Journal of Cellular Automata, Vol. 13, pp. 325–346
Reprints available directly from the publisher
Photocopying permitted by license only

Conservative Computing in a One-dimensional Cellular Automaton with Memory

GENARO J. MARTÍNEZ[1,2,*] AND KENICHI MORITA[3]

[1] *Unconventional Computing Center, University of the West of England, Bristol, United Kingdom*

[2] *Escuela Superior de Cómputo, Instituto Politécnico Nacional, México*

[3] *Hiroshima University, Higashi Hiroshima, Japan*
E-mail: km@hiroshima-u.ac.jp

Received: June 2, 2017. Accepted: June 27, 2017.

We propose a scheme to simulate Fredkin gates in a one-dimensional cellular automaton with memory by collision of particles, which is a moving pattern in this cellular space. Operations by collisions are confined in a black box with ballistic interaction, solitons and other collisions. We made a systematic analysis of binary collisions, i.e., collisions of two particles with different phases. They are used for handling these particles and obtaining the final outputs.

Keywords: Fredkin gates, elementary cellular automata, memory, particles, collisions, unconventional computing.

1 PRELIMINARIES

In the study of universal cellular automata (CA) — similar to the problem of small universal Turing machines — a number of parameters are reduced constantly: the number of states, the neighbourhood radius, the size of a configuration, and the dimension. The present elementary CA with memory (ECAM) gives a framework for implementing Fredkin gates based on a one-dimensional two-state CA having the neighborhood of radius 1. There, only one kind of glider is used to implement logical operations.

* Contact author: E-mail: genaro.martinez@uwe.ac.uk

On one-dimensional universal CA, there exist a number of relevant and interesting reductions [23].* Let (s, r) denote the order of a CA, where s is the number of states, and r is the neighborhood radius. Smith III proved in 1971 [25] that a one-dimensional CA of order (18, 1) is able to simulate a universal Turing machine. Albert and Culik II in 1987 [1] showed another universal CA of (14, 1) with a totalistic local function. Later, Lindgren and Nordahl in 1990 [10] have reduced this order to (7, 1). All of these CAs simulate a universal Turing machine, and their characteristic is that the spatial constructions were done by the interaction of signals not by the collisions of gliders. In 2004 Cook proved that ECA rule 110 (order (2,1)) is universal, where collisions of hundred of gliders are used to control a cyclic tag system [6,30]. As for reversible CAs, a one-dimensional universal reversible CA of order (24, 1/2) controlled by signals was reported by Morita in 2011 [22].

In this paper, we research a complex ECAM rule $\phi_{R22maj:4}$ (discovered in [11]). This automaton is able to simulate computable devices, such as: elastic collisions, solitons, and logic gates. Also, we present a proposal to simulate Fredkin gates in one dimension using this automaton. This ECAM is a derivative of the ECA rule 22 that is the projection of the Game of Life in one dimension [19]. The ECAM rule $\phi_{R22maj:4}$ uses just one kind of particle (and its reflection) to develop computable devices.

This paper is organized as follows. Section two presents basic concepts. Section three offers a brief discussion of ECA rule 22. Section four discusses ECAM rule $\phi_{R22maj:4}$ including its dynamics, basic gates, and the proposal of Fredkin gates in one dimension. Section five displays conclusions and future works.

2 ONE-DIMENSIONAL CELLULAR AUTOMATA

2.1 Basic notation

One-dimensional CA can be represented as a tuple as follows: $< \Sigma, \varphi, \mu, c_0 >$. The system evolve on an array of *cells* x_i, where $i \in Z$ (integer set) and each cell takes a value from the *finite alphabet* Σ, thus $x_i \in \Sigma$. A finite sequence of cells $\{x_i\}$ represents a *global configuration* c, such that $c \in \Sigma^*$. This way, the set of finite configurations of length n is represented as Σ^n.

Cell states in a configuration $c(t)$ are updated at the next configuration $c(t+1)$ simultaneously by a *local function* as:

$$\varphi(x_{i-r}^t, \ldots, x_i^t, \ldots, x_{i+r}^t) = x_i^{t+1}. \quad (1)$$

* Complex Cellular Automata Repository http://uncomp.uwe.ac.uk/genaro/Complex_CA_repository.html

Thus the local function determines the relation $\varphi : \Sigma^{\mu} \to \Sigma$, where μ represents the *neighbourhood* of φ with $2r + 1$ *neighbours*, and $r \in Z^{+}$. So, the set of states of neighbour cells is Σ^{μ}, and the space of evolution rules is $\Sigma^{\Sigma^{\mu}}$. ECAs are defined for the parameters $|\Sigma| = 2$, and $r = 1$ [29].

An *evolution* is represented by a sequence of finite configurations $\{c^t\}$ given by the global mapping, $\Phi : \Sigma^n \to \Sigma^n$. Thus the *global relation* is given as the function between configurations $\Phi(c^t) = c^{t+1}$.

2.2 Cellular automata with memory

Conventional CA are ahistoric (memoryless): i.e., the new state of a cell depends on the neighbourhood state solely at the preceding time step of φ. A CA with *memory* (CAM) 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. CAMs were introduced by Sanz and condensed basically in a book exploring an ample kind of memories on different kinds of CAs [24].

When using the memory we need to specify a kind of memory function ϕ as follows:

$$\phi(x_i^{t-\tau}, \ldots, x_i^{t-1}, x_i^t) = s_i. \tag{2}$$

The parameter $\tau < t$ specifies the degree of memory backwards, and the state $s_i \in \Sigma$ is determined by the function ϕ of the series of states x_i in the memory up to the time step t. To make an evolution step we apply the original rule as:

$$\varphi(s_{i-r}^t, \ldots, s_i^t, \ldots, s_{i+r}^t) = x_i^{t+1}. \tag{3}$$

The main feature in CAM is that the mapping φ remains unaltered, while historic memory of all the past iterations is retained by featuring each cell as a summary of its past states by ϕ. This way, cells *canalise* memory to the map φ.

