Laboratoire **B**ordelais de **R**echerche en **I**nformatique,
URA CNRS 1304,
Université Bordeaux I,
351, cours de la Libération,
33405 TALENCE Cedex,
FRANCE.

Rapport de Recherche

Numéro 1177-97

Reversible space-time simulation of cellular automata

Jérôme O. Durand-Lose [1]

LaBRI, URA CNRS 1304, Université Bordeaux I, 351, cours de la Libération, F-33405 TALENCE Cedex, FRANCE.

We briefly recall the definitions of Cellular automata (CA), simulation, reversibility and Partitioned cellular automata (PCA) as defined by Morita. We call the sequence of the iterated configurations of a configuration a *space-time diagram*. We define an *embedding* relation between space-time diagrams and a *space-time simulation* relation between CA. We built a space-time simulation of any CA by a reversible PCA (R-PCA). Finally, we state our main result: there are reversible CA able to space-time simulate any CA of the same dimension.

Key words: Cellular automata, space-time simulation, intrinsic universality and reversibility.

1 Introduction

Reversibility corresponds to the conservation of information and energy. It allows unambiguous backtracking. In computer science, reversible is studied in order to design computers which would waste less energy. In 1973, Bennett [2] proved that any Turing machine can be simulated by another one which is reversible. Recently, Morita [15] proved that any two-counter automata can by simulated by a reversible one.

Cellular automata (CA) model massively parallel computation and physical phenomena. They work over matrices of infinite size but finite dimension; the underlying lattice is $\mathbb{Z}^d$. The elements of the matrices are called *cells*. Each cell takes a value from a finite set of states $\mathcal{S}$. A

[1] jdurand@labri.u-bordeaux.fr,
http://dept-info.labri.u-bordeaux.fr/~jdurand.

configuration is a valuation of a whole matrix. A CA updates a configuration by synchronously changing the state of each cell according to the states of the cells around it and a local function f. All cells use the same local function. This is a parallel, synchronous, local and uniform process.

A CA is reversible when its global function $\mathcal{G}$ is a bijection and its inverse $\mathcal{G}^{-1}$ is it-self the global function of some CA. Research on reversible CA (R-CA) begun in the 60's: Moore [12] and Myhill [16] proved that if $\mathcal{G}$ is one-to-one then it is a bijection. Hedlund [6] and Richardson [17] proved that any function over $\mathcal{S}^{\mathbb{Z}^d}$ which commutes with any shift and which is continuous for the product topology is the global function of some CA. As a consequence, for any CA it is enough to be one-to-one to be reversible. In 1972, Amoroso and Patt [1] proved that CA reversibility is decidable in dimension 1. In 1990, Kari [9] proved that it is not decidable any more in dimension 2 and above.

In 1977, Toffoli [18] proved that any CA can be simulated by a reversible CA (R-CA) one dimension higher and that there are R-CA of dimension 2 and above which are computation universal. It was only in 1992 that the existence of a computation universal R-CA was proved in dimension 1 by Morita [13]. To do it he introduced Partitioned cellular automata (PCA). For PCA, only some part of the state is send to neighboring cells. Immediately, PCA are CA and reversible PCA (R-PCA) are R-CA. On the other way round, PCA (R-PCA) are able to simulate any CA (R-CA) [4].

In 1995 the author [3] proved that there are R-CA able to simulate any R-CA of the same dimension (greater than 2), over any configuration (finite or not), in linear time. This result has been extended to dimension 1 in 1997 [4] with the use of PCA.

Yet, it is unknown whether it is possible to simulate any CA with a R-CA of the same dimension. In 1995, Morita [14] proved that it is possible over finite configurations, i.e. configurations such that there exists some state q such that there is a finite number of cell which are not in state q. Finite configurations form a strict subset of recursive configurations which is it-self far from being the whole set of configurations. Finiteness is also too restrictive for physicians and mathematicians.

Generally, to *simulate* means that, up to some encoding, the result corresponds to any possible initial configuration of the simulated machine. On iterative systems, for induction purposes, one wants any iteration of the simulated machine to be totally encoded in one iteration of the simulating machine. Generally, it is not formally considered that a simulated iteration might be encoded over various, maybe an infinite number of, simulating iterations.

In the theory of computation, one speaks about the simulation of a machine/automaton/rewriting system/... by another and defines equivalence among programming systems. Intrinsic universality inside a class is defined as the ability to simulate any machines of the class. The action of a machine is generally defined by induction. A machine simulates another if it goes through the same steps of computation, which is more than just yielding the same result.

There are various ways to compute with CA. Input and result of a computation are usually encoded in a portion of the initial and final configurations. Considering the succession of configurations, data can be set and result retrieved in sequential or in parallel (see [11] for a discussion about this). One can define a language to be the words such that a given cell, or any cell, enters a given state.

