Composing a rotary element in reversible elementary square partitioned cellular automata to make reversible computers

Kenichi Morita[*]

April 2023

Abstract

We show computational universality of very simple two-dimensional reversible cellular automata. The model considered here is an *elementary square partitioned cellular automaton* (ESPCA). A square cell of ESPCA consists of four parts each of which has two states, and thus it is a 16-state CA. A local function of ESPCA is described by six local transition rules. We consider three reversible ESPCAs. We show that, in each of these ESPCAs, a *rotary element* (RE) is constructed utilizing only a few kinds of small patterns and their interactions. An RE is a typical reversible logic element with one-bit memory, whose operation is easily understood. Once an RE is implemented, any reversible Turing machine (RTM) is embedded in its cellular space more easily than to use only reversible logic gates. Full computing processes of RTMs in these ESPCAs can be viewed using the file `Universal_ESPCAs.zip` on the general purpose CA simulator *Golly*.

1 Introduction

A reversible cellular automaton (RCA) is an abstract model of a physically reversible space. Since reversibility is one of the fundamental properties of the microscopic physical law, it is an important question how higher functions, such as capability of universal computation, emerge from basic reversible phenomena. In this paper, we investigate this problem using the framework of a reversible elementary square partitioned cellular automaton (ESPCA), which obeys a very simple evolution law. In particular, we study how reversible Turing machines (RTMs), which are a model of universal reversible computers, can be composed in an easy way from a small number of basic reversible phenomena.

There have been various studies to find simple two-dimensional reversible CAs having computational universality. Margolus [4] proposed a two-state reversible block cellular automaton in which Billiard Ball Model (BBM) of computation [2] is simulated. Since a Fredkin gate, a universal reversible logic gate, can be realized in BBM [2], computational universality of Margolus' CA follows. The block CA has the so-called Margolus neighborhood, and thus it is different from the standard framework of CAs. Morita and Ueno [12] proposed two kinds of 16-state reversible partitioned cellular automata, which are currently called ESPCA-02c5bf and ESPCA-02c5df (see Sect. 4), and showed that a Fredkin gate is embeddable in them. Though each of them has 16 states, it is described by six local transition rules, and thus very simple. In addition, a partitioned cellular automaton (PCA) is a subclass of a standard CA. Later, Imai and Morita [3] gave an 8-state reversible triangular cellular automaton, which is now called a reversible elementary triangular partitioned cellular automaton (ETPCA) No. 0157 [6], described by only four local transition rules. They proved its computational universality by showing that a Fredkin is realized in it.

To give a proof of computational universality of RCAs, it is sufficient to show that a universal reversible logic gate is embeddable in them as in the above studies. However, if we want to construct reversible computing systems, such as RTMs, and to observe their computing processes in the RCAs, it is not a good method to use only reversible logic gates. This is because RTMs constructed in this method becomes very complicated. When using a logic gate with two or more input ports, input signals must reach it exactly at the same time. Hence, adjustment of signal timing is necessary at each gate, and it makes the circuit design cumbersome.

A reversible logic element with memory (RLEM) [6] is another type of reversible element. It is a kind of reversible finite automaton with both input and output symbols. A rotary element (RE) [5] is a typical RLEM whose operation is easily understood (see Sect. 2.2). In an RLEM, an input signal interacts with its state, and thus there is little problem on synchronizing two or more signals, and it makes it easy to design a circuit composed of RLEMs. In fact, RTMs can be constructed out of RLEMs very simply [5, 6]. Hence, if an RLEM is implemented in an RCA, we can construct RTMs in a systematic way, and can observe their computing processes.

In [7] and [8], it is shown that a specific RLEM No. 4-31 is implemented in reversible ETPCAs Nos. 0137 and 0347, respectively. There, concrete examples of RTMs are constructed out of RLEM 4-31. Likewise, in [10], an RE is

[*]Currently Professor Emeritus of Hiroshima University, morita.rcomp@gmail.com

implemented in reversible ESPCA-01caef, and RTMs are constructed by it. These ETPCAs and ESPCAs are simulated on a general purpose CA simulator *Golly* [14]. We can observe whole computing processes of the RTMs in these RCAs.

In this paper, we study ESPCA-01c5ef, ESPCA-02c5bf and ESPCA-02c5df, and give implementation methods of an RE in them. Then, examples of RTMs composed of REs are given. The reason why we choose an RE from RLEMs is that operations of an RE, as well as those of RTMs composed of it, are easy to understand intuitively.

InSect. 2, definitions on ESPCAs, an RE and RTMs are given. Known properties of them are also described.

In Sect. 3, an RE is constructed out of three kinds of small patterns in ESPCA-01c5ef. In particular a pattern called a "position marker" is used to keep the state of an RE. A state-change is performed by colliding a space-moving pattern with the position marker. Thus, no reversible logic gate is used for composing an RE module.

In Sect. 4, we investigate a construction method of an RE in ESPCA-02c5bf and ESPCA-02c5df. Here, an RE is realized using a reversible logic gate called an interaction gate (I-gate) and its inverse gate (I^{-1}-gate), since no pattern that works as a position marker has been found so far. Therefore, adjustment of signal timing is necessary inside the RE pattern. However, once an RE is created as a module, it becomes very easy to construct RTMs using the RE modules.

Examples of RTMs are constructed for these ESPCAs. Their computing processes in the ESPCAs can be seen on *Golly* [14] using the emulator available in the file `Universal_ESPCAs.zip`.

2 Preliminaries

In this section, we give basic definitions and known properties on an elementary square partitioned cellular automaton (ESPCA), a reversible logic element with memory (RLEM), in particular a rotary element (RE), and a reversible Turing machine (RTM). We also show an outline of composing RTMs out of REs.

2.1 Elementary square partitioned cellular automaton (ESPCA)

A 4-neighbor *square partitioned cellular automaton* (SPCA) is a two-dimensional CA consisting of square cells. A cell is divided into four parts as in Fig. 1, and each part has its own state set. The next state of a cell is determined depending on the present states of the four adjacent parts of the neighboring cells as shown in Fig. 2. Note that, in the 4-neighbor SPCA, the next state of a cell does not depend on the previous state of the cell itself.

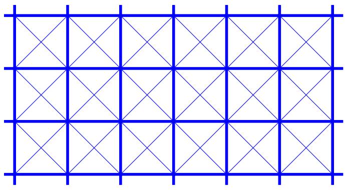


Figure 1: Cellular space of a 4-neighbour square partitioned cellular automaton (SPCA)

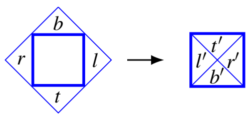


Figure 2: Local transition rule of an SPCA

Definition 2.1 A 4-neighbor *square partitioned cellular automaton* (SPCA) is defined by

$$P = (\mathbb{Z}^2, (T,R,B,L), ((0,-1),(-1,0),(0,1),(1,0)), f).$$

Here, $\mathbb{Z}^2$ is the set of all points with integer coordinates where cells are placed. The items T, R, B and L are non-empty finite sets of states of the top, right, bottom and left parts of a cell. The set of states of a cell is thus $Q = T \times R \times B \times L$. The quadruple $((0,-1),(-1,0),(0,1),(1,0))$ is the *neighborhood* of P. The item $f : Q \to Q$ is a *local (transition) function*.

If $f(t,r,b,l) = (t',r',b',l')$ holds for $(t,r,b,l),(t',r',b',l') \in Q$, this relation is called a *local transition rule* of P. It is also written pictorially as in Fig. 2. The local function f is thus defined by a set of local transition rules.

