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.

Reversible space-time simulation of cellular automata*

Jérôme Durand-Lose[1]

Laboratoire I3S, CNRS UPREES-A 6070, 930 Route des Colles, BP 145, 06903 SOPHIA ANTIPOLIS Cedex, FRANCE.

Abstract

The goal of this paper is to design a reversible d-dimensional cellular automaton which is capable of simulating the behavior of any given d-dimensional cellular automaton over any given configuration (even infinite) with respect to a well suited notion of simulation we introduce. We generalize a problem which was originally addressed in a paper by Toffoli in 1977. He asked whether a d-dimensional reversible cellular automaton could simulate d-dimensional cellular automata. In the same paper he proved that there exists a $d+1$-dimensional reversible cellular automaton which can simulate a given d-dimensional cellular automaton.

To prove our result, we use as an intermediate model partition cellular automata defined by Morita *et al.* in 1989.

Keywords: Cellular Automata, Intrinsic universality, Partitioned Cellular Automata, Space-time simulation, Reversibility

1. Introduction

"Is it possible to capture the behavior of a discrete system with another one which is reversible?" We answer affirmatively in the case of cellular automata.

Reversible systems are of interest since they preserve information and energy and allow unambiguous backtracking. They are studied in computer science in order to design computers which would consume less energy.

Cellular automata (CA), a model of computation introduced by Ulam and von Neumann in the fifties, model massively parallel computations and physical phenomena. A CA can be considered as a bi-infinite array of dimension d whose elements are called *cells*. Each cell takes a value from a finite set of states Q. A configuration is a valuation of the whole array. A CA updates its configuration by synchronously changing the state of each cell according to its neighbor cells with respect to a local transition function. All the cells use the same local transition function. Thus the update process is parallel, synchronous, local and uniform. From the local transition function f, one can define a corresponding global transition function $\mathcal{G}$ over the configurations.

*2017 version, minor changes.

Email address: jerome.durand-lose@univ-orleans.fr (2017) (Jérôme Durand-Lose)

[1]This work was done while the author was in LaBRI, Université Bordeaux I, France.

A CA is *reversible* when its global transition function $\mathcal{G}$ is bijective and $\mathcal{G}^{-1}$ is the global transition function of some CA. The study of CA reversibility started in the sixties. It is known that if $\mathcal{G}$ is one-to-one then it is bijective [10, 13] and the corresponding CA is reversible [5, 14]. If the reversibility of CA is proved decidable in dimension 1 [1], this is not anymore true for greater dimensions [7].

In 1977, Toffoli [15] proved that any CA can be simulated by a reversible CA (R-CA) one dimension higher and that there are 2-dimensional R-CA 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 [11]. To do this he introduced *partitioned cellular automata* (PCA). In PCA the state of each cell is partitioned according to the neighborhood. Each cell exchanges parts of its state with the neighbor cells and then computes its new state.

In [2], we proved that there are R-CA able to simulate in linear time any R-CA of the same dimension (greater than 2), over any configuration (even infinite). This result has been extended to dimension 1 in 1997 [3] 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 [12] proved that it is possible over finite configurations, i.e. configurations such that there are finitely many cells which are not in a given 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.

Intuitively, a simulation of $\mathcal{B}$ by $\mathcal{A}$ is some construction which shows that everything machine $\mathcal{B}$ can do on input x can be performed as well by machine $\mathcal{A}$ on the same input. We generalize this intuitive notion to *space-time* simulation. For a CA $\mathcal{A}$, a *space-time diagram* depicts the whole computation of $\mathcal{A}$ on an input c_0. It corresponds to the sequence of all the configurations of A starting with c_0. Space-time diagrams serves as a tool for designing CA (e.g. see [4, 9, 8, 6]). Our notion of space-time simulation of $\mathcal{B}$ by $\mathcal{A}$ defines an embedding relation between the space-time diagram of $\mathcal{B}$ and the space-time diagram of $\mathcal{A}$.

Our main result states that any d-dimensional CA (d-CA) can be space-time simulated by a d-R-PaCA. We give a proof for the 1-dimensional case and sketch the generalization to higher dimensions. As a corollary, we prove that any d-CA can be simulated by a d-R-CA since any d-R-PaCA is indeed a particular d-R-CA. Then, as a conclusion, we state that there exists a d-R-CA which is capable of space-time simulating any d-CA.

