Jérôme Durand-Lose. Simulation and Intrinsic Universality Among Reversible Cellular Automata, the Partition Cellular Automata Leverage. In Andrew Adamatzky, editor, *Reversibility and Universality, Essays Presented to Kenichi Morita on the Occasion of his 70th Birthday*, number 30 in Emergence, Complexity and Computation, pages 61–93. Springer, 2018. ISBN 978-3-319-73215-2. doi: 10.1007/978-3-319-73216-9_4.

Simulation and Intrinsic Universality among Reversible Cellular Automata, the Partition Cellular Automata Leverage

Jérôme Durand-Lose

Abstract This chapter presents the use of Partitioned Cellular Automata —introduced by Morita and colleagues— as the tool to tackle simulation and intrinsic universality in the context of Reversible Cellular Automata.

Cellular automata (CA) are mappings over infinite lattices such that all cells are updated synchronously according to the states around each one and a common local function. A CA is reversible if its global function is invertible and its inverse can also be expressed as a CA. Kari proved in 1989 that invertibility is not decidable (for CA of dimension at least 2) and is thus hard to manipulate. Partitioned Cellular Automata (PCA) were introduced as an easy way to handle reversibility by partitioning the states of cells according to the neighborhood. Another approach by Margolus led to the definition of Block CA (BCA) where blocks of cells are updated independently. Both models allow easy check and design for reversibility.

After proving that CA, BCA and PCA can simulate each other, it is proven that the reversible sub-classes can also simulate each other contradicting the intuition based on decidability results. In particular, it is proven that any d-dimensional reversible CA (d-R-CA) can be expressed as a BCA with $d+1$ partitions. This proves a 1990 conjecture by Toffoli and Margolus (*Physica D* 45) improved and partially proved by Kari in 1996 (*Mathematical System Theory* 29). With the use of signals and reversible programming, a 1-R-CA that is intrinsically universal —able to simulate any 1-R-CA— is built. Finally, with a peculiar definition of simulation, it is proven that any CA (reversible or not) can be simulated by a reversible one. All these results extend to any dimension.

Jérôme Durand-Lose
Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022, FR-45067 Orléans, France
LIX, CNRS-INRIA-École Polytechnique, France
e-mail: `jerome.durand-lose@univ-orleans.fr`

Key words: Block Cellular Automata ; Cellular Automata ; Intrinsic Universality ; Invertibility ; Margolus neighborhood ; Partitioned Cellular Automata ; Reversibility ; Reversible Cellular Automata

Contents

Simulation and Intrinsic Universality among Reversible Cellular Automata, the Partition Cellular Automata Leverage 1
Jérôme Durand-Lose
1 Introduction 4
2 Definitions 7
2.1 Cellular Automata 8
2.2 Block Cellular Automata 8
2.3 Partitioned Cellular Automata 9
2.4 Reversibility 10
2.5 Simulation and Intrinsic Universality 11
3 Simulations Between classes of CA 11
3.1 Simulation of BCA by CA (and R-BCA by R-CA) 11
3.2 Simulation of CA by PCA 13
3.3 Simulation of CA by BCA 13
3.4 Simulation of R-CA by R-BCA 14
3.5 Simulation of R-BCA (and R-CA) by R-PCA 18
4 Intrinsic Universality of 1-R-PCA 18
4.1 Macroscopic Level 19
4.2 States, Layers and Configurations at Microscopic Level . 20
4.3 Microscopic Algorithm 21
4.4 Local Function of $\mathscr{U}$ 24
4.5 Simulation 24
5 Space-time Simulation of Irreversible CA by Reversible Ones 26
5.1 Space-time Approach 26
5.2 Macro Dynamics 27
5.3 Micro Dynamics 29
5.4 State Function 29
5.5 Generalization Sketch 31
6 Conclusion 32
References 33

1 Introduction

In this chapter, it is shown how Partitioned Cellular Automata (PCA) have been the key to tackle simulation with Block Cellular Automata (BCA) and intrinsic universality of Reversible Cellular Automata (R-CA). Partitioned Cellular Automata were introduced [Morita and Harao, 1989, Morita, 1992a,b] to prove computation universality of 1-dimensional R-CA (1-R-CA). Before that, for lack of ways to handle 1-R-CA, computation universality of R-CA was only known in dimension 2 and above [Toffoli, 1977].

Cellular automata (CA) model parallel phenomena and architectures since their introduction by Ulam and von Neumann in the fifties. They form a model for massively parallel computations and physical phenomena. They have been widely studied for decades and there is a lot of results about them [Burks, 1970, Wolfram, 1986, Sarkar, 2000, Kari, 2005].

They operate as iterative systems on d-dimensional infinite arrays of *cells* (the underlying space is $\mathbb{Z}^d$). Each cell takes a value from a finite set of *state* (Q). A configuration is a valuation of the whole array. An iteration of a CA is the synchronous replacement of the state of every cell by the image of the states of the cells around it (following a finite local *neighborhood* $\mathscr{N}$). This replacement is done according to a unique *local function*. The update is local, uniform, parallel and synchronous.

Reversibility is the capability of a dynamical system to be invertible and to have its inverse in the same class of dynamical systems. This is interesting for physics and computation [Bennett, 1988, Toffoli and Margolus, 1990]. It allows to unambiguous backtrack a phenomenon to its origin. It preserves information and entropy. It may offer a guide to design computers that consume less energy.

A CA is *reversible* when its global function $\mathscr{G}$ is bijective and its inverse ($\mathscr{G}^{-1}$) is the global transition function of some CA. It is known that if $\mathscr{G}$ is one-to-one then it is bijective [Moore, 1962, Myhill, 1963] and the corresponding CA is reversible [Hedlund, 1969, Richardson, 1972]. The reversibility of a CA is decidable in dimension 1 [Amoroso and Patt, 1972] whereas it is not true anymore for greater dimensions [Kari, 1990, 1994].

Lecerf [1963] and Bennett [1973] proved that reversible Turing machines can simulate any Turing Machine and are thus computationally universal. In 1977, Toffoli [1977] proved that any CA can be simulated by a reversible CA (R-CA) one dimension higher. In particular, this proves the existence of 2-dimensional R-CA which are computationally universal. The computing power of R-CA as well as their simulation capability was particularly investigated in Toffoli and Margolus [1990] and Morita [2008].

Reversible CA are quite tricky to design and handle in their general form so that other forms were introduced. To built computationally universal R-CA, (in dimension 2 and above) Block CA (BCA) and (in dimension 1) Partitioned CA (PCA) were independently introduced as special CA for which reversibility is decidable. Like regular CA, they work on infinite regular lattices where each point has a value in a finite set of states.

Block CA (BCA) were introduced in the 80's as a model for lattice gases and other reversible physical phenomena [Margolus, 1984, 1988, Toffoli and Margolus, 1987]. A specific one called the *Billiard Ball Model* was defined. It has only 2 states but is yet computationally universal.

Like for CA, the global function of a BCA is locally defined. The underlying lattice is partitioned into identical hypercubic *blocks* regularly displayed. A partition is fully determined by the *size* of the blocks and the position of a block (or *origin*). A *block transition* is the parallel replacement of all the blocks of a given partition by their images by the *block function*. The *global function* is the sequential composition of various block transitions with the same size and block function.

Since the block function operates over a finite set (blocks and states are finite in number), it can be bijective. The global function is reversible if and only if the block function is a permutation, which is decidable.

Originally, BCA were named "Partitioning CA" and are also known as "CA with the Margolus neighborhood". To avoid any confusion with Morita's Partitioned CA, they are referred to as "Block CA" following Kari [1996] that named "Block Permutations" bijective block functions.

Morita and colleagues introduced *Partitioned CA* (PCA) to prove that R-CA are computationally universal in dimension 1 [Morita and Harao, 1989, Morita, 1992b, 1995]. In PCA, the states are partitioned according to the neighborhood. Each cell swaps its sub-states with neighboring cells and then computes its new state. The local function operates over the finite set of states and can be bijective. The global function is reversible if and only if the local function is a permutation, which is decidable.

Another important topic developed in this chapter is the relations between the different kinds of CA, especially in terms of capability to simulate one another. Following the survey on universalities in CA [Ollinger, 2012], one wants to consider a homogeneous type of simulation: cellular automata simulated by cellular automata in a shift invariant, time invariant way. Trivially, PCA (R-PCA) are CA (R-CA). By considering macro-cells corresponding to the blocks of the first partition, CA (R-CA) simulates BCA (R-BCA).

Block CA can simulate CA by using partitions to progressively add its next state to each cell. PCA can simulate CA by copying the original state in each sub-part. These constructions always generate non reversible BCA and PCA. Nevertheless the following conjecture was made:

Conjecture 1 [Toffoli and Margolus, 1990, Conjecture 8.1] *All invertible cellular automata are structurally invertible, i.e., can be (isomorphically) expressed in space-time as a uniform composition of finite logic primitives.*

A "finite logic primitives" is a representation of a local permutation of blocks t. Kari [1996] proved Conj. 1 for dimensions 1 and 2. The construction is complex but does not need extra states. At the end, Kari conjectures that:

Conjecture 2 [Kari, 1996, Conjecture 5.3] *For every $d \geq 1$, all reversible d-dimensional cellular automata are compositions of block permutations and partial shifts.*

(Partial shift means that the blocks can be shifted which is included in the present definition of BCA.) Durand-Lose [1995, 1996] proved that R-BCA can simulate R-CA in any dimension with extra states and then that R-PCA can also simulate R-CA [Durand-Lose, 1997].

An important concept that stems from simulation is *intrinsic universality*: the capability of a single CA to simulate all the others in a class. This is different from computation universality because it addresses infinite configurations. There exist intrinsically universal (regular) CA [Albert and Čulik II, 1987, Martin, 1994, Ollinger, 2001]. Durand-Lose [1995, 1998, 2001b] proved that the Billiard Ball model is intrinsically universal among the 2-R-CA. Using PCA, Durand-Lose [1997] extended the result to 1-R-CA. Both results extend to higher dimensions.

One natural question is whether R-CA can simulate any (non-reversible) CA. As already mentioned, any d-CA can be simulated by a $d+1$-R-CA [Toffoli, 1977].

In 95, Morita [1992a, 1995] proved with PCA that any CA can be simulated by R-CA of the same dimension over *finite* configurations but the construction does not extend to infinite configurations. A configuration is finite if all but a finite number of cells are in a defined stable state. This is enough for computing since it only treats finite information. But for physical modeling and as mathematical abstractions, there is no reason to restrict to such configurations. Durand-Lose [2000] provided a simulation of any CA by a R-CA but the simulation relation is so peculiar (it is not homogeneous at all) that the problem is still open.

This chapter first provides the formal definition of all kind of CA, of their reversible sub-classes, of simulation and of intrinsic universality.

The simulations between the various kinds of CA are presented. They come naturally but only preserve reversibility when the target is a regular CA. Simulating CA with BCA is done by progressively adding its next state to every cells before discarding all the previous states.

Simulating R-CA with R-BCA is more involving and corresponds to solving conjectures 1 and 2. It uses the local function of the inverse automaton to ensure reversibility. In this construction, a previous state is only erased when it can be regenerated from the next ones in the block. The construction in Durand-Lose [1995] uses $2^{d+1}-1$ partitions with blocks of size $4r$ (r is the greater of the coordinates of the elements of the neighborhood of the CA and of its inverse) in dimension d. The construction presented here is taken from [Durand-Lose, 2001a]. It needs $d+1$ partitions with blocks of size $3(d+1)r$. One gets from a partition to the next by a shift of $(3r, 3r, \cdots, 3r)$.

By considering blocks as cells, BCA can be simulated by PCA preserving reversibility.

Since simulation is a transitive relation, it is enough to prove intrinsic universality on one kind of CA. The construction works on 1-R-PCA and comes from

[Durand-Lose, 1997]. The intrinsically universal 1-R-PCA is organized in 10 layers (for delimitation, identification, table, value, signals, and translation of data). The dynamic is totally driven by signals which exchange values, test for equality, update when it should be done and move data around. It uses a posteriori tests to ensure reversibility.

It is still an open problem whether any CA can be simulated by a R-CA of the same dimension. Nevertheless, for a particular notion of simulation, it is possible.

A *space-time diagram* depicts the whole (infinite) computation of a CA on an initial configuration. It corresponds to the sequence of all the configurations, the orbit of the system. *Space-time simulation* defines an embedding relation between the space-time diagrams of different CA. This is a peculiar simulation relation since configurations can be encoded across infinitely many configurations.

Any CA can be space-time simulated by a R-CA of the same dimension. Unbounded delays are used to provide extra storage for the information needed for reversibility. The proof is given in dimension 1 and generalized to higher dimensions. As a corollary, using the existence of intrinsically universal R-CA, there exists a R-CA which is capable of space-time simulating any CA of the same dimension.

This chapter is based on Durand-Lose [1995, 1997, 2000, 2001a]. All definitions and proofs can be read without any previous knowledge of the subject.

Section 2 formally defines the various models, simulation and intrinsic universality. Section 3 constructs various simulations between the different classes of (reversible) CA. Section 4 details an intrinsically universal 1-R-CA. Section 5 considers space-time simulation and provides CA simulation by R-CA. Section 6 gathers some concluding remarks.

2 Definitions

In this chapter, the following notations are used: $[\![a,b]\!]$ denotes the integers from a to b included; and $<$ and $\leq$, $+$, $-$, mod, div and . also denote respectively the component-wise comparisons, ordering, addition, modulo, Euclidean division and multiplication over $\mathbb{Z}^d$.