Definition 2.2 Let $P = (\mathbb{Z}^2, (T,R,B,L), ((0,-1),(-1,0),(0,1),(1,0)), f)$ be an SPCA. A *configuration* of P is defined by a function $\alpha : \mathbb{Z}^2 \to Q$. The set of all configurations of P is denoted by $\mathrm{Conf}(P)$, i.e., $\mathrm{Conf}(P) = \{\alpha \mid \alpha : \mathbb{Z}^2 \to Q\}$. Let $\mathrm{pr}_T : Q \to T$ be the *projection function* that satisfies $\mathrm{pr}_T(t,r,b,l) = t$ for all $(t,r,b,l) \in Q$. The projection functions $\mathrm{pr}_R : Q \to R$, $\mathrm{pr}_B : Q \to B$ and $\mathrm{pr}_L : Q \to L$ are defined similarly. The *global function* $F : \mathrm{Conf}(P) \to \mathrm{Conf}(P)$ of P is defined as the one that satisfies the following.

$$\begin{aligned}&\forall \alpha \in \mathrm{Conf}(P), \forall (x,y) \in \mathbb{Z}^2 :\\ &\quad F(\alpha)(x,y) \;=\; f(\mathrm{pr}_T(\alpha(x,y-1)), \mathrm{pr}_R(\alpha(x-1,y)), \mathrm{pr}_B(\alpha(x,y+1)), \mathrm{pr}_L(\alpha(x+1,y)))\end{aligned}$$

Definition 2.3 An SPCA P is called *reversible* if its global function is injective.

A more detailed explanation on the definition of reversibility is found in Sect. 10.3 of [6]. The next Theorem shows that injectivity of the global function of a PCA is equivalent to that of the local function [6, 11]. By this, we can easily obtain a reversible CA, since it is sufficient to design a PCA whose local function is injective.

Theorem 2.4 *Let P be an SPCA. Its global function F is injective if and only if its local function f is injective.*

Next, we define the simplest subclass of SPCAs such that its local function is rotation-symmetric, and each of four parts has only two states. It is called an *elementary* SPCA (ESPCA) as in the case of a one-dimensional *elementary cellular automaton* (ECA) [15]. We first define the notion of rotation-symmetry.

Definition 2.5 Let $P = (\mathbb{Z}^2, (T,R,B,L), ((0,-1),(-1,0),(0,1),(1,0)), f)$ be an SPCA. The SPCA P is called *rotation-symmetric* (or *isotropic*)if the following conditions (1) and (2) hold.

(1) $T = R = B = L$

(2) $\forall (t,r,b,l),(t',r',b',l') \in T \times R \times B \times L :\ f(t,r,b,l) = (t',r',b',l') \Rightarrow f(r,b,l,t) = (r',b',l',t')$

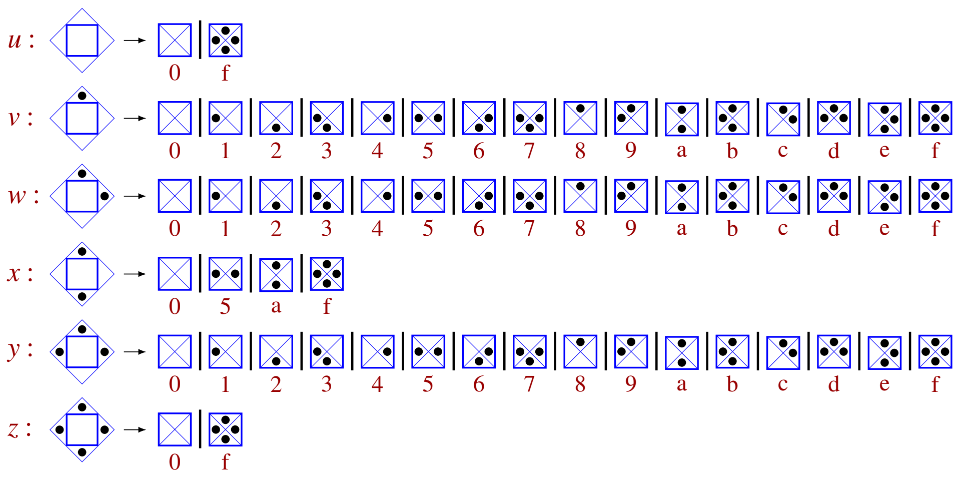
Definition 2.6 Let $P = (\mathbb{Z}^2, (T,R,B,L), ((0,-1),(-1,0),(0,1),(1,0)), f)$ be an SPCA. We say P is an *elementary square partitioned cellular automaton* (ESPCA), if $T = R = B = L = \{0,1\}$, and it is rotation-symmetric.

Since an ESPCA is rotation-symmetric, its local function $f : \{0,1\}^4 \to \{0,1\}^4$ is defined by only six local transition rules. Namely, it is described by the six values:

$$f(0,0,0,0),\ f(0,0,1,0),\ f(0,0,1,1),\ f(1,0,1,0),\ f(0,1,1,1),\ f(1,1,1,1)$$

Here, $f(0,0,1,0)$, $f(0,0,1,1)$, and $f(0,1,1,1)$ may have any value in $\{0,1\}^4$. On the other hand, $f(1,0,1,0) \in \{(0,0,0,0),$ $(0,1,0,1),\ (1,0,1,0),\ (1,1,1,1)\}$ and $f(0,0,0,0), f(1,1,1,1) \in \{(0,0,0,0),(1,1,1,1)\}$, since it is rotation-symmetric. Hence, there are $16^3 \times 4 \times 2^2 = 65{,}536$ ESPCAs in total.

Reading the 4-bit values of $f(0,0,0,0)$, $f(0,0,1,0)$, $f(0,0,1,1)$, $f(1,0,1,0)$, $f(0,1,1,1)$, $f(1,1,1,1)$ as six binary numbers, we can express an ESPCA by a 6-digit hexadecimal identification number $uvwxyz$ as shown in Fig. 3. For example, if $f(0,0,1,0) = (t,r,b,l)$, then $v = 2^3t + 2^2r + 2^1b + 2^0l$. An ESPCA with the identification number $uvwxyz$ is denoted by ESPCA-$uvwxyz$. For example, ESPCA-01c5ef is defined by the six local transition rules shown in Fig. 7.



u :
0
f
v :
0 1 2 3 4 5 6 7 8 9 a b c d e f
w :
0 1 2 3 4 5 6 7 8 9 a b c d e f
x :
0 5 a f
y :
0 1 2 3 4 5 6 7 8 9 a b c d e f
z :
0 f

Figure 3: Representing an elementary SPCA (ESPCA) by a 6-digit hexadecimal number $uvwxyz$. Here, the states 0 and 1 in each part are represented by a blank and a particle, respectively

2.2 Rotary element, a typical reversible logic element with memory

A *reversible logic element with memory* (RLEM) is a kind of reversible finite automaton having both input and output symbols [6]. In reversible computing, reversible logic gates such as a Fredkin gate [2], a Toffoli gate [13], and others are often used as logical primitives. However, RLEMs are also useful in reversible computing. This is because various reversible computing machines can be constructed out of some kinds of RLEMs very easily [6].

A *rotary element* (RE) [5] is a two-state RLEM that has four input symbols n, e, s and w, and four output symbols n', e', s' and w'. It is depicted as in Fig. 4. To each input (output, respectively) symbol, there corresponds a unique input (output) port to (from) which a signal is given (comes out). Intuitively, an RE has a rotatable bar inside, and an incoming signal is controlled by the bar. It takes either of the state V or the state H, depending on the direction of the bar. As shown in Fig. 5, if the direction of a coming signal is parallel to the bar, the signal goes straight ahead, and the state does not change. If the direction of a coming signal is orthogonal to the bar, the signal turns right, and the state changes. It is easy to see that an RE is reversible in the sense that from the next state and the output we can know its previous state and the input uniquely. Note that signals should not be given to two or more input ports at the same time.

When connecting many REs to form an RE-circuit, each output port of an RE can be connected to at most one input port of another (or may be the same) RE. Furthermore, two or more output ports should not be connected to one input port. Therefore, neither branching nor merging of signal lines is permitted.

It is known that any reversible Turing machine can be constructed by an RE (see Sect. 2.4), and that any other RLEM is simulated by it [5, 6].

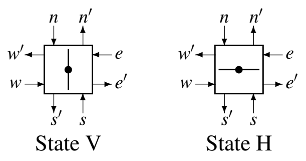


Figure 4: Two states of a rotary element (RE)

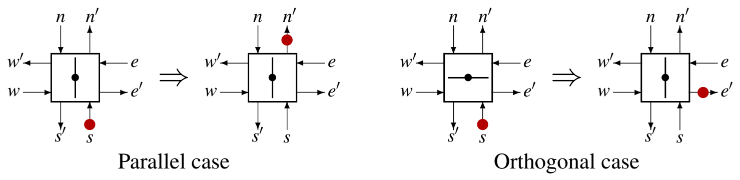


Figure 5: Operations of an RE

2.3 Reversible Turing machines

A one-tape Turing machine (TM) consists of a finite control, a read-write head, and a two-way infinite tape. Formal definition of a TM is as follows.