Some authors consider the space-time matrix as a tool for construction. Heen [7,8] developed a statistical positioning to accelerate computations. Mazoyer [11] uses dynamical generation of lattices to lead computation inside diagrams. Other authors consider the whole space-time diagram (or orbit of a configuration) as the result and not just the final configuration. Functions constructible or Fisher-constructible [5,10] correspond to geometrical properties of space-time diagrams. Constructed objects are not given in the output but constructed through the iterations. The whole diagram hold the result or the proof of correctness.

Based on these observations, we define an *embedding* relation between space-time diagrams. We say that a CA A *space-time simulates* another CA B when any space-time diagram generated by B can be embedded in one generated by A. Within this definition, we prove that there are R-CA which are able to space-time simulate any CA of the same dimension, reversible or not.

We prove this theorem in dimension 1 with a construction of a R-CA which progressively generates any diagram of a given CA on any starting configurations. This construction is based on the the movements of a signal on a limited portion of the configuration. When the signal goes toward the center of the portion, it moves over more and more iterated cells that it updates. When it goes away from the center to a border, it moves over less and less iterated cells without modifying them.

The article is articulated as follows. The definitions of Cellular Automata (CA), Partitioned CA (PCA) and reversibility are gathered in sect. 2. In sect. 3, we recall the usual definition of simulation and explain how to simulate any CA by a CA of neighborhood $\{0,1\}^d$, then space-time diagrams and space-time simulation are defined. In sect. 4, we construct a space-time simulation of any 1-dimensional CA (1-CA) by some 1-R-PCA.

2 Definitions

Configurations are infinite matrices of finite dimension d. Points of configurations are called *cells*. Each cell takes a value from a finite set of *states* $\mathcal{S}$. The set of all configurations is denoted $\mathcal{C}$ ($\mathcal{C} = \mathcal{S}^{\mathbb{Z}^d}$). Cellular automata (CA) and Partitioned cellular automata (PCA) update all the cells of a configuration in a parallel and synchronous way.

2.1 Cellular automata

A d-dimensional *Cellular automaton* (d-CA) is defined by $(\mathcal{S}, \mathcal{N}, f)$. The *neighborhood* $\mathcal{N}$ is a finite subset of $\mathbb{Z}^d$. The *local function* $f : \mathcal{S}^{\mathcal{N}} \to \mathcal{S}$ yields the new state of a cell in function of the states of the cells in its neighborhood. The *global function* $\mathcal{G} : \mathcal{C} \to \mathcal{C}$ updates configurations 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 only depends on the states around it.

2.2 Partitioned cellular automata

According to Morita's definition [13,14], a d-dimensional *Partitioned cellular automaton* (d-PCA) is a CA defined by $(\mathcal{S}, \mathcal{N}, \Phi)$. The set of states is a product of sets indexed by the neighborhood, i.e. $\mathcal{S} = \prod_{\nu \in \mathcal{N}} \mathcal{S}^{(\nu)}$. The ν component of a state s is denoted $s^{(\nu)}$. The function Φ operates over $\mathcal{S}$. The local function f is defined by:

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

Equivalently, each state is the product of the information to be send around. Each component is send to only one cell. The function Φ uses what is left and what is received. The cell only keeps a partial knowledge about its own state and only receives a partial knowledge about the states of the cells in its neighborhood, as depicted in the right part of fig. 1.

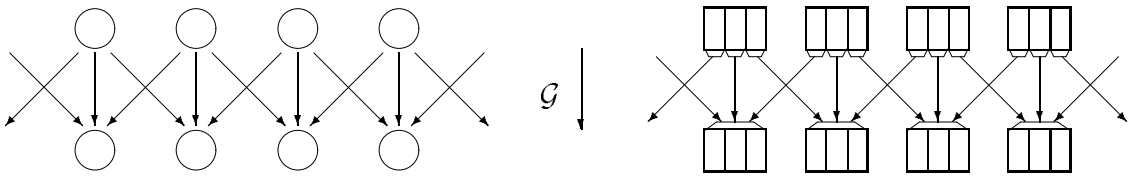


Fig. 1. Updating of CA and PCA.

2.3 Reversibility

A CA (PCA) is *reversible* if its global function $\mathcal{G}$ is a bijection and its inverse $\mathcal{G}^{-1}$ is the global function of some CA (PCA). We denote R-CA (R-PCA) reversible CA (PCA). For PCA the following lemma is true in any dimension.