As an example, we can consider memory function ϕ as a *majority memory* ϕ_{maj}, where in case of a tie we will take the last value x_i. In the case of three variables, the function ϕ_{maj} represents the classic majority function [20] as follows:

$$\phi_{maj}(a, b, c) = (a \wedge b) \vee (b \wedge c) \vee (c \wedge a). \tag{4}$$

Note that memory is a simple function added to a CA. Particularly, an ample and systematic analysis on ECAM with majority, minority, and parity memories was done in [11].

3 ELEMENTARY CELLULAR AUTOMATON RULE 22

ECA rule 22 is a projection in one dimension of the famous two-dimensional CA Conway's Game of Life (for details see [19]). Although its global behaviour does not display much information to deduce some complex dynamics, rule 22 is classified as a chaotic rule (class III) in the Wolfram's classification [29,30].

Rule 22 can be stated as a relation similar to the Game of Life [4] given in the next conditions [19]:

I. Born: a dead cell x_i at the time t will be born in $t+1$ if there is just one live neighbour.

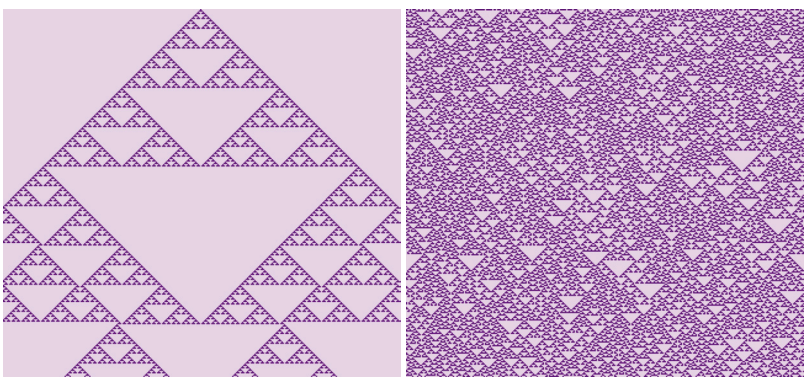
II. Survival: an alive cell x_i at the time t will survive in $t+1$ if there is not live neighbours.

III. Dead by overcrowding: an alive cell x_i at the time t will be dead in $t+1$ if there is just two or one live neighbours.

Such a relation covers each condition of the Game of Life. Nevertheless, from a quick exploration in the one-dimensional dynamics, it does not exhibit complex behaviour. Figure 1 shows classic dynamics in ECA rule 22 beginning with a cell in state one and a random initial condition.

Rule 22 is an ECA evolving in one dimension of order $|\Sigma| = 2$ and neighbourhood radius $r = 1$. Thus the local rule φ is defined as follows:

$$\varphi_{R22} = \begin{cases} 1 & \text{if} \quad 100, 010, 001 \\ 0 & \text{if} \quad 111, 110, 101, 011, 000 \end{cases} . \tag{5}$$



(a) (b)

FIGURE 1
Classic dynamics in ECA rule 22 from (a) the configuration with only one cell in state 1, and (b) a random configuration with the ratio of 50% of state 1, on a ring of 323 cells to 300 generations.

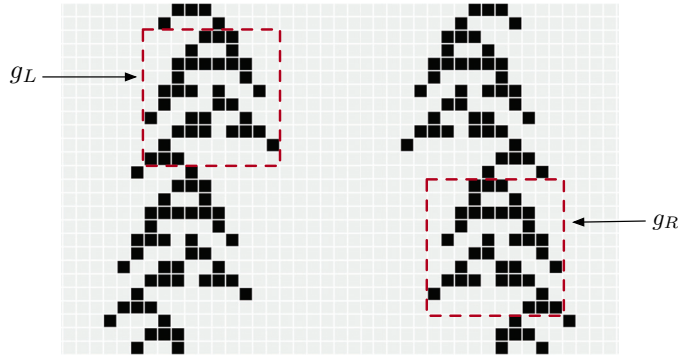


FIGURE 2
g_L and g_R particles evolving in $\phi_{R22maj:4}$.

The local function φ_{R22} has a probability of 37.5% to get states 1 in the next generation and consequently a higher probability to get state 0 in the next generation. Of course, it is the same fixed point value for the Game of Life [18].

4 ELEMENTARY CELLULAR AUTOMATON WITH MEMORY RULE $\phi_{R22MAJ:4}$

In [11, 16] ECA rule 22 with memory is identified as a 'strong' class into ECAM classification. ECAM rule $\phi_{R22maj:4}$ is defined by the ECA rule 22 ($R22$), the majority memory (maj) and degree of memory backward $\tau = 4$. This way, to represent functions with memory and a ECA is as follows: $\phi_{ECAm:\tau}$. Such that ECA represents the decimal notation of a specific rule, m is the kind of memory, and τ the degree of memory backward.

The main characteristic is that rule $\phi_{R22maj:4}$ has only one kind of particle (and its reflection) traveling on its evolution space, as is illustrated in Figure 2. The set of particles $\mathcal{G}_{\phi_{R22maj:4}} = \{g_L, g_R\}$ are defined in a perfect square volume of 11×11 cells, its properties are described in the Table 1.

Figure 3 displays a typical evolution from a random initial condition in ECAM $\phi_{R22maj:4}$. In this evolution, we can see how non-trivial patterns travel

particle	period	shift	velocity	mass
g_R	11	2	2/11	38
g_L	11	−2	−2/11	38

TABLE 1
Properties of particles of ECAM $\phi_{R22maj:4}$.