2. Definitions

Configurations are bi-infinite arrays of finite dimension d. Points of configurations are called *cells*. Each cell takes a value from a finite set of *states* Q. The set of all configurations is denoted by $\mathcal{C}$ $(= Q^{\mathbb{Z}^d})$. The state of cell x in configuration c is denoted c_x. Cellular automata (CA) and partitioned cellular automata (PCA) update all the cells of a configuration in a parallel, synchronous local and uniform way.

2.1. Cellular automata

A d-dimensional *cellular automaton* (d-CA) is defined by $(Q, \mathcal{N}, f)$. The *neighborhood* $\mathcal{N}$ is a finite subset of $\mathbb{Z}^d$. The *local transition function* $f : Q^{\mathcal{N}} \to Q$ yields the new state

of a cell according to the states of the cells in its neighborhood. The *global transition function* $\mathcal{G} : \mathcal{C} \to \mathcal{C}$ updates configurations as follows:

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

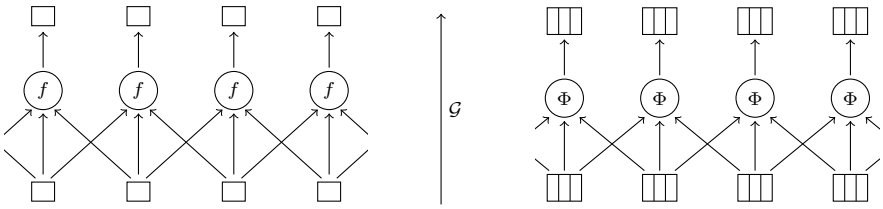
The new state of a cell only depends on the neighbor cells as described on Fig. 1 (left).

2.2. Partitioned cellular automata

According to Morita's definition [11, 12], a d-dimensional *partitioned cellular automaton* (d-PCA) is a special form of CA defined by $(Q, \mathcal{N}, \Phi)$. The set of states is a cartesian product of sets indexed by the neighborhood: $Q = \prod_{\nu \in \mathcal{N}} Q^{(\nu)}$. The ν component of a state s is denoted by $s^{(\nu)}$. The *state transition function* Φ operates over Q. The global transition function $\mathcal{G}$ is defined by:

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

Equivalently, each state is the product of the information to be 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 transition 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 neighbor cells, as depicted in the right part of Fig. 1.



(a) Cellular automata (b) Partitioned CA

Figure 1: Updating of CA and PCA.

2.3. Reversibility

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

Lemma 1 *A PCA is reversible if and only if its state transition function Φ is a permutation, which is decidable.*

Proof. (sketch)
If Φ is a permutation then the inverse PCA is $\left(\prod_{\nu\in-\mathcal{N}} Q^{(-\nu)}, -\mathcal{N}, \Phi^{-1}\right)$ where $-\mathcal{N} = \{-\nu|\nu \in \mathcal{N}\}$. The action of Φ is undone and the different pieces are sent 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 that are mapped to the same configuration.
Decidability comes from the finiteness of Q. *q.e.d.*

Amoroso and Patt [1] proved that the reversibility of 1-dimensional CA is decidable. Kari [7] proved that it is not decidable any more in dimension greater or equal to 2. 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 iterations over a configuration an *iterated image*. In this context, simulation means that for any initial configuration c of the *simulated* CA $\mathcal{A}$:, one wants to find each iterated image of c entirely encoded inside an iterated image of a configuration e of the *simulating* CA $\mathcal{B}$.

Definition 2 (Toffoli [15]) A CA $\mathcal{A}$ *(iteratively) simulate* a CA $\mathcal{B}$ if there exist three functions $\psi : \mathcal{C}_\mathcal{B} \times \mathbb{N} \to \mathbb{N}$, $\alpha : \mathcal{C}_\mathcal{B} \to \mathcal{C}_\mathcal{A}$ and $\beta : \mathcal{C}_\mathcal{A} \to \mathcal{C}_\mathcal{B}$ such that:

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

The functions ψ, α and β must be of a lower complexity than the simulated CA $\mathcal{B}$ 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 infinitely many 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, for any configuration c, $\psi(c,t) = \tau\, t$. 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-PaCA can be simulated by d-R-CA. In [3], there is a construction of a simulation of d-CA (d-R-CA) by d-PCA (d-R-PaCA) 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.*

Proof. [sketched] The cells are gathered in blocks of adjacent cells as illustrated in Fig. 2 for $\mathcal{N} = \{-2,0,1\}$. The simulation is in real time. *q.e.d.*

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.