Lemma 1 (Morita) *A* PCA *is reversible if and only if its function* Φ *is a permutation, which is decidable.*

PROOF. If Φ is a permutation then the inverse PCA is $\left(\prod_{\nu\in-\mathcal{N}} \mathcal{S}^{(-\nu)}, -\mathcal{N}, \Phi^{-1}\right)$ where $-\mathcal{N} = \{-\nu | \nu \in \mathcal{N}\}$. The action of Φ is undone and the different pieces are send back to where they came from.
Otherwise since Φ works over a finite set, it is not one-to-one. It is easy to construct two configurations which are mapped in the same configuration.
Decidability comes from the finiteness of $\mathcal{S}$. $\square$

In 1990, Kari [9] proved that the reversibility of CA of dimension greater or equal to 2 and above is not decidable. As far as reversibility is concerned, CA and PCA fundamentally differ.

3 Simulation

3.1 Iterative approach

Cellular automata iteratively update configurations. We call any configuration generated by a finite number of iteration over a configuration an *iterated image*. For any initial configuration c, one wants to find each iterated image of c by the simulated CA A wholly encoded inside an iterated image of a configuration e by the simulating CA B. The definition of Toffoli [18] is:

Definition 2 *A* CA A simulate (iteratively) *another* B *if there exist three functions* $\psi : \mathcal{C}_\mathrm{B} \times \mathbb{N} \to \mathbb{N}$, $\alpha : \mathcal{C}_\mathrm{B} \to \mathcal{C}_\mathrm{A}$ *and* $\beta : \mathcal{C}_\mathrm{A} \to \mathcal{C}_\mathrm{B}$ *such that:*

$$\forall b \in \mathcal{C}_\mathrm{B},\ \forall t \in \mathbb{N},\quad \mathcal{G}_\mathrm{B}^t(c) = \beta \circ \mathcal{G}_\mathrm{A}^{\psi(c,t)} \circ \alpha\ (b)\ .$$

The functions ψ, α and β must be of a lower complexity than the simulated CA B in order to insure that they are not doing the computation. Generally α and β are projections or injections. When c is fixed, $\psi(c,t)$ may be undefined for many t as long as it is defined for an infinity of t. This is required to allow speed-up: to simulate $3n$ iterations with n iterations, there are $2n$ iterations which cannot be defined.

A simulation is in *linear time* τ if $\psi(c,t) = \tau\, t$ for any configuration c. If $\tau = 1$ the simulation is in *real time*. Since d-PCA are d-CA, they can be simulated in real time by d-CA. Identically, d-R-PCA can be simulated by d-R-CA. In [4], there is a construction of a simulation of d-CA (d-R-CA) by d-PCA (d-R-PCA) in linear time.

The following lemma gives an example of a simulation.

Lemma 3 *Any* d-CA *can be simulated by a* d-CA *with neighborhood* $\{-1,0,1\}^d$ *in real time.*

$\mathcal{N} = \{-2, 0, 1\}$

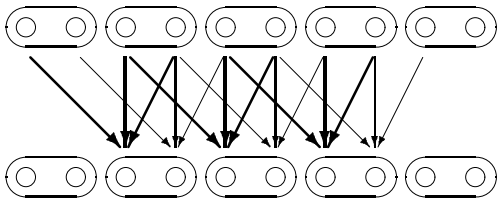


Fig. 2. Grouping cells 2 by 2.

PROOF. The cells are gathered in blocks of adjacent cells. Let $\mathrm{B} = (\mathcal{S}_\mathrm{B}, \mathcal{N}, f_\mathrm{B})$ be any d-CA. Let r be the *radius* of B, i.e. the maximum absolute value of all the components of all the elements in $\mathcal{N}$. Let $\mathrm{A} = (\mathcal{S}_\mathrm{B}^{r^d}, \{-1, 0, 1\}^d, f_\mathrm{A})$. The function α is defined by $(\alpha(c)_x)_i = c_{rx+i}$, then $\beta = \alpha^{-1}$ and f_B is defined by induction from f_A as illustrated in fig. 2 for $\mathcal{N} = \{-2, 0, 1\}$. A simulates B in real time. □

3.2 Space-time approach

Definition 4 *A* space-time diagram $\mathbb{A}$ *is the sequence of the iterated images of a configuration by a* CA.

Put it differently, let $\mathcal{G}$ be the global function of some d-CA A and c a configuration ($c \in \mathcal{S}^{\mathbb{Z}^d}$). The associated space-time diagram $\mathbb{A} : \mathbb{Z}^d \times \mathbb{N} \to \mathcal{S}$ is defined by $\mathbb{A}_{x,t} = (\mathcal{G}^t(c))_x$. It is denoted $(\mathcal{G}, c)$ or (A, c).

