LATIN '95, LNCS 911, pp. 230–244

Reversible Cellular Automaton Able to Simulate Any Other Reversible One Using Partitioning Automata

Jérôme Olivier DURAND-LOSE*

Departamento de Ingeniería Matemática,
Facultad de Ciencias Físicas y Matemáticas,
Universidad de Chile, Santiago, Chile.
e-mail: jdurand@llaima.dim.uchile.cl

Abstract. Partitioning automata (PA) are defined. They are equivalent to cellular automata (CA). Reversible sub-classes are also equivalent. A simple, reversible and universal partitioning automaton is described. Finally, it is shown that there are reversible PA and CA that are able to simulate any reversible PA or CA on any configuration.

1 Introduction

The main interest of reversibility in computation is backtracking a phenomenon to its source and in relation with physics, isoentropic phenomena modelization and saving energy, and had have various interests in relation to physics as explained by Toffoli and Margolus in [15]. It is well known that, given any d-dimensional cellular automata (CA), it can be simulated by one $(d+1)$-dimensional CA which is reversible [12]. It is still an open problem if it can be simulated by a reversible CA of the same dimension. For example, Morita showed in [7] that this is true in dimension one but only over finite configurations.

In the present paper, some definitions and basic results about partitioning automata (PA) and mutual simulations with cellular automata are shown before an example of a reversible and universal PA is detailed. Then the main result is deduced: for any d, with $2 \leq d$, there exists d-dimensional reversible PA and CA able to simulate any d-dimensional reversible PA or CA over both finite and infinite configurations.

Partitioning automata were first introduced by Margolus and Toffoli's in the middle of the 80's as models of lattice gases and other reversible physical phenomena [14]. Like cellular automata, they work on an infinite lattice. A *node* is a point of the lattice and has a value in a finite set of *states*. A *tile* is a $h \times v$ rectangle of nodes. Like CA, the global function of a PA is defined by a local rule called *elementary transition function* from and to the set of tiles (for CA, a neighborhood is mapped into a cell). The plane is cut into different regular

* This research was partially supported by ECOS and the French Cooperation in Chile

partitions of tiles (a partition is fully determined by h, v and its origin). An *elementary transition* is the parallel replacement of all the tiles of a given partition by their images by the elementary transition function. The *global transition function* is the sequential composition of various elementary transitions. A PA is reversible iff its global function is invertible and is the global function of some PA. It is equivalent to bijectivity of the elementary transition function which is decidable (Lemma 8).

Cellular automata (CA) are the most famous model of parallel phenomena and architectures. They have been widely studied for decades and there is a lot of results about them [16]. After a brief definition of CA, both simulations between CA and PA and between reversible CA and reversible PA are constructed. Thus, as far as computation is concerned the class of PA (resp. reversible PA) is equivalent to the one of CA (resp. reversible CA) and the class of PA is computational universal (able to simulate any Turing machine). The fact that CA and PA can simulate each other was already mentioned by Toffoli and Margolus in [14]. Here, full constructive demonstrations that care about conservation of reversibility are given.

In the second part, an example of a simple and reversible PA is given. Margolus and Toffoli described a more simple one (with only two states and two (2,2)-tiling) and showed both reversibility and universality in [14,4]. The one presented here has the interest of not having virtual **0** signals and somehow needs less space. This PA, noted P_u, has four states, for the ether, **0** and **1** signals and to built the architecture (signals routing and gates). It is able to simulate any boolean circuit. Fredkin and Toffoli studied a binary logic/functionality called *conservative logic* where all functions are reversible and keep the number of **1** [13,2]. K. Morita demonstrated in [6] that it can do any finite reversible computing without constant inputs or drop-off. P_u can simulate any circuit of this logic. Using results of this logic and some constructions, it is shown that P_u is able to simulate any reversible PA.

Finally, gathering the result of both parts, it is concluded that there exist reversible CA and PA able to simulate any reversible PA or CA. It is explained how to turn this universal PA into one that can simulate any other one but is not reversible any more. Those results can be extended to higher dimensions.