Cellular automata (CA) define mappings over d-dimensional infinite arrays over a finite *set of states* Q. The supporting lattice is denoted by $\mathbb{L}$ $(=\mathbb{Z}^d)$. The points of $\mathbb{L}$ are called *cells* and each has a value in Q. The state of cell x in configuration c is denoted by c_x. The set of configurations is denoted by $\mathscr{C}$ $(=Q^{\mathbb{L}})$. Functions on one state/cell are naturally extended into functions over arrays of states/cells and configurations.

For any configuration c and subset E of $\mathbb{L}$, $c_{|E}$ is the restriction of c to E. For any $\mathbf{x} \in \mathbb{L}$, $\sigma_{\mathbf{x}}$ is the shift by $\mathbf{x}$ over configurations $(\forall c \in \mathscr{C}, \forall \mathbf{i} \in \mathbb{L}, (\sigma_{\mathbf{x}}(c))_{\mathbf{i}} = c_{\mathbf{i}-\mathbf{x}})$.

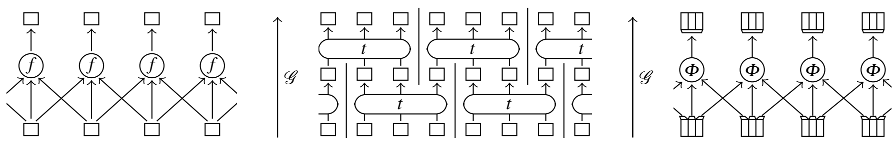
2.1 Cellular Automata

A *Cellular Automaton* of dimension d (d-CA) is defined by $(Q, \mathscr{N}, f)$. The *neighborhood* $\mathscr{N}$ is a finite subset of $\mathbb{L}$. The *local function* $f : Q^{\mathscr{N}} \to Q$ maps the states of a neighborhood into one state. The *global function* $\mathscr{G} : \mathscr{C} \to \mathscr{C}$ maps configurations into themselves as follows:

$$\forall c \in \mathscr{C},\ \forall \mathbf{x} \in \mathbb{L},\ \mathscr{G}(c)_{\mathbf{x}} \ =\ f\left(\left(c_{\mathbf{x}+\mu}\right)_{\mu \in \mathscr{N}}\right) \ .$$

The new state of a cell depends only on the states of neighboring cells as depicted in Fig. 1(a).

The *radius* of a cellular automaton, r, is the maximum absolute value of any coordinate of any element of $\mathscr{N}$. It is the smallest integer r such that: $\mathscr{N} \subseteq [\![-r, r]\!]^d$. By adding dummy entries, the local function can be extended to the domain $[\![-r, r]\!]^d$. Neighborhood and radius can be used equivalently.



(a) Cellular Automata (b) Block CA (c) Partitioned CA

Fig. 1 Schematic CA, BCA and PCA updatings in dimension 1.

2.2 Block Cellular Automata

A *Block CA* of dimension d (d-BCA) is defined by: $(Q, \mathbf{v}, n, (\mathbf{o}^{(j)})_{1 \le j \le n}, t)$. The *size* $\mathbf{v}$ is an element of $\mathbb{L}$ such that $0 < \mathbf{v}$. The *volume* V is the subset $[\![0, \mathbf{v}_1 - 1]\!] \times [\![0, \mathbf{v}_2 - 1]\!] \times \cdots \times [\![0, \mathbf{v}_d - 1]\!]$ of $\mathbb{L}$. A *block* is a mapping from V to Q, or, equivalently, an array of states whose underlying lattice is V. The set of all blocks is Q^V. The *block function* t is a function over blocks. The number of partitions used is n. The origins of the n partitions, $(\mathbf{o}^{(j)})_{1 \le j \le n}$, are elements of V.

The *block transition* T is the following mapping over $\mathscr{C}$: for any $c \in \mathscr{C}$ and $\mathbf{i} \in \mathbb{L}$, let $\mathbf{a} = \mathbf{i} \operatorname{div} \mathbf{v}$ and $\mathbf{b} = \mathbf{i} \bmod \mathbf{v}$ ($\mathbf{a} \in \mathbb{L}$ and $0 \le \mathbf{b} < \mathbf{v}$) so that $\mathbf{i} = \mathbf{a}.\mathbf{v} + \mathbf{b}$, then $t(c)_{\mathbf{i}} = t(c_{|\mathbf{a}.\mathbf{v}+V})_{\mathbf{b}}$. In other words, the block containing $\mathbf{i}$ in the regular partition with blocks of size $\mathbf{v}$ is updated according to t. The same happens for all the blocks of this partition. The configuration is partitioned into regularly displayed blocks, then each block is replaced by its image by the block function t as in Figs. 1(b) and 2.

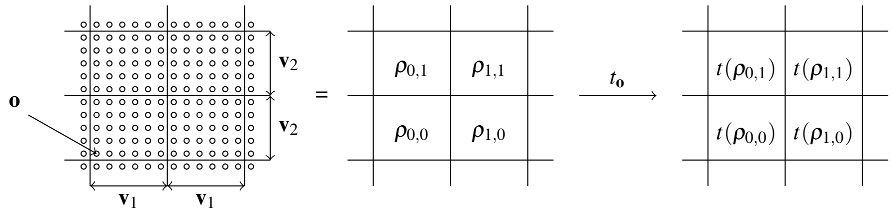


Fig. 2 $t_{\mathbf{o}}$: the block permutation of size $\mathbf{v}$ and origin $(\mathbf{o})$.

The block transition of origin $\mathbf{o}^{(j)}$, T_j is $\sigma_{\mathbf{o}^{(j)}} \circ T \circ \sigma_{-\mathbf{o}^{(j)}}$. It is the original one with the partition shifted by $\mathbf{o}^{(j)}$. The global function is the composition of the block transitions of origins $\mathbf{o}^{(j)}$: $\mathscr{G} = T_n \circ T_{n-1} \circ \cdots \circ T_1$. This is illustrated in Fig. 1(b) with 2 partitions and $\mathbf{v} = (3)$. The new state of a cell depends only on the states around it.

To see that BCA are indeed CA, consider the blocks of the first partition to be cells. At this scale, the global function commutes with any shift and is continuous for the product topology, according to a theorem of [Hedlund, 1969, Richardson, 1972], it is a CA. A constructive proof is provided in Subsect. 3.1.

2.3 Partitioned Cellular Automata

A *Partitioned Cellular Automaton* of dimension d (d-PCA) is defined by: $(Q, \mathscr{N}, \Phi)$. The set of states is a sub-set product indexed by the neighborhood: $Q = \prod_{\mu \in \mathscr{N}} Q^{(\mu)}$. The μ component of a state q is noted $q^{(\mu)}$. The *state function* Φ operates over Q. The global transition function $\mathscr{G}$ is defined by:

$$\forall c \in \mathscr{C}, \forall \mathbf{x} \in \mathbb{L},\ \mathscr{G}(c)_{\mathbf{x}} = \Phi \left(\prod_{\mu \in \mathscr{N}} c^{(\mu)}_{\mathbf{x}+\mu} \right) \quad .$$

The local function works only with what remains and what is received. Only partial information is accessible to a cell, even about its own state as depicted in Fig. 1(c).

Equivalently, each state is the product of the information to be exchanged. Each component is sent to a single cell. An intermediate state is formed by grouping what is left and what is received. The state function Φ yields the new state from the intermediate state. The cell only keeps a partial knowledge about its own state and only receives a partial knowledge about the states of the neighboring cells, as depicted in Fig. 1(c).

A PCA is indeed a CA: the formalization only prevents it from accessing the full states of its neighbors.

A *space-time diagram* $\mathbb{A} : \mathbb{L} \times \mathbb{N} \to Q$ is the sequence of the iterated images of a configuration by a CA $\mathscr{A}$ from an initial configuration c_0. It is defined by $\mathbb{A}_{\mathbf{x},t} = \left(\mathscr{G}^t(c_0)\right)_{\mathbf{x}}$ and denoted by $(\mathscr{G}, c_0)$ or $(\mathscr{A}, c_0)$.

2.4 Reversibility

A CA (resp. BCA, PCA) is *reversible* if and only if its global function $\mathscr{G}$ is bijective and $\mathscr{G}^{-1}$ is the global function of some CA (BCA, PCA). Let R-CA (R-PCA, R-BCA) denote the class of reversible CA (BCA, PCA). Myhill [1963] and Moore [1962] proved that for CA injectivity is equivalent to reversibility. The main decidability result is:

Theorem 3 (Amoroso & Patt, Kari) *The reversibility of CA is decidable in dimension* 1 *[Amoroso and Patt, 1972] but it is undecidable for higher dimension [Kari, 1990, 1994].*

Whereas for BCA and PCA, the following lemmas hold in any dimension.

Lemma 4 (Margolus) *A BCA is reversible iff its block function t is a permutation (which is decidable).*

Proof. If the block function t is a permutation, by construction, any block transition is reversible. The global transition as a composition of transitions, is reversible. Otherwise, t is not one-to-one, then neither is any transition, and neither is the global transition.

Decidability comes from the finiteness of the domain of t. □

Lemma 5 (Morita) *A PCA is reversible iff its state function Φ is a permutation (which is decidable).*

Proof. If Φ is a permutation, then the inverse is obtained by reversing Φ and then sending back the pieces to corresponding neighbors. Otherwise, since Φ works on a finite set, it is not one-to-one and it is easy to construct 2 configurations which have the same image.

Decidability comes from the finiteness of the domain of Φ. □

The inverse PCA is not presented since the proof only assert that, as a CA it is reversible. The inverse is $\left(\prod_{\mu \in -\mathscr{N}} Q^{(-\mu)}, -\mathscr{N}, \Phi^{-1}\right)$ where the state function is computed *before* the sub-states are exchanged. If need, the constructions in the next section can be used to provide the expression of the inverse as a PCA.

As far as reversibility is concerned, BCA and PCA fundamentally differ from CA. It is known that bijectivity for CA is equivalent to reversibility [Hedlund, 1969, Richardson, 1972] and that there exists CA that are surjective but not reversible. By a local inspection, it is easy to prove that for any surjective BCA or PCA the block or state function must be a permutation.

2.5 Simulation and Intrinsic Universality

The local updating process differs for the various kind of CA. Thus simulation has to be defined at global level. To simplify, the definition is presented in 2 steps. The first one does not allow any shift nor scaling. The second introduces them.

Definition 6 (Direct simulation) $\mathscr{A}$ *is directly simulated* by $\mathscr{B}$ if there is some onto partial function α from $Q_{\mathscr{B}}$ to $Q_{\mathscr{A}}$ such that:

$$\forall c \in \mathscr{C}_{\mathscr{A}}, \alpha \circ \mathscr{G}_{\mathscr{B}} \circ \alpha^{-1}(c) = \{ \mathscr{G}_{\mathscr{A}}(c) \} \quad .$$

This is denoted by $\mathscr{A} \preccurlyeq \mathscr{B}$ or when dealing with functions by $\mathscr{G}_{\mathscr{A}} \preccurlyeq \mathscr{G}_{\mathscr{B}}$.

In this definition, space-time diagrams must match exactly. Any computation that is started on a $\mathscr{B}$-configuration that maps to the initial $\mathscr{A}$-configuration (there exist at least one since α is onto) generates the whole space-time diagram.

The next definition adds scaling and shifting. Let $\mathbf{m}$ be any element of $\mathbb{L}$ with positive coordinates. The $\mathbf{m}$-packing, $p_{\mathbf{m}}$, is the bijective mapping from $Q^{\mathbb{L}}$ to $(Q^{\mathbf{m}})^{\mathbb{L}}$ that correspond to identifying the blocks of the $\mathbf{0}$-partition with cells.

Definition 7 (Simulation) $\mathscr{A}$ *is simulated* by $\mathscr{B}$ if there are a positive vector $\mathbf{m}$, an integer τ and a vector $\mathbf{s}$ such that:

$$\mathscr{G}_{\mathscr{A}} \preccurlyeq p_{\mathbf{m}} \circ \mathscr{G}_{\mathscr{B}}^{\tau} \circ p_{\mathbf{m}}^{-1} \circ \sigma_{\mathbf{s}} \quad .$$

This is denoted by $\mathscr{A} \lll \mathscr{B}$ or when dealing with functions by $\mathscr{G}_{\mathscr{A}} \lll \mathscr{G}_{\mathscr{B}}$.

Unpacking is used so that the direct simulation works with macro-cells, i.e. blocks. This is used directly in Subsect. 3.1 as an example. As the chapter goes, the different elements of the simulation are more and more implicit.

From the definition of simulation comes the following definition:

Definition 8 An XCA is *intrinsically universal* if it can simulate any XCA.

3 Simulations Between classes of CA

PCA (resp. R-PCA) are CA (resp. R-CA). The remaining simulations have to be expressed or generated by transitivity.

3.1 Simulation of BCA by CA (and R-BCA by R-CA)

Theorem 9 *Any d-BCA can be simulated by a d-CA. This simulation preserves reversibility.*

Proof. Let $\mathscr{A} = (Q_{\mathscr{A}}, \mathbf{v}, n, (\mathbf{o}^{(j)})_{1\leq j\leq n}, t)$ be any BCA. The set of states of the CA $\mathscr{B}$ is the set of blocks of $\mathscr{A}$ (i.e. $Q_{\mathscr{A}}^{\overline{V}}$). The cells represent the blocks of the first partition. The traces of the partitions are identical in every cell. The neighborhood is $\mathscr{N} = [\![-n, n]\!]^d$ (n is the number of partitions). The cells in $\mathscr{N}$ correspond to the blocks $[\![-n, n]\!]^d$ of the first partition centered on the cell.