Definition 2.7 A *one-tape Turing machine (TM)* is defined by

$$T = (Q, S, q_0, F, s_0, \delta),$$

where Q is a non-empty finite set of states, S is a non-empty finite set of tape symbols, q_0 is an *initial state* ($q_0 \in Q$), F is a set of *final states* ($F \subseteq Q$), and s_0 is a special *blank symbol* ($s_0 \in S$). Here, δ is a move relation, which is a subset of $(Q \times S \times S \times \{L, N, R\} \times Q)$. The symbols "$L$", "$N$", and "$R$" are *shift directions* of the head, which stand for "left-shift", "no-shift", and "right-shift", respectively. Each element of δ is a *quintuple* of the form $[p, s, t, d, q]$, which is called a *rule* of T. It means if T reads the symbol s in the state p, then write t, shift the head to the direction d, and go to the state q. We assume each state $q_f \in F$ is a *halting state*, i.e., there is no quintuple of the form $[q_f, s, t, d, q]$ in δ. In this paper, we assume T is deterministic. Hence, for any pair of distinct quintuples $[p_1, s_1, t_1, d_1, q_1]$ and $[p_2, s_2, t_2, d_2, q_2]$ in δ, the relation $(p_1 = p_2) \Rightarrow (s_1 \neq s_2)$ holds.

Reversibility of a TM is defined as below [6].

Definition 2.8 Let $T = (Q, S, q_0, F, s_0, \delta)$ be a TM. We call T a *reversible TM* (RTM), if the following holds for any pair of distinct quintuples $[p_1, s_1, t_1, d_1, q_1]$ and $[p_2, s_2, t_2, d_2, q_2]$ in δ.

$$(q_1 = q_2) \ \Rightarrow \ (d_1 = d_2 \ \wedge \ t_1 \neq t_2)$$

It means that for any pair of distinct rules, if the next states are the same, then the shift directions are the same, and the written symbols are different.

Example 1 An RTM T_{parity} given below is a very simple example.

$$T_{\text{parity}} = (Q_{\text{parity}}, \{0,1\}, q_0, \{q_{\text{a}}\}, 0, \delta_{\text{parity}})$$

Here, $Q_{\text{parity}} = \{q_0, q_1, q_2, q_{\text{a}}, q_{\text{r}}\}$, and $\delta_{\text{parity}} = \{\, [q_0, 0, 1, R, q_1],\ [q_1, 0, 1, L, q_{\text{a}}],\ [q_1, 1, 0, R, q_2],\ [q_2, 0, 1, L, q_{\text{r}}],\ [q_2, 1, 0, R, q_1] \,\}$.

It is easy to see that T_{parity} is reversible. Consider the pair of rules $[q_0, 0, 1, R, q_1]$ and $[q_2, 1, 0, R, q_1]$. The next states in them are the same (i.e., q_1). We can see the shift directions are the same (i.e., R), and the written symbols are different (i.e., 1 and 0). Thus the pair satisfies the reversibility condition in Definition 2.8. No other pair of distinct rules have the same next state. Therefore T_{parity} is reversible. For a given string 01^n0, the RTM T_{parity} tests whether n is even or not. If it is even, It halts in the accepting state q_{a}. Otherwise it halts in q_{r}. All the read symbols are complemented.

Bennett [1] showed the following result, which states that the class of three-tape reversible TMs are computationally universal.

Theorem 2.9 *For any (irreversible) one-tape TM, we can construct a reversible three-tape RTM that simulates the former and leaves no garbage information on its tape.*

It is possible to show the following: (1) Any RTM with k two-way infinite tapes can be simulated by an RTM with k one-way infinite tapes ($k = 1, 2, \ldots$), (2) any RTM with k one-way infinite tapes can be simulated by an RTM with only one one-way infinite tape ($k = 2, 3, \ldots$), and (3) any k-symbol RTM can be simulated by a two-symbol RTM ($k = 3, 4, \ldots$). In the case of irreversible TMs, it is easy to prove the above properties. However, in the case of RTMs, the simulating TMs should be carefully constructed so that they satisfy the reversibility condition. Their details are found in Sect. 5.3.5 of [6]. By above, we obtain the following. Thus, hereafter, we consider only two-symbol RTMs with a one-way infinite tape.

Theorem 2.10 *The class of two-symbol RTMs with a one-way infinite tape is computationally universal.*

2.4 Composing reversible Turing machines out of REs

Using only REs, we can compose any two-symbol RTM having a rightward infinite tape. A composing method was first given in [5]. Here, we use a revised method. See [10] for the detailed explanations of the revised method, and how the circuit works.

The RE-circuit shown in Fig. 6 simulates the RTM T_{parity} in Example 1. It is composed of two kinds of functional modules: a tape cell module and a state module. The left part of Fig. 6 is the finite control of T_{parity}, which consists of several state modules. The right part is a tape unit, which is an infinite array of tape cell modules.

A tape cell has two columns. The left column consists of nine REs, while the right column consists only of one RE. The RE in the right column keeps a tape symbol. If it is in the state H (state V, respectively), then we assume that it keeps the symbol 1 (0). The bottom RE in the left column keeps the head position. If it is in the state H (state V, respectively), then we consider that the head is (is not) on this tape cell. The remaining eight REs in the left column process four kinds of instructions given from the finite control. They are shift-left (SL), shift-right (SR), write 0 (W0), and write 1 (W1) instructions. After executing an instruction, it gives a response signal SLc, SRc, R0, or R1 to the finite control. They mean that shift-left is completed (SLc), shift-right is completed (SRc), the read symbol is 0 (R0), and the read symbol is 1 (R1). Note that by giving a write instruction, a read operation is also executed.

A state module has three REs (though a module for the initial state or a halting state has fewer REs). The left, center, and right REs perform write, shift and read operations, respectively. We prepare one state module for each state of the RTM, and interconnect them according to the move function of the RTM.

If we give a signal to the port "Start", then it is sent to the state module for the initial state. By this, T_{parity} begins to compute. If T_{parity} halts, then the signal from the state module corresponding to the accepting or rejecting state is sent to the port "Accept" or "Reject" showing that the computation is completed.