2 Definitions

Partitioning automata (PA) work over a 2-dimensional infinite *lattice* $\mathcal{L}$ ($= \mathbb{Z}^2$). The points of $\mathcal{L}$ are called *nodes*. Each node has a value in a finite *set of states* $\mathcal{Q}$. A *configuration* is a lattice with a value in each node. $\mathcal{C}$ ($= \mathcal{Q}^{\mathcal{L}}$) is the set of configurations.

Definition 1. Let x, y be two non-zero natural numbers and α and β be two integers. The *tile* $t_{\alpha,\beta} \in \mathcal{Q}^{h \times v}$, of coordinates (α, β), size (h, v) and origin (x, y) of a configuration $c \in \mathcal{C}$ is the following rectangle part of c:

$$t_{\alpha,\beta} \;=\; {}^{c}|_{(\,x+\alpha.h,\;\,y+\beta.v\,)} \;+\; [0, h-1] \times [0, v-1] \;.$$

Definition 2. The *partition* of *size* (h, v) and *origin* (x, y) is the partition of $\mathcal{L}$ with the tiles of size (h, v) and origin (x, y) . It is rectangular and regular as shown in Fig 1.

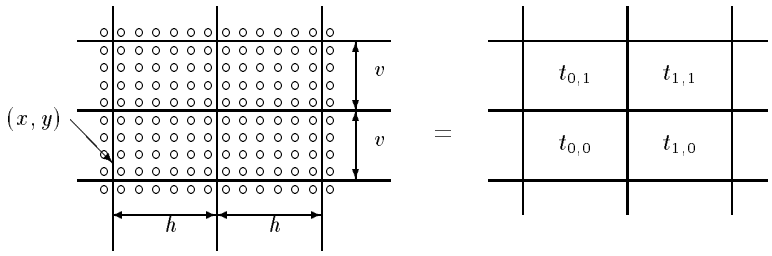

Fig. 1. $(6, 4)$ partition of origin (x, y) .

Definition 3. The *Elementary Transition Function* (e.t.f.) e is a function from and to the set of tiles ($e : \mathcal{Q}^{h \times v} \longrightarrow \mathcal{Q}^{h \times v}$).

For a given PA, all partitions have the same size. (h, v) is a constant of the PA called its *size*.

Definition 4. An *Elementary Transition* (e.t.) $\mathcal{T}$ is, for a given partition, the simultaneous parallel replacement of all its tiles by their images by e as in Fig. 2.

An elementary transition $\mathcal{T}_{x,y}$ is fully determined by its origin (x, y) .

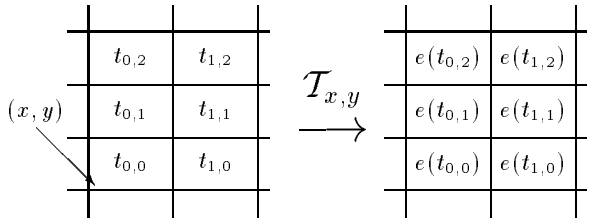

Fig. 2. $\mathcal{T}_{x,y}$: elementary transition of origin (x, y) .

Definition 5. A partitioning automaton is defined by:

$$P = \{\ \mathcal{Q},\ (h, v),\ n,\ (\,(x_i, y_i)\,)_{1 \le i \le n},\ e\ \}\ .$$

Several partitions are used in order to let information spread from tile to tile. Their origins are $(x_i, y_i)_{1 \le i \le n}$.

Definition 6. The updating function of a PA, the *global transition* $\mathcal{G}$, maps configurations into configurations. It is the sequential composition of all the parallel elementary transitions associated with each of the partitions.

$$\mathcal{G} \ : \ \mathcal{C} \to \mathcal{C} \ = \ \mathcal{T}_{x_n,y_n} \circ \mathcal{T}_{x_{n-1},y_{n-1}} \circ \ldots \circ \mathcal{T}_{x_1,y_1} \ .$$

Definition 7. A PA P is *reversible* iff its global function $\mathcal{G}$ is invertible and $\mathcal{G}^{-1}$ is the global function of a PA, the *inverse* PA. Reversible PA are noted R-PA.