The local function $f : Q^{(2n+1)^d} \to Q$ makes the first block transition on an hypercube of size $(2n)^d$. It contains the blocks $[\![-n+1, n-1]\!]^d$ of the first partition centered on the cell. Then f makes the second block transition on the $(2(n-1))^d$ blocks containing the blocks $[\![-n+2, n-2]\!]^d$ of the first partition centered on the cell. Each subsequent block partition is applied on a smaller part but the central block remains in the middle.

The values corresponding to the updated cells are taken as the image by f of the whole neighborhood. Different partitions in dimension 2 are shown in Fig. 3. All the block partitions are considered in one application of f. In one step of $\mathscr{B}$, the images of the cells in the block are computed.

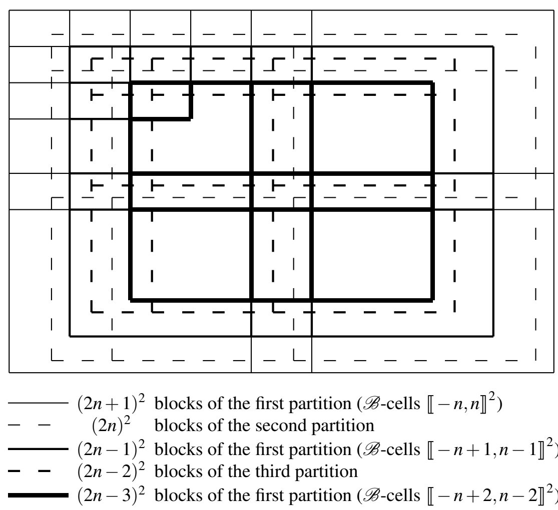


Fig. 3 First and second cuttings.

To be formal with Def. 7 (simulation): $p_{\mathbf{v}}$ is grouping by block, τ is 1, $\mathbf{s}$ is the shift of the first partition and α (for the direct simulation) is the identity. With this simple encoding and this f, there is a natural identification between $\mathscr{A}$ and $\mathscr{B}$: $\mathscr{G}_{\mathscr{A}} = p_{\mathbf{v}} \circ \mathscr{G}_{\mathscr{B}}^{\tau} \circ p_{\mathbf{v}}^{-1} \circ \sigma_{\mathbf{s}}$. Since $p_{\mathbf{v}}$ and σ are reversible, $\mathscr{G}_{\mathscr{B}}$ is reversible if $\mathscr{G}_{\mathscr{A}}$ is. Reversibility is preserved. □

The size of the blocks defines the number of states of the CA while the radius only depends on the number of partitions.

3.2 Simulation of CA by PCA

Proposition 10 *Any d-CA can be directly simulated by a d-PCA.*

Proof. The idea is to duplicate the states in every part. Let $\mathscr{A} = (Q, \mathscr{N}, f)$ be any d-CA. It is simulated by the following d-PCA: $\mathscr{B} = \left(Q^{\mathscr{N}}, \mathscr{N}, \Phi\right)$, with:

$$\forall \nu \in \mathscr{N},\ \Phi\left(\prod_{\mu \in \mathscr{N}} s_\mu^{(\mu)}\right)^{(\nu)} = f\left(\left(s_\mu^{(\mu)}\right)_{\mu \in \mathscr{N}}\right) .$$

The onto partial function α is defined by: $\forall s \in Q, \alpha(s^{\mathscr{N}}) = s$. It is undefined for every other value in $Q^{\mathscr{N}}$. □

Since Φ only maps onto the diagonal of $Q^{\mathscr{N}}$, the simulating PCA is never reversible.

3.3 Simulation of CA by BCA

Theorem 11 *Any d-CA can be directly simulated by a d-BCA.*

Proof. Let $\mathscr{A} = (Q, \mathscr{N}, f)$ be a CA and r be its *radius*: the maximum absolute coordinate of the elements of $\mathscr{N}$. It represents half the size of the “windows” required to gather the information needed to update a cell.

Let $\mathscr{B}$ be the following BCA: $(Q \cup Q^2, (4r, \cdots, 4r), d^2, \{0, 2r\}^d, t)$. The order of the partitions is irrelevant as shows the definition of t below. The state of a $\mathscr{B}$-cell represents either the previous state of a $\mathscr{A}$-cell ($\in Q$) or its previous and next states ($\in Q^2$). The previous configuration is preserved until the next configuration is completely generated.

In each block a part is singled out: the core. It correspond to $[\![r, 3r-1]\!]^d$: the cells that have their full neighborhood in the block. The block function first adds to every cell in the core its next state and then, if all cell in the block have states in Q^2, all previous states are removed and the configuration is ready for next iteration. The choice of block size and partitions ensures that every cell is in the core of exactly one partition.

The onto partial function α is defined by the identity on Q and is undefined on Q^2. □

This simulating BCA is not reversible because of the erasement of the previous state.

3.4 Simulation of R-CA by R-BCA

To preserve reversibility, erasing is done progressively. The inverse d-R-CA of $\mathscr{A}$, $\mathscr{A}^{-1}$, is used to impose a condition (using $f_{\mathscr{A}^{-1}}$) to erase injectively.

The inverse CA can be computed: the CA can be effectively enumerated, composition of CA as well as identity test are computable. The algorithm stops in finite time; but in any dimension greater than one this time cannot be bounded by any computable function because of the undecidability of reversibility.

The inverse of a R-BCA is very simple to built as described in the proof of Lem. 4. When simulating a R-CA, the simulation of it inverse is somehow built.

Theorem 12 *Any d-R-CA $\mathscr{A}$ can be simulated by a d-R-BCA $\mathscr{B}$ with states $Q \cup Q^2$, size $3(d+1)\mathbf{r}$ and $d+1$ partitions. The origins of the partitions are: $\mathbf{0}$, $3\mathbf{r}$, $6\mathbf{r}$, $9\mathbf{r}, \ldots, 3d\mathbf{r}$ where $\mathbf{r}$ is the vector $(r, r, \cdots, r)$ and r is the radius of the CA.*

In the construction, the state of a $\mathscr{B}$-cell represents either the previous or next state of a $\mathscr{A}$-cell ($\in Q$) or both its previous and next states ($\in Q^2$). Before proving the theorem, some lemmas are provided to ensure the distinction between previous and next state for single-state $\mathscr{B}$-cell (i.e. in Q).

The following subsets of $\mathbb{Z}$ and $\mathbb{Z}^d$ are used to locate previous and next states during the iterations. For every θ in $[\![0, d+1]\!]$ and $\kappa \in [\![0, d]\!]$, let

$$
\begin{aligned}
F_\kappa &= 3\kappa r + [\![r, 3(d+1)r - r - 1]\!] + (3(d+1)r)\,\mathbb{Z} \ , \\
\overline{F_\kappa} &= 3\kappa r + [\![-r, r-1]\!] + (3(d+1)r)\,\mathbb{Z} \ , \\
F_\kappa^d &= 3\kappa \mathbf{r} + [\![r, 3(d+1)r - r - 1]\!]^d + (3(d+1)r)\,\mathbb{Z}^d \ , \\
E_\theta^{\mathsf{P}} &= \bigcup_{\theta \le \kappa < d+1} F_\kappa^d \quad \text{and} \\
E_\theta^{\mathsf{N}} &= \bigcup_{0 \le \kappa < \theta} F_\kappa^d \ .
\end{aligned}
$$

These sets are closed under all $\pm 3(d+1)r$ shifts in every direction. This is not always indicated to ease the presentation.

Lemma 13 *For any $0 \le \theta \le d+1$, these sets verify the symmetries $E_\theta^{\mathsf{N}} = E_{d+1-\theta}^{\mathsf{P}} - 3(d+1-\theta)\mathbf{r}$ and $E_\theta^{\mathsf{N}} = -3d\mathbf{r} - E_{d+1-\theta}^{\mathsf{P}} - \mathbf{1}$ and the equalities: $E_{d+1}^{\mathsf{P}} = E_0^{\mathsf{N}} = \emptyset$ and $E_0^{\mathsf{P}} = E_{d+1}^{\mathsf{N}} = E_\theta^{\mathsf{P}} \cup E_\theta^{\mathsf{N}} = \mathbb{Z}^d$.*

Proof. The symmetries, the equality with $\emptyset$ and $E_0^{\mathsf{P}} = E_{d+1}^{\mathsf{N}} = E_\theta^{\mathsf{P}} \cup E_\theta^{\mathsf{N}}$ are obvious.

It remains to prove that $E_{d+1}^{\mathsf{N}} = \mathbb{Z}^d$. Let $\mathbf{x}$ be any element of $\mathbb{Z}^d$. The $d+1$ sets (of $\mathbb{Z}$) $\overline{F_\theta}$ are non-empty and disjoint. Since $\mathbf{x}$ has d coordinates, there exists θ_0 such that none of the coordinates of $\mathbf{x}$ belongs to $\overline{F_{\theta_0}}$. This means that $\mathbf{x}$ belongs to $F_{\theta_0}^d$, thus to E_{d+1}^{N}. $\square$

Let $\bowtie$ be the operator that returns the tuple of defined operand (it returns a pair, a value or undefined). Let following configurations over the alphabet $Q \cup Q^2$ are defined:

$$\forall c \in \mathscr{C},\ \forall \theta \in [\![0, d{+}1]\!], \quad \mathscr{E}_\theta(c) \ = \ c_{|E^{\mathsf{P}}_\theta} \bowtie \mathscr{G}(c)_{|E^{\mathsf{N}}_\theta} \quad .$$

Lemma 13 implies that: $\mathscr{E}_0(c) = c$, $\mathscr{E}_{d+1}(c) = \mathscr{G}(c)$ and that, for all θ, $\mathscr{E}_\theta(c)$ is everywhere defined. In the following $d+1$ reversible block transitions, $(B_\theta)_{0 \le \theta < d+1}$, are defined so that the diagram in Fig. 4 commutes. Then their block functions are proven compatible to be merge into one reversible block function.

$$\begin{tikzcd}
c \arrow[rrrr, "\mathscr{G}_{\mathscr{A}}"] \arrow[dr, "B_0"'] & & & & \mathscr{G}_{\mathscr{A}}(c) \\
& \mathscr{E}_1(c) \arrow[r, "B_1"] & \mathscr{E}_2(c) \arrow[r, dotted] & \mathscr{E}_d(c) \arrow[ur, "B_d"'] &
\end{tikzcd}$$

Fig. 4 Simulation commuting diagram.

Let B_θ be a block transition of size $3(d{+}1)\mathbf{r}$ and origin $3\theta\mathbf{r}$. The size of B_θ matches the length of the shift closure of the sets E^{P}_θ and E^{N}_θ. The 2 partitions for dimension 1 are given in Fig. 5 (they correspond to the construction of Kari [1996]).

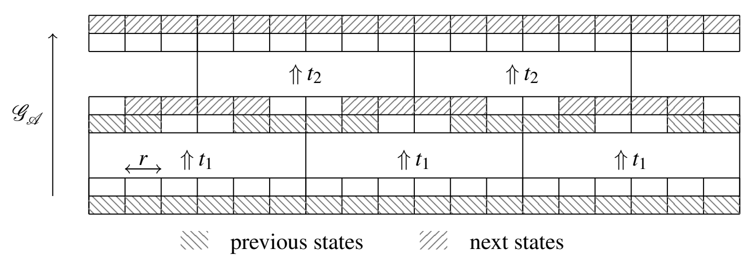


Fig. 5 The 2 steps in dimension 1.

The block functions t_θ have to be defined so that B_θ maps reversiblely $\mathscr{E}_\theta(c)$ into $\mathscr{E}_{\theta+1}(c)$. Each one adds next states and erases previous states. The following lemma states that there is enough information to compute and add the next states.

Lemma 14 *For any θ in $[\![0, d]\!]$, there is enough information in $\mathscr{E}_\theta(c)$ to compute $\mathscr{E}_{\theta+1}(c)$ in each block of the partition of B_θ.*

Proof. The next states added belong to:

$$\begin{aligned}
\Delta_\theta &= E^{\mathsf{N}}_{\theta+1} \setminus E^{\mathsf{N}}_\theta \\
&= \left(3\theta\mathbf{r} + [\![r, 3(d{+}1)r{-}r{-}1]\!]^d\right) \setminus \bigcup_{0 \le \kappa < \theta} \left(3\kappa\mathbf{r} + [\![r, 3(d{+}1)r{-}r{-}1]\!]^d\right) \quad .
\end{aligned}$$

For any $\mathbf{x} \in \Delta_\theta$, $\mathbf{x} \in 3\theta\mathbf{r}+[\![r,3(d+1)r-r-1]\!]^d$. All the cells of the neighborhood of $\mathbf{x}$ should still hold their previous states in order to compute the next state of $\mathbf{x}$. Cell $\mathbf{x}$ and its neighbors are all in the block $3\theta\mathbf{r}+[\![0,3(d+1)r-1]\!]^d$. It corresponds to the block of the partition of B_θ since its origin is $3\theta\mathbf{r}$. It remains to verify that the previous states needed to compute the next state of $\mathbf{x}$ are still present.

For any κ in $[\![0,\theta-1]\!]$, since $\mathbf{x} \notin 3\kappa\mathbf{r}+[\![r,3(d+1)r-r-1]\!]^d$, there is some index j_κ such that $\mathbf{x}_{j_\kappa} \notin 3\kappa r+[\![r,3(d+1)r-r-1]\!]$. So $\mathbf{x}_{j_\kappa}$ is in $3\kappa r+[\![-r,r-1]\!]$ (all is $3(d+1)r$ periodic following any direction). Since the sets $3\kappa r+[\![-r,r-1]\!]$ are disjoint, all j_κ must be different and there are θ of them.

