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.

Intrinsic Universality of a 1-Dimensional Reversible Cellular Automaton*

Jérôme Durand-Lose**

LaBRI, Université Bordeaux I,
351, cours de la Libération,
33 405 Talence Cedex, France.

Abstract. This paper deals with simulation and reversibility in the context of Cellular Automata (CA). We recall the definitions of CA and of the Block (BCA) and Partitioned (PCA) sub-classes. We note that PCA simulate CA. A simulation of reversible CA (R-CA) with reversible PCA is built contradicting the intuition of known undecidability results. We build a 1-R-CA which is intrinsically universal, i.e., able to simulate any 1-R-CA. These results extend to any dimension.

Keywords Block Cellular Automata; Cellular Automata; Partitioned Cellular Automata; Reversibility; Reversible Cellular Automata; Intrinsic Universality

1 Introduction

Cellular Automata (CA) model parallel computing as well as physical phenomena. They operate over regular infinite discrete lattices of finite dimension ($\mathbb{Z}^d$). Points are called *cells* and take a value from a finite set of *state* (Q). A CA iteration is the replacement of every state according to the states of the neighboring cells and a unique local function. This replacement is a local, uniform, parallel and synchronous update.

Reversibility is used for backtracking a phenomenon to its origin as well as for preserving information and energy. The posibilities of reversible CA (R-CA) have been investigated from the 60s: the equivalence between bijectivity and injectivity by Moore [1962], Myhill [1963]; in the 70s: the equivalence of reversibility and bijectivity by Hedlund [1969] and Richardson [1972] and the decidability of reversibility in dimension 1 by Amoroso and Patt [1972]; to its undecidability in higher dimension by Kari [1990, 1994].

The computing power of R-CA as well as their simulation powers was particularly investigated in Toffoli and Margolus [1990]. Bennett [1973] proved that reversible Turing machines can simulate any Turing Machine. In 1977, Toffoli

[*] 2017 version, minor changes.

[**] (2017) `jerome.durand-lose@univ-orleans.fr`,

[1977] proved that R-CA of dimension d +1 are able to simulate any d-CA and thus are computationally universal in dimension 2 and higher. To built universal R-CA, (in dimension 2) Partitioned CA and (in dimension 1) Block CA were independently defined as special CA for which reversibility was decidable.

As defined by Morita [1992, 1995], the states of *Partitioned CA* (PCA) are partitioned according to the neighborhood and only the corresponding pieces of states are available to any cell. Sub-states are gathered to form one state to be updated. The local function operates over the finite set of states Q rather that over two sets of different cardinality. Thus it can be bijective and this directly defines the reversibility.

? and Toffoli and Margolus [1987] define *Block CA* (BCA) to built the Billiard Ball Model (BBM), a computation universal 2-R-CA. For BCA, the underlying lattice—not the states—is partitioned into identical blocks regularly displayed. A transition is the replacement of all the blocks of a partition by their images by a unique local transition function. Again, since the local function operates over one finite set, it can be bijective. Originally, BCA were named: "Partitioning CA". Morita defined independently Partitioned CA. To avoid confusion in this article, we refer to Partitioning CA as *Block CA (BCA)* following Kari [1996] that names Block Permutations bijective transitions.

A configuration is finite if all but a finite number of cells are in a define state. In 95, Morita [1995] proved with PCA that any CA can be simulated by R-CA over finite configurations. 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. Moreover, finite configurations are a strict subset of recursive configurations (recursive mapping from $\mathbb{Z}^d$ to Q) which are far from covering all configurations.

Trivially, BCA and PCA are CA. Although BCA and PCA are sub-classes of CA, they are able to simulate any CA. It was proved in Durand-Lose [1995, 1996] that R-BCA can simulate R-CA (in any dimension with extra states). This was a 1990 conjecture by Toffoli and Margolus [1990] which was independently proved (for dimension 1 and 2 only) in Kari [1996]. One of the results in the present paper is that R-PCA can also simulate R-CA.

The main result of this paper is the existence of an intrinsic universal R-CA: i.e. an R-CA capable of simulating all R-CA of the same dimension. This is one step ahead of the results in Durand-Lose [1995], where BBM is proven to be intrinsically universal for 2-R-CA, i.e., able to simulate any 2-R-CA. The results and methods developed in Durand-Lose [1995] extend to higher dimensions, but do not hold for dimension 1. Our extension for dimension 1 is done differently, using PCA and an explicit R-PCA code which was not the case before. The intrinsic universality of U is proven from the existence of simulations between R-CA and R-PCA and then on R-PCA. It extends to higher dimensions.

In the intrinsically universal 1-R-PCA is constructed in layers: sub-states are organized in 10 layers for identification, delimitation, table, signals, value, 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 is a signal like approach to computation with a posteriori tests to ensure reversibility.

In Sect. 2, we define the three CA models and show the decidability of reversibility for PCA and BCA. In Sect. 3, we build simulations of CA with PCA, BCA with PCA and R-CA with R-PCA. In Sect. 4, we built the intrinsically universal 1-R-CA.