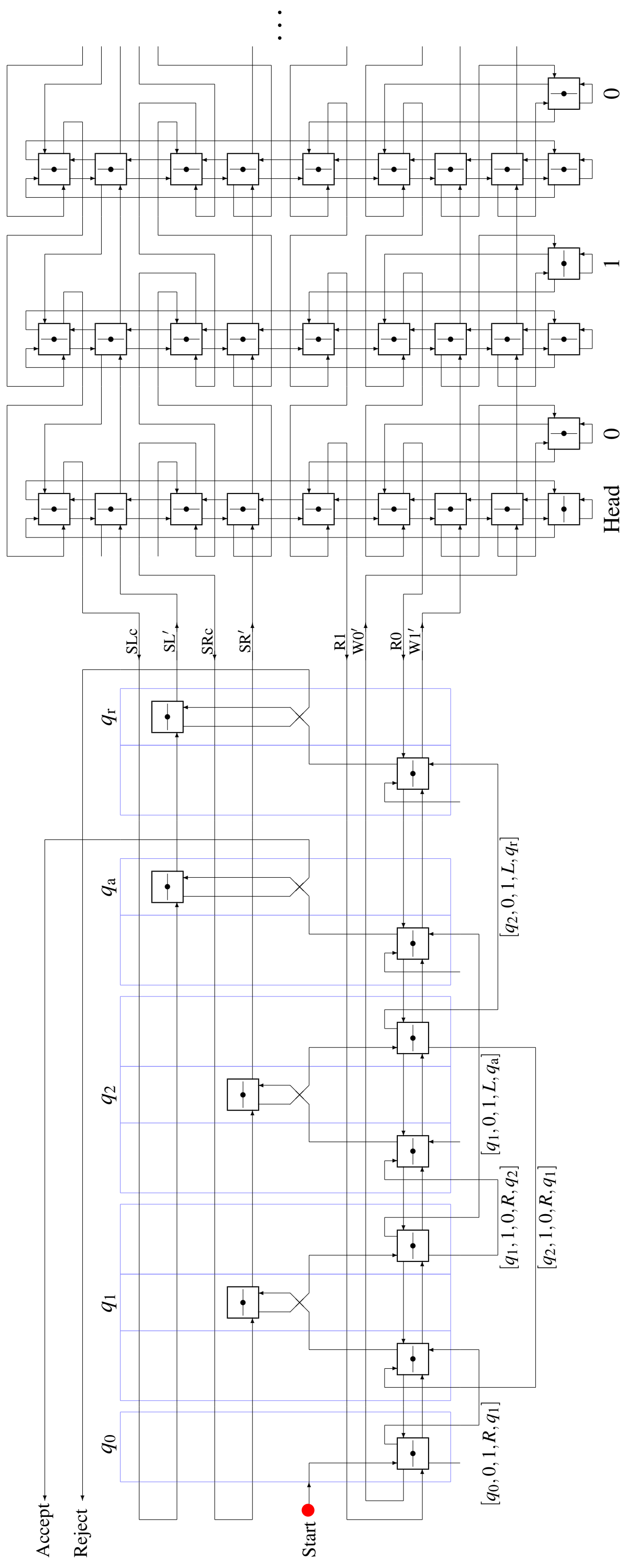


Figure 6: RTM T_{parity} composed of REs [10]

3 Composing RE and RTMs in ESPCA-01c5ef

We consider ESPCA-01c5ef, whose local function f_{01c5ef} is shown in Fig. 7. It is easy to see that there is no pair of distinct local transition rules that have the same right-hand sides, and thus f_{01c5ef} is injective. It means ESPCA-01c5ef is reversible.

Here, we prove that an RE is implemented in this cellular space. Therefore, any RTM can be simulated in it, and thus ESPCA-01c5ef is computationally universal.

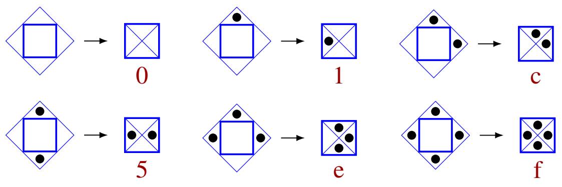


Figure 7: Six local transition rules that define the local function f_{01c5ef} of ESPCA-01c5ef

3.1 Useful patterns in ESPCA-01c5ef

A *pattern* is a finite segment of a configuration (see [6] for the precise definition). In ESPCA-01c5ef, there are three useful patterns called a blinker, a rotor and a glider-12. Interacting these patterns, interesting phenomena are observed. We shall see that an RE is constructed using the three patterns and three useful phenomena.

A *periodic pattern* is one such that the same pattern appears at the same position after some time steps. A *blinker* is a periodic pattern of period 2 shown in Fig. 8. A *rotor* is also a periodic pattern of period 4 shown in Fig. 9.

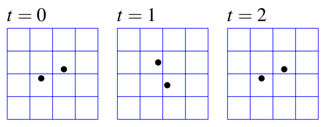


Figure 8: Blinker, a periodic pattern of period 2 in ESPCA-01c5ef

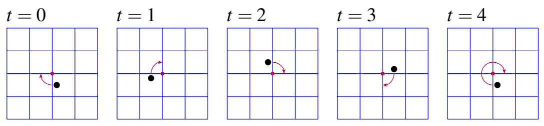


Figure 9: Rotor, a periodic pattern of period 4 in ESPCA-01c5ef

A *space-moving pattern* is one such that the same pattern appears at a different position after some time steps. A *glider-12* is a space-moving pattern of period 12 shown in Fig. 10. It consists of four particles and travels one cell diagonally in 12 steps. The pattern at time p ($0 \leq p \leq 11$) (or its rotated one by a multiple of 90 degrees) is called a glider-12 of phase p. A glider-12 will be used as a signal when we construct an RE.

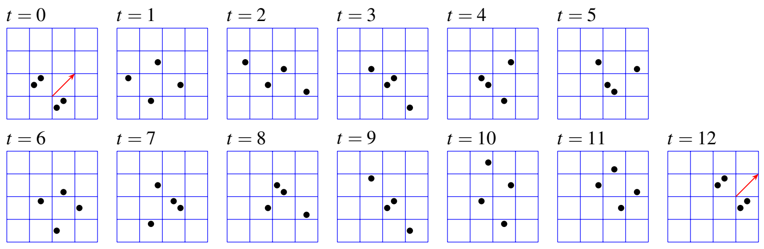


Figure 10: Glider-12, a space-moving pattern of period 12 in ESPCA-01c5ef

A *block* is a stable pattern (i.e., a periodic pattern of period 1) shown in Fig. 11. When composing an RE and an RTM in ESPCA-01c5ef, it will be used only for writing comments and indicating border of a module. Hence it has no functional role for processing signals. Note that in ESPCA-02c5bf a block will be used to control the moving direction of a signal (see Sect. 4).

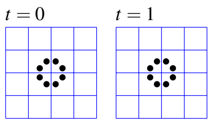


Figure 11: Block, a stable pattern in ESPCA-01c5ef. It is not used for logical operations, but for indicating boundaries of modules and writing comments

3.2 Useful phenomena in ESPCA-01c5ef

Interacting a blinker, a rotor and a glider-12, we obtain useful basic phenomena that can be used to construct an RE. There are three such phenomena.

The first is shown in Fig. 12. Colliding a glider-12 with a rotor in this way, the position of the rotor is shifted by 4 cells from the left to the right. This phenomenon is used to implement a kind of memory, where the memory states are distinguished by the positions of a rotor. Changing the state of the memory is performed by this process. Such a rotor is called a *position marker*. Shifting the position marker from the right to the left is, of course, done by colliding a glider-12 from the opposite side.

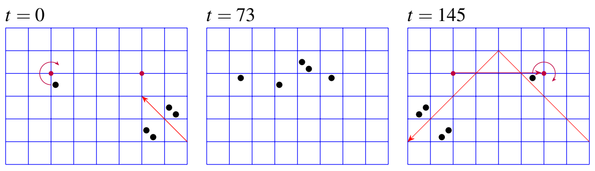


Figure 12: Shifting a rotor by a glider-12 in ESPCA-01c5ef. The rotor plays the role of a position marker for implementing a memory

The second is in Fig. 13. Colliding a glider-12 with a rotor in this way, the glider-12 makes a left-turn, and the rotor is not affected at all as a result. This phenomenon is used to control the moving direction of a glider-12. It is also used to know the memory state (it will be explained later).

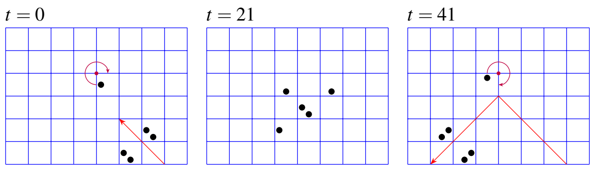


Figure 13: Left-turn of a glider-12 by a rotor in ESPCA-01c5ef

The third is shown in Fig. 14. Colliding a glider-12 with a blinker, the glider-12 makes a right-turn, and the blinker is not affected at all. Although a right-turn is possible by three successive left-turns shown in Fig. 13, this method needs a more space. When implementing a memory using the phenomenon of Fig. 12, we need both right-turns and left-turns, since many access paths to a position marker must be placed in a small area. It is also used to adjust the phase of a glider-12 by combining right-turns with left-turns.

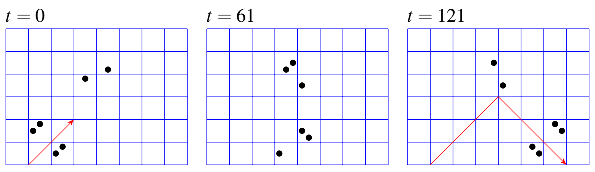


Figure 14: Right-turn of a glider-12 by a blinker in ESPCA-01c5ef

3.3 RE and RTM in ESPCA-01c5ef

Using the three small patterns and the three phenomena given above, an RE is realized as in Fig. 15. Two rotors are used as position markers to keep the state V or H of the RE. The four small circles near the center of the RE indicate the possible positions of the rotors. This figure shows the case of state V, since the left and the right position markers are in the circles labeled by V. If they are in the circles labeled by H, it is in the state H.