Let $\mathbf{y}$ be any cell needed to compute the next value in $\mathbf{x}$. It belongs to $\mathbf{x}+[\![-r,r]\!]^d$, then, for all κ in $[\![0,d-1]\!]$, $\mathbf{y}_{j_\kappa}$ must be in $3\kappa r+[\![-2r,2r-1]\!]$. By contradiction, let us assume that there exists such a $\mathbf{y}$ which does not belong to E^{P}_θ then for all $\lambda \in [\![\theta,d+1]\!]$, there exists some k_λ such that $\mathbf{y}_{k_\lambda}$ does not belong to $\lambda r+[\![r,3(d+1)r-r-1]\!]$, or equivalently, $\mathbf{y}_{k_\lambda} \in 3\lambda r+[\![-r,r-1]\!]$. Since the sets $3\lambda r+[\![-r,r-1]\!]$ are disjoint, all the k_λ must be different and there are $d+1-\theta$ of them.

Altogether, there are $d+1$ (distinct) j_κ and (distinct) k_λ for d values so there exist κ_0 and λ_0 such that $j_{\kappa_0} = k_{\lambda_0}$. Then the intersection of $3\kappa_0 r+[\![-2r,2r-1]\!]$ and $3\lambda_0 r+[\![-r,r-1]\!]$ is not empty. This means that $\kappa_0 = \lambda_0$, but by construction, $\kappa_0 < \lambda_0$.

Thus $\mathbf{y}$ belongs to E^{P}_θ and all the previous states needed to compute the next state of $\mathbf{x}$ are still present in the block. The next state of $\mathbf{x}$ can be computed with the information held inside the block. □

From the symmetry between E^{N} and E^{P}, follows:

Corollary 15 *For any θ in $[\![0,d]\!]$, there is enough information in $\mathscr{E}_{\theta+1}(c)$ to compute $\mathscr{E}_\theta(c)$ in each block of the partition of B_θ.*

This means that the corresponding blocks of $\mathscr{E}_\theta(c)$ and $\mathscr{E}_{\theta+1}(c)$ in the partition of B_θ can be uniquely determined one from the other. The partial function t_θ is one-to-one.

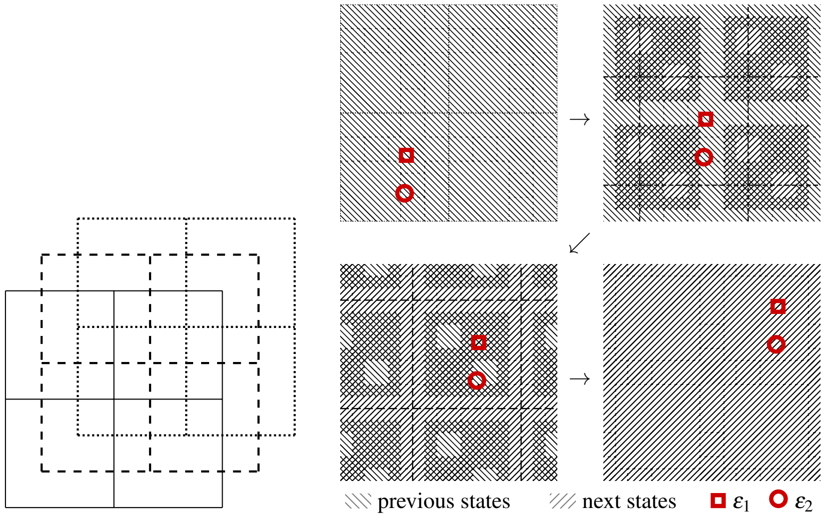
In dimension 2, the partitions are given in Fig. 6(a) and the positions of previous and next states are detailed in Fig. 6(b) (they do not correspond to the construction of Kari any more).

The following lemma shows that the partial definitions of functions t_θ are compatible so that they can be merged into a unique t to define a reversible BCA.

Lemma 16 *The current block transition B_θ can be identified by the position of the double states inside the blocks of partition.*

Proof. If all cells are single, then $\theta = 0$.

For κ in $[\![1,d]\!]$, let ε_κ be the following vector inside the blocks:



(a) the 3 partitions (b) the 3 steps

Fig. 6 Simulating R-CA by R-BCA in dimension 2.

$$
\begin{aligned}
\varepsilon_1 &= 3(d{+}1)\mathbf{r} + (\,-3r, \cdots, -3r\,) \ , \\
\varepsilon_2 &= 3(d{+}1)\mathbf{r} + (\,-3r, -6r, \cdots, -6r\,) \ , \\
\varepsilon_\kappa &= 3(d{+}1)\mathbf{r} + (\,-3r, -6r, -9r, \cdots, -3(\kappa{-}1)r, -3\kappa r, \cdots, -3\kappa r\,) \ , \\
\varepsilon_d &= 3(d{+}1)\mathbf{r} + (\,-3r, -6r, -9r, \cdots, -3dr\,) \ .
\end{aligned}
$$

To get the coordinate in $\mathbb{L}$, a translation by $3\theta\mathbf{r}$ have to be applied. The vectors in dimension 2 are indicated in Fig. 6(b).

For κ in $[\![1,d]\!]$, no coordinate of ε_κ belongs to $[\![-r,r-1]\!]$, so that $\varepsilon_\kappa + 3\theta\mathbf{r}$ belongs to F_θ and thus to E^{P}_θ.

For all κ in $[\![1,\theta-1]\!]$, no coordinate of ε_κ belongs in $\overline{F_0}$, so that ε_κ belongs to F_0^d and thus to E^{N}_θ.

For all κ in $[\![1,d]\!]$, the coordinate value $-3ir+3\theta r = 3(\theta-\kappa)r$ prevents ε_κ from being in $F_{\theta-\kappa}$. So that ε_θ does not belong to $F^d_{\theta-1} \cup F^d_{\theta-2} \cup \cdots \cup F^d_0 = E^{\mathsf{N}}_\theta$.

Altogether θ is the maximum κ such that ε_κ holds 2 states plus one. If there is no such κ then $\theta = 1$. □

Corollary 17 *Thanks to the symmetry (Lem. 13), the positions of double states after the block transition indicate which t_θ was used.*

Above Lemma and Corollary show that all the partial definitions of the block permutations of the block transitions are compatible for domains and ranges. They can be grouped and completed in a unique bijective block function.

Altogether, Th. 12 is proved.

3.5 Simulation of R-BCA (and R-CA) by R-PCA

Lemma 18 *Any d-BCA can be simulated by a d-PCA. This simulation preserves reversibility.*

Proof. The idea is to identify cells with blocks. Let $\mathscr{A}=(Q_{\mathscr{A}},\mathbf{v},n,(\mathbf{o}^{(j)})_{1\leq j\leq n},t)$ be any d-BCA. Let $\mathscr{B}=\left(\prod_{\mu\in\mathscr{N}}Q_{\mathscr{B}}^{(\mu)},\mathscr{N},\Phi\right)$ be a PCA where $\mathscr{N}=\{-1,0,1\}^d$, i.e., coordinates which differ by at most one in any direction. The block of coordinates $\mathbf{x}$ (at block scale) of the j^{th} partition is $\rho_{\mathbf{x}}^{j}$. The block $\rho_{\mathbf{0}}^{j}$ holds the cell of coordinates $\mathbf{0}$. The sets of states are defined by:

$$Q_{\mathscr{B}}^{(\mathbf{0})}=\bigcup_{1\leq j\leq n}\left(\{j\}\times Q_{\mathscr{A}}^{\left(\rho_{\mathbf{0}}^{j-1}\cap\rho_{\mathbf{0}}^{j}\right)}\right)\quad\text{, and}$$

$$\forall\mu\in\mathscr{N},\mu\neq\mathbf{0},\ Q_{\mathscr{B}}^{(\mu)}=\bigcup_{1\leq j\leq n}Q_{\mathscr{A}}^{\left(\rho_{\mathbf{0}}^{j-1}\cap\rho_{-\mu}^{j}\right)}\quad .$$

It holds the partition number together with the intersection of the block that holds the cell of coordinates $\mathbf{0}$ for a partitions and of the one of the next partition holding the cell $\mathbf{0}$ translated by $\mu.\mathbf{v}$. Any intersection may be empty. Blocks are partitioned according to the next partition so that every part is sent to the corresponding cell to form whole blocks of the next partition. Identically, each cell retrieves a full block, uses the local transition and sends the corresponding parts to the neighbors for the next transition. The $\mathbf{0}$-sub-state identifies the partition number which indicates how to split the image block into sub-states.

Configurations are encoded by setting the first components to 1 and by putting states in the corresponding intersections between the last and the first partitions. On the first iteration of $\mathscr{B}$, each cell gets one entire block of the first partition and make the first transition. Then all pieces are sent to the corresponding cells and 2 is recorded in the cell. Each iteration of $\mathscr{B}$ makes a successive transition of $\mathscr{A}$. After n iterations of $\mathscr{B}$, one iteration of $\mathscr{A}$ is made and the first component is 1 again.

This construction preserves reversibility: the partial definition of the PCA state function Φ is one-to-one if the local transition t of the BCA is reversible. □

From above Lemma and the transitivity of simulation comes:

Theorem 19 *Any d-R-CA can be simulated by a d-R-PCA.*

4 Intrinsic Universality of 1-R-PCA

In this section, a 1-R-PCA $\mathscr{U}=(Q_{\mathscr{U}},\{-1,0,1\},\Phi_{\mathscr{U}})$ is built such that:

Theorem 20 *The* 1*-R-PCA* $\mathscr{U}$ *is intrinsically universal, i.e., able to simulate any* 1*-R-PCA.*

Let $\mathscr{A} = (Q, \mathscr{N}, \Phi)$ be any 1-R-PCA. With cells grouping, $\mathscr{A}$ can be simulated by a 1-R-PCA with neighborhood $\{-1,0,1\}$. From now on, $\mathscr{N} = \{-1,0,1\}$.

The construction is first done at macroscopic level (macro-cells at $\mathscr{A}$ scale) to show the reversible process. Then at the microscopic level (at $\mathscr{U}$ scale), the steps of the process are detailed. Macro-cells as well as $\mathscr{U}$-cells are products of different layers. The states and the local function $\Phi_{\mathscr{U}}$ are defined in tables 1 and 3.

4.1 Macroscopic Level

Let B_x be the x^{th} element of Q modulo $|Q|$. An $\mathscr{A}$-configuration is encoded by macro-cell as in Fig. 7 in 4 layers: an index to identify the cell in the loop, an entry of the table of $\Phi_{\mathscr{A}}$ (with the same id), the $\mathscr{A}$-state and a signal and mode (lower/upper case) to know the current step of the simulation. The initial configuration presented in Fig. 7 extends infinitely on each sides. The value is denoted by V_x at the beginning and W_x after exchanging parts with neighbors (and $\Phi_{\mathscr{A}}(W_x)$ after updating).

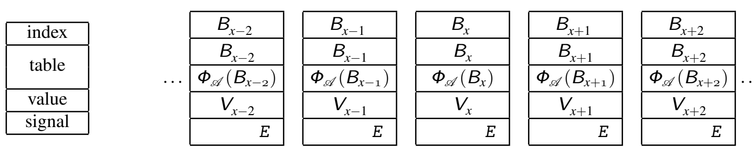
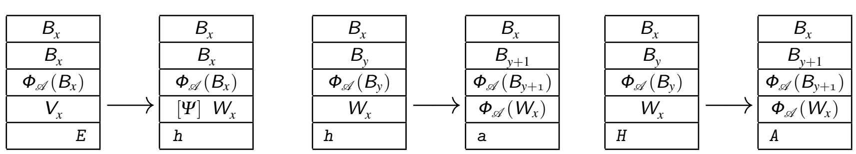


Fig. 7 Initial configuration encoding at $\mathscr{A}$-cell level.

From PCA definition, all $\mathscr{A}$-cells first exchange their -1 and 1 parts and the signal changes from `E` on right to `h` on left to denote this (the rule in Fig. 8(a)). The mode changes from uppercase to lowercase.

The inner loop of the simulation starts then. First the layers holding B_y and $\Phi_{\mathscr{A}}(B_y)$ are shift to the left with the rules in Fig. 8(a). The mode is preserved.

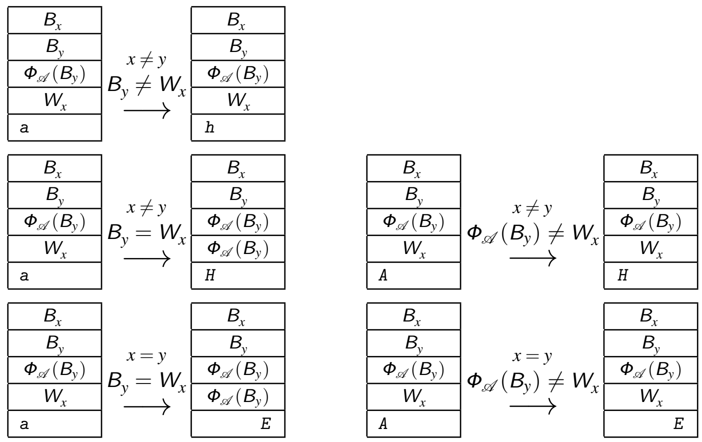


(a) exchanging sub-states (b) shifting the table

$[\Psi]$ means -1 and 1 parts exchanged with adjacent cells.

Fig. 8 Begining of cycle and table shifting at $\mathscr{A}$-cell level.

