinois.
3. 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.
4. Cook, M. (1999) Introduction to the activity of rule 110 (copyright 1994-1998 Matthew Cook), `http://w3.datanet.hu/~cook/Workshop/CellAut/Elementary/Rule110/110pics.html`
5. Gardner, M. (1970) Mathematical Games - The fantastic combinations of John H. Conway's new solitaire game Life, *Scientific American* **223** 120–123.
6. Wolfram, S. (1986) *Theory and Aplications of Cellular Automata*, World Scientific Press, Singapore.
7. Lindgren, K. & Nordahl M. (1990) Universal Computation in Simple One-Dimensional Cellular Automata, *Complex Systems* **4** 229–318.
8. Griffeath D. & Moore C. (eds.) (2003) *New Constructions in Cellular Automata*, Oxford University Press.
9. McIntosh, H.V. (1999) Rule 110 as it relates to the presence of gliders, `http://delta.cs.cinvestav.mx/~mcintosh/oldweb/pautomata.html`

[18] `http://uncomp.uwe.ac.uk/genaro/rule110/ctsRule110.html`

10. Martínez, G.J., McIntosh, H.V. & Seck Tuoh Mora, J.C. (2003) Production of gliders by collisions in Rule 110, *Lecture Notes in Computer Science* **2801** 175–182
11. 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.
12. 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.html`
13. McIntosh, H.V. (2000) A Concordance for Rule 110, `http://delta.cs.cinvestav.mx/~mcintosh/oldweb/pautomata.html`
14. Martínez, G.J., McIntosh, H.V., Seck Tuoh Mora, J.C. & Chapa Vergara, S.V. (2007) Rule 110 objects and other collision-based constructions, *Journal of Cellular Automata* **2 (3)** 219–242.
15. McIntosh, H.V. (1991) Linear cellular automata via de Bruijn diagrams, `http://delta.cs.cinvestav.mx/~mcintosh/oldweb/pautomata.html`
16. Voorhees, B.H. (1996) *Computational analysis of one-dimensional cellular automata*, World Scientific Series on Nonlinear Science, Series A, Vol. 15.
17. Voorhees, B.H. (2008) Remarks on Applications of De Bruijn Diagrams and Their Fragments, *Journal of Cellular Automata* **3 (3)** 187–204.
18. von Neumann, J. (1966) *Theory of Self-reproducing Automata* (edited and completed by A. W. Burks), University of Illinois Press, Urbana and London.
19. Codd, E.F. (1968) *Cellular Automata*, Academic Press, Inc. New York and London.
20. Banks, E.R. (1971) Information and transmission in cellular automata, *PhD Dissertion*, Cambridge, MA, MIT.
21. Smith III, A.R. (1971) Simple computation-universal cellular spaces, *J. of the Assoc. for Computing Machinery* **18** 339–353.
22. Berlekamp, E.R., Conway, J.H. & Guy, R.K. (1982) *Winning Ways for your Mathematical Plays*, Academic Press, vol. 2, chapter 25.
23. Lindgren, K. & Nordahl, M.G. (1990) Universal Computation in Simple One-Dimensional Cellular Automata, *Complex Systems* **4** 229–318.
24. Shannon, C.E. (1956) A Universal Turing Machine with Two Internal States, *Automata Studies*, Princeton University Press, 157–165.
25. Adamatzky, A. (ed.) (2003) *Collision-Based Computing*, Springer.
26. Davis, M. (ed.) (1994) *Solvability, Provability, Definability: The Collected Works of Emil L. Post*, Birkhäuser Boston.
27. Post, E.L., Absolutely Unsolvable Problem and Relativity Undecidable Propositions, 345–441, in [26].
28. Davis, M. (ed.) (1965) *The Undecidable*, Raven Press Books.
29. Turing, A.M. (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.
30. Minsky, M. (1967) *Computation: Finite and Infinite Machines*, Prentice Hall.
31. Cocke, J. & Minsky, M. (1964) Universality of Tag Systems With P = 2, *Journal of the Association for Computing Machinery* **11 (1)** 15–20.