Lemma 8. *A PA is reversible iff its elementary transition function is invertible.*

Proof. By definition, if the PA P is reversible, $\mathcal{G}$ is invertible. If $\mathcal{G}$ is invertible then, since $\mathcal{G} = \mathcal{T}_{x_n,y_n} \circ \mathcal{T}_{x_{n-1},y_{n-1}} \circ \ldots \circ \mathcal{T}_{x_1,y_1}$, $\mathcal{T}_{x_1,y_1}$ must be injective. Because of the construction of $\mathcal{T}$, the elementary transition function e must be injective and as it works over a finite set, it is bijective.

Conversely, if e is bijective, let $\mathcal{T}'_{x_1,y_1}$ be the e.t. of origin (x_1, y_1) and e.t.f. e^{-1} . $\mathcal{T}'_{x_1,y_1}$ is the inverse of $\mathcal{T}_{x_1,y_1}$. In the same way, all the e.t. are invertible. As a consequence, the global transition $\mathcal{G}$ is invertible and:

$$\mathcal{G}^{-1} = \mathcal{T}'_{x_1,y_1} \circ \mathcal{T}'_{x_2,y_2} \circ \ldots \circ \mathcal{T}'_{x_n,y_n} \ .$$

P is reversible and its inverse is:

$$P^{-1} = \{ \ \mathcal{Q}, (h, v), \ n, \ (\ (x_{n+1-i}, y_{n+1-i}) \ )_{1 \le i \le n}, \ e^{-1} \ \} \ . \qquad \square$$

The inverse PA only differs by the use of e^{-1} instead of e and the opposite order of the partitions. In the above proof, it was also shown that injectivity, surjectivity and reversibility are equivalent for PA.

Definition 9. For any two functions $f : F \to F$ and $g : G \to G$. g *simulates* f (in real time) iff there exists two encoding functions $\alpha : F \to G$ and $\beta : G \to F$, recursive, space and time inexpensive compared to f and g, such that: $f^n = \beta \circ g^n \circ \alpha$ for all non-zero natural n. g can be used instead of f for iterating the global function.

The simulation is *strong* iff f and g are invertible and $\forall z \in \mathbb{Z}$, $f^z = \beta \circ g^z \circ \alpha$. It is *uniform* iff α and β are bijective and $\alpha^{-1} = \beta$ or in other words f also simulates g. An automaton simulates another iff its global function simulates the global function of the other.

3 Relations to Cellular Automata

Cellular automata (CA) work on the same kind of lattice L $(= \mathbb{Z}^2)$. The points of L are called *cells* and they take their values in a finite set of *states* Q. The *neighborhood* is defined by a finite set of relative coordinates $N = \{x_1, \ x_2, \ \ldots, x_n\}$ $(\forall i, x_i \in L)$. Making an iteration is changing the value of each cell according to the states of its neighbors and a *local function* $f : Q^n \to Q$.

Definition 10. A CA is defined by: $A \ = \ (\ Q, \ N, \ f \ )$. $C = Q^L$ is the set of *configurations*. Let c be a configuration. The *global function*, $F : C \to C$ is defined by:

$$F \ (c)(x) = f \ (\ c(x + x_1), \ c(x + x_2), \ \ldots \ , \ c(x + x_n) \ ) \ .$$

A CA is *reversible* iff, its global function F is invertible and its inverse is the global function of some CA. Moore and Myhill proved that for CA injectivity is equivalent to reversibility in [9,5]. Kari showed in [3] that as opposed to PA it is not decidable. Reversible CA are noted R-CA.

3.1 Simulating PA with CA

Theorem 11. *Any PA can be uniformly simulated by a CA whose cells are cartesian products of nodes.*[1]

Proof. Let $P = \{\mathcal{Q}, (h, v), n, ((x_i, y_i))_{1 \leq i \leq n}, e\}$ be any PA. The first partition of $\mathcal{L}$ is identified with L. The cells are the tiles of the first partition and their values belong to $Q = \mathcal{Q}^{h \times v}$. This encoding helps to get rid of the origins of the partitions because the intersections of the partitions correspond to the same portion of every tile/cell. The origins are 'encoded' in the local function f.