If the mode is lowercase (signal `a`) then if W_x to B_y are equal, W_x is replaced by $\Phi_{\mathscr{A}}(B_y)$ and the signals turn to uppercase (the rules in Fig. 9(a)). If the mode is an uppercase signal (`A`) then it ensures that $\Phi_{\mathscr{A}}(B_y)$ to B_y are different (the rules in Fig. 9(b)). Finally, B_x and B_y are check for equality to know whether the loop is ended (the bottom rules in Figs. 9(a) and 9(b)).



(a) lowercase mode (b) uppercase mode

Fig. 9 Update inside the loop at $\mathscr{A}$-cell level.

4.2 States, Layers and Configurations at Microscopic Level

The $\mathscr{U}$-cells are organized in 10 layers as detailed in Tab. 1. Architecture layer (A) holds delimiters for the $\mathscr{A}$-cells ([and]) and for the -1, 0 and 1 parts (**$**). Layer I holds an index to store where the reading of the table started. Layers B and F hold one entry of the table B_y and its image $\Phi_{\mathscr{A}}(B_y)$. The value of the $\mathscr{A}$-cell (V_x or W_x) is stocked on layer V. Signals are found on layer S. Layers L_1 to L_4 work like conveyor belts to transfer data. The values in layers A and I never change.

Table 1 The 10 layers and corresponding sub-states.

Layer	Name	States -1 0 1	Use
1	A	**_[$]**	Architecture: limits of cells and parts
2	I	01	$\mathscr{A}$-cell identification B_x
3	B	01	Table entry B_y
4	F	01	Image of the table entry B_y, $\Phi_{\mathscr{A}}(B_y)$
5	V	01	Value of the $\mathscr{A}$-cell (V_x or W_x)
6	S	Σ Σ Σ	Control signals as detailed in Tab. 2
7–10	L_1–L_4	01 01	Shift the table of Φ and exchange values (W_x^{-1} & W_x^1)

Capital (B, $\Phi_{\mathscr{A}}(B)$, W) are used to address the macroscopic level ($\mathscr{A}$-cells) and small symbols (i, b, f, v) for microscopic level ($\mathscr{U}$-cells). All $\mathscr{A}$-cells are binary encoded. For the exchange, the codes of -1 and 1 parts must have the same length (0's are added if necessary).

The signals are 23 symbols `typed in this police` as described in Tab. 2. Uppercase and lowercase signals behave similarly except for the table testing. This mode distinguishes between before and after the replacement. During the simulation, signals are turned from uppercase to lowercase when parts are exchanged and back to uppercase when the value is replaced by its image.

Table 2 Signals of $\mathscr{U}$.

lowercase	uppercase	Use
`a-h`		Loop which tests if $W_x = B_y$ and $B_x = B_y$
	`A-H`	Loop which tests if $W_x = \Phi_{\mathscr{A}}(B_y)$ and $B_x = B_y$
`k`		Write $\Phi_{\mathscr{A}}(B_y)$ over W_x
`m`, `n`	`M`, `N`	Shift of the table: B_y and $\Phi_{\mathscr{A}}(B_y)$
	`S`, `T`	Exchange of Parts W_x^{-1} and W_x^1

The encoding of $\mathscr{A}$-cells is given in Fig. 10. It takes care of the particular positions of the -1 and 1 parts from the beginning. Since V_x^{-1} is exchanged with V_{x+1}^1 (V_x^1 with V_{x-1}^{-1}), B_x^1 (B_x^{-1}) should be above it. When $\Phi(W_x)$ replaces W_x, the -1 and 1 parts are directly on the corresponding sides.

		B_x^{-1}	B_x^0	B_x^1	I	B_x^{-1}	B_x^0	B_x^1		
B_x		$[\quad \$$		$\$ \quad]$	A	$[\quad \$$		$\$ \quad]$		B_x
B_x		B_x^{-1}	B_x^0	B_x^1	B	B_x^{-1}	B_x^0	B_x^1		B_x
$\Phi_{\mathscr{A}}(B_x)$	$\Longleftrightarrow$	$\Phi_{\mathscr{A}}(B_x)^1$	$\Phi_{\mathscr{A}}(B_x)^0$	$\Phi_{\mathscr{A}}(B_x)^{-1}$	F	$\Phi_{\mathscr{A}}(B_x)^1$	$\Phi_{\mathscr{A}}(B_x)^0$	$\Phi_{\mathscr{A}}(B_x)^{-1}$	$\Longleftrightarrow$	$\Phi_{\mathscr{A}}(B_x)$
V_x		V_x^1	V_x^0	V_x^{-1}	V	V_{x-1}^{-1}	V_x^0	V_{x+1}^1		W_x
`E`				`E`	S	`h`				`h`
		4 layers for displacements			L1-L4	4 layers for displacements				

Fig. 10 Encoding of $\mathscr{A}$-cell at coordinate x, before and after the exchange.

4.3 Microscopic Algorithm

It is defined by space-time diagrams driven by signals. Signals in the different $\mathscr{A}$-cells are always exactly synchronized. The duration is the same whether or not a test succeed or fail. The end of loop case is shorter but the test is uniformly satisfied (or not). All the rules for lowercase signals are indicated in Tab. 3; the rules for the uppercase are similar. The algorithm starts with `E` signal arriving in the rightmost $\mathscr{U}$-cell of each $\mathscr{A}$-cell.

First, the -1 and 1 parts of the $\mathscr{A}$-cell are exchanged and the signal is switched to $\mathtt{h}$ as depicted in Fig. 11. The initial value of the $\mathscr{A}$-Cell is V_x. The bits of V_x^{-1} and V_{x+1}^{1} are swapped on the layer L_1 by signals $\mathtt{S}$ and $\mathtt{T}$ on they way from **]**. On crossing **]**, the flows are transferred on L_2 (to avoid superposition as explained below). Signals $\mathtt{S}$ and $\mathtt{T}$ turn back at **\$** and on their way back, they retrieve the bits from L_2 and put them in their destination slot with another swap. Synchronization is very important. Signals $\mathtt{S}$ and $\mathtt{T}$ finally get back together as $\mathtt{h}$ at **]** (switching the mode) and go back to the left end of the $\mathscr{A}$-cell. This is implemented with 11 rules in Tab. 3.

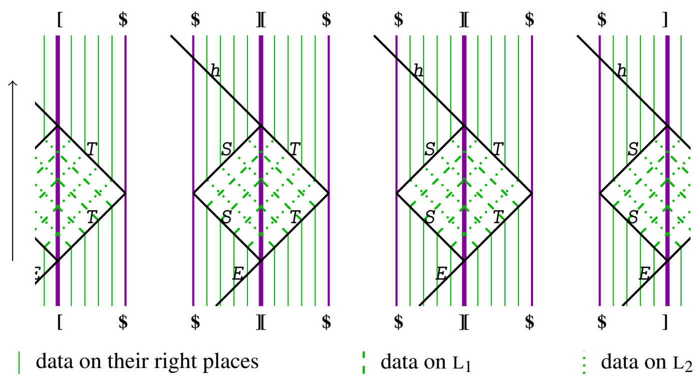


Fig. 11 Exchanging V_x^{-1} and V_{x+1}^{1} to realize the rule in Fig. 8(a).

Signal $\mathtt{h}$ crosses the $\mathscr{A}$-cell and asserts that $B_x = B_y$ (for reversibility). On arriving at **[**, $\mathtt{h}$ splits into $\mathtt{m}$ and $\mathtt{n}$. These signals manage the shift of the table by one $\mathscr{A}$-cell rightward using layers L_1 to L_4 as illustrated in Fig. 12. Signal $\mathtt{m}$ sets B_{y-1} and $\Phi_{\mathscr{A}}(B_{y-1})$ on movement by swapping then on layers L_1 and L_3. On passing **]**, bits go down a layer so as not to interfere with the moving ones of the next $\mathscr{A}$-cell. On its way back, $\mathtt{n}$ sets B_{y-1} and $\Phi_{\mathscr{A}}(B_{y-1})$ on their final places by swapping them from layers L_2 and L_4. Signals $\mathtt{m}$ and $\mathtt{n}$ gather and form $\mathtt{a}$ which starts the test part of the loop. This corresponds to the last 12 rules in Tab. 3.

The test part of the loop works as follows for the lowercase mode. Value W_x and table entry B_y are in place, bit below bit, to be compared. Signal $\mathtt{a}$ crosses the whole $\mathscr{A}$-cell to compare them. If they differ, a marker $\mathtt{b}$ is put on the first different bit, and $\mathtt{a}$ turns to $\mathtt{b}$. On the way back, $\mathtt{b}$ marks $\mathtt{d}$ the last bit which differs and collects the marker $\mathtt{b}$ back (first column in Fig. 13). If W_x and B_y are equal, the signal reaches **[** as $\mathtt{a}$, turns to $\mathtt{k}$, writes $\Phi_{\mathscr{A}}(B_y)$ over W_x on the way back and switch mode (second column in Fig. 13). This special behavior takes as much time as the regular one, keeping the synchronization. Equality (with $\Phi_{\mathscr{A}}(B_y)$) is tested on the way back for reversibility: going backward in time, $\mathscr{U}$ must make the correct change at the adequate time, so it needs this as well as inequality for the rest of the iterations (last two columns in Fig. 13).

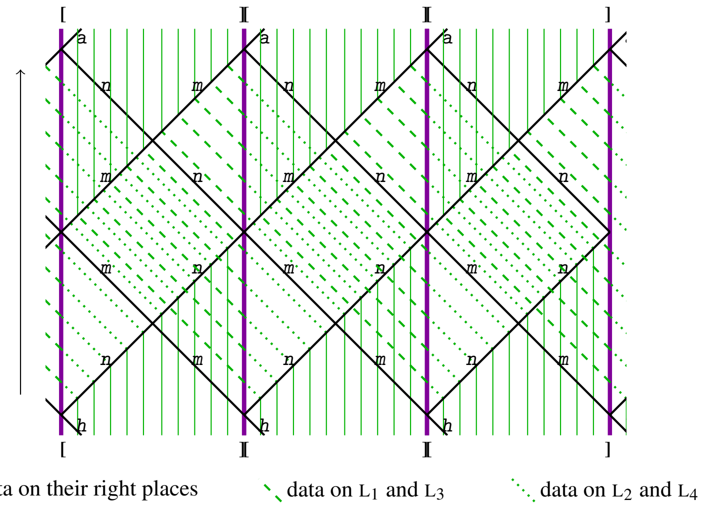


Fig. 12 Shifting the table to realize the rule in Fig. 8(b).

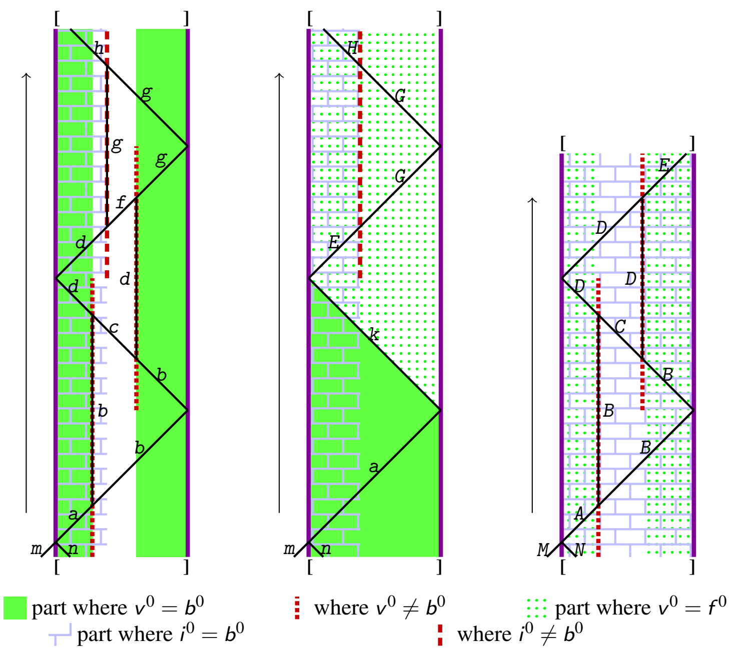


Fig. 13 Test for replacement and for the end of $\mathscr{A}$-iteration (basic cases).

In uppercase mode, it is exactly the same except that the equality is tested between W_x and $\Phi_{\mathscr{A}}(B_y)$ instead of B_y, and B_y as a requirement. Since $\mathscr{A}$ is reversible, each value of $\Phi_{\mathscr{A}}(B_y)$ appears once and only once in the table. After being copied, $\Phi_{\mathscr{A}}(B_y)$ is never met again.

To know that the table was completely scanned, signal must test whether B_x and B_y are equal. On the second left to right crossing, signal d (or e) gets back the previous marker (if any) and turn to g and marks g the first different bit between B_x and B_y (last column in Fig. 13). If they differ, g comes back and gets the marker (first two columns in Fig. 13). If there are equal, it turns (or remains) e and the process is restarted. Uppercase signals behave identically.

4.4 Local Function of $\mathscr{U}$

Most of the definition of $\Phi_{\mathscr{U}}$ is given in Tab. 3. The values of the layers that hold 0 and 1 are not indicated. These values are tested as requirement for rules and are not modified otherwise noted in the last column. These modifications are either swapping or writing on v^0. In the latter case, the previous value is held somewhere else as indicated by a condition. In uppercase mode, the differences are only for the lines with an '$*$': the test made is $v^0 = f^0$ instead of $v^0 = b^0$, and b^0 (instead of f^0) is copied over v^0. Since the rules are one-to-one, $\Phi_{\mathscr{U}}$ can be completed bijectively.