32. Post, E.L. (1943) Formal reductions of the general combinatorial decision problem, *American Journal of Mathematics* **65 (2)** 197–215.
33. Aandreraa, S. & Belsnes, D. (1971) Decision problems for tag systems, *Journal of Symblic Logic* **36 (2)** 229–239.
34. Minsky, M. (1961) Recursive unsolvability of Post's problem of tag and other topics in the theory of Turing machines, *Annals of Mathematics* **74** 437–455.
35. Morita, K. (2007) Simple universal one-dimensional reversible cellular automata, *Journal of Cellular Automata* **2** 159–166.
36. Martínez, G.J. (2004) Introduction to OSXLCAU21 System, `http://uncomp.uwe.ac.uk/genaro/papers.html`
37. Kolakoski, W. (1966) Problem 5304. *Amer. Math. Monthly* **73** 681–682.
38. Giles, J. (2002) What kind of science is this?, *Nature* **417** 216–218.
39. Martínez, G.J. (2004) Introduction to Rule 110, `http://uncomp.uwe.ac.uk/genaro/papers.html`
40. Arbib, M.A. (1969) *Theories of Abstract Automata*, Prentice-Hall Series in Automatic Computation.
41. Kudlek, M. & Rogozhin, Y. (2001) New Small Universal Post Machine, *Lecture Notes in Computer Science* **2138** 217–227.
42. Kudlek, M. & Rogozhin, Y. (2001) Small Universal Circular Post Machine, *Computer Science Journal of Moldova* **9 (25)** 34–52.
43. McIntosh, H.V. (2002) Rule 110 Is Universal!, `http://delta.cs.cinvestav.mx/~mcintosh/oldweb/pautomata.html`
44. Turlough, N. & Woods, D. (2006) P-completeness of cellular automaton Rule 110, *Lecture Notes in Computer Science* **4051** 132–143.
45. Wuensche, A. (1999) Classifying Cellular Automata Automatically, *Complexity* **4 (3)** 47–66.

Appendix. Initial condition to reproduce a CTS in Rule 110

This initial condition is a long chain moreover it is available from a digital file (ctsR110.txt) in `http://uncomp.uwe.ac.uk/genaro/rule110/ctsRule110.html`.

Then the succession of components is given in the next table. This succession must be reading up-down and left-right. Every letter e represents an ether sequence in phase f_1_1 therefore just we must write it as e (where e represents $e = 11111000100110$).[19]

[19] `http://uncomp.uwe.ac.uk/genaro/rule110/listPhasesR110.txt`

left	center	right
. . .-217e-	1Ele_C_2(A,f$_1$_1)-	SepInit_$E\bar{E}$(C,f$_3$_1)-
4_A^4(F2)-	e-	1Blo_$\bar{E}$P(C,f_4_1)-
649e-	3A(f$_1$_1)-	SepInit_$E\bar{E}$(C,f$_3$_1)-
4_A^4(F1)-		1Blo_$\bar{E}$P(C,f$_4$_1)-
649e-		0Blo_$\bar{E}$(C,f$_4$_1)-
4_A^4(F3)-		1Blo_$\bar{E}$S(A,f$_4$_1)-
649e-		SepInit_$E\bar{E}$(A,f$_3$_1)(2)-
4_A^4(F2)-		1Blo_$\bar{E}$P(F,f$_1$_1)-
649e-		SepInit_$E\bar{E}$(A,f$_3$_1)(2)-
4_A^4(F1)-		1Blo_$\bar{E}$P(F,f$_1$_1)-
649e-		0Blo_$\bar{E}$(E,f$_4$_1)-
4_A^4(F3)-		1Blo_$\bar{E}$S(C,f$_4$_1)-
216e-		e-
		SepInit_$E\bar{E}$r(B,f$_1$_1)(2)-
		1Blo_$\bar{E}$P(F,f$_3$_1)-
		e-
		SepInit_$E\bar{E}$(B,f$_1$_1)(2)-
		217e-. . .

Finally our reconstruction has the next explicit sequence of bits; displayed by each particular element. We are omitting in the type trivial repetitions of the same sequence, for instance when there are ether configurations. In this case, ne means n copy of e ether chains.