The neighborhood is $N = [-n, n] \times [-n, n]$, where n is the number of partitions of P. The local function $f : Q^{(2n+1)^2} \to Q$ is defined as follows:

The cells in $[-n, n] \times [-n, n]$ correspond to the tiles $[-n, n] \times [-n, n]$ of the first partition. The e.t.f. e can operate over them, making the first e.t. . In the images of the tiles, $2n \times 2n$ tiles of the second partition can be found, the nodes of the updated cell are in the middle. The second e.t. can be made over these tiles.

In two steps, the images of the tiles $[-n+1, n-1] \times [-n+1, n-1]$ after two e.t. are generated. Each step this process is iterated, one more e.t. is made and the width of the 'window' is reduced by one.

In n steps, there are only left 2×2 tiles with the nodes of the updated cell in the middle. The values of these nodes are their images by the global transition of P. The nodes corresponding to the updated cells are taken as the image by f of the whole neighborhood. Different cuttings are shown in Fig. 3.

With this simple encoding and this function f, there is a natural identification between A and P. □

This is a strong, uniform, and real time simulation. The size of the tiles defines the number of states of the CA while the number of neighbors depends only on the number of partitions. The computation power of PA is at most equal to the one of the CA.

3.2 Simulating CA with PA

Theorem 12. *Any CA can be simulated by a PA.*

Proof. Let $A = (Q, N, f)$ be a CA. The *radius* r is the maximum absolute coordinate of the vectors of the neighborhood N. It represents half the size of the 'windows' required to gather information to update a cell. The tile is four

[1] Richardson proved in [10] that CA are equivalent to continuous, shift commuting functions over $\mathcal{Q}^{\mathcal{L}}$. This gives a direct, non-constructive proof of the Theorem.

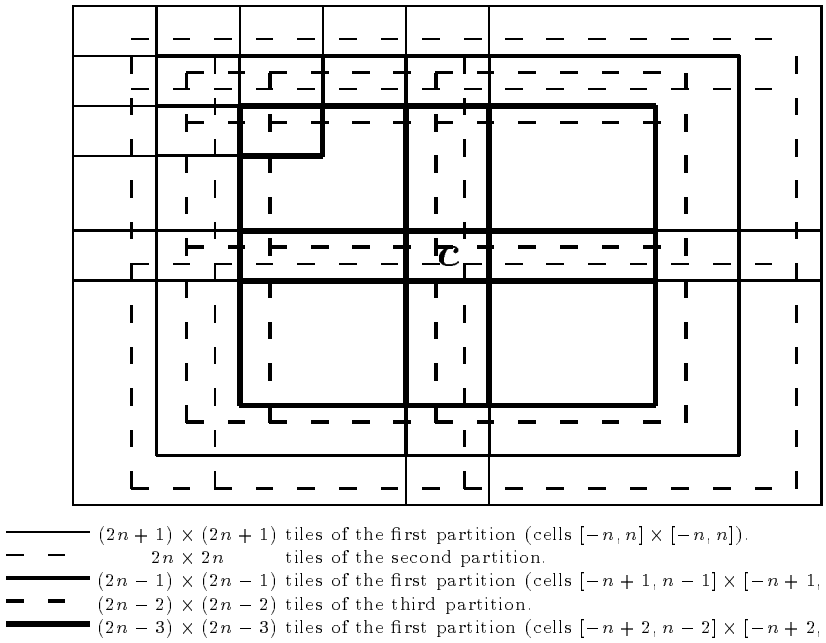

Fig. 3. first and second cuttings.

time bigger, *i.e.*, $h = v = 4r$. The number of cells in the windows is $s = 16r^2$. Let

$$P = \{\ Q \cup Q^2,\ (\ 4r,\ 4r\ ),\ 4,\ (\ (0,0),\ (2r,0),\ (0,2r),\ (2r,\ 2r)\ ),\ e\ \}\ .$$

The value of a node represents either the actual value of a cell or its actual and next values. Q is used to encode the configurations and Q^2 to keep information during the update.

Each $(4r, 4r)$ tile is cut in four parts (as shown in Fig. 4) named M (middle), V (vertical), H (horizontal) and C (corners). A tile is now noted (M, H, V, C) .