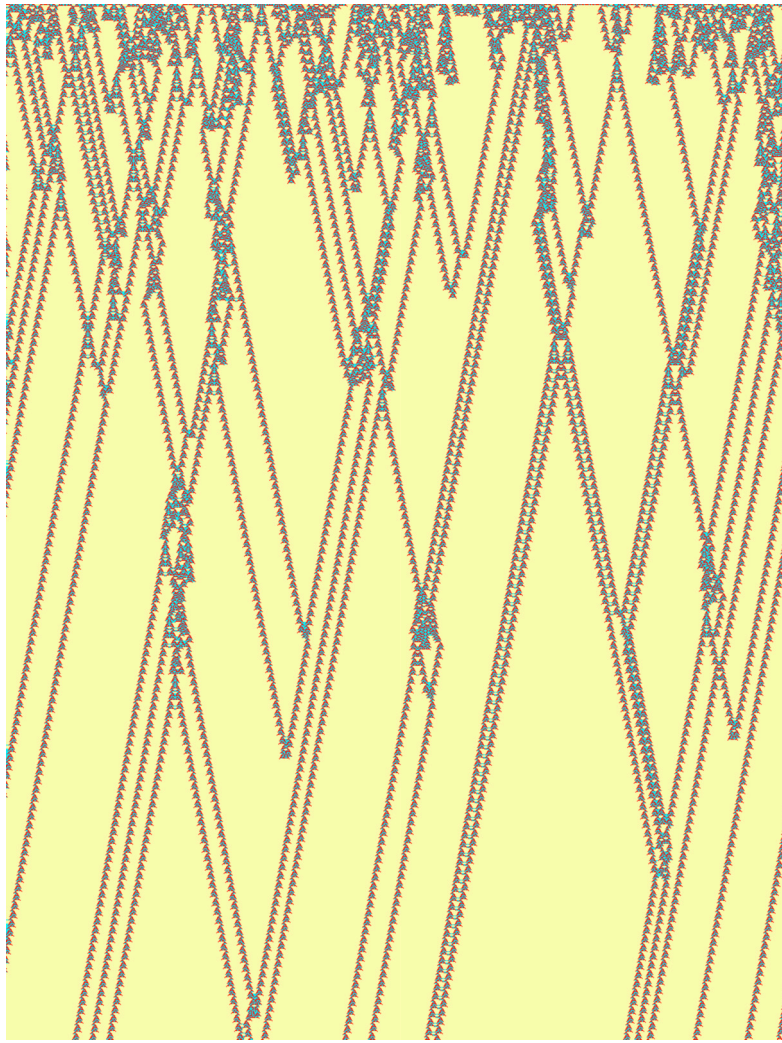


FIGURE 3
Typical dynamics of $\phi_{R22maj:4}$ from a random configuration with the ratio of 37% on a ring of 968 cells for 1274 generations. Filter and a selection of colors are used to improve the view of particles.

and interact by collisions. Also, collisions continue for a large number of generations. In this case the evolution lasts than 1000 steps, and their reactions do not become stationary.

We will start to design constructions based on collisions of particles. Initially, we have done a systematic analysis of binary collisions that is given

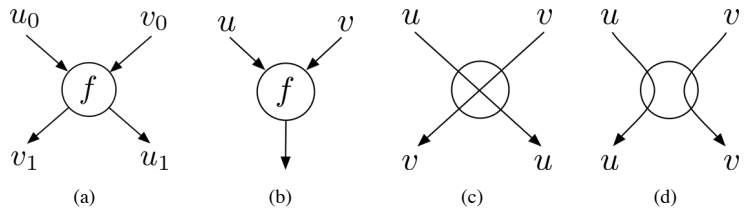


FIGURE 4
Toffoli's notation for basic ballistic collision with particles [28].

at the appendix. In the appendix we have 44 different binary reactions that offer an attractive number of possibilities for design devices, as computable functions, nano-devices, or collision theory in $\phi_{R22maj:4}$.

4.1 Ballistic collisions

Fredkin thought that two Boolean states were independently conserved by the logic gates with the intention that this could be represented as *balls* or atoms shall preserve their identity through of collisions. Thus Fredkin came up the *Billiard-Ball Model of Computation* consisting of elastic balls and flat mirrors [28]. Later Margolus developed a CA to implement the billiard-ball model, such CA is a block CA (BCA) which shows universal computing by collisions of soft spheres simulating a Fredkin gate in [13] and as crystalline computation in [14].

Ballistic collisions of particles permit to represent functions of two arguments (general signal-interaction scheme is conceptualized in Figure 4a), as follows:

1. $f(u, v)$ is a product of one collision (Figure 4b);
2. $f_i(u, v) \mapsto (u, v)$ identity (Figure 4c);
3. $f_r(u, v) \mapsto (v, u)$ reflection (Figure 4d);

where Figure 4a represents a result without preserving some identity of u and v, while f (in Figure 4b) displays the general scheme where two arguments are preserved.

ECAM $\phi_{R22maj:4}$ can reproduce ballistic collisions and preserve them given a good synchronisation. Figure 5 simulates identity collisions. All collisions preserve a reaction like soliton fixing distances between these particles or changing the number and position of them and they are preserved forever. Computing with solitons was explored amply by Steiglitz in [9,27], a computation to implement a carry-ripple adder [26] can be developed in $\phi_{R22maj:4}$ by solitonic reactions.