. . . - 217e = 217e(f$_1$_1)

4_A^4(F2) = 11111000111011100010011011111000100110111110001001101111100011
00110111110001001101111100010011011111000100110111110001001100
11111000100110111110001001101111100010011011111000100110111111
00010011011111000100110111110001001101111100010011011111000100
01101111100010011011111000100110111110001001101111100010011101
11110001001101111100010011011111000100110111110001001101111110
00100110111110001001101111100010011011111011011111000100110111
11110001001101111100010011011111000100110111110001001101111110
00100110111110001001101111100010011011111000100110111110001001100
11011111000100110111110001001101111100010011011111000100110111
11100010011011111000100110111110001001101111100010011011111100
01001101111100010011011111000100110111110001001101111100010011
10111110001001101110100110111110001001101111100010011011111100
01001101111100010011011111000100110111110001001101111100010011
10111110001001101111100010011011111000100110111110001001101111
11000100110111110001001101111100010011011111000100110111111000

$$
\begin{aligned}
&1001101111100010011011111000100110111110001001101111100010011\\
&0111110001001101111100010011011111000100110111110001001101111\\
&1000100110111110001110110001001 10
\end{aligned}
$$

$649e = 649e(\mathrm{f}_1\_1)$

$$
\begin{aligned}
4\_A^4(\mathrm{F1}) = {} & 1111101110111110001001101111100010011011111000100110111110001\\
&0011011111000100110111110001001101111100010011011111000100110\\
&1111100010011011111000100110111110001001101111100010011011111\\
&0001001101111100010011011111000100110111110001001101111100010\\
&0110111110001001101111100010011011111000100110111110001001101\\
&1111000100110111110001001101111100010011011111000100110111110\\
&0010011011111000100110111111000100110111010011011111000100 1101\\
&1111000100110111110001001101111100010011011111000100110111110\\
&0010011011111000100110111110001001101111100010011011111000100\\
&1101111100010011011111000100110111110001001101111100010011011\\
&1110001001101111100010011011111000100110111110001001101111100\\
&0100110111110001001101111100010011011111000100110111110001001\\
&1011111000111011100010011011111000100110111110001001101111100\\
&0100110111110001001101111100010011011111000100110111110001001\\
&1011111000100110111110001001101111100010011011111000100110111\\
&1100010011011111000100110111110001001101111100010011011111000\\
&1001101111100010011011111000100110111110001001101111100010011\\
&0111110001001101111100010011011111000100110111110001001101111\\
&1000100110111110 1110
\end{aligned}
$$

$649e = 649e(\mathrm{f}_1\_1)$

$$
\begin{aligned}
4\_A^4(\mathrm{F3}) = {} & 1111100010011011101001101111100010011011111000100110111110001\\
&0011011111000100110111110001001101111100010011011111000100110\\
&1111100010011011111000100110111110001001101111100010011011111\\
&0001001101111100010011011111000100110111110001001101111100010\\
&0110111110001001101111100010011011111000100110111110001001101\\
&1111000100110111110001001101111100010011011111000100110111110\\
&0010011011111000100110111111000100110111000111011100010011 01\\
&1111000100110111110001001101111100010011011111000100110111110\\
&0010011011111000100110111110001001101111100010011011111000100\\
&1101111100010011011111000100110111110001001101111100010011011\\
&1110001001101111100010011011111000100110111110001001101111100\\
&0100110111110001001101111100010011011111000100110111110001001\\
&1011111000100110111110111011011111000100110111110001001101111100\\
&0100110111110001001101111100010011011111000100110111110001001
\end{aligned}
$$

$$
\begin{aligned}
&10111110001001101111100010011011111000100110111110001001101111\\
&11000100110111110001001101111100010011011111000100110111111000\\
&10011011111000100110111110001001101111100010011011111000100110011\\
&01111100010011011111000100110111110001001101111100010011011111\\
&10001001101111100010011011101001 10
\end{aligned}
$$