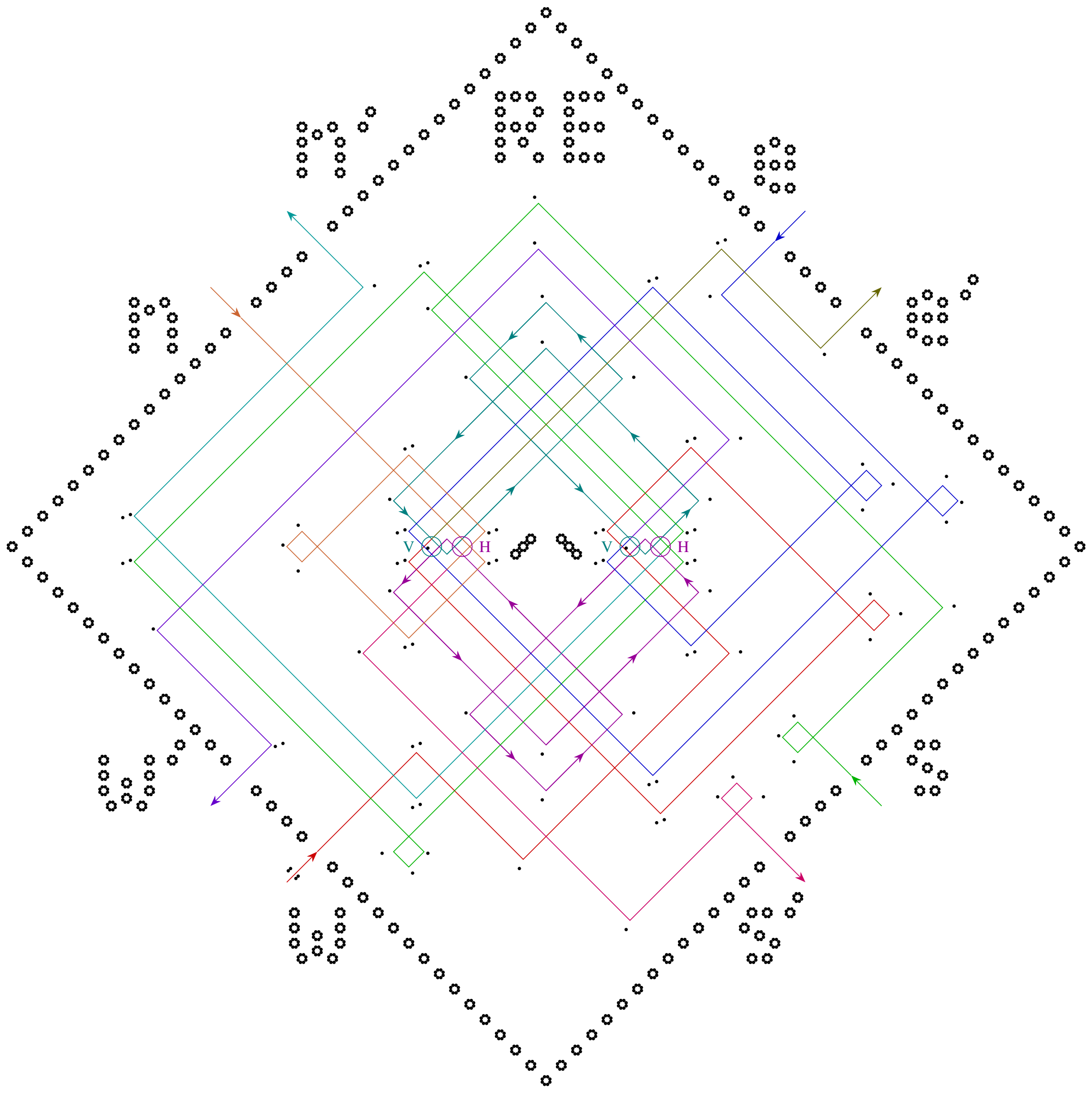


Figure 15: Rotary element (RE) implemented in ESPCA-01c5ef

We now explain how the RE pattern works using Fig. 16, which is an enlarged figure of the center part of Fig. 15.

First, consider the case where the state is V and the input is w. The glider-12 (i.e., an input signal) moves along the following path:

$$\text{Case (V, } w): \; w \to \underline{\text{A}} \to \text{B} \to \text{C} \to \text{D} \to \text{E} \to \text{A} \to \text{B} \to \underline{\underline{\text{C}}} \to \text{F} \to s'$$

The signal first moves from w to A. Since the right position marker is in the circle V, the signal makes a left-turn at A, and goes to B, then C. At C, the signal shifts the left position marker from V to H. After that the signal goes to D, and then E. The signal shifts the right position marker from V to H. Then it travels along the path A $\to$ B $\to$ C. At C it makes a left-turn, since the left position marker is at H at this moment. Finally, the signal goes out from s' via F. Note that at $\underline{\text{A}}$ and $\underline{\underline{\text{C}}}$, branching and merging of signal paths occur, which will be explained later.

Second, consider the case where the state is H and the input is w. The input signal moves along the following path:

$$\text{Case (H, } w): \; w \to \underline{\text{A}} \to \text{J} \to \text{K} \to \underline{\text{L}} \to e'$$

The signal goes straight ahead at A, since the right position marker is at the position H. It then goes along the path J $\rightarrow$ K $\rightarrow$ L. Since the left position marker is also at the position H, the signal goes straight ahead at L, and thus goes out from e'.

In both cases of (V,w) and (H,w) the input signal first visits A. At this point the signal branches, and takes different paths depending on the state V or H. Hence, A is a branching point for the input w. Likewise, for each of the inputs n, e and s, there is a branching point. Thus, four input lines branch into eight different paths in total.

The eight different paths must be reversibly merged into four to create four output lines n', e', s' and w'. It is explained below.

Consider the case where the state is H and the input is s. As in the case of (V,w), the signal from s first changes the state of the RE from H to V (explanation of this process is omitted here). Then the signal appears at the point M in Fig. 16. After that the signal moves along the following path:

$$\text{Case (H, } s) : \ M \rightarrow \underline{\text{L}} \rightarrow e'$$

At the point L the signal makes a left-turn, since the state is V at this moment. Hence, the signal goes out from e'. Comparing the above with Case (H,w), we can see that the point L is a merge point for the output e'. This merging process is reversibly done by using the information of the new state of the RE.

Consider the case (V, n). As in the case of (H, w), after knowing the state is V, the signal from n appears at the point N. Then the signal moves as follows:

$$\text{Case (V, } n) : \ N \rightarrow \underline{\text{C}} \rightarrow F \rightarrow s'$$

At the point C, the signal goes straight ahead, since it knows the state is V. Thus, the signal goes out from the output line s'. Hence, C is a merge point for the output s'.

By above, we can see that Fig. 15 correctly simulates an RE.