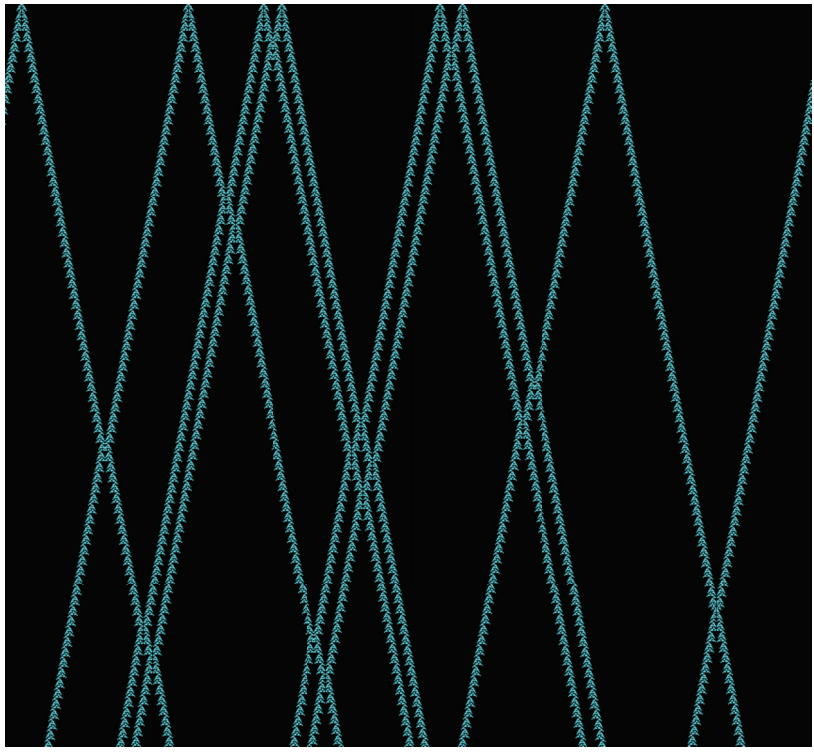


FIGURE 5
Solitonic collisions in ECAM $\phi_{R22maj:4}$. 14 particles are synchronised to simulate the function identity $f_i(u, v) \mapsto (u, v)$ from different intervals. Solitons are preserved forever.

Figure 6 simulates elastic collisions $f_r(u, v) \mapsto (v, u)$ with a pair of particles. This way, a pair of gliders reflect identically during all reactions. These elastic collisions are robust on any change of phase, position or number of them; 13 pair of particles in different positions preserve the elastic collisions forever.

Ballistic ball reactions are *interaction gates*. The interaction gate is a function with two inputs and four possible outputs (Figure 7a), this one has its inverse (Figure 7b) to return at the original inputs, it is reversible computation [8].

4.2 Fredkin gate

An schematic diagram and implementation to get a Fredkin gate in one dimension is proposed in this section.

Typically, two-dimensional CAs were studied to simulate Fredkin gates [8], rotatory elements [21], and reversible computing in general. In this case,

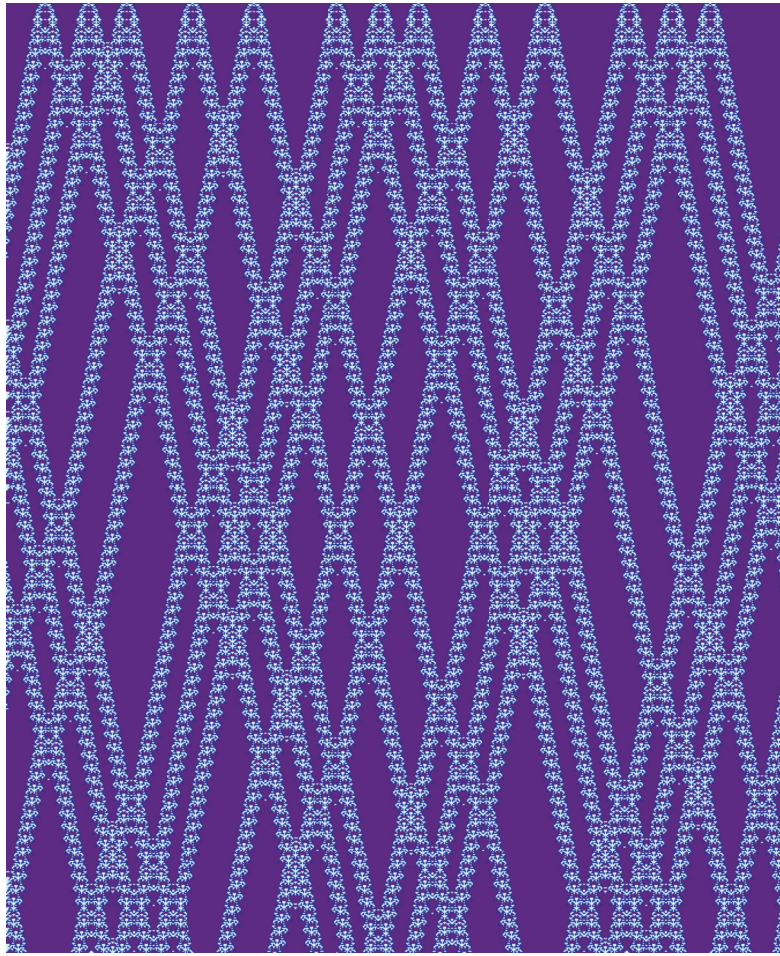


FIGURE 6
Ballistic collisions in ECAM $\phi_{R22maj:4}$. 13 pairs of particles are synchronised to simulate the function reflection $f_r(u, v) \mapsto (v, u)$ from different intervals. Reflections are preserved forever.

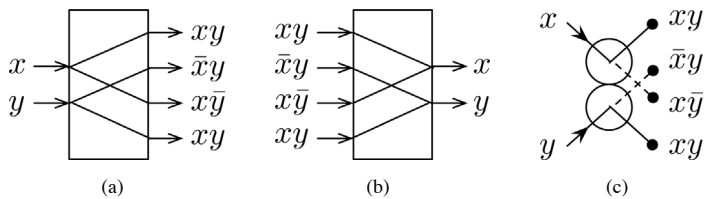


FIGURE 7
With ballistic collisions we can handle an (a) interaction gate, (b) its inverse, and (c) its billiard ball model realization.

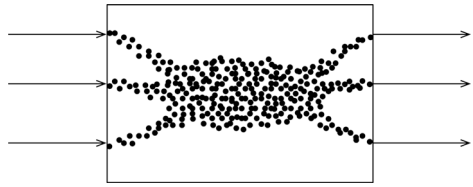


FIGURE 8
Physical computing device with non-linear interaction from several signals (original conceptualization by Fredkin and Toffoli in [7]).