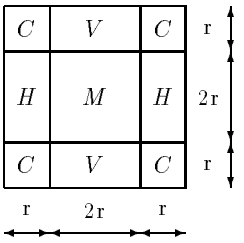

Fig. 4. partition of the tiles.

The values of the tiles are in $\mathcal{Q}^{4.r^2}$ ($\mathcal{Q} = Q \cup Q^2$). There are two special cases of $2r \times 2r$ sub-tiles, the ones with only Q values ($\in \Theta = Q^{4.r^2}$) and the ones with only Q^2 values ($\in \Delta = (Q^2)^{4.r^2}$).

Each time, if the current values of the cells are held in the tile, since all the neighbors of the middle cells (in M) are in the tile, their next values can be computed. The function γ computes the next values of the cells in M according to the current ones and sets them as second values of the nodes of M.

For each partition, the next values of the middle cells are added ($Q \to Q^2$). After four elementary transitions, all next values are computed, the actual are discarded and replaced by the next. The e.t.f. e is defined in Fig. 5 where π_2 is the second projection ($\pi_2(x_1, x_2) = x_2$) of all nodes of a tile.

$$
\begin{array}{ll}
\textbf{if} & (M,H,V,C) \in \Theta \times \Theta \times \Theta \times \Theta \\
 & \qquad \cup\ \Theta \times \Delta \times \Theta \times \Theta \\
 & \qquad \cup\ \Theta \times \times\Theta \times \Delta \times \Delta \\
\quad \textbf{then} & e(M,H,V,C) = (\gamma(M,H,V,C),H,V,C) \\
\textbf{else if} & (M,H,V,C) \in \Theta \times \Delta \times \Delta \times \Delta \\
\quad \textbf{then} & e(M,H,V,C) = \pi_2(\gamma(M,H,V,C),H,V,C) \\
\textbf{else} & e(A) = A \quad .
\end{array}
$$

Fig. 5. elementary transition function for CA simulation.

Altogether, each tile goes first from $\Theta \times \Theta \times \Theta \times \Theta$ to $\Delta \times \Theta \times \Theta \times \Theta$. Then, by changing partition, it goes to $\Theta \times \Delta \times \Theta \times \Theta$. It finally reaches $\Theta \times \Delta \times \Delta \times \Delta$ before being completed to $\Delta \times \Delta \times \Delta \times \Delta$ and projected to $\Theta \times \Theta \times \Theta \times \Theta$.

Let α be the natural injection of Q in $Q \cup Q^2$. A CA configuration is naturally coded in a PA configuration for simulation. □

In the first case, only information is added. The second case clearly adds non-injective rules. The simulating PA P cannot be reversible. The simulation is neither strong nor uniform. The class of PA and CA are equivalent.

3.3 Simulating Reversible CA

Theorem 13. *Any reversible CA can be strongly simulated by a reversible PA.*

Proof. Let A be a reversible CA. In the previous construction, the first case only generates injective rules. The definition of the second case is changed in order to generate injective rules. Then it will be possible to bijectively complete the set or rules.

The inverse of a reversible PA is very simple to built as described in the proof of lemma 8, when simulating strongly a CA A, the simulation of the inverse of the CA A, A^{-1} is somehow built. A^{-1} is used to bijectively erase the first coordinates.

Let r be large enough for the neighborhoods of both A and A^{-1}. The next values are still built in four steps (action of A). But the current values are now erased in four steps (reverse-action of A^{-1}).

Erasing depends on A^{-1}. The predicate $\mathcal{P}(M,H,V,C)$ is true iff $M \in \Delta$ and the old values in the middle correspond to what they should be according to A^{-1} and the next values encoded in the whole tile.