All rules are combined with the following: for the last 4 layers L_1 to L_4, the -1 and 1 parts are swapped so that -1 (1) parts move at speed 1 to the right (left). For all rules with **[**: layers L_1 and L_2 (L_3 and L_4) are swapped. This is technical for the flows of the table shift not to collide in the middle in Fig. 12 where 2 flows are traveling together.

The design BCA is reversible: the provided rules are one-to-one. If the simulated BCA is not reversible, then the simulation just does not work because of the backward tests and the duplicate values in images.

4.5 Simulation

For Def. 7, the packing function p is the grouping by macro-cell, $\mathbf{s}$ is the null shift; the onto function α is one-to-one as described in Fig. 10.

Let a be the width of a $\mathscr{A}$-cell and b the width of the exchanged parts ($0 \leq 2b \leq a$ and $\lceil \log|Q| \rceil \leq a \leq 2\lceil \log|Q| \rceil + 2$). The inner loop needs $4(a-1)$ iterations for the tests and $2a$ for the shift of the table. It is done for every $\mathscr{A}$-state, i.e., $|Q|$ times. To make a $\mathscr{A}$-iteration, values are exchanged between neighboring cells, this needs $2b+1$ iterations. All together, τ is a constant bounded by $12\,|Q|\log(|Q|) + o(|Q|\log(|Q|))$.

The number of states of $\mathscr{U}$ is $2^{13}.24^3$, a little above 113.10^6.

The construction can be extended to greater dimension. The table and test are done in one direction and sub-states exchanged on every directions must be added.

Table 3 Table of $\Phi_{\mathscr{U}}$.

	Structure	Value Signals			Value Condition		Image Signals			Image Modification
*	[		a		$v^0 = b^0$				a	
					$v^0 \neq b^0$			b	b	
*	‘ ’,$	a			$v^0 = b^0$				a	
					$v^0 \neq b^0$			b	b	
*	]	a			$v^0 = b^0$			k		$v^0 \leftarrow f^0$
					$v^0 \neq b^0$		d	d		
	‘ ’,$	b							b	
	‘ ’,$		b					b		
*	]	b			$v^0 = b^0$		b			
					$v^0 \neq b^0$		c	d		
*	‘ ’,$			b	$v^0 = b^0$		b			
					$v^0 \neq b^0$		c	d		
*	‘ ’,$		b	b	$v^0 \neq b^0$		d	d		
*	[		b	b	$v^0 \neq b^0$	$i^0 = b^0$		e		
						$i^0 \neq b^0$		g	g	
*	‘ ’,$		b	c	$v^0 \neq b^0$		d			
*	[		b	c	$v^0 \neq b^0$	$i^0 = b^0$			d	
						$i^0 \neq b^0$		g	f	
	‘ ’,$			c			c			
*	‘ ’,$			d	$v^0 = b^0$		d			
*	[			d	$v^0 = b^0$	$i^0 = b^0$			d	
						$i^0 \neq b^0$		g	f	
*	‘ ’,$	d			$v^0 = b^0$	$i^0 = b^0$			d	
						$i^0 \neq b^0$		g	f	
*	‘ ’,$	d	d		$v^0 \neq b^0$	$i^0 = b^0$			e	
						$i^0 \neq b^0$		g	g	
*	]	d	d		$v^0 \neq b^0$	$i^0 = b^0$		S	T	
						$i^0 \neq b^0$	h			
*	‘ ’,$	e			$v^0 = b^0$	$i^0 = b^0$			e	
						$i^0 \neq b^0$		g	g	
*	]	E			$v^0 = b^0$	$i^0 = b^0$		S	T	
						$i^0 \neq b^0$	H			
	‘ ’,$	f							f	
*	‘ ’,$	f	d		$v^0 \neq b^0$				g	
*	]	f	d		$v^0 \neq b^0$		g			
*	‘ ’,$	g			$v^0 = b^0$				g	
*	]	g			$v^0 = b^0$		g			
	[, ‘ ’,$		g					g		
	‘ ’,$			g			g			
	‘ ’,$		g	g		$i^0 \neq b^0$	h			
	[		g	g		$i^0 \neq b^0$	m	n		
	‘ ’,$			h		$i^0 = b^0$	h			
	[			h		$i^0 = b^0$	m	n		
	]		S				S			$\mathrm{swap}(v^0, l_1^1)$
	‘ ’			S			S			$\mathrm{swap}(v^0, l_1^1)$
	$			S				S		$\mathrm{swap}(v^0, l_1^1)$
	$		S						S	$\mathrm{swap}(v^0, l_2^1)$
	‘ ’	S							S	$\mathrm{swap}(v^0, l_2^1)$
	]	S						S		$\mathrm{swap}(v^0, l_2^1)$
	[, ‘ ’	T							T	$\mathrm{swap}(v^0, l_1^{-1})$
	$	T						T		$\mathrm{swap}(v^0, l_1^{-1})$
	$		T				T			$\mathrm{swap}(v^0, l_2^{-1})$
	[, ‘ ’			T			T			$\mathrm{swap}(v^0, l_2^{-1})$
	]		S	T			h			
*	‘ ’,$			k	$v^0 = b^0$		k			$v^0 \leftarrow f^0$
*	[			k	$v^0 = b^0$				E	$v^0 \leftarrow f^0$
	], ‘ ’			m			m			$\mathrm{swap}(b^0, l_1^1)$, $\mathrm{swap}(f^0, l_3^1)$
	‘ ’,$			m			m			$\mathrm{swap}(b^0, l_1^1)$, $\mathrm{swap}(f^0, l_3^1)$
	‘ ’,$	n		m			m		n	$\mathrm{swap}(b^0, l_1^1)$, $\mathrm{swap}(f^0, l_3^1)$
	[			m					m	
	‘ ’,$]	m							m	
	[		n						n	
	‘ ’,$	n							n	
	]	n						n		
	]		n				n			$\mathrm{swap}(b^0, l_2^{-1})$, $\mathrm{swap}(f^0, l_4^{-1})$
	‘ ’,$			n			n			$\mathrm{swap}(b^0, l_2^{-1})$, $\mathrm{swap}(f^0, l_4^{-1})$
	[			n				n		$\mathrm{swap}(b^0, l_2^{-1})$, $\mathrm{swap}(f^0, l_4^{-1})$
	[	m	n						a	

5 Space-time Simulation of Irreversible CA by Reversible Ones

5.1 Space-time Approach

A space-time diagram $\mathbb{A}$ is *embedded* into another space-time diagram $\mathbb{B}$ when it is possible to "reconstruct" $\mathbb{A}$ from $\mathbb{B}$ and the way that $\mathbb{A}$ is embedded into $\mathbb{B}$.

The recovering of an embedded $\mathscr{A}$-configuration is done in the following way. A $\mathscr{B}$-configuration is constructed by taking each cell at a given iteration. This $\mathscr{B}$-configuration is decoded to get an iterated configuration for $\mathscr{A}$. More precisely, it is defined as follows:

Definition 21 A space-time diagram $\mathbb{A} = (\mathscr{A}, a)$ is *space-time embedded* into another space-time diagram $\mathbb{B} = (\mathscr{B}, b)$ when there exist two functions $\chi : \mathbb{L} \times \mathbb{N} \to \mathbb{N}$ and $\zeta : \mathscr{C}_{\mathscr{B}} \to \mathscr{C}_{\mathscr{A}}$ such that:
- $\forall (\mathbf{x}, t) \in \mathbb{L} \times \mathbb{N}$, let c^t be the configuration of $\mathscr{B}$ such that $c^t_{\mathbf{x}} = \mathbb{B}_{\mathbf{x}, \chi(\mathbf{x}, t)}$ and
- $\forall t \in \mathbb{N}$, $\mathscr{G}^t_{\mathscr{A}}(a) = \zeta(c^t)$.

To recover an iterated image of a, the function χ indicates which iteration is to be considered for each cell and ζ decodes the assembled configuration. The generation of c^t and then $\mathscr{G}^t_{\mathscr{A}}(a)$ is illustrated in Fig. 14.

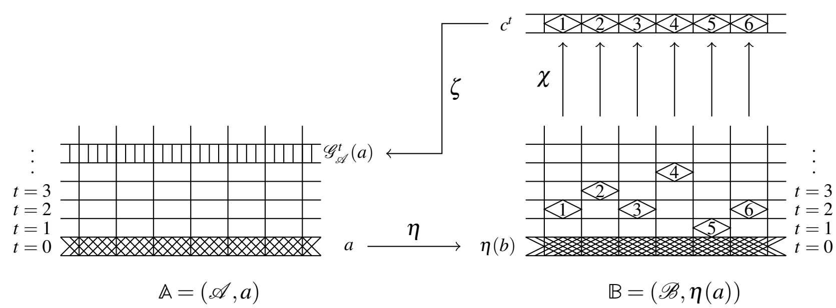


Fig. 14 Space-time diagram $\mathbb{A}$ is embedded into $\mathbb{B}$.

The functions χ and ζ could be complex and provide all the computation. To avoid this, in the following definition, they are independent from the initial configuration, thus unable to do much of the $\mathscr{A}$ computation.

Definition 22 A CA $\mathscr{B}$ *space-time simulates* a CA $\mathscr{A}$ when there exists a function $\eta : \mathscr{C}_{\mathscr{A}} \to \mathscr{C}_{\mathscr{B}}$ such that any space-time diagram $(\mathscr{A}, a)$ is embedded into the space-time diagram $(\mathscr{B}, \eta(a))$ and all embeddings use the same functions χ and ζ.

This section provides a construction to prove the following lemma:

Lemma 23 *Any d-CA with neighborhood $\{-1, 0, 1\}^d$ can be space-time simulated by a d-R-PCA.*

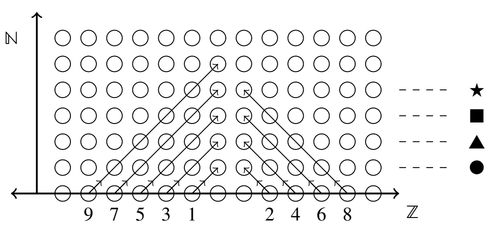
The proof is only detailed in dimension 1. The generalization to greater dimensions is sketched at the end of this section.

5.2 Macro Dynamics

Let $\mathscr{A}$ =$(Q_{\mathscr{A}}, \{-1,0,1\}, f)$ be any 1-CA. The 1-R-PCA $\mathscr{B} = (Q_{\mathscr{B}}, \{-1,0,1\}, \Phi)$ which space-time simulates $\mathscr{A}$ is progressively constructed. Let a be any configuration in $\mathscr{C}_{\mathscr{A}}$ and $\mathbb{A}$ the associated space-time diagram. The space-time diagram $\mathbb{B}$ is generated in order to embed $\mathbb{A}$ as follows.

A signal moves forth and back and updates the cells on a finite part of the configuration called the *updating zone*. Outside of this zone, the $\mathscr{B}$-cell are at $\mathscr{A}$-iteration 0. Inside, $\mathscr{A}$-iteration number increases as $\mathscr{B}$-cell are closer to the center of the zone. As $\mathscr{B}$-iterations go, the updating zone is enlarged on both sides (space) and in iteration numbers (time) so that each cell will eventually enter the zone and reach any iteration.

The simulated diagram $\mathbb{A}$ is generated according to diagonal lines, one after the other. The updating lines of $\mathbb{A}$ are depicted in Fig. 15 where the numbers, the arrows and the geometrical symbols on the last column correspond respectively to the order in which updates are made, to their directions and to the identifications of the $\mathscr{A}$-iterations (as on $\mathbb{B}$ in Fig. 16).



The symbol in the last column identifies the $\mathscr{A}$-iteration.
It corresponds to the embedding in Fig. 16.

Fig. 15 Order of generation inside the simulated diagram $\mathbb{A}$.

The state of a cell x at iteration t in the embedded diagram $\mathbb{A}$ is denoted by ${}_x\backslash^t$ (${}_x\backslash^t = \mathscr{G}^t_{\mathscr{A}}(a)_x$) and the information needed to compute ${}_x\backslash^t$ is denoted by $[{}_x\backslash^t]$ ($[{}_x\backslash^t] = ({}_{x-1}\backslash^{t-1}, {}_x\backslash^{t-1}, {}_{x+1}\backslash^{t-1})$). Each time a cell is updated, a $[{}_x\backslash^t]$ is generated to keep the data needed for undoing the update. The generated data cannot be disposed off because $\mathscr{G}_{\mathscr{A}}$ is not necessarily one-to-one and the previous configuration might be uncomputable from the actual one. These needed but cumbersome data are evacuated by being sent away outside of the updating zone.

When a signal goes from the left to the right for the n^{th} time on the updating zone, as in Fig. 16, its dynamics work as follows:

Starting from the far left of the updating zone, the first cell encountered by the signal holds $[{}_{x}\backslash^{1}]$. The signal sets this data moving to the left to save it and evacuate it while generating ${}_{x}\backslash^{1}$. The next cell holds $[{}_{x+1}\backslash^{2}]$ which is also set on movement to the right while ${}_{x+1}\backslash^{2}$ is generated. This goes on until the signal reaches the middle of the updating zone (vertical line), then no more updating is done until the signal reaches the right end. On its way back, the signal updates the other half of the updating zone.

The signal makes n updates one way and n updates on its way back. Then it makes $n+1$ and $n+1$ updates, then $n+2$ and so on. The cells corresponding to the iteration 1 (2, 3 and 4 respectively) in $\mathbb{A}$ are generated on a parabola indicated by ● (▲, ■, ★ respectively) on the simulating diagram $\mathbb{B}$ in Fig. 16. This corresponds to the layer-construction of $\mathbb{A}$ depicted in Fig. 15. Figure 16 depicts the evacuation of the $[{}_{x}\backslash^{y}]$ away from the updating zone for the first 100 iterations. Evacuated data never interact.