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

To recover an embedded B-configuration each cell has to be taken at a given iteration. A A-configuration is thus constructed. This A-configuration is decoded to get an iterated configuration for B. More precisely, we define this as follows:

Definition 5 *A space-time diagram* $\mathbb{B} = (\mathrm{B}, b)$ *is* inserted *in another space-time diagram* $\mathbb{A} = (\mathrm{A}, a)$ *when there exist three functions* $\chi : \mathbb{Z}^d \times \mathbb{N} \to \mathbb{N}$, $\alpha : \mathcal{C}_\mathrm{B} \to \mathcal{C}_\mathrm{A}$ *and* $\beta : \mathcal{C}_\mathrm{A} \to \mathcal{C}_\mathrm{B}$ *such that:*
- $a = \alpha(b)$;
- $\forall (x, t) \in \mathbb{Z}^d \times \mathbb{N}$, *let* c^t *be the configuration of* A *such that* $c^t_x = \mathbb{A}_{x,\chi(x,t)}$;
- $\forall t \in \mathbb{N}$, $\mathcal{G}^t_\mathrm{B}(a) = \beta(c^t)$.

The configuration c is encoded into d according to α. To recover an iterated value of c, the function χ indicates which iteration is to be considered for each cell and β decodes the generated configuration.

As before, the functions χ, α and β must be of a lower complexity than the ones of the diagrams and c^t may be undefined for many t as long as it is defined for infinitely many t.

Definition 6 *A* CA A space-time simulates *a* CA B *when any space-time diagram generated by* B *can be inserted inside a space-time diagram generated by* A *and all insertions use the same functions α and β.*

The function χ may depend on the initial configuration c of the inserted diagram. If it only depends on the time and the initial configuration, it is an iterative simulation. This new simulation relation is an extension of the former.

We refer the reader to the simulation in the next section and the simulation in [7] for examples.

4 Space-time simulation by reversible CA

In this section we give an explicit construction to prove the following lemma:

Lemma 7 *Any d-*CA *with neighborhood $\{-1,0,1\}^d$ can be space-time simulated by a d-*R-PCA.

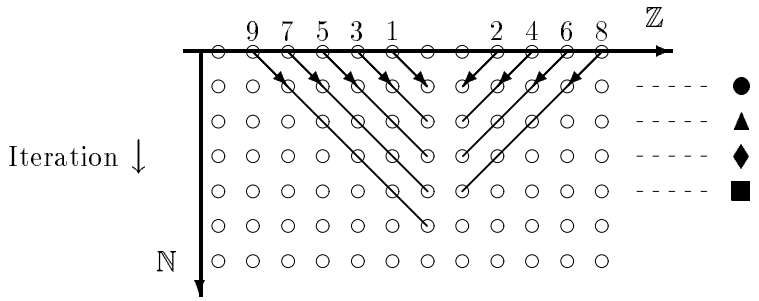
PROOF. The proof is only detailed in dimension 1. We generalize it to any dimension at the end. We explain how the simulation works before going deeper and deeper into details.

4.1 Macro dynamics

Let B=$(\mathcal{S}_B, \{-1,0,1\}, f)$ be any 1-CA with the given neighborhood. We build a 1-R-PCA P = $(\mathcal{S}_P, \{-1,0,1\}, \Phi)$ which space-time simulate B. Let b be any configuration in $\mathcal{C}_B$ and $\mathbb{B}$ the associated space-time diagram. We explain how the simulated diagram $\mathbb{B}$ is inserted into the diagram $\mathbb{P}$ generated by P and an initial configuration p.

The idea is to update only finitely many cells each loop. Thus only a finite part of the configuration is not at iteration 0. Inside it, the closer to the center a cell is the higher its iteration number is. As iteration goes by, this part is increasing on both sides (space) and in iteration numbers (time).

The simulated diagram $\mathbb{B}$ is generated according to diagonal lines, one after the other. The updating lines of $\mathbb{B}$ are depicted in figure 3 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 B-iterations (as on $\mathbb{P}$ in fig. 4).



The symbol in the last column identifies the iteration.
It corresponds to the insertion in fig. 4.

Fig. 3. Order of generation inside the simulated diagram $\mathbb{B}$.

The state of a cell x at iteration t in the inserted diagram $\mathbb{B}$ is denoted ${}_x\backslash^t$ (${}_x\backslash^t = \mathcal{G}_{\mathrm{B}}^t(c)_x$) and the information needed to compute ${}_x\backslash^t$ is denoted $[{}_x\backslash^t]$ ($[{}_x\backslash^t] = ({}_{x-1}\backslash^{t-1}, {}_x\backslash^{t-1}, {}_{x+1}\backslash^{t-1})$). Each time a cell is updated, a $[{}_x\backslash^t]$ is generated to keep the data needed for undoing the update. Recordings of $[{}_x\backslash^t]$ are accumulating. They cannot be disposed off because $\mathcal{G}_{\mathrm{B}}$ is not one-to-one and the previous configurations cannot be guessed from the actual one. These needed but cumbersome data are evacuated on both sides of the configuration.