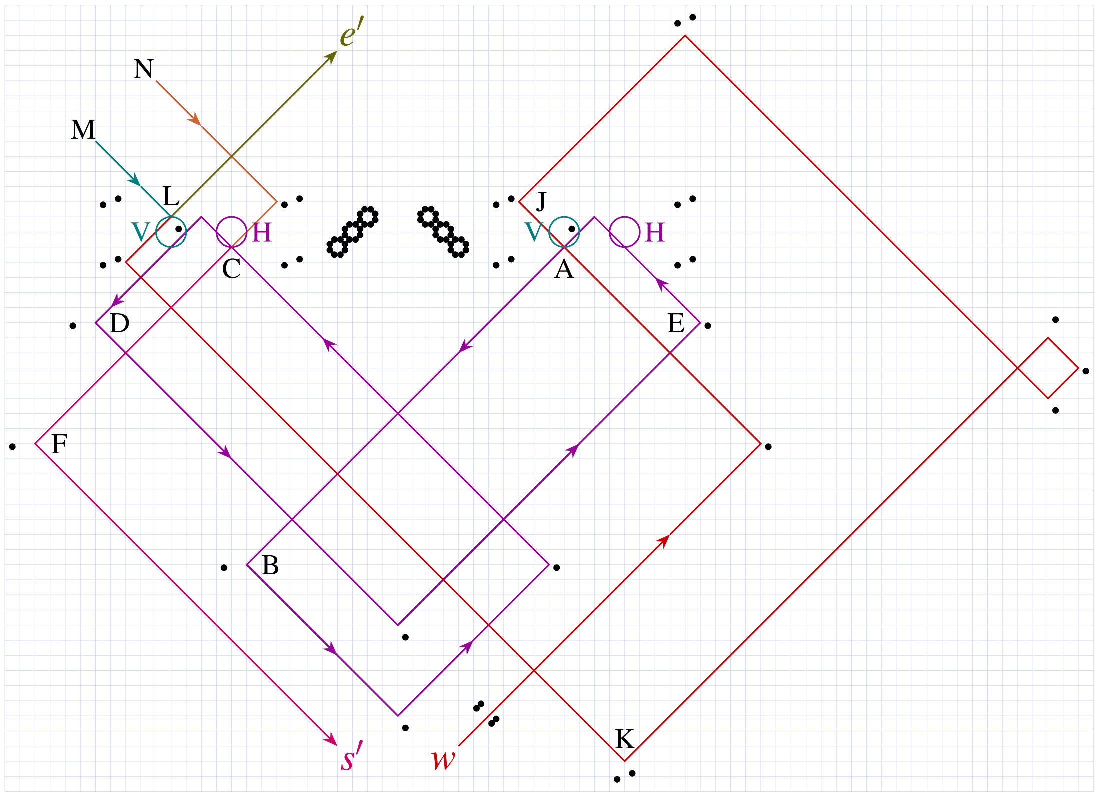


Figure 16: Trajectories of a signal in the RE implemented in ESPCA-01c5ef (Fig. 15) in the case where the input is w. The point A is a branch point, while C and L are merge points

Putting copies of the RE pattern at the positions corresponding to the REs in Fig. 6 and connecting them appropriately, we have a configuration that simulates T_{parity}. Full computing processes of T_{parity} in ESPCA-01c5ef can be seen on *Golly* using the emulator given in the file `Universal_ESPCAs.zip`.

4 Composing RE and RTMs in ESPCA-02c5bf and ESPCA-02c5df

We first consider ESPCA-02c5bf. Its local function f_{02c5bf} is given in Fig. 17. It is easy to see that ESPCA-02c5bf is reversible. In ESPCA-01c5ef (Sect. 3) a rotor is used as a position marker, since it is shifted by colliding a glider-12 (Fig. 12). In ESPCA-02c5bf, however, no such pattern has been found. Therefore, we have to use another method to compose an RE in this cellular space.

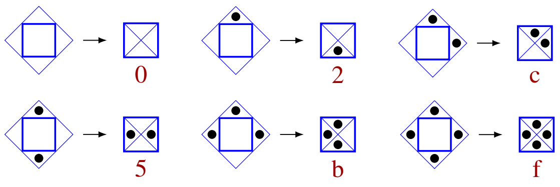


Figure 17: Six local transition rules that define the local function f_{02c5bf} of ESPCA-02c5bf

4.1 Useful patterns in ESPCA-02c5bf

There are two useful patterns in ESPCA-02c5bf. They are a glider-1 and a block. We shall see that an RE is realized in this cellular space using only these two patterns and two basic phenomena shown in Sect. 4.2.

A *glider-1* is a space-moving pattern of period 1 shown in Fig. 18. It will be used as a signal. Although even a one-particle pattern acts as a space-moving pattern, the glider-1 consists of two particles. This is because a kind of logical operation is possible as shown in Fig. 22.

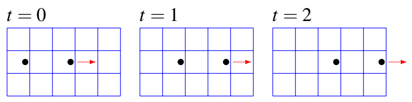


Figure 18: Glider-1, a space-moving pattern of period 1 in ESPCA-02c5bf

A *block* is a stable pattern. It is the same pattern as the one in Fig. 11. However, here, it is used for changing the moving direction of a glider-1. It is also used for indicating boundaries of modules and writing comments as in the case of ESPCA-01c5ef.

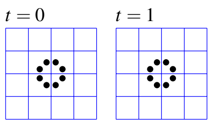


Figure 19: Block, a stable pattern in ESPCA-02c5bf.

4.2 Useful phenomena in ESPCA-02c5bf

Colliding a glider-1 with a block, a left-turn is realized as shown in Fig. 20. However, no right-turn is possible if we use only one block (Fig. 21). Thus, a right-turn will be implemented by three successive left-turns.

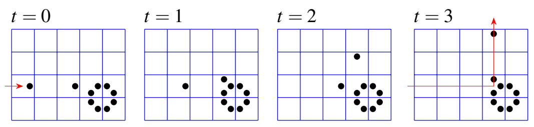


Figure 20: Left-turn of a glider-1 by a block in ESPCA-02c5bf [12]

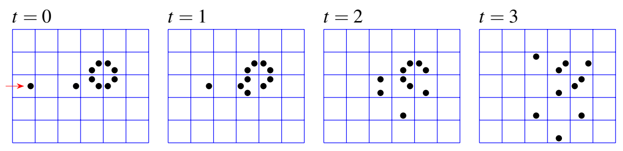


Figure 21: Right-turn of a glider-1 by a block is not possible in ESPCA-02c5bf

Colliding two glider-1's as in Fig. 22 a kind of logical operation is performed. This process mimics a collision of two elastic balls in the Billiard-Ball Model shown in Fig. 23 proposed by Fredkin and Toffoli [2].

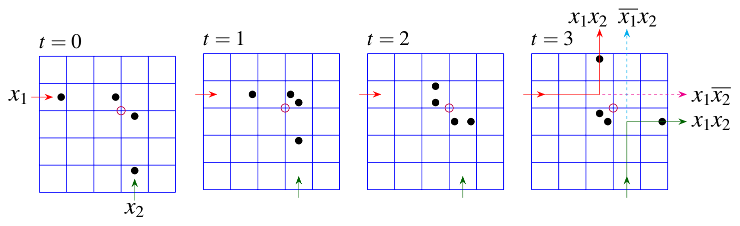


Figure 22: Collision of two glider-1's in ESPCA-02c5bf [12]

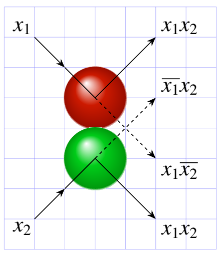


Figure 23: Collision of two balls in the Billiard Ball Model [2]

We can see that the processes in Figs. 22 and 23 realize a reversible logic gate called an *interaction gate* (I-gate) [2] shown in Fig. 24 **(a)**. An I-gate is a 2-input 4-output logic gate having the following injective logic function $f_\mathrm{I} : (0,0) \mapsto (0,0,0,0),\ (0,1) \mapsto (0,1,0,0),\ (1,0) \mapsto (0,0,1,0),\ (1,1) \mapsto (1,0,0,1)$.

By the inverse process of a collision of two glider-1's or two balls, we obtain a 4-input 2-output reversible logic gate called an *inverse interaction gate* (I^{-1}-gate). It is a logic gate having the following injective logic function f_I^{-1} : $(0,0,0,0) \mapsto (0,0),\ (0,1,0,0) \mapsto (0,1),\ (0,0,1,0) \mapsto (1,0),\ (1,0,0,1) \mapsto (1,1)$ (Fig. 24 **(b)**).

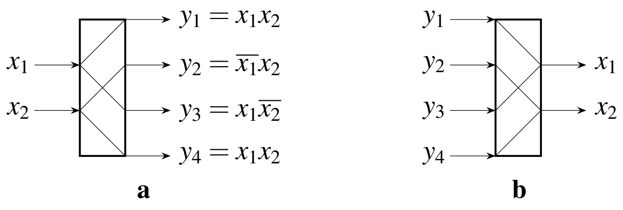


Figure 24: **(a)** Interaction gate (I-gate), and **(b)** its inverse gate (I^{-1}-gate) [2]

4.3 RE and RTMs in ESPCA-02c5bf

Here, we newly show that an RE is constructed out of four I-gates and four I^{-1}-gates. The circuit that simulates an RE is given in Fig. 25. In it many *delay elements* are also used to adjust the signal timing. They are indicated by small triangles, in which a delay time is shown by an integer.