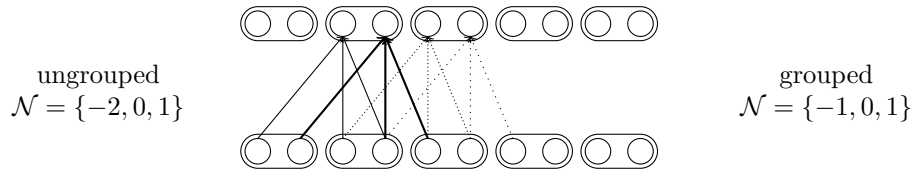


Figure 2: Grouping cells 2 by 2.

In other words, let $\mathcal{G}$ be the global transition function of some d-CA $\mathcal{A}$ and a a configuration. The space-time diagram $\mathbb{A} : \mathbb{Z}^d \times \mathbb{N} \to Q$ associated to $\mathcal{A}$ and a is defined by $\mathbb{A}_{x,t} = \left(\mathcal{G}^t(a)\right)_x$. It is denoted by $(\mathcal{G}, a)$ or $(\mathcal{A}, a)$.

A space-time diagram $\mathbb{B}$ is *embedded* into 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 into $\mathbb{A}$.

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

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

The configuration b is encoded into a with respect to α. To recover an iterated image of b, the function χ indicates which iteration is to be considered for each cell and β decodes the generated configuration. The generation of c^t and then $\mathcal{G}^t_\mathcal{B}(b)$ is illustrated in Fig. 3.

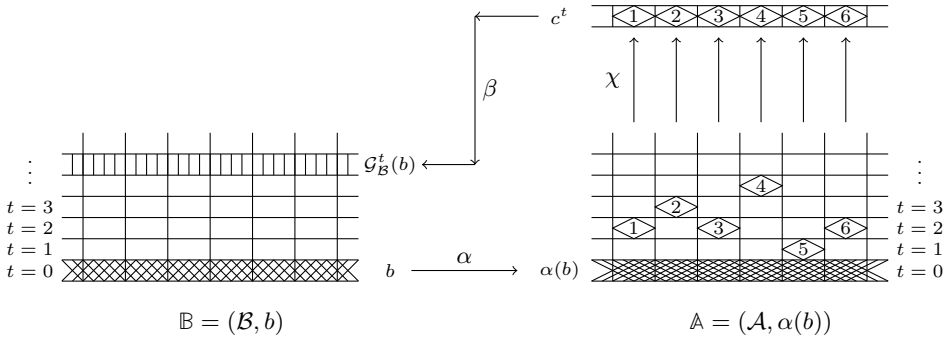


Figure 3: Space-time diagram $\mathbb{B}$ is embedded into $\mathbb{A}$.

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 $\mathcal{A}$ *space-time simulates* a CA $\mathcal{B}$ when any space-time diagram generated by $\mathcal{B}$ can be embedded into a space-time diagram generated by $\mathcal{A}$ and all insertions use the same functions α and β.

The function χ may depend on the initial configuration b of the embedded diagram. If χ only depends on the time and the initial configuration (and not on the location of the cell) simulation is iterative. Space-time simulation is an extension of iterative simulation.

4. Space-time simulation by reversible CA

In this section we give an explicit construction which proves the following lemma:

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

The proof is only detailed in dimension 1. We show how the simulation works before going into details. At the end of this section, we sketch how to generalize it to any dimension.

4.1. Macro dynamics

Let $\mathcal{B}$ =$(Q_{\mathcal{B}}, \{-1,0,1\}, f)$ be any 1-CA with neighborhood $\{-1,0,1\}^1$. We build a 1-R-PaCA $\mathsf{P} = (Q_{\mathsf{P}}, \{-1,0,1\}, \Phi)$ which space-time simulate $\mathcal{B}$. Let b be any configuration in $\mathcal{C}_{\mathcal{B}}$ and $\mathbb{B}$ the associated space-time diagram. We show how the space-time diagram $\mathbb{P}$ is generated in order to embed $\mathbb{B}$.

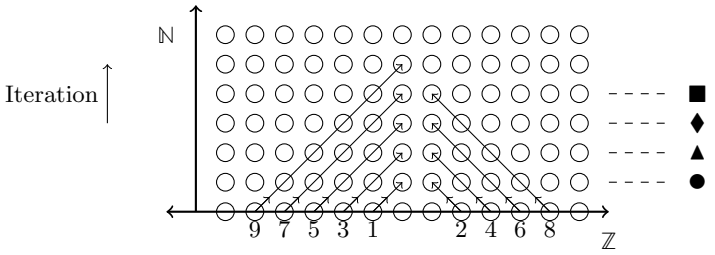
A signal moves forth and back on a finite part of the configuration. It updates the cells when it passes over them. Outside the part where the signal moves, the P-cell are at $\mathcal{B}$-iteration 0. We call the *updating zone* the part where the signal moves. Inside it, P-cell are at $\mathcal{B}$-iteration is more than 0 and the closer to the center a cell is the higher its $\mathcal{B}$-iteration number is. As iteration goes by, the updating zone is enlarged on both sides (space) and in iteration numbers (time) so that any cell will eventually enter the zone and reach any iteration.