$649e = 649e(\mathrm{f_1\_1})$

$$
\begin{aligned}
4\_A^4(\mathrm{F2}) = {}&11111000111011100010011011111000100110111110001001101111100011\\
&00110111110001001101111100010011011111000100110111110001001101 10\\
&11111000100110111110001001101111100010011011111000100110111111\\
&00010011011111000100110111110001001101111100010011011111000101\\
&01101111100010011011111000100110111110001001101111100010011011\\
&11110001001101111100010011011111000100110111110001001101111110\\
&00100110111110001001101111100010011011111011101111100010011011\\
&11110001001101111100010011011111000100110111110001001101111110\\
&00100110111110001001101111100010011011111000100110111110001001 00\\
&11011111000100110111110001001101111100010011011111000100110011\\
&11100010011011111000100110111110001001101111100010011011111100\\
&01001101111100010011011111000100110111110001001101111110001001\\
&10111110001001101110100110111110001001101111100010011011111100\\
&01001101111100010011011111000100110111110001001101111110001001\\
&10111110001001101111100010011011111000100110111110001001101111\\
&11000100110111110001001101111100010011011111000100110111111000\\
&10011011111000100110111110001001101111100010011011111000100110011\\
&01111100010011011111000100110111110001001101111100010011011111\\
&10001001101111100011101110001001 10
\end{aligned}
$$

$649e = 649e(\mathrm{f_1\_1})$

$$
\begin{aligned}
4\_A^4(\mathrm{F1}) = {}&11111011101111100010011011111000100110111110001001101111100011\\
&00110111110001001101111100010011011111000100110111110001001101 10\\
&11111000100110111110001001101111100010011011111000100110111111\\
&00010011011111000100110111110001001101111100010011011111000101\\
&01101111100010011011111000100110111110001001101111100010011011\\
&11110001001101111100010011011111000100110111110001001101111110\\
&00100110111110001001101111100010011011101001101111100010011011\\
&11110001001101111100010011011111000100110111110001001101111110\\
&00100110111110001001101111100010011011111000100110111110001001 00\\
&11011111000100110111110001001101111100010011011111000100110011\\
&11100010011011111000100110111110001001101111100010011011111100\\
&01001101111100010011011111000100110111110001001101111110001001
\end{aligned}
$$

$$
\begin{aligned}
&1011111000111011100010011011111000100110111110001001101111100\\
&0100110111110001001101111100010011011111000100110111110001001\\
&1011111000100110111110001001101111100010011011111000100110111\\
&1100010011011111000100110111110001001101111100010011011111000\\
&1001101111100010011011111000100110111110001001101111100010011\\
&0111110001001101111100010011011111000100110111110001001101111\\
&1000100110111110111 0
\end{aligned}
$$

$649e = 649e(\mathrm{f_1\_1})$

$$
\begin{aligned}
4\_A^4(\mathrm{F3}) = {}&1111100010011011101001101111100010011011111000100110111110001\\
&0011011111000100110111110001001101111100010011011111000100110\\
&1111100010011011111000100110111110001001101111100010011011111\\
&0001001101111100010011011111000100110111110001001101111100010\\
&0110111110001001101111100010011011111000100110111110001001101\\
&1111000100110111110001001101111100010011011111000100110111110\\
&0010011011111000100110111110001001101111100011101110001001101\\
&1111000100110111110001001101111100010011011111000100110111110\\
&0010011011111000100110111110001001101111100010011011111000100\\
&1101111100010011011111000100110111110001001101111100010011011\\
&1110001001101111100010011011111000100110111110001001101111100\\
&0100110111110001001101111100010011011111000100110111110001001\\
&1011111000100110111110111011111000100110111110001001101111100\\
&0100110111110001001101111100010011011111000100110111110001001\\
&1011111000100110111110001001101111100010011011111000100110111\\
&1100010011011111000100110111110001001101111100010011011111000\\
&1001101111100010011011111000100110111110001001101111100010011\\
&0111110001001101111100010011011111000100110111110001001101111\\
&1000100110111110001001101110100110
\end{aligned}
$$