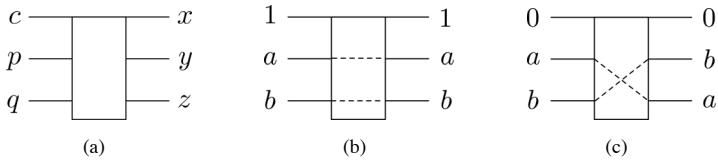


FIGURE 9
Fredkin gate. (a) Scheme of the gate, (b) operation for the control value 1, (c) operation for the control value 0.

we use the ECAM $\phi_{R22maj:4}$ to simulate Fredkin gates in one dimension. For this implementation we need to adjust the timing with additional collisions of localizations that can be confined to a dark box. The physical computing device illustrated in Figure 8 (presented originally in [7]) represents a conservative computation with multiple inputs and the same number of outputs. So, precisely we will exploit this feature to adjust a scheme where reversible computing could be manipulated in an ECAM as an unconventional computing paradigm.

A *conservative logic gate* is a Boolean function that is invertible and preserves signals [8]. Fredkin gate is a classical conservative logic gate. The gate realises the transformation $(c, p, q) \mapsto (c, cp + \bar{c}q, cq + \bar{c}p)$, where $(c, p, q) \in \{0, 1\}^3$. Schematic functioning of Fredkin gate is shown in Figure 9 and the truth table is in Table 2.

Fredkin gate is universal because one can implement a functionally complete set of logical functions with this gate (Figure 10). Other gates implemented with Fredkin gate are shown below:

- $c = u, p = v, q = 1$ (or $c = u, p = 1, q = v$) yields the IMPLIES gate $y = u \to v$ ($z = u \to v$).
- $c = u, p = 0, q = v$ (or $c = u, p = v, q = 0$) yields the NOT IMPLIES gate $y = \overline{(v \to u)}$ ($z = \overline{(v \to u)}$) [8]

c	p	q		x	y	z
0	0	0		0	0	0
0	0	1		0	1	0
0	1	0		0	0	1
0	1	1	→	0	1	1
1	0	0		1	0	0
1	0	1		1	0	1
1	1	0		1	1	0
1	1	1		1	1	1

TABLE 2
Truth table of a Fredkin gate.

4.3 Basic collisions

To simulate a Fredkin gate in a non-invertible ECAM $\phi_{R22maj:4}$, we will take a set of basic collisions to preserve the reactions and the persistence of data, these basic collisions have a specific task and actions which are specified as follows.

- A *mirror* might reflect a mobile localisation, but the mirror should be deleted.
- A *doubler* might copy a value into two.
- A *soliton* crosses two mobile localisations preserving its identity with a small change of phase.
- A *splitter* separate two mobile localisations traveling together in opposite directions.
- A *flag* is a mobile localisation that will be activated from the initial condition depending on an input value given.
- A *displacer* might move forward a mobile localisation.

Fredkin gates implemented in non-invertible systems were proposed and simulated by Adamatzky in a non-linear medium as an oregonator model of Belousov-Zhabotinsky [3].

4.4 Fredkin gates in one dimension

Figure 11 displays the schematic diagram proposed to simulate Fredkin gates in ECAM rule $\phi_{R22maj:4}$. There are three inputs (c, p, q) and three outputs (x, y, z). During the computation auxiliary gliders are generated. They are deleted before reaching outputs. Gliders travel in two directions in a 1D chain of cells, they can be reused only when crossing periodic boundaries or via combined collisions. We use two gliders as flags, travelling from the left 'L_f'

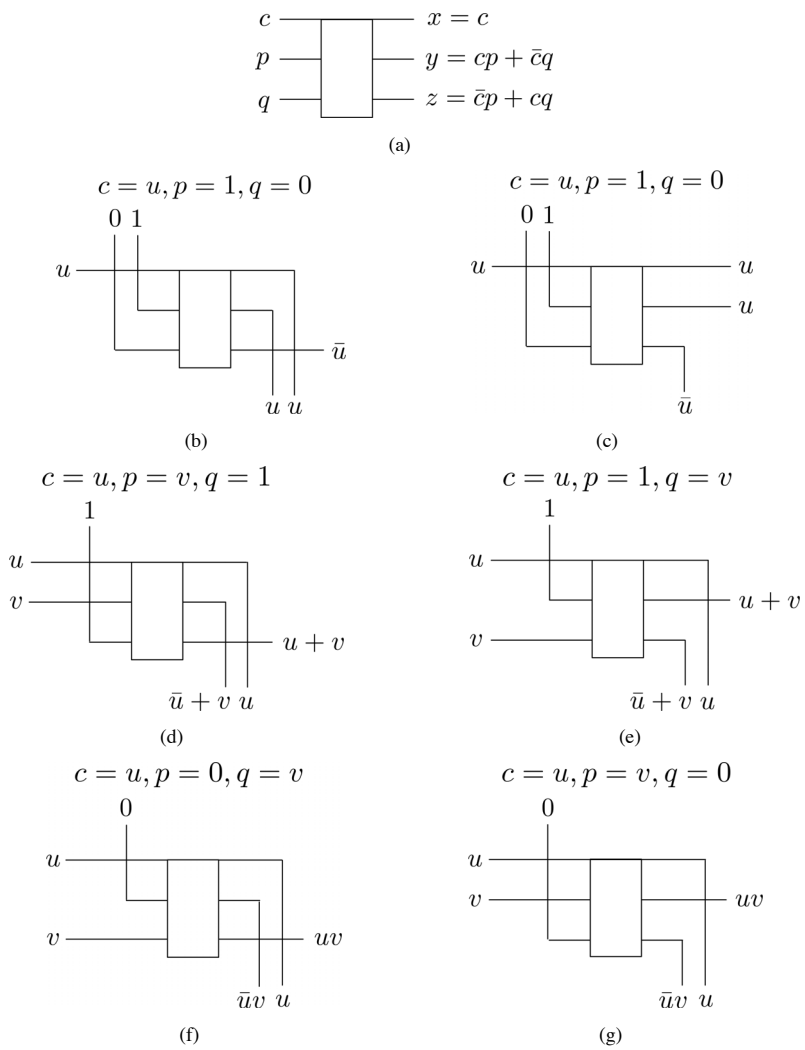


FIGURE 10
Realisation of Boolean functions using Fredkin gate. (a) Fredkin gate, (b) NOT gate, (c) FANOUT gate, (d-e) OR gate, and (f-g) AND gate [8].

and from the right 'R_f'. Flags are activated depending on initial values for c or q as follows:

If $c = 0$ then $R_f = 1$,
If $q = 0$ then $L_f = 1$,
In any other case L and $R = 0$.