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 4 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 $\mathcal{B}$-iterations (as on $\mathbb{P}$ in Fig. 5).

The state of a cell x at iteration t in the embedded diagram $\mathbb{B}$ is denoted by ${}_x\backslash^t$ (${}_x\backslash^t = \mathcal{G}^t_{\mathcal{B}}(c)_x$) 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 are accumulating. They cannot be disposed off because $\mathcal{G}_{\mathcal{B}}$ is not necessarily one-to-one and the previous configurations cannot be guessed from the actual one. These needed but cumbersome data are evacuated by being sent away on both sides of the configuration.

When a signal goes from the left to the right for the n^{th} time on the updating zone, as in Fig. 5, 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 it and save it while it



The symbol in the last column identifies the iteration.
It corresponds to the embedding in Fig. 5.

Figure 4: Order of generation inside the simulated diagram $\mathbb{B}$.

generates ${}_x\backslash^1$. 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 $\mathbb{B}$ are generated on an hyperbola indicated by circles (triangles, diamonds and squares respectively) on the simulating diagram $\mathbb{P}$ in Fig. 5. This corresponds to the layer-construction of $\mathbb{B}$ depicted in Fig. 4. Figure 5 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

Cells are organized in three 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 $[{}_x\backslash^t]$.

The first 26 iterations are depicted in Fig. 6. In the upper layer, the cells alternatively holds 3 times the same state $({}_x\backslash^t)$ or the states of the cell and its two closest neighbors at the same iteration $({}_{x-1}\backslash^{t-1}, {}_x\backslash^{t-1}, {}_{x+1}\backslash^{t-1})$, otherwise some mix over 2 or 3 iterations. A cell can only be updated when it has the information $[{}_x\backslash^t]$. By induction from Fig. 6, 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 that rules the dynamics. We call it the *"suit signal"*. Depending on its position, it takes the values ♣, ♠, ♥ and ♦ in $\mathbb{P}$. The suit signal only moves forth and back in the updating zone and thus appears as a zigzag on figures 5 and 6. It is delayed by one on the left side to keep it synchronized with the presence of $[{}_x\backslash^t]$.

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, three values, ${}_{x-1}\backslash^{t-1}$, ${}_x\backslash^{t-1}$ and ${}_{x+1}\backslash^{t-1}$, are 'used up' and become useless. They are gathered in $[{}_x\backslash^t]$ and moved to the lower layer to

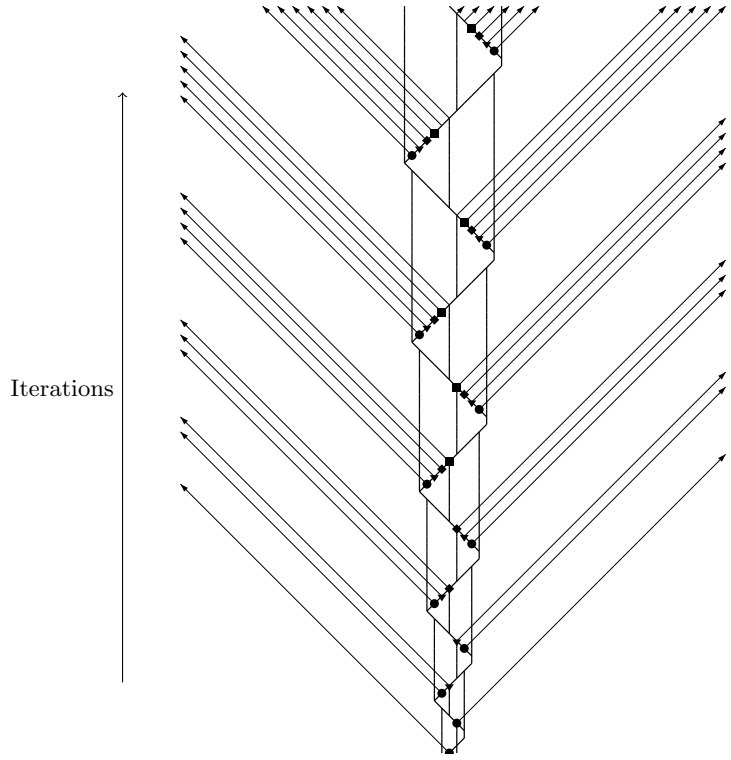


Figure 5: Scheme of the evacuation of data ($[_x\backslash^t]$) on the simulating diagram $\mathbb{P}$.

be evacuated. Three copies of the new state ${}_x\backslash^t$ are made. They will be used for the next update of the simulated cell and of its two 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. 6.