2 Definitions

Cellular automata (CA) define mappings over d-dimensional infinite arrays over a finite *set of states* Q. The set of configurations is $c = Q^{\mathbb{Z}^d}$.

Let $[\![a,b]\!]$ denotes the integers from a to b included. Let $+$, mod, div and . denote respectively the coordinate-wise ordering, addition, modulo, Euclidean division and multiplication over $\mathbb{Z}^d$ i.e.: $\forall x,y \in \mathbb{Z}^d$, $\forall k \in [\![1,d]\!]$, $(x+y)_k = x_k+y_k$, $(x \bmod y)_k = x_k \bmod y_k$, $(x \operatorname{div} y)_k = x_k \operatorname{div} y_k$ and $(x.y)_k = x_k\, y_k$. For any configuration c and subset E of $\mathbb{Z}^d$, $c_{|E}$ is the restriction of c to E. For any $x \in \mathbb{Z}^d$, σ_x is the shift by x ($\forall c \in c, \forall i \in \mathbb{Z}^d, \left(\sigma_x(c)\right)_i = c_{i+x}$). Periodicity is to be understood as on every direction.

A **Cellular Automaton** of dimension d (d-CA) is defined by $(Q, \mathcal{N}, f)$. The *neighborhood* $\mathcal{N}$ is a finite subset of $\mathbb{Z}^d$. The *local function* $f : Q^{\mathcal{N}} \to Q$ maps the states of a neighborhood into one state. A configuration is a mapping from $\mathbb{Z}^d$ onto Q. The sets of all configurations is denoted $\mathcal{C}$ $(= Q^{\mathbb{Z}^d})$. The *global function* $\mathcal{G} : \mathcal{C} \to \mathcal{C}$ maps configurations into themselves as follows:

$$\forall c \in \mathcal{C},\ \forall x \in \mathbb{Z}^d,\ \mathcal{G}(c)_x \ = \ f\left((c_{x+\nu})_{\nu \in \mathcal{N}}\right) \ .$$

The new state of a cell depends only on the neighbor states as depicted by Fig. 1(a).

A **Partitioned Cellular Automaton** of dimension d (d-PCA) is defined by: $(Q, \mathcal{N}, \Phi)$. The set of states is a sets product indexed by the neighborhood: $Q = \prod_{\nu \in \mathcal{N}} Q^{(\nu)}$. The ν component of a state q is noted $q^{(\nu)}$. The local function f is defined with function $\Phi : Q \to Q$ as follows:

$$\forall c \in \mathcal{C}, \forall x \in \mathbb{Z}^d,\ \mathcal{G}(c)_x \ = \Phi\left(\prod_{\nu \in \mathcal{N}} c_{x+\nu}^{(\nu)}\right) \ .$$

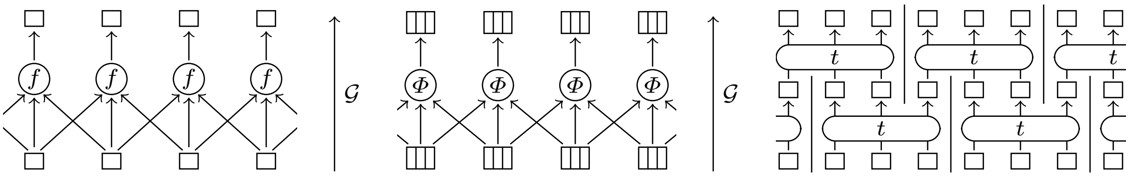
The local function works only with what remains and what is received. Only partial information is accessible to a cell, even about its own state as depicted by Fig. 1(b).

A **Block Cellular Automaton** of dimension d (d-BCA) is defined by: $(Q, v, n, (o^{(j)})_{1 \leq j \leq n}, t)$. The *size* v is an element of $\mathbb{Z}^d$ such that $\forall k \in [\![1,d]\!]$, $0 < v_k$. All $o^{(j)}$ are coordinates modulo v: $o^{(j)} \in \mathbb{Z}^d$ and $\forall k \in [\![1,d]\!], 0 \leq o_k^{(j)} < v_k$. Block V is the subset $[\![0, v_1-1]\!] \times [\![0, v_2-1]\!] \times \cdots \times [\![0, v_d-1]\!]$ of $\mathbb{Z}^d$. The *local transition* t is a function over Q^V.

The *block transition* T is the following mapping over $\mathcal{C}$: for any $c \in \mathcal{C}$ and $i \in \mathbb{Z}^d$, let $a = i \operatorname{div} v$, and $b = i \bmod v$, so that $i = a.v + b$, then $T(c)_i = t(c_{|a.v+V})_b$. In other words, the block containing i in the regular partition with blocks of size v originated from 0 is updated according to t. The same happens for all the blocks of this partition. The block transition of origin o, T_o is $\sigma_o \circ T \circ \sigma_{-o}$. It is the original one with the partition shifted by o. The global function is the composition of the transitions of origins $o^{(j)}$: $\mathcal{G} = T_{o^{(n)}} \circ T_{o^{(n-1)}} \circ \cdots \circ T_{o^{(1)}}$. This is illustrated on the right part of Fig. 1 with 2 partitions and $v = (3)$. Since partitions come in a cycle, we assimilate $n{+}1$ with 1, and 0 with n from now on.