There are 7 partitions: $(0,0), (2r,0), (0,2r), (2r,2r), (0,0), (2r,0)$ and $(0,2r)$. During the first four elementary transitions, the tiles go from $\Theta \times \Theta \times \Theta \times \Theta$ to $\Delta \times \Delta \times \Delta \times \Delta$ by adding information in second coordinates and directly to $\Theta \times \Delta \times \Delta \times \Delta$. In the last four ones, only information redundant (for A^{-1}) is deleted. A^{-1} is applied in reverse. The algorithm is detailed in Fig. 6.

if	$(M,H,V,C) \in \Theta \times \Theta \times \Theta \times \Theta$	
	$\cup\ \Theta \times \Delta \times \Theta \times \Theta$	
	$\cup\ \Theta \times \Theta \times \Delta \times \Delta$	
then	$e(M,H,V,C) = (\gamma(M,H,V,C),H,V,C)$	
if	$\mathcal{P}(\gamma(M,H,V,C),H,V,C)$ **and** $(M,H,V,C) \in \Theta \times \Delta \times \Delta \times \Delta$	
then	$e(M,H,V,C) = (\pi_2(\gamma(M,H,V,C)),H,V,C)$	
else if	$\mathcal{P}(M,H,V,C)$ **and** $(M,H,V,C) \in \Delta \times \Delta \times \Delta \times \Theta$	
	$\cup\ \Delta \times \Theta \times \Theta \times \Delta$	
	$\cup\ \Delta \times \Theta \times \Theta \times \Theta$	
then	$e(M,H,V,C) = (\pi_2(M),H,V,C)$	
else	... { completed bijectively } .	

Fig. 6. elementary transition function for R-CA simulation by R-PA.

Because of the discriminating use of $\mathcal{P}$ all produced rules are injective. Completing bijectively is not a problem. □

The classes of reversible PA and CA are equivalent. In each simulation of CA, the configuration of the PA goes from (c) to $(c, F(c))$ to $(F(c))$.

4 A Reversible and Universal PA

Let $$P_u = (\ \mathcal{Q}_u,\ (2,2),\ 2,\ (\,(0,0),(1,1)\,),\ e_u\ )\ .$$

The set of states is $\mathcal{Q}_u = \{\_, \mathbf{0}, \mathbf{1}, \bullet\}$. '$\_$' is the ether, space where signals go through in straight line. **0** and **1** are binary signals and '$\bullet$' is used to make gates and set paths. The $\bullet$ are neither created, nor moved, nor withdrawn. They stay fixed.

The size of the tiles is $(2,2)$. There are two partitions of origins are $(0,0)$ and $(1,1)$. The partial definitions of e_u given in Fig. 7 is to be completed by symmetries and rotations. e_u is only partially stated because there is no need to describe more to prove universality. Since e_u is injective, it is possible to complete it in an invertible way.

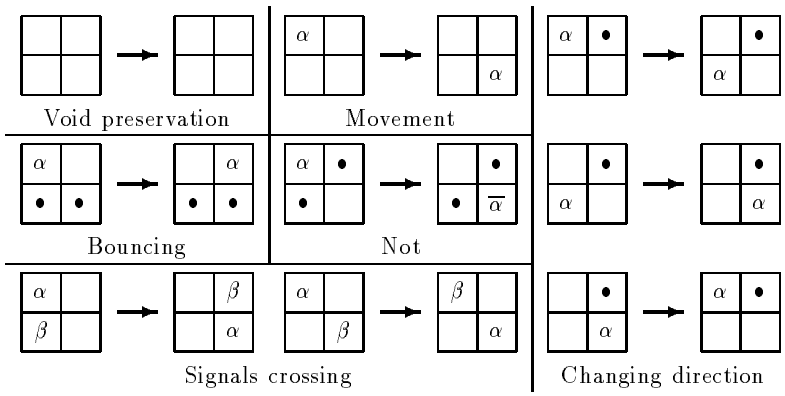

Fig. 7. partial elementary transition function e_u (α, $\beta \in \{\mathbf{0}, \mathbf{1}\}$).

The construction is based on having **0** and **1** signals and 'bumpers' to guide them. The three rules with one bumper (changing direction) are use to avoid back-propagation of signals. The center rule is a simple implementation of a NOT gate.

Signals are encoded by their binary values **0** and **1**. They are traveling diagonally. Their positions in the tiles of the first partition give their directions. In the next figures, the first partition is the one with the thin lines. **0** and **1** have the same behavior. Figure 8 shows signal propagation and deviation.