A finite part of the P-configuration represents B-cells which are not in the initial state, i.e. the configuration c. We call this part the *updating zone*. The construction is driven by a signal moving forth and back on the updating zone. When a signal goes from the left to the right for the n^{th} time as in fig. 4, its dynamics are 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 evacuate and save it while it generates ${}_x\backslash^1$. The next cell holds $[{}_x\backslash^2]$ which is also set on movement to the right while ${}_x\backslash^2$ 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 $\mathbb{B}$ are generated on an hyperbola indicated by circles (triangles, lozenges, squares respectively) on the simulating diagram $\mathbb{P}$ in fig. 4. This corresponds to the layers construction of $\mathbb{B}$ depicted in fig. 3. Figure 4 depicts the evacuation of the $[{}_x\backslash^t]$ away from the updating zone for the first 100 iterations. Evacuated data never interact.

4.2 Micro dynamics

Let us go into the details of how this mechanism works. Cells are organized in three layers: the upper layer holds the state of the simulated cell, the middle one holds signals driving the dynamics and the lower one acts like a conveyor belt to evacuate the $[{}_x\backslash^t]$.

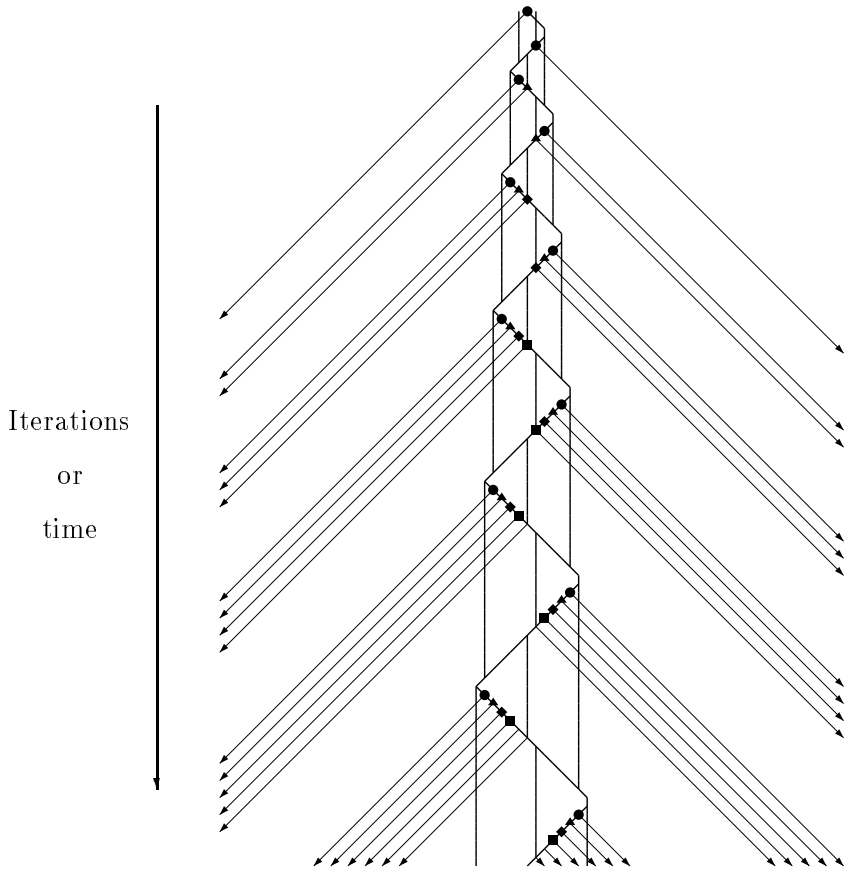


Fig. 4. Scheme of the evacuation of data($[\backslash^t_x]$) on the simulating diagram $\mathbb{P}$.

The first 26 iterations are depicted in fig. 5. In the upper layer, the cells alternatively holds 3 times the same state ($\backslash^t_x$) or the states of the cell and its two closest neighbors at the same iteration ($\backslash^{t-1}_{x-1}, \backslash^{t-1}_x, \backslash^{t-1}_{x+1}$), otherwise some mix over 2 or 3 iterations. A cell can only be updated when it has the information $[\backslash^t_x]$. By induction from fig. 5, the possibility to update a cell only depends on the parity of the sum of simulating and simulated iteration numbers (the full demonstration is easy but very long because of the many cases to consider).

Let us define the signal which leads the dynamics. We call it the *"suit signal"*. It is the traces of the sub-states ♣, ♠, ♥ and ♦ in $\mathbb{P}$. The suit signal only moves forth and back in the updating zone and thus appears as a zigzag on figures 4 and 5. It is delayed by one on the left side to keep synchronism with the presence of $[\backslash^t_x]$.