A new state of a cell depends only on the neighbor states accessed by blocks as depicted by Fig. 1(c).

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 Richardson [1972], it is a CA. A constructive proof can be found in Durand-Lose [1995].



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

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

2.1 Reversibility

A CA (resp. PCA, BCA) is *reversible* if and only if its global function $\mathcal{G}$ is bijective and $\mathcal{G}^{-1}$ is the global function of some CA (PCA, BCA). Let R-CA (R-BCA, R-PCA) denote the class of reversible CA (PCA, BCA). The main decidability result is:

Theorem 1. *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 PCA and BCA, the following lemmas hold in any dimension.

Lemma 1 (Morita). *A PCA is reversible iff its local function Φ is a permutation (which is decidable).*

Proof If Φ is a permutation, then the inverse PCA is $\left(\prod_{\nu \in -\mathcal{N}} Q^{(-\nu)}, -\mathcal{N}, \Phi^{-1}\right)$ where $-\mathcal{N} = \{-\nu | \nu \in \mathcal{N}\}$, Φ is undone and pieces of states send back. 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. *q.e.d.*

Lemma 2 (Margolus). *A BCA is reversible iff its local transition t is a permutation (which is decidable).*

Proof If the local transition t is a permutation, by construction, any 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. *q.e.d.*

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 local transition must be a permutation.

3 Simulations

An automaton $\mathcal{A}$ *simulates* another B in linear time τ if there exist two functions α and β such that $\forall c \in \mathcal{C}_{\mathsf{B}},\ \forall t \in \mathbb{N},\ \mathcal{G}_{\mathsf{B}}^{t}(c){=}\beta \circ \mathcal{G}_{\mathcal{A}}^{\tau t} \circ \alpha\ (c)$. If $\tau{=}1$, the simulation is *real time.*

Since PCA and BCA (resp. R-PCA and R-BCA) are CA (resp. R-CA), they can obviously be simulated in real time by CA (resp. R-CA). In Durand-Lose [1995], it is proved that CA (R-CA) can be simulated in real time by BCA (R-BCA). We built the missing simulations.

Proposition 1. *Any d-CA can be simulated by a d-PCA in real time.*

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

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

Any configuration c of $\mathcal{C}_{\mathcal{A}}$ is naturally mapped in d of $\mathcal{C}_{\mathsf{P}}$ with $\forall x \in \mathbb{Z}^d, \forall \nu \in \mathcal{N}, d_x^{(\nu)}{=}c_x$. Information is duplicated, resources are wasted. *q.e.d.*

Since Φ only maps onto the diagonal of $Q^{\mathcal{N}}$, the simulating PCA is never reversible. Nevertheless, it is possible to simulate R-CA with R-PCA with the following result:

Lemma 3. *Any d-R-BCA can be simulated by a d*-R-PCA *in linear time n.*

Proof The idea is to let one cell represent one block. This is a change of scale. Let $\mathsf{B}{=}(Q_{\mathsf{B}}, v, n, (o^{(j)})_{1\leq j\leq n}, t)$ be any d-BCA. Let $\mathsf{P} = \left(\prod_{\nu\in\mathcal{N}} Q_{\mathsf{P}}^{(\nu)}, \mathcal{N}, \Phi\right)$ where $\mathcal{N}{=}\{-1,0,1\}^d$, i.e., coordinates which differ by at most one in any direction. The block of coordinates x (at block scale) of the j^{th} partition is b_x^j. The block b_0^j holds the cell of coordinates $\mathbf{0}$. The sets of states are defined by:

$$\forall \nu \in \mathcal{N},\ Q_{\mathsf{P}}^{(\nu)} = \bigcup_{1\leq j\leq n} \left(\{j\} \times Q_{\mathsf{B}}^{\left(b_0^{j-1} \cap b_{-\nu}^{j}\right)}\right) \ .$$

It is the intersection of the block that holds the cell of coordinates 0 for a partitions and of the one of the next partition holding the cell 0 translated by $\nu.v$. Any intersection may be empty. Blocks are partitioned in function of the next partition so that every part is sent to the corresponding cells 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 local function is defined by:

$$\forall \mu \in \mathcal{N},\ \forall c \in \mathcal{C},\ \varPhi \left(\prod_{\nu \in \mathcal{N}} x_{x+\nu}^{(\nu)} \right)^{(\mu)} = \varPhi \left(k, b_0^k \right)^{(\mu)} = \left(k+1, t\left(b_0^k \right)_{|b_{-\mu}^{k+1}} \right) \ .$$

The first coordinates identify the current transition. They must match, otherwise $\varPhi$ is not yet defined. 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 P, each cell gets one entire block of the first partition and make the first transition. Then every pieces are sent to the corresponding cells with 2 in the first component. Each iteration of P makes a successive transition of B. After n iterations of P, one iteration of B is made and the first component is 1 again.