$216e = 216e(\mathrm{f_1\_1})$

$$
\begin{aligned}
1\mathrm{Ele}\_C_2(\mathrm{A,f_1\_1}) = {}&1111100000010011011111000100110111110001001101111100000\\
&0100110111110001001101111100010011011110000001001101111\\
&1100010011011111000111011010
\end{aligned}
$$

$e = e(\mathrm{f_1\_1})$

$A^3(\mathrm{f_1\_1}) = 111110111000100110$

$$
\begin{aligned}
\mathrm{SepInit}\_E\bar{E}(\mathrm{C,f_3\_1}) = {}&1111100010011001000111001100011100110111110001001101\\
&0011111110101111100010011011111000100110111110001001
\end{aligned}
$$

10111110001001100100011100110001111100010011011111001
01001100111011100110001001101111100010011010001101001
11000100110111110001001101111100010011011000101111101
00100110111110011111011100110111110001001101111100011
0011011101101110 1011100110

1Blo_$\bar{E}$P(C,f$_4$_1) = 11111100111100010011011111000111011111010011111000100111
01111100010011011100011101100010011011111000101101000011
10111110001001101111100010011011111100001101110011011111
10001001101111100010011011111001110000010011011101101111
01011100110111110001001101111100010011011101110011000011
00110111110001011010000110111110111111011111010111110001
10011011111000100110111110011111011100110111110001001100
11111000100110111011011101011100110

SepInit_$E\bar{E}$(C,f$_3$_1) = 11111000100110010001110011000111001101111100010011011
00111111101011111000100110111110001001101111100010011
10111110001001100100011100110001111100010011011111001
01001100111011100110001001101111100010011010001101001
11000100110111110001001101111100010011011000101111101
00100110111110011111011100110111110001001101111100011
00110111011011101011100110

1Blo_$\bar{E}$P(C,f$_4$_1) = 11111100111100010011011111000111011111010011111000100111
01111100010011011100011101100010011011111000101101000011
10111110001001101111100010011011111100001101110011011111
10001001101111100010011011111001110000010011011101101111
01011100110111110001001101111100010011011101110011000011
00110111110001011010000110111110111111011111010111110001
10011011111000100110111110011111011100110111110001001100
11111000100110111011011101011100110

0Blo_$\bar{E}$(C,f$_4$_1) = 11111100111100010011011111000100110111110001001101111100
01110110010110111110111111011111010111110000101100100110
11111000100110111110001001101110111001100001001101111100
01001101111100010011011111100111100010011011111000100110
11111000100110111110011100000100110111110001001101111100
01001101111100111110111001101110001110110001001101111100
01011010000110111110001001101111100010011011111100001101
11001101111100010011011111000100110111110011100000100110

1Blo_$\bar{E}$S(A,f$_4$_1) = 11101101110101110011011111000100110111110001001101110111

$$
\begin{aligned}
&1001100001001101111100010110100001101111101111110111110\\
&1011111000100110111110001001101111100111111011001101111\\
&1000100110111111000100110111011011101011100110111111 0011\\
&1100010011011111000100110111110001001101111100011101100\\
&1011011111011111101111101011111000010110010011011111000\\
&1001101111100010011011011100110000100110111110001001 10110\\
&1111100010011011111100111000100110
\end{aligned}
$$

$$
\begin{aligned}
\text{SepInit\_}E\bar{E}\text{(A,f}_3\text{\_1)(2)} = {}&11111000100110000001111100000111110101111100010011\\
&01111111111000111110001001101111100010011011111000\\
&10011011111000100110000001111100000111110001001101\\
&11110001001101100010111110001001101111100010011001\\
&10001111110001001101111100010011011111000100110111\\
&00011101100010011011101110011000010011011111000100\\
&11011111000100110111111001111000100110
\end{aligned}
$$