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 simulated updates of the cells until it reaches ★. Afterwards it is ♠ and just moves to the other ▮.

Each time a simulated update is done, three values, $\backslash^{t-1}_{x-1}$, $\backslash^{t-1}_x$ and $\backslash^{t-1}_{x+1}$, are 'used up' and become useless. They are gathered in $[\backslash^t_x]$ and moved to the lower layer to be evacuated. Three copies of the new state $\backslash^t_x$ are made. They will be used for the next update of the simulated cell and of its two closest neighbors.

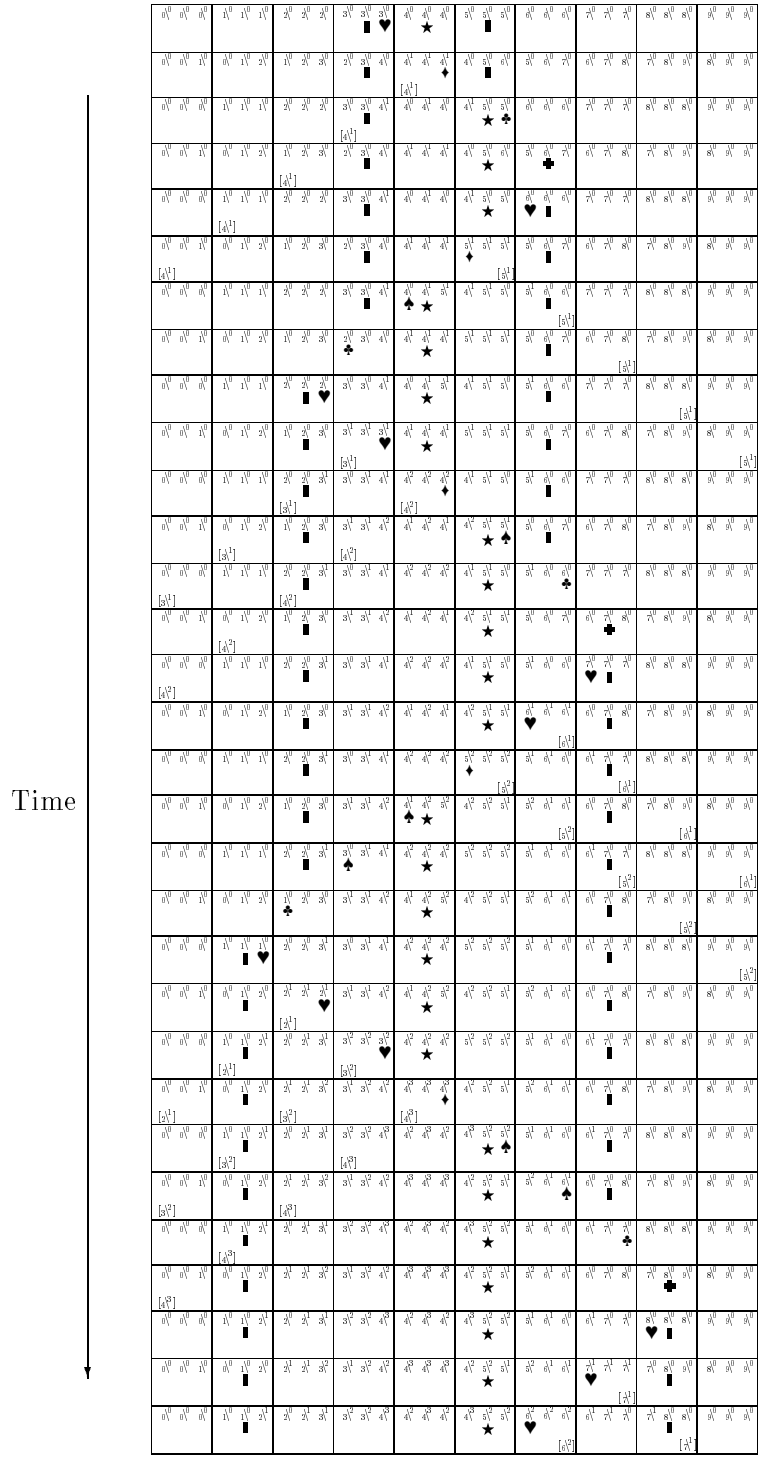


Fig. 5. The first 26 iterations of the space-time simulation.

The endless movement of the suit signal and updates (at the right parities) can be 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. 5.