This construction preserves reversibility: the partial definition of the PCA local function $\varPhi$ is one-to-one if the local transition t of the BCA is reversible. *q.e.d.*

It is proven in Durand-Lose [1995] that any R-CA can be simulated by a R-BCA in real time. From above Lemma and the transitivity of simulation comes:

Theorem 2. *Any d-R-CA can be simulated by a* d-R-PCA *in linear time.*

With the construction in Durand-Lose [1995], 2^{d+1}-1 partitions are needed and the number of B-states is quadratic. In dimension 2 and above, the size v is not bounded by any computable function (from the decidability results 1 and 2). Therefore, neither is the number of P-states.

4 Intrinsic Universality of 1-R-PCA

We built the following 1-R-PCA: U=$(1, Q_{\mathsf{U}}, \{-1,0,1\}, \varPhi_{\mathrm{U}})$. The states and the local function $\varPhi_{\mathrm{U}}$ are defined on Figs. 4 and 10. To avoid using $\{-1,0,1\}$, we denote l, c and r the left, center and right part. We prove that:

Theorem 3. U *is intrinsically universal, i.e., able to simulate any* 1-R-PCA*.*

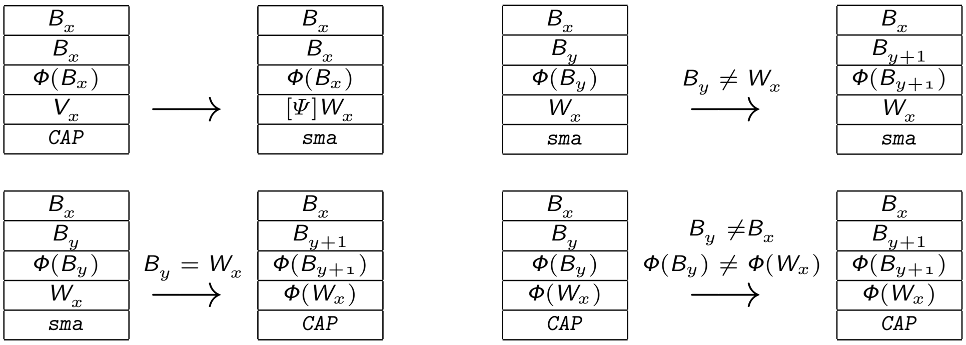
For any 1-R-PCA P=$(Q, \mathcal{N}, \varPhi)$. With classical techniques of cells grouping, P can be simulated in real time with a 1-R-PCA with neighborhood $\{l,c,r\}$. From now on, we suppose that $\mathcal{N} = \{l,c,r\}$.

Macroscopic Level. Let B_x be the x^{th} element of Q modulo $|Q|$. We encode a configuration and the table of $\varPhi$ in a configuration of U with the P-cell architecture of Figs. 2 and 6. The initial configuration given on Fig. 2 extends infinitely on both sides.

Index		B_{x-2}	B_{x-1}	B_x	B_{x+1}	B_{x+2}	
Table		B_{x-2}	B_{x-1}	B_x	B_{x+1}	B_{x+2}	
	...	$\Phi(B_{x-2})$	$\Phi(B_{x-1})$	$\Phi(B_x)$	$\Phi(B_{x+1})$	$\Phi(B_{x+2})$	...
Value		V_{x-2}	V_{x-1}	V_x	V_{x+1}	V_{x+2}	
Mode		CAP	CAP	CAP	CAP	CAP	

Fig. 2. Initial configuration.

From PCA definition, all P-cells first exchange their l and r parts. The state is denoted V_x before the exchange and W_x after. The inner loop of the simulation is: shift the layers holding B_y and $\Phi(B_y)$ to the left, compare W_x to B_y and if they are equal, replace W_x by $\Phi(B_y)$. The table is fully scanned when the index B_x is equal to the B_y of the table, and then P-cells are updated and one P-iteration is done. The dynamics of the loop is given in Fig. 3.



$[\Psi]$ means that the l and r parts are exchanged with adjacent cells.

Fig. 3. Update of P-cells.

States, Layers and Initial Configuration at Microscopic Level. U-cells are organized in 10 layers as detailed in Fig. 4. Layer I holds an index to store where the reading of the table started. Architecture layer A holds delimiters for P-cells ([and]) and for l, c and r parts ($). Layers B and F hold one entry of the table B_y and its image $\Phi(B_y)$. Signals are found on layer S. The value of the P-cell (V_x or W_x) is stocked on layer V. Layers L1 to L4 work like conveyorbelts to transfer data. The values in layers I and A never change.