$$
\begin{aligned}
\text{1Blo\_}\bar{E}\text{P(F,f}_1\text{\_1)} = {}&11111010011011110011011111000100110111110011100000100 11\\
&01111100010011011111000101101000011011111011111101111 10\\
&10111110001001101111100010011011111001111101110010 01011111\\
&10001001101111100010011011101101110101110011011111100 11\\
&11000100110111110001001101111100010011011111000110110 0\\
&10110111110111111011111010111110000101100100110111110 00\\
&10011011111000100110111011100110000100110111110001001 00110\\
&11111000100110111111001111000100110
\end{aligned}
$$

$$
\begin{aligned}
\text{SepInit\_}E\bar{E}\text{(A,f}_3\text{\_1)(2)} = {}&11111000100110000001111100000111110101111100010011\\
&01111111111000111110001001101111100010011011111000\\
&10011011111000100110000001111100000111110001001101\\
&11110001001101100010111110001001101111100010011001\\
&10001111110001001101111100010011011111000100110111\\
&00011101100010011011101110011000010011011111000100\\
&11011111000100110111111001111000100110
\end{aligned}
$$

$$
\begin{aligned}
\text{1Blo\_}\bar{E}\text{P(F,f}_1\text{\_1)} = {}&11111010011011110011011111000100110111110011100000100 11\\
&01111100010011011111000101101000011011111011111101111 10\\
&10111110001001101111100010011011111001111101110010 01011111\\
&10001001101111100010011011101101110101110011011111100 11\\
&11000100110111110001001101111100010011011111000110110 0\\
&10110111110111111011111010111110000101100100110111110 00\\
&10011011111000100110111011100110000100110111110001001 00110\\
&11111000100110111111001111000100110
\end{aligned}
$$

$0\text{Blo\_}\bar{E}(\text{E},\text{f}_4\text{\_1}) =$ 111110100110111100110111110001001101111100010011011111100
0100011001100011111000010110010011011111000111011111110010
111110001001101111100010011011111000111011001011011111100
0100110111110001001101111101001101111001101111100010011 0
1111100010011011101101110101110011011111000100110111111 00
0100110111011100110000100110111110001011010000110111110 1
1111101111101011111000100110111110001001101111100111110 1
1100110111110001001101111100010011011101101110101110011 0

$1\text{Blo\_}\bar{E}\text{S}(\text{C},\text{f}_4\text{\_1}) =$ 111111001111000100110111110001001101111100010011011111 0
0011101100101101111101111110111110101111100001011001001
1011111000100110111110001001101110111001100001001101111
1000100110111110001001101111100111100010011011111 01001
1011110011011111000100110111110001001101111100010001100
1100011111000010110010011011111000111011111001011111000
1001101111100010011011111000111011001011011111000100110
111110001001101111101001101111 00110

$e = e(\text{f}_1\text{\_1})$

$\text{SepInit\_}E\bar{E}(\text{B},\text{f}_1\text{\_1})(2) =$ 1111101000011100110001110011000100110111110001000 0
0001101110011011111000100110111110001001101111100 0
1001101111101000011100110001110011011111000100110 1
1111000010110010011011111000100110111110111010111 0
0011011111000100110111110001001101111100011101111 1
0010111110001110111110100111110001001101111100010 0
1101111100010001111 10110

$1\text{Blo\_}\bar{E}\text{P}(\text{F},\text{f}_3\text{\_1}) =$ 111110001001101110001011110001001101111100010011111110 0
011111000100110111110001001101100000100110001001101000 1
101001100010011011111000100110111110001001111100110111 1
100010011011111000100110111110001110000111000111110001 0
001111101101111100010011011111000100110111110001001101 1
111001110000010011010001101001100010011001100011111100 0
100110111110001001101111100011101111101001111100010011 0
1111100010011011111000100011111 0110

$e = e(\text{f}_1\text{\_1})$

$\text{SepInit\_}E\bar{E}(\text{B},\text{f}_1\text{\_1})(2) =$ 1111101000011100110001110011000100110111110001000 0
0001101110011011111000100110111110001001101111100 0
1001101111101000011100110001110011011111000100110 1

11110000101100100110111110001001101111101110101110
00110111110001001101111100010011011111000111011111
00101111100011101111101001111100010011011111000100
1101111100010001111110110

$217e = 217e(\mathrm{f_1\_1})$ - ...