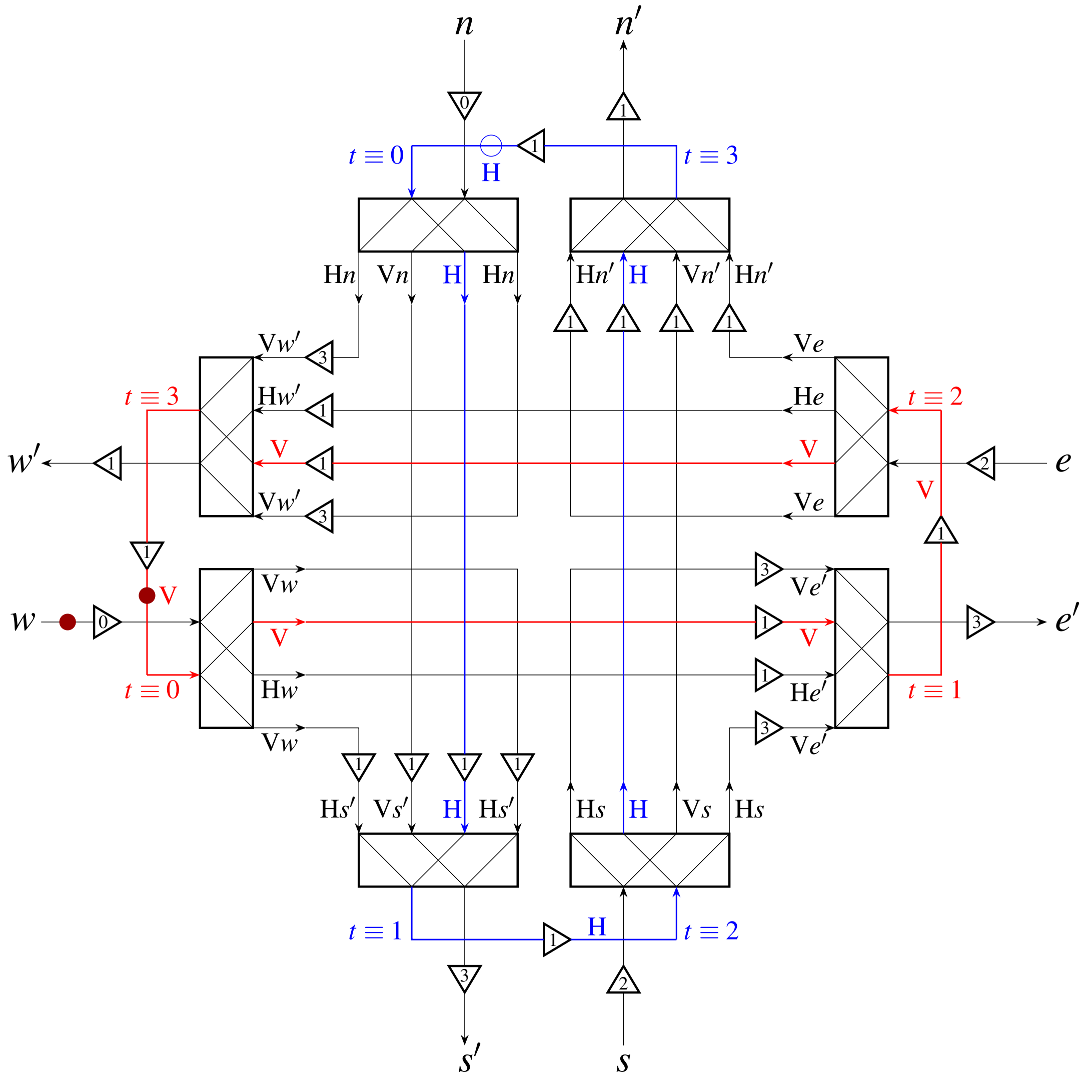


Figure 25: Implementation of an RE using four I-gates, four I^{-1}-gates and delay elements

To keep the state V or H of an RE, we use a *state signal* that moves along the circular path labeled by V or H in Fig. 25. If there is no input signal, the state signal circulates with the period of 4 units of time. When we give an input signal, it must be done at time $t = 0 \bmod 4$.

Consider the case where the state is V and an input signal is given to w at $t = 0 \bmod 4$. The state signal and the input signal interact at the west I-gate. The two signals come out from the two output ports of the west I-gate labeled by Vw. They reach the two input ports Hs' of the south I^{-1}-gate. Then, one of the signal enters the cycle labeled by H. By this, the state changes from V to H correctly. The other signal goes out from the output port s' at $t+4$.

Next, consider the case where the state is H and an input signal is given to w. Since the state signal moves along the cycle labeled by H, the input signal does not interact with it. Hence, the state does not change. The input signal comes out from the output port Hw of the west I-gate. This signal reaches the input port He' of the east I^{-1}-gate at $t+1$, and then goes out from e' at $t+4$.

Since other cases are similar, we can conclude that the circuit simulates an RE.

We now explain a method of implementing the circuit in ESPCA-02c5bf.

Although an I-gate is realized in ESPCA-02c5bf by a collision of glider-1's as shown in Fig. 22, it is convenient to make it as an *I-gate module* so that the delays between the inputs and the outputs become constant. Figure 26 is an I-gate module whose delay is 48 steps.

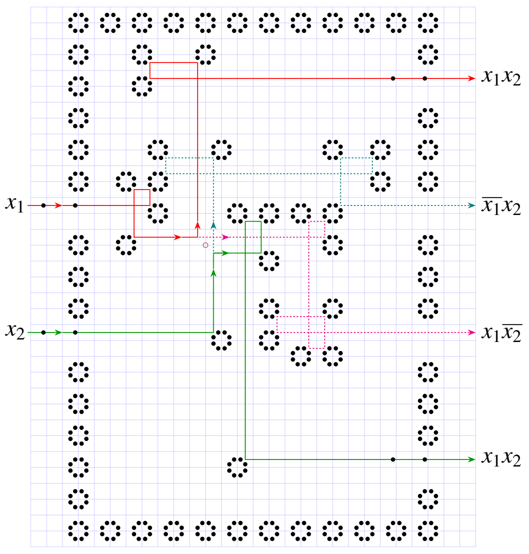


Figure 26: I-gate module in ESPCA-02c5bf. The delay between an input and an output is 48

Likewise, an I^{-1}*-gate module* is implemented as shown in Fig. 27. Actually, it is a mirror image of the I-gate module.

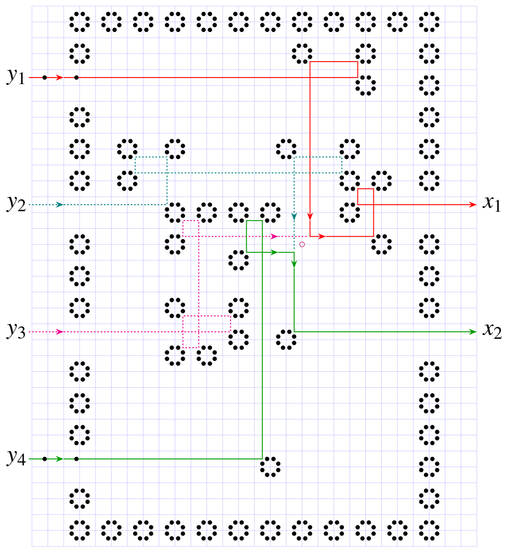


Figure 27: I^{-1}-gate module in ESPCA-02c5bf

We need various kinds of delay modules. Figure. 28 shows an example of a module with an additional 500-step delay.

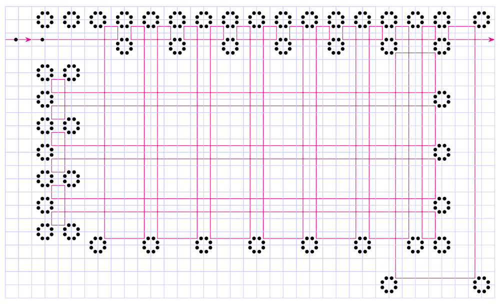


Figure 28: Example of a delay module in ESPCA-02c5bf. It has an additional delay of 500 steps

Assembling copies of the I-gate, the I^{-1}-gate and delay modules and connecting them as in Fig. 25 we have a pattern that simulates an RE in ESPCA-02c5bf. In this implementation, the state signal circulates with the period 1000. Therefore, an input signal must be given at $t = 0 \bmod 1000$.