4.3 Local function

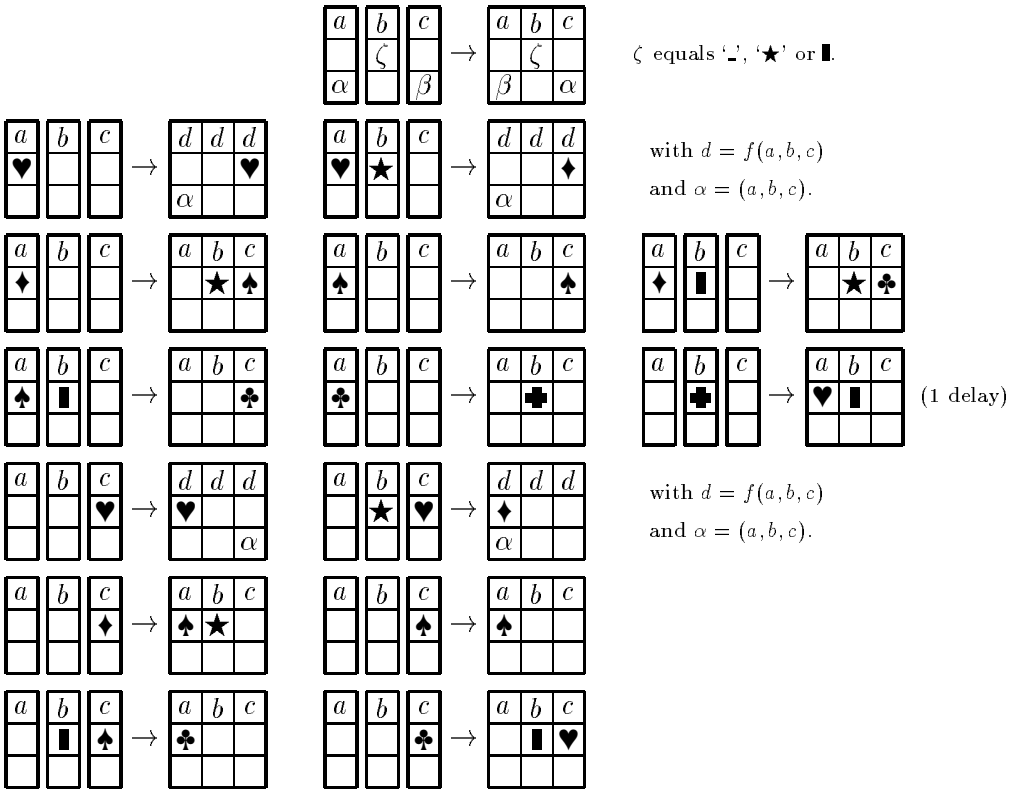
The set of states of P is detailed in fig. 6. If the CA B has m states, P has $100\,m^3\,(m^3+1)^2$ states. This represents a big increasing in the size of the table of the local function and in complexity.

1	0	-1
$\mathcal{S}_{\mathrm{B}}$	$\mathcal{S}_{\mathrm{B}}$	$\mathcal{S}_{\mathrm{B}}$
♣ ♠ ♥ ♦ _	✚ ▮ ★ _	♣ ♠ ♥ ♦ _
$\mathcal{S}_{\mathrm{B}}^3 \cup \{\_\}$	_	$\mathcal{S}_{\mathrm{B}}^3 \cup \{\_\}$

Fig. 6. Set of states of P: $\mathcal{S}_{\mathrm{P}}$.

Cells are depicted as 3×3 matrices as in the first line of fig. 5. The upper layer holds the states of P; a configuration of B is encoded there as depicted in fig. 4. The middle layer holds the suit signal and the lower layer is used to store the data away from the updating zone.

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



a, b, c and d belong to $\mathcal{S}_{\mathrm{B}}$, α and β belong to $\mathcal{S}_{\mathrm{B}}^3 \cup \{\_\}$.

Fig. 7. Definition of Φ.

The transition rules are given in fig. 7. The first rule correspond to the lack of any signal. On the lower layer, the two 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 the cumbersome data. The rules which corresponds to the updating are on the lines 2 and 5.

The second and third lines of fig. 7 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 turns to ♠.

Let us detail how the signal ♠ turns on the right side, as depicted on the fourth line of fig. 7. On arriving on ▮ from the left, ♠ grab it and turns to ♣. On the next iteration, ♣ turns to ✚ and does nothing else. This is the delay of 1 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 of fig. 7. The state ✚ does not appear.

The defined rules are one-to-one, thus they can be completed so that Φ is a permutation, B is then reversible (lem. 1).

The initial configuration is depicted on the first line of fig. 5. The state of each cell is copied thrice in the upper layer. Markers ▮, ★ and ▮ are laid in the center of 3 adjacent cells and the ♥ is together with the left ▮. □

4.4 Generalization

This construction can be generalized to any dimension greater than 1. The bent of the space-time diagram is always done on the first direction. On this direction, the dynamics are exactly the same as explained above. On the other directions, the same space-time location and signals are found. The updating is still conditioned by the parity of the sum of the numbers of the iterations. There is an infinity of ▮, ★ and suit signals. They are arranged on hyperplanes orthogonal to the first direction and are exactly synchronized.