4.3. State transition function

The set of states of P is detailed in Fig. 7. If the CA $\mathcal{B}$ 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 state transition function and in complexity.

Cells are depicted as 3×3 arrays as in the first line of Fig. 6. The upper layer holds the states of P; a configuration of $\mathcal{B}$ is encoded there as depicted in Fig. 5. 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 ★.

The transition rules are given in Fig. 8. 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

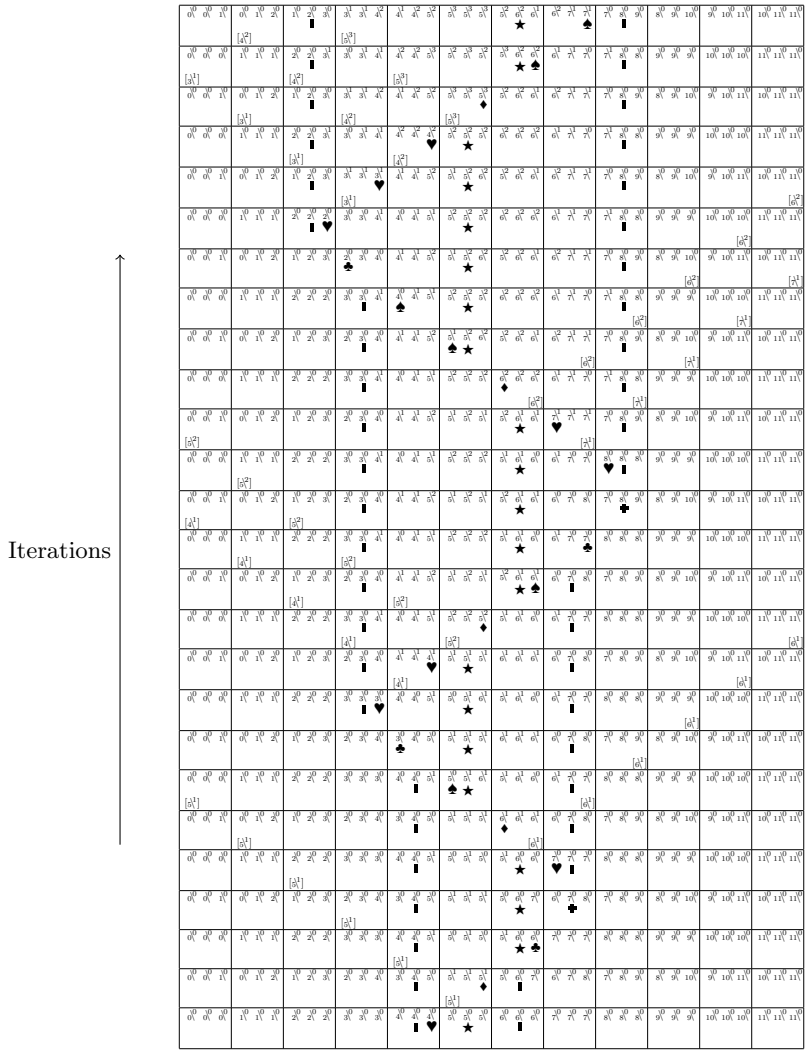


Figure 6: The first 26 iterations of the space-time simulation.

at each iteration. This is used to evacuate data. The rules which corresponds to the updating are on the lines 2 and 5.

The second and third lines of Fig. 8 depicted how ♥ moves to the right and updates

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

Figure 7: Set of states of P: Q_{P}.