Layer	Name	States l c r	Use
1	I	0 1	P-cell identification B_x
2	A	_ [$]	Architecture: limits of cells and parts
3	B	0 1	Table entry B_y
4	F	0 1	Image of the table entry B_y, $\Phi(B_y)$
5	S	Σ Σ Σ	Control signals detailed on Fig. 5
6	V	0 1	Value of the P-cell (V_x or W_x)
7–10	L1–L4	0 1 0 1	Shift the table of Φ and exchange values (W_x^l & W_x^r)

Fig. 4. The 10 layers and corresponding sub-states.

We use capital (B, $\Phi(B)$, W) to address the macroscopic level (P-cells) and small letters (i, b, f, v) for microscopic level (U-cells). All P-cells are binary encoded. For the exchange, the codes of l and r parts must have the same length (0's are added if necessary).

Signals are 26 symbols of `this police` as described on Fig. 5. Small (`sma`) and capital (`CAP`) letters behave similarly. The `sma` or `CAP` mode distinguishes between before and after the replacement. During the simulation, signals are turn from `CAP` to `sma` when parts are exchanged and from `sma` to `CAP` when the value is replaced by its image. The number of states of U is $2^{13}.27^3 < 2^{28}$.

`sma`	`CAP`	Use
`a`-`h`	`A`-`H`	Loop which tests if [$W_x = B_y$ — $W_x = \Phi(B_y)$]
`k`	`K`	Write [$\Phi(B_y)$ — B_y] over W_x
`m`, `n`	`M`, `N`	Shift of the table: B_y and $\Phi(B_y)$
`s`, `t`	`S`, `T`	Exchange of Parts W_x^l and W_x^r

Fig. 5. Signals use.

The encoding of P-cells is given on Fig. 6. It takes care of the particular positions of the l and r parts from the beginning. Since V_x^l is exchanged with V_{x+1}^r (V_x^r with V_{x-1}^l), we want B_x^r (B_x^l) above it. When $\Phi(W_x)$ replaces W_x, we want the l and r parts to be directly on the corresponding sides.

I	B_x^l	B_x^c	B_x^r
A	[$		$]
B	B_x^l	B_x^c	B_x^r
F	$\Phi(B_x)^r$	$\Phi(B_x)^c$	$\Phi(B_x)^l$
S			[—`S` —`T`]
V	V_x^r	V_x^c	V_x^l
L1-L4	4 layers for displacements		

I	B_x^l	B_x^c	B_x^r
A	[$		$]
B	B_x^l	B_x^c	B_x^r
F	$\Phi(B_x)^r$	$\Phi(B_x)^c$	$\Phi(B_x)^l$
S	`a`		
V	V_{x-1}^l	V_x^c	V_{x+1}^r
L1-L4	4 layers for displacements		

$\Longleftrightarrow$

B_x
B_x
$\Phi(B_x)$
W_x
`sma`

Fig. 6. Encoding of P-cell at coordinate x, before and after the exchange.

The Microscopic Algorithm is defined by space-time diagrams driven by signals. The corresponding rules are indicated on Fig. 10. The algorithm starts in `CAP` mode with [|`S` |`T`] signals in the rightmost cell. Signals in the different P-cells are always exactly synchronized.

First, the l and r parts of the P-cell are exchanged and the mode is switched to `sma` as depicted on Fig. 7. The initial value of the P-Cell is V_x. The bits of V_x^l and V_{x+1}^r are swapped on the layer L1 by signals `S` and `T` on they way from]. On crossing], the flows are transferred on L2 (for technical reasons explained below). Signals `S` and `T` turn back at $ and on their way back, they retrieve the bits from L2 and put them back on their destination slot. Synchronization is very important. Signals `S` and `T` finally get back together as `h` at], switch mode and go back to the left end of the P-cell. This is implemented with 11 rules of Fig. 10.

Let us describe in details the inner loop. Value W_x and table entry B_y are in place, bit below bit, to be compared. Signal `a` crosses the whole P-cell to compare them. If they differ, a marker `b` is put on the first different bit, and `a` turns to `b`. On the way back, `b` marks `d` the last bit which differs and gets the marker `b` back as illustrated on the first column of Fig. 8. If W_x and B_y are equal, the signal reaches] as `a`, turns to `k`, writes $\Phi(B_y)$ over W_x on the

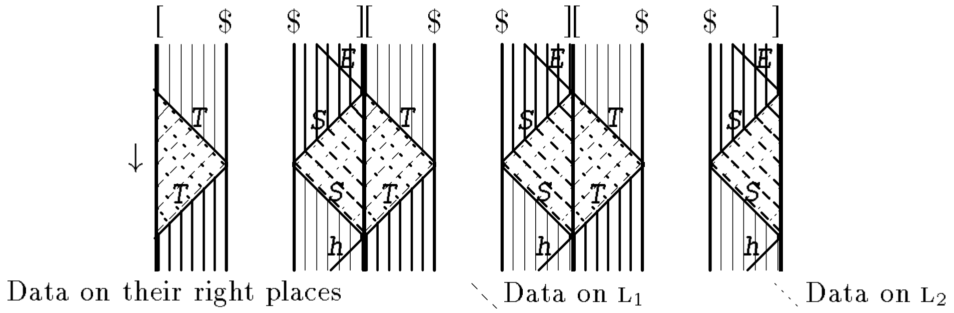


Fig. 7. Exchanging V_x^l and V_{x+1}^r.

way back and switch mode. This special behavior takes as much time as the regular one, keeping the synchronization. Equality is tested on the way back for reversibility: going backward in time, U must make the correct change at the adequate time, so it needs this test and it must change back $\Phi(W_x)$ into W_x.