Using the RE pattern, we can compose any RTM in a systematic manner as in the case of T_{parity} (Fig. 6). In the file `Universal_ESPCAs.zip`, an RTM T_{power} that accepts the unary language $\{1^n \mid n = 2^k \ (k = 0, 1, 2, \dots)\}$ is given. It has the following set of quintuples. Full computing processes of T_{power} in ESPCA-02c5bf can be seen on *Golly* using the emulator in the file `Universal_ESPCAs.zip`.

$$\begin{array}{llllll}\{ & [q_0,0,0,R,q_1], & [q_1,0,0,R,q_2], & [q_2,0,0,L,q_6], & [q_2,1,0,R,q_3], & [q_3,0,1,L,q_4], \\ & [q_3,1,1,R,q_3], & [q_4,0,0,L,q_7], & [q_4,1,0,L,q_5], & [q_5,0,1,R,q_2], & [q_5,1,1,L,q_5], \\ & [q_6,0,0,L,q_r], & [q_6,1,1,R,q_1], & [q_7,0,0,L,q_a], & [q_7,1,1,L,q_r] & \}\end{array}$$

4.4 Composing RE in ESPCA-02c5df

Here, we shortly consider ESPCA-02c5df, since several properties of it are the same as the ones in ESPCA-02c5bf. Hence, an RE is realized in a similar manner as in ESPCA-02c5bf. Its local function f_{02c5df} is given in Fig. 29.

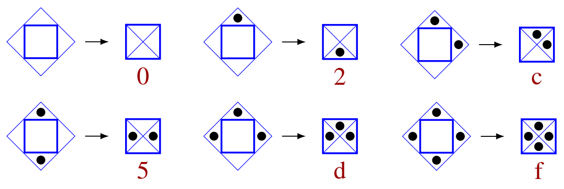


Figure 29: Six local transition rules that define the local function f_{02c5df} of ESPCA-02c5df

A glider-1 (Fig. 18) and a block (Fig. 19) also exist and are useful in ESPCA-02c5df. In addition, interaction of two glider-1's is the same as Fig. 22. Only a left/right-turn of a signal is different. Figure 30 shows a left-turn in ESPCA-02c5df. A right-turn is realized by the mirror image of this figure. Using these phenomena, an I-gate module in ESPCA-02c5df is constructed as shown in Fig. 31.

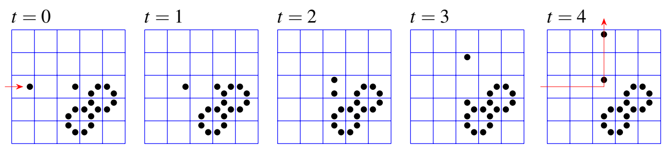


Figure 30: Left-turn of a glider-1 by blocks in ESPCA-02c5df [12]. Right-turn is possible similarly

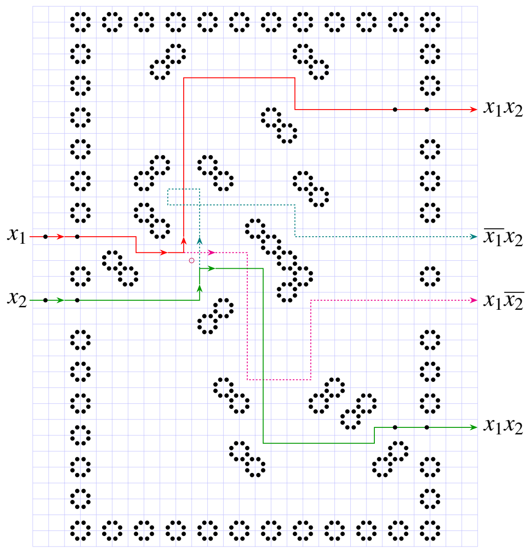


Figure 31: I-gate module in ESPCA-02c5df. The delay between an input and an output is 48

By the I-gate and I^{-1}-gate modules we can construct an RE in ESPCA-02c5df as in the case of ESPCA-02c5bf. Thus, any RTM can be composed of it. In `Universal_ESPCAs.zip`, configurations of the RE and an RTM T_{prime} that are executable in the simulator *Golly* are included, where T_{prime} is a 54-state 2-symbol RTM that accepts the unary language $\{1^n \mid n \text{ is a prime}\}$.

5 Concluding Remarks

In this paper, we showed that an RE is implemented in very simple RCAs, which are ESPCAs Nos. 01c5ef, 02c5bf and 02c5df, having six local transition rules. In addition, in each ESPCA, an RE is designed by utilizing only a few small patterns and a few basic phenomena. Once an RE is implemented, it is easy to compose RTMs out of it in a systematic manner. By this, we not only have a proof of computational universality of each ETPCAs, but also obtain concrete examples of RTMs in the ESPCA, whose computing processes can be seen by a simulator.

Besides the above ESPCAs, it has been shown that an RE is constructed in ESPCA-01caef [10]. In addition, for each ESPCA there are "dual" ESPCAs [9]. For example, if we consider ESPCA-02c5bf, then ESPCAs Nos. 02c5ef, 01c57f and 04c57f are dual ones under reflection (i.e., taking a mirror image), 0-1 complementation, and both, respectively. Hence, an RE is embedded also in them. Since an RE is realizable using only a few reversible basic phenomena in the above ESPCAs, we expect that there still exist other reversible ESPCAs in which an RE is constructed.

References

[1] Bennett, C.H.: Logical reversibility of computation. IBM J. Res. Dev. **17**, 525–532 (1973). doi:10.1147/rd.176.0525

[2] Fredkin, E., Toffoli, T.: Conservative logic. Int. J. Theoret. Phys. **21**, 219–253 (1982). doi:10.1007/BF01857727

[3] Imai, K., Morita, K.: A computation-universal two-dimensional 8-state triangular reversible cellular automaton. Theoret. Comput. Sci. **231**, 181–191 (2000). doi:10.1016/S0304-3975(99)00099-7

[4] Margolus, N.: Physics-like model of computation. Physica D **10**, 81–95 (1984). doi:10.1016/0167-2789(84)90252-5

[5] Morita, K.: A simple reversible logic element and cellular automata for reversible computing. In: Proc. MCU 2001 (eds. M. Margenstern, Y. Rogozhin), LNCS 2055, pp. 102–113 (2001). doi:10.1007/3-540-45132-3_6

[6] Morita, K.: Theory of Reversible Computing. Springer, Tokyo (2017). doi:10.1007/978-4-431-56606-9

[7] Morita, K.: Constructing reversible Turing machines in a reversible and conservative elementary triangular cellular automaton. J. Automata, Languages and Combinatorics **26**, 125–144 (2021). doi:10.25596/jalc-2021-125

[8] Morita, K.: How can we construct reversible Turing machines in a very simple reversible cellular automaton? In: Proc. RC 2021 (eds. S. Yamashita, T. Yokoyama) LNCS 12805, Springer, pp. 3–21 (2021). doi:10.1007/978-3-030-79837-6_1

[9] Morita, K.: Time-reversal symmetries in two-dimensional reversible partitioned cellular automata and their applications. Int. J. Parallel Emergent & Distributed Syst. **37**, 479–511 (2022). doi:10.1080/17445760.2022.2102169

[10] Morita, K.: Making reversible computing machines in a reversible cellular space. Bulletin of EATCS **140** (to appear). https://eatcs.org/index.php/on-line-issues

[11] Morita, K., Harao, M.: Computation universality of one-dimensional reversible (injective) cellular automata. Trans. IEICE **E72**, 758–762 (1989). http://ir.lib.hiroshima-u.ac.jp/00048449

[12] Morita, K., Ueno, S.: Computation-universal models of two-dimensional 16-state reversible cellular automata. IEICE Trans. Inf. & Syst. **E75-D**, 141–147 (1992). http://ir.lib.hiroshima-u.ac.jp/00048451

[13] Toffoli, T.: Reversible computing. In: Automata, Languages and Programming (eds. J.W. de Bakker, J. van Leeuwen), LNCS 85, pp. 632–644 (1980). doi:10.1007/3-540-10003-2_104

[14] Trevorrow, A., Rokicki, T., Hutton, T., et al.: Golly: an open source, cross-platform application for exploring Conway's Game of Life and other cellular automata. http://golly.sourceforge.net/ (2005)

[15] Wolfram, S.: A New Kind of Science. Wolfram Media Inc. (2002)