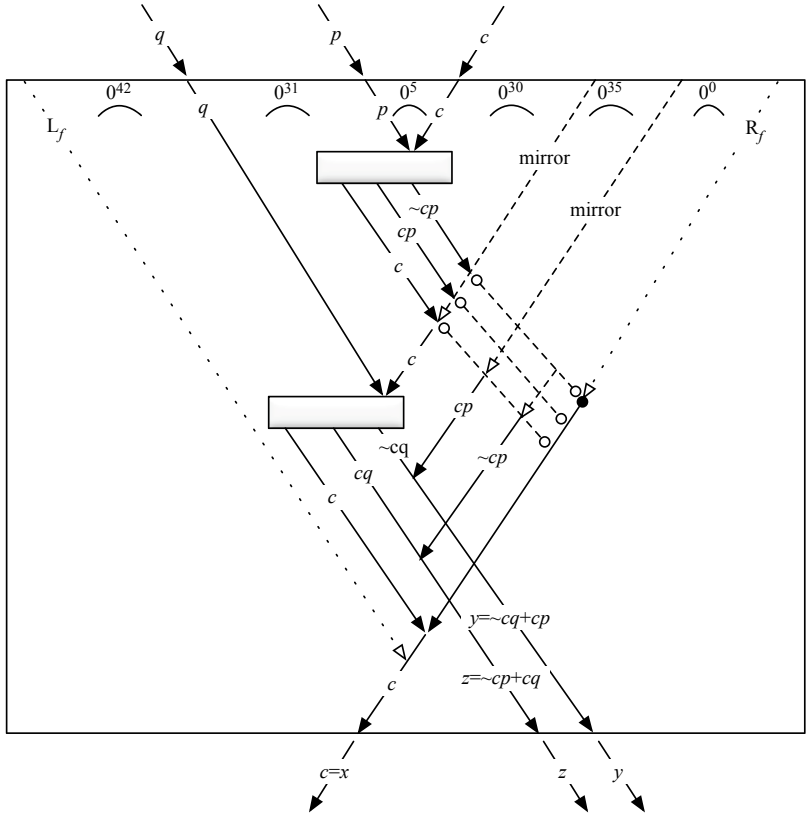


FIGURE 11
A scheme of Fredkin gate implementation in ECAM rule $\phi_{R22maj:4}$ via glider collisions.

Mirrors M are defined as follows:

If $c = p = q = 1$ then $M = 2$,
If $c = p = 1$ then $M = 2$,
If $c = q = 1$ then $M = 1$,
If $p = 1$ then $M = 1$,
In any other case $M = 0$.

Distances between gliders are fixed as positive integers determined for a number of cells in the state '0', as $\overset{0^n}{\frown}\ \forall\ n \geq 0$. During the computation we split gliders when two gliders travel together, the split gliders travel in opposite directions. We use two gliders as mirrors to change the direction of

cpq	*xyz*	L_f	R_f	M
111	111	0	0	2
110	110	1	0	2
101	101	0	0	1
100	100	1	0	0
011	011	0	1	0
010	001	1	1	1
001	010	0	1	0

TABLE 3
Following the scheme in Figure 11 we specify a sequence of collisions that are controlled with flag gliders (L_f, R_f) and mirrors (M) glider.

(a) (b)

FIGURE 12
Fredkin gate in ECAM rule $\phi_{R22maj:4}$. (a) INPUT $c = 1, p = 1, q = 1$, OUTPUT $x = 1, y = 1, z = 1$, (b) INPUT $c = 1, p = 0, q = 1$, OUTPUT $x = 1, y = 0, z = 1$.

FIGURE 13
Fredkin gate in ECAM rule $\phi_{R22maj:4}$. (a) INPUT $c = 1, p = 1, q = 0$, OUTPUT $x = 1, y = 1, z = 0$, (b) INPUT $c = 1, p = 0, q = 0$, OUTPUT $x = 1, y = 0, z = 0$.

an argument movement. We use displacer to move an output glider and adjust its distance with respect to other.

Table 3 shows values of inputs, outputs, flags, and mirrors. First column represents INPUTS, the second column OUTPUTS, third column are values of a flag activated in the left side, fourth column are values of the flag activated in the right side, and the last column shows the number of necessary mirrors.

Space-time configurations of ECAM rule $\phi_{R22maj:4}$ implementing Fredkin gate for all non-zero combinations of inputs are shown in Figures 12–15.

FIGURE 14
Fredkin gate in ECAM rule $\phi_{R22maj:4}$. (a) INPUT $c = 0, p = 1, q = 1$, OUTPUT $x = 0, y = 1, z = 1$, (b) INPUT $c = 0, p = 0, q = 1$, OUTPUT $x = 0, y = 1, z = 0$.

A way to connect more Fredkin gates in two dimensions to implement more complex computations is shown in Figure 16 that uses an array of these physical computing devices.