To know that the table was completely scanned, signal must test whether B_x and B_y are equal. On the second 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, as illustrated by Fig. 8. If they differ, g comes back and gets the marker. If there are equal, before returning e exchanges the l and r parts and switches mode and return as h.

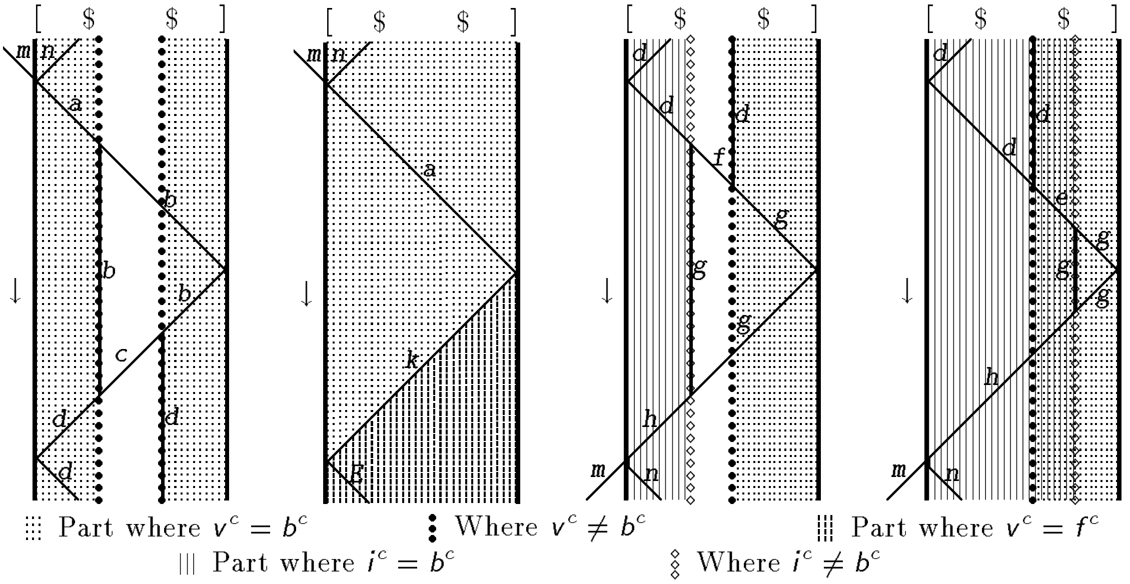


Fig. 8. Inner loop: test for replacement and for the end of P-iteration.

On returning to [, h splits into m and n (right half of Fig. 8). These signals manage the shift of the table by one P-cell rightward using layers L$_1$ to L$_4$ as illustrated by Fig. 9. Signal m sets B_{y-1} and $\Phi(B_{y-1})$ 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 P-cell. On its way back, n sets B_{y-1} and $\Phi(B_{y-1})$ on their final places by swapping them from layers L$_2$ and L$_4$. Signals m and n gather and form a which starts the loop again. This corresponds to the last 12 rules of Fig. 10.

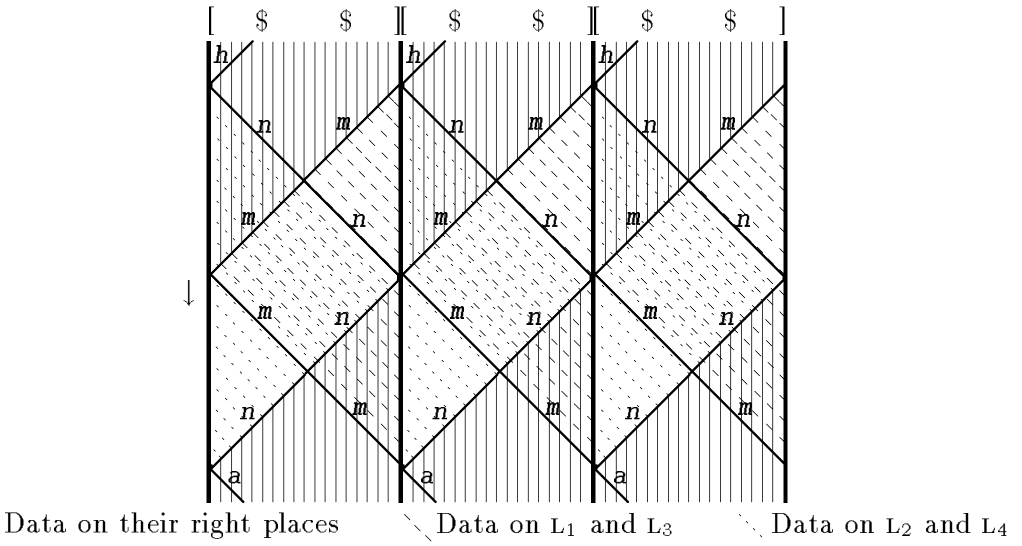


Fig. 9. Shifting the table.