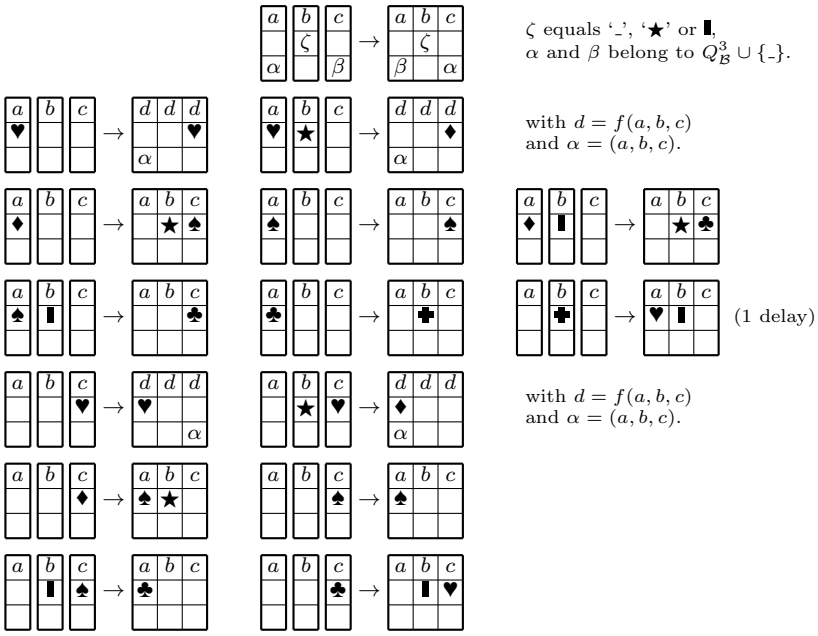


Figure 8: Definition of Φ.

cells. When it reaches the middle frontier ★, it moves it one step to the right as ♦ and then turns to ♠.

Let us detail how the signal ♠ turns on the right side, as depicted on the fourth line of Fig. 8. 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 of Fig. 8. The state ✚ does not appear.

The rules defined are one-to-one, thus they can be completed so that Φ is a permutation, $\mathcal{B}$ is then reversible (Lem 1).

The initial configuration is depicted on the first line of Fig. 6. The state of each cell is copied three 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 ⋁. This makes it very hard to access geometrical properties like Fisher-constructibility.

4.4. Sketch of generalization

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 the same 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 in real time by a d-CA whose neighborhood is $\{-1,0,1\}^d$ (lemma 3). The theorem comes from Lemma 7 and the fact that d-R-PaCA are d-R-CA.

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

There are d-R-CA able to simulate (iteratively) all d-R-CA over any configuration [3].

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

5. Conclusion

With our space-time simulation, it is not possible to go backward before the first configuration if no previous configuration were previously encoded in the initial configuration. Moreover, there is no guarantee that any previous configuration does exist.

An infinite time is required 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 like in [11, 12].

Although space-time simulation is an extension of iterative simulation, it differs. For example, it keeps the locality of the information processing but is not shift invariant (at least in our construction).

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 is bounded when t is fixed. We believe that it corresponds to an iterative simulation via some kind of grouping.

References

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

[2] J. 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.

[3] J. 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.

[4] P. C. Fischer. Generation of primes by a one-dimensional real-time iterative array. *J ACM*, 12(3): 388–394, 1965.

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

[6] O. Heen. Linear speed-up for cellular automata synchronizers and applications. *Theoret Comp Sci*, 188(1-2):45–57, 1997.

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

[8] J. Mazoyer. Computations on one dimensional cellular automata. *Ann Math Artif Intell*, 16: 285–309, 1996. doi: 10.1007/BF02127801.

[9] J. Mazoyer and V. Terrier. Signals in one-dimensional cellular automata. *Theoret Comp Sci*, 217 (1):53–80, 1999. doi: 10.1016/S0304-3975(98)00150-9.

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

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

[12] K. Morita. Reversible simulation of one-dimensional irreversible cellular automata. *Theoret Comp Sci*, 148:157–163, 1995.
[13] J. R. Myhill. The converse of Moore's garden-of-eden theorem. In *Proceeding of the American Mathematical Society*, volume 14, pages 685–686, 1963.
[14] D. Richardson. Tessellations with local transformations. *J Comput System Sci*, 6(5):373–388, 1972.
[15] T. Toffoli. Computation and construction universality of reversible cellular automata. *J Comput System Sci*, 15:213–231, 1977.

bibtex entry

```
@article{durand-lose00tcs,
  author = {Durand-{L}ose, J{\'e}r{\^o}me},
  journal = {Theoretical Computer Science},
  number = {1--2},
  pages = {117--129},
  title = {Reversible space-time simulation of cellular automata},
  volume = {246},
  year = {2000},
  doi = {10.1016/S0304-3975(99)00075-4},
  language = {english}
}
```