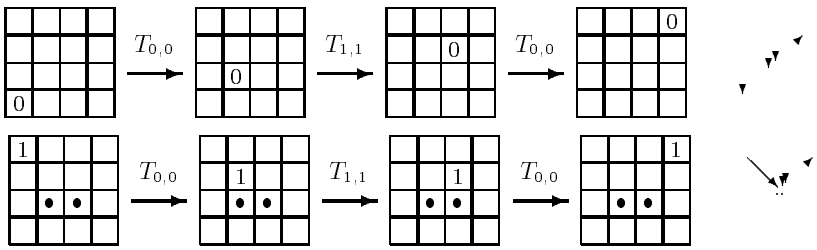

Fig. 8. movement and bouncing.

Figure 9 is a direct application of the rule NOT. It is the more simple gate, it has only 2 bumpers. It can easily be integrated to any logical function.

Figure 10 shows oriented deviations, *i.e.*, going backward, a signal will not go to the direction it came from.

The time for a signal to go through a path is equal to half the length of the path (one step per partition iteration). To obtain a delay the signal travels through a longer path as in Fig. 11 (only the path of the signal is indicated).

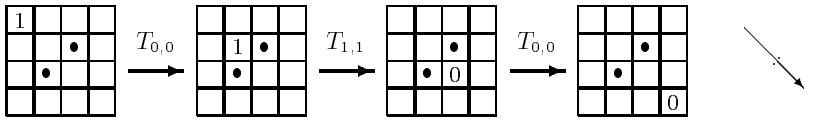

Fig. 9. NOT gate.

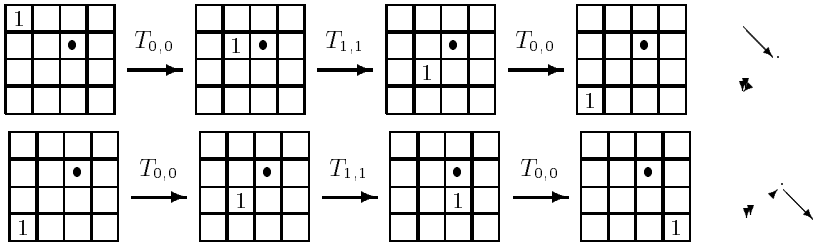

Fig. 10. unreversible direction change.

All of this is not enough to make computations. The two rules of Fig. 12 are added to e_u in order to built Fredkin gate, basis of *conservative logic*. These definitions are not to be completed by rotation and symmetry.

The gate function and implementation are given in Fig. 13. The defined part remains injective and e_u can be completed bijectively. e_u is now assumed totally and bijectively defined. P_u is reversible.

The principles of conservative logic are to use only reversible gates that keeps the numbers of **1** and to forbid duplication of signals in the ether. It was introduced and studied by Fredkin and Toffoli in [2,4,13]. It can be construct out of a single type of logical gate: the Fredkin gate, described in Fig. 13.

Provided constant inputs and garbage outputs are added AND, OR, NOT, artificial signal duplications, memory... can be built. P_u can compute any 'infinite' boolean function and is thus universal like the one described by Serizawa in [11] or Morita in [8].

5 Universality of P_u

5.1 Simulating Reversible PA

Theorem 14. *P_u is able to strongly simulate any reversible PA.*

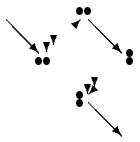

Fig. 11. simple delay.

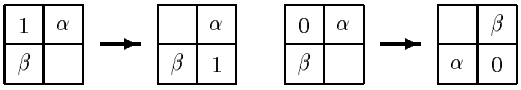

Fig. 12. added rules to built the Fredkin gate.

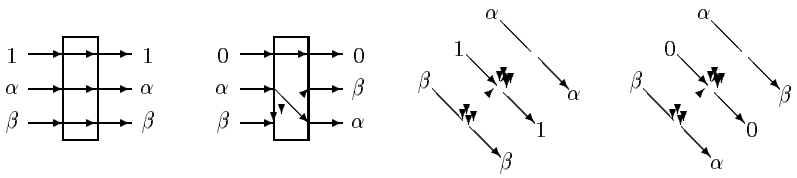

Fig. 13. Fredkin gate and tracing of its implementation ($\alpha, \beta \in \{\mathbf{0}, \mathbf{1}\}$).