5 FINAL REMARKS

We have proposed a scheme to simulate Fredkin gates in one dimension in ECAM rule $\phi_{R22maj:4}$. But, although this can be confined in a physical

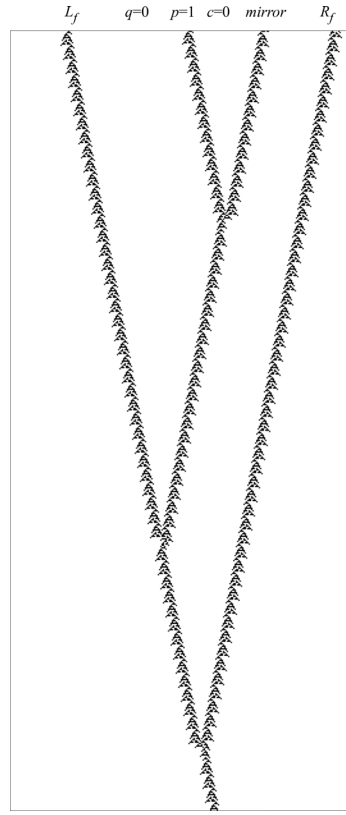


FIGURE 15
Fredkin gate in ECAM rule $\phi_{R22maj:4}$. INPUT $c = 0$, $p = 1$, $q = 0$, OUTPUT $x = 0$, $y = 0$, $z = 1$.

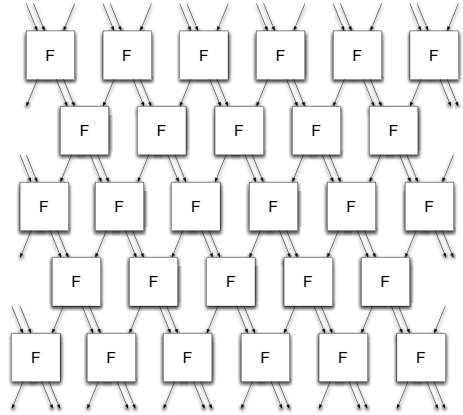


FIGURE 16
Cascaded Fredkin gates (It is a modification of Fredkin array proposed in [15]).

computing device several limitations should be improved: avoid the use of mirrors and the use of flags. An option is that they could be manipulated into in a ring (or virtual CA collider [12,17]).

REFERENCES

[1] Albert, J. & Culik II, K. (1987) A simple universal cellular automaton and its one-way and totalistic versions, *Complex Systems* **1(1)** 1–16.

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

[3] Adamatzky, A. (2017) Fredkin and Toffoli gates implemented in oregonator model of Belousov-Zhabotinsky medium, *Int. J. Bifurcation and Chaos* **27(3)** in press.

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

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

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

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

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

[9] Jakubowski, M.H., Steiglitz, K. & Squier, R. (2001) Computing with Solitons: A Review and Prospectus, *Multiple-Valued Logic* **6 (5-6)**. (also republished in [2])

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

[11] Martínez, G.J., Adamatzky, A. & Sanz, R.A. (2013) Designing Complex Dynamics with Memory, *Int. J. Bifurcation and Chaos* **23(10)** 1330035–131.

[12] Martínez, G.J., Adamatzky, A., & McIntosh, H.V. (2016) A Computation in a Cellular Automaton Collider Rule 110, In: *Advances in Unconventional Computing Volume 1: Theory*, A. Adamatzky (Ed.), Springer, chapter 15, 391–428.

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

[14] Margolus, N.H. (1998) Crystalline Computation, In: *Feynman and computation: exploring the limits of computers*, A.J.G. Hey (Ed.), Perseus Books, 267–305.

[15] Margolus, N.H. (2002) Universal Cellular Automata Based on the Collisions of Soft Spheres, In: *Collision-Based Computing*, A. Adamatzky (Ed.), Springer, chapter 5, 107–134.

[16] Martínez, G.J. (2013) A Note on Elementary Cellular Automata Classification, *Journal of Cellular Automata* **8(3-4)** 233–259.

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

[18] McIntosh, H.V. (1990) Wolfram's Class IV and a Good Life, *Physica D* **45** 105–121.

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

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

[21] Morita, K. (2001) A simple universal logic element and cellular automata for reversible computing, *Proc. MCU 2001* (eds. M. Margenstern, Y. Rogozhin), LNCS 2055, 102–113.

[22] Morita, K. (2011) Simulating reversible Turing machines and cyclic tag systems by one-dimensional reversible cellular automata, *Theoret. Computer Science* **412** 3856–3865.

[23] Martínez, G.J., Mora, J.C.S.T. & Zenil, H. (2013) Computation and Universality: Class IV versus Class III Cellular Automata, *Journal of Cellular Automata* **7(5-6)** 393–430.

[24] Sanz, R.A. (2009) *Cellular Automata with Memory*, Old City Publishing.

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

[26] Steiglitz, K., Kamal, I. & Watson, A. (1988) Embedding Computation in One-Dimensional Automata by Phase Coding Solitons, *IEEE Transactions on Computers* **37(2)** 138–145.

[27] Steiglitz, K. (2016) Soliton-Guided Quantum Information Processing, In: *Advances in Unconventional Computing Volume 2: Prototypes, Models and Algorithms*, A. Adamatzky (Ed.), Springer, chapter 13, 297–307.

[28] Toffoli, T. (2002) Symbol Super Colliders, In: *Collision-Based Computing*, A. Adamatzky (Ed.), Springer, chapter 1, 1–22.