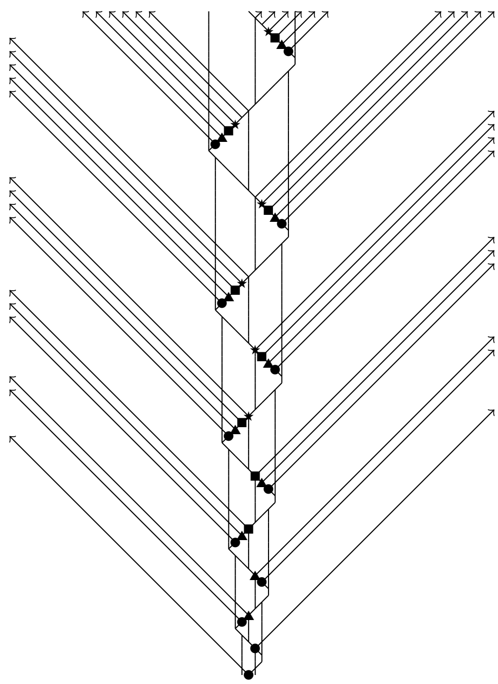


Fig. 16 Scheme of the evacuation of data ($[{}_{x}\backslash^{y}]$) on the simulating diagram.

5.3 Micro Dynamics

Cells are organized in 3 layers: the upper layer holds the state of the simulated cell, the middle one holds the signal that drives the dynamics and the lower one acts like a conveyor belt to evacuate the $[{}_{x}\diagdown^{t}]$.

The first 26 iterations are depicted in Fig. 17. In the upper layer, the cells alternatively hold 3 times the same state $({}_{x}\diagdown^{t})$ or the states of the cell and its 2 closest neighbors at the same iteration $({}_{x-1}\diagdown^{t-1}, {}_{x}\diagdown^{t-1}, {}_{x+1}\diagdown^{t-1})$, otherwise some mix over 2 or 3 iterations. A cell can only be updated when it has the information $[{}_{x}\diagdown^{t}]$. By induction from the dynamics in Fig. 17, the possibility to update a cell only depends on the parity of the sum of simulating and simulated iteration numbers.

The signal that rules the dynamics is called the *suit signal*. Depending on its position, it takes the values ♣, ♠, ♥ and ♦ in $\mathbb{B}$. The suit signal only moves forth and back in the updating zone and thus appears as a zigzag in Figs. 16 and 17. It is delayed by one on the left side to keep it synchronized with the presence of $[{}_{x}\diagdown^{t}]$.

The updating zone is delimited by a pair of ❚ and its middle is indicated by a ★. The ❚ progressively move away from each other while the ★ oscillates in the middle. Starting on the left ❚, the suit signal is ♥. While passing, it makes the updates of the simulated cells until it reaches ★. Afterwards it is ♠ and just moves to the other ❚.

Each time a simulated update is done, 3 values, ${}_{x-1}\diagdown^{t-1}$, ${}_{x}\diagdown^{t-1}$ and ${}_{x+1}\diagdown^{t-1}$, are "used up" and become useless. They are gathered in $[{}_{x}\diagdown^{t}]$ and moved to the lower layer to be evacuated. Three copies of the new state ${}_{x}\diagdown^{t}$ are made. They will be used for the next update of the simulated cell and of its 2 neighbors.

The endless movement of the suit signal and updates (at correct parities) are deduced by induction. Since the interaction is only local and has radius 1, global properties are not otherwise modified. All the necessary steps for the induction can be found on the two and a half loops of the suit signal in Fig. 17.

5.4 State Function

The R-BCA $\mathscr{B}$ has $100\,|Q_{\mathscr{A}}|^3\,(|Q_{\mathscr{A}}|^3+1)^2$ states detailed in Tab. 4.

Table 4 States of $\mathscr{B}$.

1	0	-1
$Q_{\mathscr{A}}$	$Q_{\mathscr{A}}$	$Q_{\mathscr{A}}$
♣ ♠ ♥ ♦ _	✚ ❚ ★ _	♣ ♠ ♥ ♦ _
$Q^3_{\mathscr{A}} \cup \{\_\}$	_	$Q^3_{\mathscr{A}} \cup \{\_\}$

Cells are depicted as 3×3 arrays as in the first line in Fig. 17. The upper layer encodes a configuration of $\mathscr{A}$. The middle layer holds the suit signal. The lower layer is used to store the data away from the updating zone.

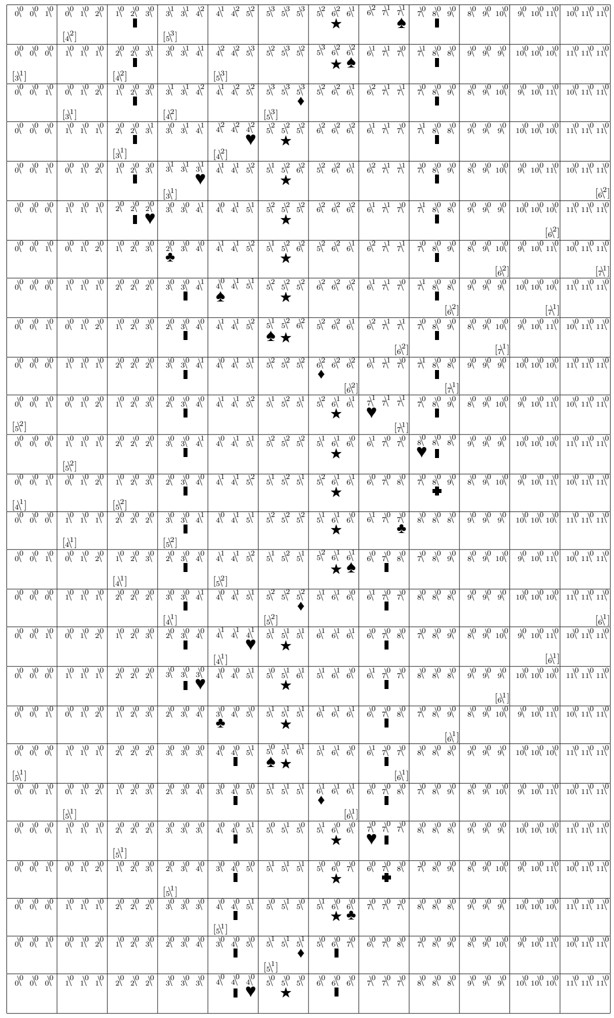


Fig. 17 The first 26 iterations of the space-time simulation.

The suit signal is alternatively equal to ♥ and ♠. When shifting, ♥ updates cells while ♠ does nothing. The signal becomes ♣ and ♦ to move respectively ▮ and ★.

The transition rules are given in Fig. 18. The first rule corresponds to the lack of any signal. On the lower layer, the 2 values on the side are swapped, this acts like a conveyor belt. As soon as something is put on the lower layer, it is shifted by one cell at each iteration. This is used to evacuate data. The updating rules are on the lines 2 and 5.

The second and third lines in Fig. 18 depicted how ♥ moves to the right and updates cells. When it reaches the middle frontier ★, it moves it one step to the right as ♦ and then turns to ♠.

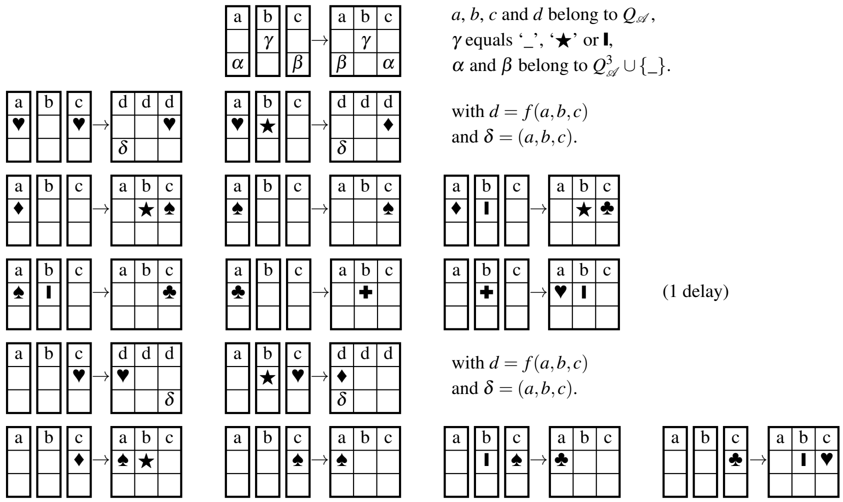


Fig. 18 Definition of $\Phi_{\mathscr{B}}$.

The signal ♠ turns on the right side, as depicted on the fourth line in Fig. 18. On arriving on ▌ from the left, ♠ grabs it and turns to ♣. On the next iteration, ♣ turns to ✚ and does nothing else. This is the delay of one iteration needed to keep up with parity. Next iteration, ✚ regenerates the ▌ and the signal ♥ which goes back to the left.

The signal turns back one iteration faster on the left side as depicted on the last line in Fig. 18: the state ✚ does not appear.

The rules defined are one-to-one, thus they can be completed so that Φ is a permutation; $\mathscr{A}$ is then reversible (Lem. 5).

The initial configuration is depicted on the first line in Fig. 17. The state of each cell is copied 3 times in the upper layer. Markers ▌, ★ and ▌ are laid in the center of 3 adjacent cells and the ♥ is together with the left ▌.

With this construction, the embedded space-time diagram is bent in a parabola shape. This makes it meaningless to access geometrical properties like, e.g., Fisher constructibility or Firing Squad Synchronization.

5.5 Generalization Sketch

This construction can be generalized to any dimension greater than 1. The simulated configurations are still "bent" according to the first direction in the simulating diagram. Along the first direction, the dynamics are exactly as explained above. The signals are duplicated along the other directions. The updatings are still conditioned by parities. There are an infinity of ▌, ★ and suit signals. They are arranged on hyperplanes orthogonal to the first direction and are exactly synchronized.

Any d-CA can be simulated by a d-CA whose neighborhood is $\{-1,0,1\}^d$ (and this simulation is transitively compatible). From Lemma 23 and the fact that d-R-PCA are d-R-CA comes:

Theorem 24 *Any d-CA can be space-time simulated by a d-R-CA.*

Since there are d-R-CA able to simulate all d-R-CA over any configuration and the simulations are compatible enough:

Theorem 25 *There are d-R-CA able to space-time simulate any d-CA.*

6 Conclusion

Conjectures 1 and 2 are true even if states in Q^2 are used in intermediate configurations during the simulation, the input and output are restricted to Q.

For any d, d-CA, d-BCA and d-PCA have the same power over infinite configurations. The same holds for d-R-CA, d-R-BCA and d-R-PCA classes. This is an important result since reversibility is decidable for BCA and PCA while it is not for CA. This is not a contradiction since the inverse CA is needed for the construction.

The proof of Th. 12 is more involved than the one in Durand-Lose [1995]. Nevertheless, the number of block transitions needed is lowered from $2^{d+1}-1$ to $d+1$. Generating and erasing are done concurrently, not one after the other. We conjecture that it is impossible to make a representation with less that $d+1$ block transitions.

The expression with block transitions allows one to use reversible circuitry in order to build R-CA. This was done in Durand-Lose [1995] to prove that, for $2 \leq d$, there exists d-dimensional R-CA (based on the the Billiard ball model) able to simulate any d-dimensional R-CA on infinite configurations. Kari [1999] provides more information on the relation between R-CA and BCA and the inner structure of R-CA in dimensions 1 and 2.

The $\mathscr{U}$ is programmed: loops, tests and conditional executions. Basic programming schemes can be embedded in R-PCA when conceived reversible: a global dynamic of move, test and replace which needs backward tests.

There exist simulations of any Turing machines with R-CA Morita [1992b] so that all partial recursive functions can be computed by R-PCA, so that $\mathscr{U}$ is computationally universal. The existence of an intrinsically universal R-PCA is proven here with the use of the source code of the R-PCA. So there should be some *S-m-n* theorem for R-PCA to prove that they form an acceptable programming system as proved for CA by Martin [1994].

It is unknown whether the class of d-CA is strictly more powerful than the class of d-R-CA on infinite configurations. Nevertheless, if a 1-R-CA can simulate a non reversible CA, then by transitivity, $\mathscr{U}$ is also able to do it, so that if $\mathscr{U}$ cannot, none can.

With space-time simulation, reversible can simulate irreversible, but this simulation is not homogeneous; it is not shift invariant nor time invariant. An infinite time is required to fully generate the configuration after one iteration. Moreover, it is not possible to go backward before the first configuration if no such configuration were encoded in the initial configuration —anyway, there is no guarantee that any previous configuration does exist. When the significant part of a configuration represents only a finite part of the space, the result of the computation is given in finite time like in Morita [1992b, 1995]. In Toffoli [1977], an extra dimension is used to store information for reversibility, here configurations are bent to provide the room.

References

Jürgen Albert and Karel Čulik II. A simple universal cellular automaton and its one-way and totalistic version. *Complex Systems*, 1:1–16, 1987.

Serafino Amoroso and Yale N. Patt. Decision procedure for surjectivity and injectivity of parallel maps for tessellation structure. *J Comput System Sci*, 6:448–464, 1972.

Charles H. Bennett. Logical reversibility of computation. *IBM J Res Dev*, 6:525–532, 1973.

Charles H. Bennett. Notes on the history of reversible computation. *IBM J Res Dev*, 32(1):16–23, 1988.