If the mode is *CAP*, it is exactly the same except that the equality tested is W_x with $\Phi(B_y)$ instead of B_y, and B_y is written over W_x instead of $\Phi(B_y)$. Since P is reversible, each value of $\Phi(B_y)$ appears once and only once in the table. After being copied $\Phi(B_y)$, it is never met again in the table as an image.

Local Function of U. The necessary definitions of Φ_{U} are given in Fig. 10. Since they are one-to-one, Φ_{U} can be completed bijectively. 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 swaps or of v^c, but in such case, the previous value is held somewhere else as indicated by a condition. For *CAP* signals, the differences are only for the lines with an '$*$': the test made is $v^c = f^c$, instead of $v^c = b^c$, and b^c, instead of f^c, is copied over v^c.

Rule (*i*) of Fig. 10 starts the replacement process: writing $\Phi(W_x)$ over W_x (exchanging pieces of states with neighbor cells). Rules (*ii*) and (*iii*) start the shifting of the table. Rules (*iv*) and (*v*) start the exchange of the l and r parts. Rules (*vi*) and (*vii*) switch the mode. The first corresponds to the end of the P-iteration, the second to the replacement of W_x by $\Phi(W_x)$. Rule (*viii*) is the end of the inner loop.

All rules are combined with the following: for the last 4 layers L_1 to L_4, the l and r parts are swapped so that l (r) 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 of Fig. 9 where 2 flows are traveling together.

Simulation Time. Let a be the width of a P-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 P-state, i.e., $|Q|$ times. To make a P-iteration, values are exchanged between neighboring cells, this needs $2b+1$ iterations. All together, the simulation is in

		Value					Image				
	Structure	Signals			Condition		Signals			Modification	
*	[		a		$v^c = b^c$				a		
					$v^c \neq b^c$			b	b		
*	‘ ’,$	a			$v^c = b^c$				a		
					$v^c \neq b^c$			b	b		
*	]	a			$v^c = b^c$			k		$v^c \leftarrow f^c$	(i)
					$v^c \neq b^c$		d	d			
	‘ ’,$	b							b		
	‘ ’,$		b					b			
*	]	b			$v^c = b^c$		b				
					$v^c \neq b^c$		c	d			
*	‘ ’,$			b	$v^c = b^c$		b				
					$v^c \neq b^c$		c	d			
*	‘ ’,$		b	b	$v^c \neq b^c$		d	d			
*	[		b	b	$v^c \neq b^c$	$i^c = b^c$		e			
						$i^c \neq b^c$		g	g		
*	‘ ’,$		b	c	$v^c \neq b^c$		d				
*	[		b	c	$v^c \neq b^c$	$i^c = b^c$			d		
						$i^c \neq b^c$		g	f		
	‘ ’,$			c			c				
*	‘ ’,$			d	$v^c = b^c$		d				
*	[			d	$v^c = b^c$	$i^c = b^c$			d		
						$i^c \neq b^c$		g	f		
*	‘ ’,$	d			$v^c = b^c$	$i^c = b^c$			d		
						$i^c \neq b^c$		g	f		
*	‘ ’,$	d	d		$v^c \neq b^c$	$i^c = b^c$			e		
						$i^c \neq b^c$		g	g		
*	]	d	d		$v^c \neq b^c$	$i^c = b^c$		s	t		(ii)
						$i^c \neq b^c$	h				
*	‘ ’,$	e			$v^c = b^c$	$i^c = b^c$			e		
						$i^c \neq b^c$		g	g		
*	]	e			$v^c = b^c$	$i^c = b^c$		s	t		(iii)
						$i^c \neq b^c$	h				
	‘ ’,$	f							f		
*	‘ ’,$	f	d		$v^c \neq b^c$				g		
*	]	f	d		$v^c \neq b^c$		g				
*	‘ ’,$	g			$v^c = b^c$				g		
*	]	g			$v^c = b^c$		g				
	[, ‘ ’,$		g					g			
	‘ ’,$			g			g				
	‘ ’,$		g	g		$i^c \neq b^c$	h				
	[		g	g		$i^c \neq b^c$	m	n			(iv)
	‘ ’,$			h		$i^c = b^c$	h				
	[			h		$i^c = b^c$	m	n			(v)
	]		s				s			swap(v^c, l^r_1)	
	‘ ’			s			s			swap(v^c, l^r_1)	
	$			s				s		swap(v^c, l^r_1)	
	$		s						s	swap(v^c, l^r_2)	
	‘ ’	s							s	swap(v^c, l^r_2)	
	]	s						s		swap(v^c, l^r_2)	
	[, ‘ ’	t							t	swap(v^c, l^l_1)	
	$	t						t		swap(v^c, l^l_1)	
	$		t				t			swap(v^c, l^l_2)	
	[, ‘ ’			t			t			swap(v^c, l^l_2)	
	]		s	t			H				(vi)
*	‘ ’,$			k	$v^c = b^c$		o			$v^c \leftarrow f^c$	
*	[			k	$v^c = b^c$				E	$v^c \leftarrow f^c$	(vii)
	], ‘ ’			m			m			swap(b^c, l^r_1), swap(f^c, l^r_3)	
	‘ ’,$			m			m			swap(b^c, l^r_1), swap(f^c, l^r_3)	
	‘ ’,$	n		m			m		n	swap(b^c, l^r_1), swap(f^c, l^r_3)	
	[			m					m		
	‘ ’,$]	m							m		
	[		n						n		
	‘ ’,$	n							n		
	]	n						n			
	]		n				n			swap(b^c, l^l_2), swap(f^c, l^l_4)	
	‘ ’,$			n			n			swap(b^c, l^l_2), swap(f^c, l^l_4)	
	[			n				n		swap(b^c, l^l_2), swap(f^c, l^l_4)	
	[	m	n						a		$(viii)$