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

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

A APPENDIX – BINARY COLLISIONS IN $\phi_{R22MAJ:4}$

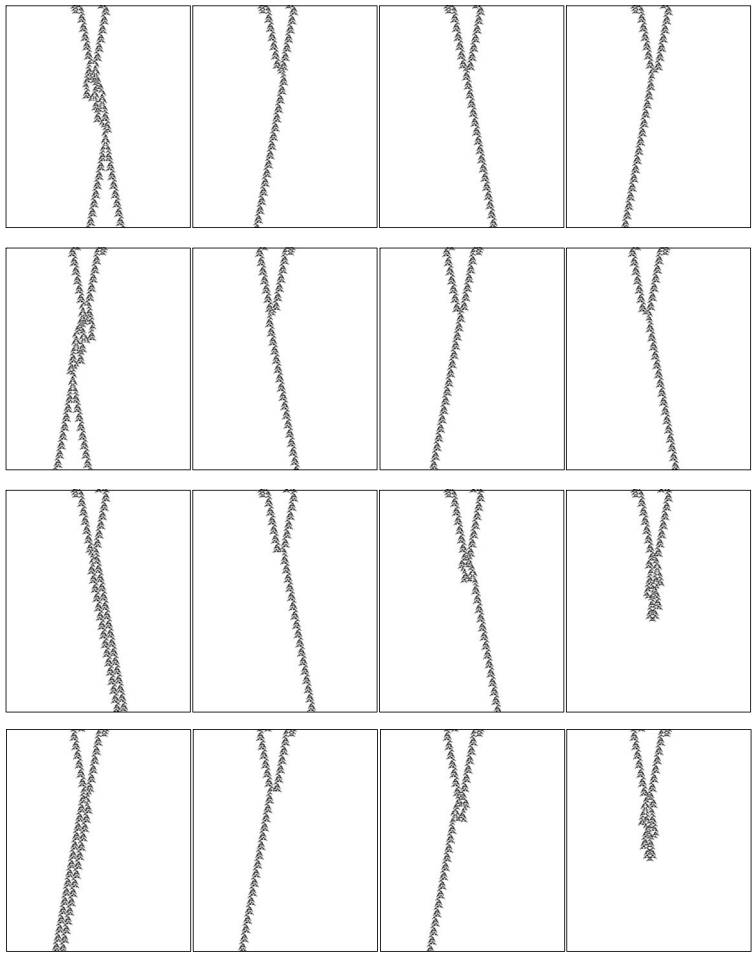


FIGURE 17
Binary collisions in Rule 22 with memory.

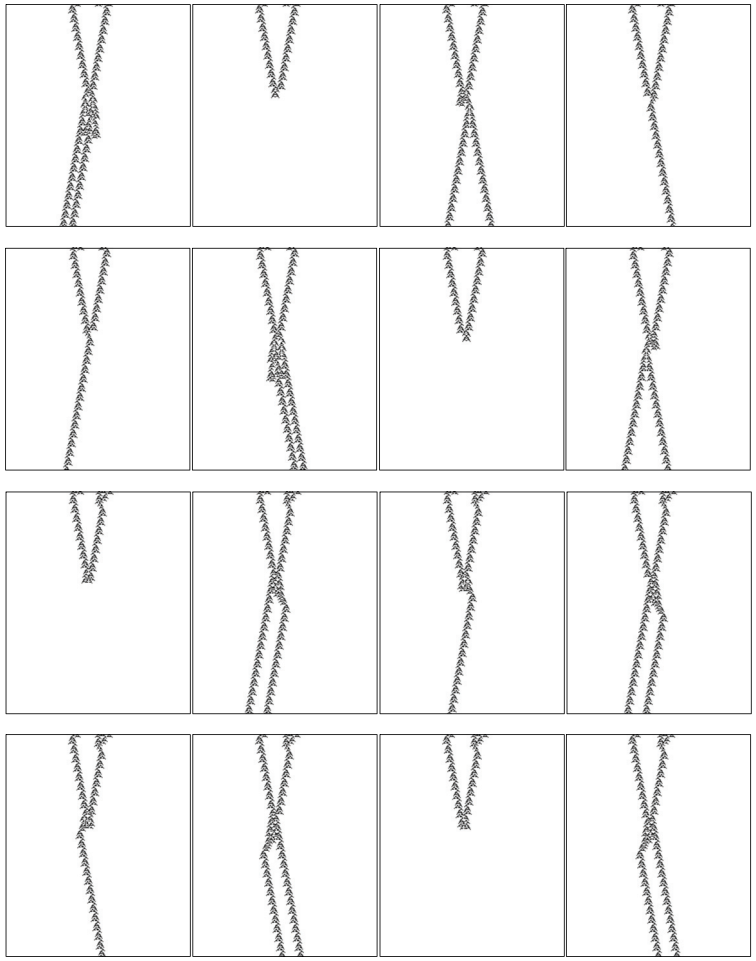


FIGURE 18
Binary collisions in Rule 22 with memory.

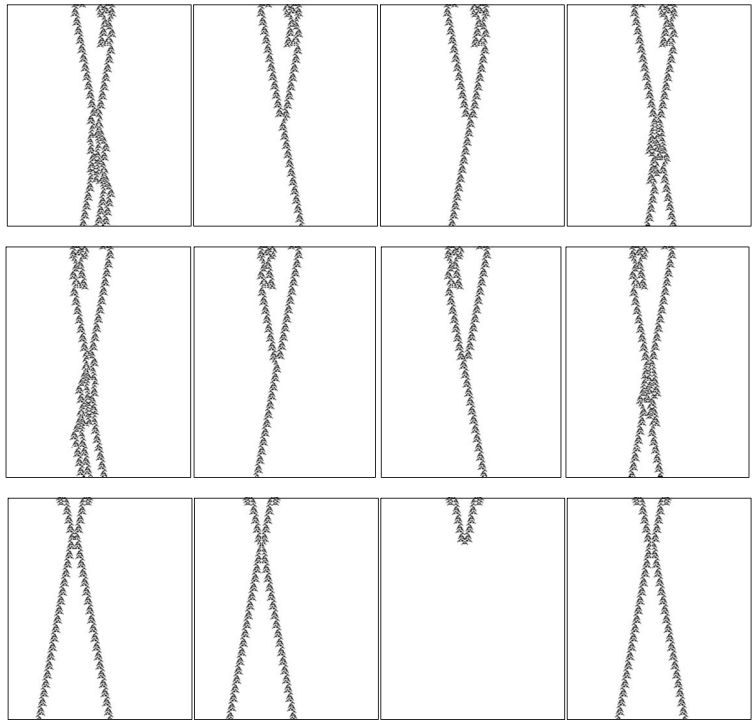


FIGURE 19
Binary collisions in Rule 22 with memory.