Arthur W. Burks. *Essays on Cellular Automata*. Univ. of Illinois Press, 1970.

Jérôme Durand-Lose. Reversible cellular automaton able to simulate any other reversible one using partitioning automata. In *LATIN 1995*, number 911 in LNCS, pages 230–244. Springer, 1995. doi: 10.1007/3-540-59175-3_92.

Jérôme Durand-Lose. *Automates Cellulaires, Automates à Partitions et Tas de Sable*. Thèse de doctorat, LaBRI, 1996. URL `http://www.univ-orleans.fr/lifo/Members/Jerome.Durand-Lose/Recherche/These/index.html`. In French.

Jérôme Durand-Lose. Intrinsic universality of a 1-dimensional reversible cellular automaton. In *STACS 1997*, number 1200 in LNCS, pages 439–450. Springer, 1997. doi: 10.1007/BFb0023479.

Jérôme Durand-Lose. About the universality of the billiard ball model. In M. Margenstern, editor, *Universal Machines and Computations (UMC 1998)*, volume 2, pages 118–133. Université de Metz, 1998.

Jérôme Durand-Lose. Reversible space-time simulation of cellular automata. *Theoret Comp Sci*, 246(1–2):117–129, 2000. doi: 10.1016/S0304-3975(99)00075-4.

Jérôme Durand-Lose. Representing reversible cellular automata with reversible block cellular automata. In Robert Cori, Jacques Mazoyer, Michel Morvan, and Rémy Mosseri, editors, *Discrete Models: Combinatorics, Computation, and Geometry, DM-CCG 2001*, volume AA of *Discrete Mathematics and Theoretical Computer Science Proceedings*, pages 145–154, 2001a. URL `http://dmtcs.loria.fr/volumes/abstracts/dmAA0110.abs.html`.

Jérôme Durand-Lose. Back to the universality of the Billiard ball model. *Multiple Valued Logic*, 6 (5–6):423–437, 2001b. doi: None,juillet2021.

Gustav A. Hedlund. Endomorphism and automorphism of the shift dynamical system. *Math System Theory*, 3:320–375, 1969.

Jarkko Kari. Reversibility of 2D cellular automata is undecidable. *Phys D*, 45:379–385, 1990.

Jarkko Kari. Reversibility and surjectivity problems of cellular automata. *J Comput System Sci*, 48(1):149–182, 1994.

Jarkko Kari. Representation of reversible cellular automata with block permutations. *Math System Theory*, 29:47–61, 1996.

Jarkko Kari. On the circuit depth of structurally reversible cellular automata. *Fundamenta Informaticae*, 38(1–2):93–107, 1999.

Jarkko Kari. Theory of cellular automata: a survey. *Theoret Comp Sci*, 334:3–33, 2005.

Yves Lecerf. Machines de Turing réversibles. Récursive insolubilité en $n \in \mathbb{N}$ de l'équation $u = \theta^n u$, où θ est un isomorphisme de codes. *Comptes rendus des séances de l'académie des sciences*, 257:2597–2600, 1963.

Norman Margolus. Physics-like models of computation. *Phys D*, 10(1–2):81–95, 1984.

Norman Margolus. *Physics and Computation*. PhD thesis, MIT, 1988.

Bruno Martin. A universal cellular automaton in quasi-linear time and its S-n-m form. *Theoret Comp Sci*, 123:199–237, 1994.

Edward F. Moore. Machine models of self-reproduction. In *Proceeding of Symposium on Applied Mathematics*, volume 14, pages 17–33, 1962.

Kenichi Morita. Any irreversible cellular automaton can be simulated by a reversible one having the same dimension. *Technical Report of the IEICE, Comp.*, 92-45 (1992-10):55–64, 1992a.

Kenichi Morita. Computation-universality of one-dimensional one-way reversible cellular automata. *Inform Process Lett*, 42:325–329, 1992b.

Kenichi Morita. Reversible simulation of one-dimensional irreversible cellular automata. *Theoret Comp Sci*, 148:157–163, 1995.

Kenichi Morita. Reversible computing and cellular automata - A survey. *Theoret Comp Sci*, 395 (1):101–131, 2008. doi: 10.1016/j.tcs.2008.01.041.

Kenichi Morita and Masateru Harao. Computation universality of one-dimensional reversible (injective) cellular automata. *Transactions of the IEICE*, E 72(6):758–762, 1989.

John R. Myhill. The converse of Moore's garden-of-eden theorem. In *Proceeding of the American Mathematical Society*, volume 14, pages 685–686, 1963.

Nicolas Ollinger. Two-states bilinear intrinsically universal cellular automata. In *Fundamentals of Computation Theory, 13th International Symposium (FCT 2001)*, number 2138 in LNCS, pages 369–399. Springer, 2001.

Nicolas Ollinger. Universalities in cellular automata*. In Grzegorz Rozenberg, Thomas Bäck, and Joost N. Kok, editors, *Handbook of Natural Computing*, pages 189–229. Springer, 2012. doi: 10.1007/978-3-540-92910-9_6.

Daniel Richardson. Tessellations with local transformations. *J Comput System Sci*, 6(5):373–388, 1972.

Palash Sarkar. A brief history of cellular automata. *ACM Computing Surveys*, 32(1):80–107, 2000.

Tommaso Toffoli. Computation and construction universality of reversible cellular automata. *J Comput System Sci*, 15:213–231, 1977.

Tommaso Toffoli and Norman Margolus. *Cellular Automata Machine — A New Environment for Modeling*. MIT press, Cambridge, MA, 1987.

Tommaso Toffoli and Norman Margolus. Invertible cellular automata: a review. *Phys D*, 45: 229–253, 1990.

Stephen Wolfram. *Theory and Applications of Cellular Automata*. World Scientific, 1986.

Table of symbols

Symbol	Definition	Page
CA	Cellular Automaton/a	1
PCA	Partitioned Cellular Automaton/a	1
BCA	Block Cellular Automaton/a	1
d	Dimension of a Cellular Automaton	1
R-CA	Reversible Cellular Automaton/a	1
$\mathbb{Z}$	Set of all integers	4
Q	Set of states of a Cellular Automaton	4
$\mathscr{N}$	Neighborhood of a Cellular Automaton	4
$\mathscr{G}$	Global function of a Cellular Automaton	4
R-PCA	Reversible Partitioned Cellular Automaton/a	5
R-BCA	Reversible Block Cellular Automata	5
t	Local function of a Block Cellular Automaton (function over blocks)	5
r	Radius of a Cellular Automaton	6
$\mathbb{L}$	CA lattice for the Cellular Automata of a given dimension	7
c	Configuration of a Cellular Automaton	7
$\mathscr{C}$	Set of all configurations of a Cellular Automaton	7
E	Some subset of $\mathbb{L}$	7
$\mathbf{x}$	Some vector in $\mathbb{L}$	7
σ	Shift over $\mathbb{L}$	7
$\mathbf{i}$	Some vector in $\mathbb{L}$	7
f	Local function of a Cellular Automaton	8
μ	Coordinates in the neighborhood of a CA	8
Φ	Local function of a partitioned Cellular Automaton (function over states expressed as a product)	8
$\mathbf{v}$	Size of the block of a BCA (vector)	8
n	Number of partitions of a BCA	8
$\mathbf{o}$	Origin of a partition of a BCA	8
V	Block of a BCA (hyper-cube)	8
T	Block transition	8
$\mathbf{a}$	Some vector in $\mathbb{L}$	8
$\mathbf{b}$	Some vector in $\mathbb{L}$	8
ρ	Block in a partition	9
q	State of a Cellular Automaton	9
$\mathbb{A}$	One space-time diagram	10
$\mathbb{L} \times \mathbb{N}$	Space-time lattice	10
$\mathbb{N}$	Set of all natural integers	10
$\mathscr{A}$	A Cellular Automaton	10
$\mathscr{B}$	Another Cellular Automaton	11
α	Partial function on states for direct simulation	11
$\preccurlyeq$	Directly simulate	11
$\mathbf{m}$	Simulate packing vector	11
p	Simulation packing function	11
τ	Simulation time delay	11
$\mathbf{s}$	Simulation shift	11
$\lll$	Simulate	11
ν	Another coordinates in the neighborhood of a CA	13

$\mathbf{r}$	Vector $(r, r, \cdots, r)$ of $\mathbb{Z}^d$ for a CA	14
θ	Index of partition in the simulation of R-CA by R-PCA	14
κ	Sub-index of partition in the simulation of R-CA by R-PCA	14
F	Set bases to identify the partition in the simulation of R-CA by R-BCA	14
E	Sets to identify the partition in the simulation of R-CA by R-BCA	14
$\bowtie$	Product if both defined, otherwise the defined one	15
$\mathscr{E}$	Set of configurations in the simulation of R-CA by R-BCA	15
B	Block transition in the simulation of R-CA by R-BCA	15
$\mathbf{y}$	Some other vector in $\mathbb{L}$	16
λ	Onother sub-index of partition in the simulation of R-CA by R-PCA	16
ε	Witness vector for the simulation of R-CA by R-BCA	17
$\mathscr{U}$	Intrinsic universal R-CA	18
$\Phi_{\mathscr{U}}$	Local rule of U	18
B	(for $\mathscr{U}$) binary encoding of $\mathscr{A}$ states	19
V	(for $\mathscr{U}$) encoding of $\mathscr{A}$-cell before exchanging	19
W	(for $\mathscr{U}$) encoding of $\mathscr{A}$-cell after exchanging with neighbors	19
$\mathtt{E}$	Meta-signal $\mathtt{E}$	19
$\mathtt{h}$	Meta-signal $\mathtt{h}$	19
$\mathtt{a}$	Meta-signal $\mathtt{a}$	19
$\mathtt{H}$	Meta-signal $\mathtt{H}$	19
$\mathtt{A}$	Meta-signal $\mathtt{A}$	19
A	Layer of $\mathscr{U}$-states	20
[	Left $\mathscr{A}$-cell for simulation with $\mathscr{U}$	20
]	Right $\mathscr{A}$-cell for simulation with $\mathscr{U}$	20
$	Sub-state separation in $\mathscr{A}$-cell for simulation with $\mathscr{U}$	20
I	Layer of $\mathscr{U}$-states	20
B	Layer of $\mathscr{U}$-states	20
F	Layer of $\mathscr{U}$-states	20
V	Layer of $\mathscr{U}$-states	20
S	Layer of $\mathscr{U}$-states	20
L	4 layers of $\mathscr{U}$-states	20
i	(for $\mathscr{U}$) encoding of $\mathscr{A}$-cell at $\mathscr{U}$-level	21
b	(for $\mathscr{U}$) encoding of $\mathscr{A}$-cell at $\mathscr{U}$-level	21
f	(for $\mathscr{U}$) encoding of $\mathscr{A}$-cell at $\mathscr{U}$-level	21
v	(for $\mathscr{U}$) encoding of $\mathscr{A}$-cell at $\mathscr{U}$-level	21
$\mathtt{k}$	Meta-signal $\mathtt{k}$	21
$\mathtt{m}$	Meta-signal $\mathtt{m}$	21
$\mathtt{n}$	Meta-signal $\mathtt{n}$	21
$\mathtt{M}$	Meta-signal $\mathtt{M}$	21
$\mathtt{N}$	Meta-signal $\mathtt{N}$	21
$\mathtt{S}$	Meta-signal $\mathtt{S}$	21
$\mathtt{T}$	Meta-signal $\mathtt{T}$	21
$\mathtt{b}$	Meta-signal $\mathtt{b}$	22
$\mathtt{d}$	Meta-signal $\mathtt{d}$	22
$\mathtt{c}$	Meta-signal $\mathtt{c}$	23
$\mathtt{f}$	Meta-signal $\mathtt{f}$	23

g	Meta-signal g	23
G	Meta-signal G	23
B	Meta-signal B	23
C	Meta-signal C	23
D	Meta-signal D	23
e	Meta-signal e	24
a	width of a $\mathscr{A}$-cell in the $\mathscr{U}$ simulation	24
b	width of the exchanged parts in the $\mathscr{U}$ simulation	24
l	(for $\mathscr{U}$) encoding of $\mathscr{A}$-cell at $\mathscr{U}$-level	25
$\mathbb{B}$	Another space-time diagram	26
χ	Space-time simulation: get simulated time	26
ζ	Space-time simulation: decoding function	26
η	Space-time simulation: encoding function	26
●	Layer 1 in space-time simulation	27
▲	Layer 2 in space-time simulation	27
■	Layer 3 in space-time simulation	27
★	Layer 4 in space-time simulation	27
♣	Club Signal for space-time simulation	29
♠	Space Signal for space-time simulation	29
♥	Heart Signal for space-time simulation	29
♦	Diamond Signal for space-time simulation	29
❙	Zone delimiter for space-time simulation	29
★	Center delimiter for space-time simulation	29
✚	Center plus delimiter for space-time simulation	29

bibtex entry

```
@incollection{durand-lose18rev-book,
  doi = {10.1007/978-3-319-73216-9_4},
  author = {Durand-{L}ose, J{\'e}r{\^o}me},
  title = {Simulation and {I}ntrinsic {U}niversality {A}mong
                  {R}eversible {C}ellular {A}utomata, the {P}artition
                  {C}ellular {A}utomata {L}everage},
  booktitle = {Reversibility and {U}niversality, Essays Presented to
                  {K}enichi {M}orita on the Occasion of his 70th
                  Birthday},
  pages = {61--93},
  editor = {Andrew Adamatzky},
  series = {Emergence, Complexity and Computation},
  number = {30},
  publisher = {Springer},
  year = {2018},
  isbn = {978-3-319-73215-2},
  language = {english}
}
```