Theorem 8 *Any d-CA can be space-time simulated by a d-R-CA.*

PROOF. Any d-CA can be simulated in real time by a d-CA which neighborhood is $\{-1,0,1\}^d$ (Lemma 3). The theorem comes from from previous lemma 7 and the fact that d-R-PCA are d-R-CA. □

5 Conclusion

We have proved that any CA can be space-time simulated by a reversible CA. There are d-R-CA able to simulate (iteratively) all d-R-CA over any configuration [4].

Theorem 9 *There are d-R-CA able to space-time simulate any d-CA.*

Unfortunately, with the construction we gave, the inserted space-time diagram is bent in $\bigvee$. This makes it very hard to access geometrical properties like Fisher-constructibility.

In our space-time simulation, it is not possible to go backward before the first configuration if no previous configuration was previously encoded in the initial configuration. This would yield an α too complicated and moreover, there is no insurance that there exists any previous configuration at all.

In our construction, an infinite time is needed to fully generate the configuration after one iteration. 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. But this is not the case for the average configuration.

Space-time simulation must have different properties that the usual simulation because it is a larger relation. Our definition keeps the locality of the information processing but is not shift invariant.

It would be interesting to know up to what extend the techniques and results of this article can be adapted to the case where χ must be bounded when t is fixed. For example, if they is an integer c such that $\forall x, y, \forall t, |\chi(x,t) - \chi(y,t)| \leq c$, we believe that there is some iterative simulation via some kind of grouping.

Our construction relies on the infiniteness of the space to store data for reversibility. It does not work on limited space like torus. We do not know how to extend our result to such underlying lattices.

References

[1] S. Amoroso and Y. Patt. Decision procedure for surjectivity and injectivity of parallel maps for tessellation structure. *Journal of Computer and System Sciences*, 6:448–464, 1972.

[2] C. H. Bennett. Logical reversibility of computation. IBM *Journal of Research and Development*, 6:525–532, 1973.

[3] J. O. Durand-Lose. Reversible cellular automaton able to simulate any other reversible one using partitioning automata. In LATIN *'95*, number 911 in Lecture Notes in Computer Science, pages 230–244. Springer-Verlag, 1995.

[4] J. O. Durand-Lose. Intrinsic universality of a 1-dimensional reversible cellular automaton. In STACS '97, number 1200 in Lecture Notes in Computer Science, pages 439–450. Springer-Verlag, 1997.

[5] P. C. Fisher. Generation of primes by a one-dimension real-time iterative array. *Journal of the* ACM, 12(3):388–394, 1965.

[6] G. A. Hedlund. Endomorphism and automorphism of the shift dynamical system. *Mathematical System Theory*, 3:320–375, 1969.

[7] O. Heen. A linear speed-up theorem for cellular automata synchronizers and applications. To appear in Theoretical Computer Science.

[8] O. Heen. *Economie de Ressources sur Automates Cellulaires.* PhD thesis, LITP, IBP, Université Paris 7, 1996. In French.

[9] J. Kari. Reversibility of 2D cellular automata is undecidable. *Physica D*, 45:379–385, 1990.

[10] J. Mazoyer. Signals in one dimensional cellular automata. Technical Report 94-50, LIP, ENS Lyon, 46 allée d'Italie, 69 364 LYON 7, 1994.

[11] J. Mazoyer. Computations on one dimensional arrays. *Annals of Mathematics and Artificial Intelligence*, 16:285–309, 1996.

[12] E. Moore. Machine models of self-reproduction. In *Proceeding of Symposium on Applied Mathematics*, volume 14, pages 17–33, 1962.

[13] K. Morita. Computation-universality of one-dimensional one-way reversible cellular automata. *Information Processing Letters*, 42:325–329, 1992.

[14] K. Morita. Reversible simulation of one-dimensional irreversible cellular automata. *Theoretical Computer Science*, 148:157–163, 1995.

[15] K. Morita. Universality of a reversible two-counter machine. *Theoretical Computer Science*, 168:303–320, 1996.

[16] J. Myhill. The converse of Moore's garden-of-eden theorem. In *Proceedings of the Symposium of Applied Mathematics*, number 14, pages 685–686, 1963.

[17] D. Richardson. Tessellations with local transformations. *Journal of Computer and System Sciences*, 6:373–388, 1972.

[18] T. Toffoli. Computation and construction universality of reversible cellular automata. *Journal of Computer and System Sciences*, 15:213–231, 1977.