Fig. 10. Table of Φ_{U}.

linear time and bounded by $12\,|Q|\log(|Q|) + o(|Q|\log(|Q|))$. This is fine for R-PCA. For R-BCA simulation, it have to be multiplied by $2^{d+1}-1$, here 3 and

Q is larger. For R-CA, $|Q|$ of the simulating R-BCA can be very big. With the construction of Durand-Lose [1995], it is $(|Q|^2 + |Q|)^{(4 \max(r_{\mathcal{A}}, r_{\mathcal{A}^{-1}}))^d}$ where $r_{\mathcal{A}}$ $(r_{\mathcal{A}^{-1}})$ is the radius of $\mathcal{A}$ $(\mathcal{A}^{-1})$, i.e., the maximum absolute value of any component of any ν in the neighborhood. Nevertheless, the simulation is still in linear time.

5 Conclusion

The inverse R-PCA is simulated by changing [$|\mathsf{S}$ $|\mathsf{T}$] by [$|\mathsf{s}$ $|\mathsf{t}$] in the initial configuration. When U runs with the code of an unreversible PCA, the Φ is not one-to-one so there are two states A and B such that $\mathsf{A} \neq \mathsf{B}$ and $\Phi(\mathsf{A}) = \Phi(\mathsf{B})$. When A is encountered before B then A will be replaced by $\Phi(\mathsf{B})$ and in CAP it is then replaced by B and remains so until the end of the P-iteration. This is wrong.

The construction can be extended to greater dimension. The table and test are done in the first direction and sub-states exchanged in the other directions must be added.

Basic programming schemes can be embedded in R-PCA when conceived reversible. We have implemented with reversible local rules a global dynamic of move, test and replace which needs backward tests. The R-PCA is programmed: we make loops, tests and branches to subroutines.

The power of computations of d-CA, d-BCA and d-PCA over infinite configurations are the same. We have proved that the d-R-CA, d-R-BCA and d-R-PCA classes are also equivalent which is an important result since reversibility is decidable for BCA and PCA while it is not for CA.

In Durand-Lose [1995], it was proved the existence of intrinsically universal R-CA of dimension 2 and above. The construction was made using the R-CA simulation by R-BCA and then by constructing a simulation of any R-BCA with the Billiard Ball Model of Margolus **?**Toffoli and Margolus [1987]. We have proved using R-PCA and their 'source code' (i.e. local function) that this result still holds in dimension 1. Since any R-CA can be simulated by a R-PCA (Th. 2) and U is able to simulate any R-PCA (Th. 3) and U is a R-CA:

Theorem 4. *There exist* 1*-R-CA able to simulate any* 1*-R-CA in linear time.*

There exist simulations of any Turing machines with R-CA Morita [1992] so that all partial recursive functions can be computed by R-PCA, so U is computation universal. The existence of a 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 **?**.

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 nonreversible CA, then by transitivity, U is also able to do it, so that if U can not, none can.

Bibliography

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.

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 These/index.html. In French.

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.

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

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

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

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.

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

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.

bibtex entry

```
@inproceedings{durand-lose97stacs,
  author = {Durand-{L}ose, J{\'e}r{\^o}me},
  booktitle = {STACS 1997},
  number = {1200},
  pages = {439--450},
  publisher = {Springer},
  series = {LNCS},
  title = {Intrinsic {U}niversality of a $1$-{D}imensional
                {R}eversible {C}ellular {A}utomaton},
  year = {1997},
  doi = {10.1007/BFb0023479},
  language = {english}
}
```