Proof. Let $P = \{\mathcal{Q}, (h, v), n, ((x_i, y_i))_{1 \leq i \leq n}, e\}$ be a reversible PA. According to Lemma 8, e is reversible. K. Morita showed in [6] that a the circuit μ that encodes e can be built in conservative logic with neither constant inputs nor garbage outputs (In fact there are **0** constant inputs but here they are recycled since the circuit is to be used periodically). Note that each state is encoded on $|\mathcal{Q}|$ binary signals and only the one corresponding to the value should be **1** (the others should be **0**) to be in conservative logic.

The input (output) tile of this circuit is divided in four blocks corresponding to the intersection with the tiles of the previous (next) partition. The last partition is the previous of the first and *vice-versa*. The tiles are parted as described in Fig. 14.

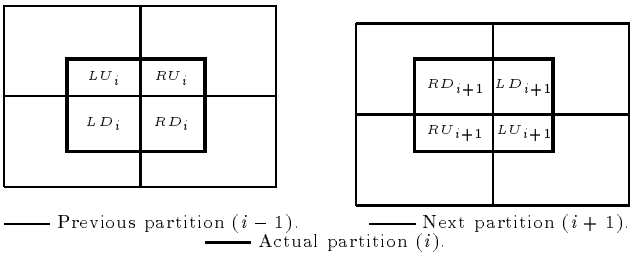

Fig. 14. partition intersection.

Different circuits must be made for each partition because the intersections can differ. Each circuit represents the action of the elementary local function e over one tile of one partition. It gets its inputs from the corresponding outputs of the circuits of the previous tilling and send its outputs to the corresponding inputs of the next circuits. The circuit in Fig. 15 as basis of architecture.

Figure 16 shows circuits and wiring for a two partitions reversible PA. The thick lines are used for wiring from the first to the second tiling and the thin

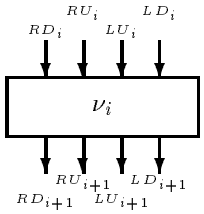

Fig. 15. circuit of the i^{th} partition.

ones for the second to the first. Any number of partitions could be added in this cycle between the last and the first one.

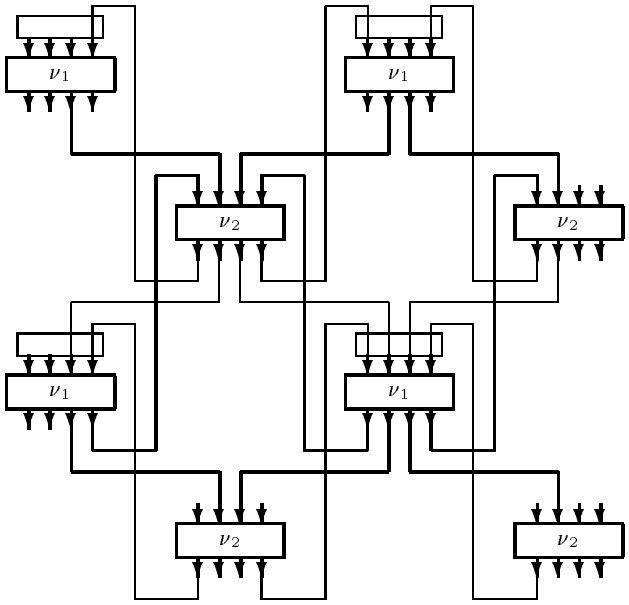

Fig. 16. wiring for reversible PA simulation.

A configuration is encoded at the entry of the circuit of the first tiling, in the thin boxes. □

5.2 Simulating Reversible CA

Since the BBM model developed by Toffoli and Margolus in [14,4] is also able to simulate Fredkin gates and all kinds of movements, it also verifies the following result:

Corollary 15. *P_u is able to strongly simulate any reversible CA.*

Proof. This is a simple application of the Theorem 13 that say that any reversible CA can be strongly simulated by a reversible PA and above Theorem 14 that say that P_u can strongly simulate any reversible PA (strong simulation is a transitive relation). □

5.3 Non Reversible all Simulator PA

Signal creators and absorbers are created by adding a new symbol $\star$ and the rules in Fig. 17. Injectivity is lost adding the absorption rules.

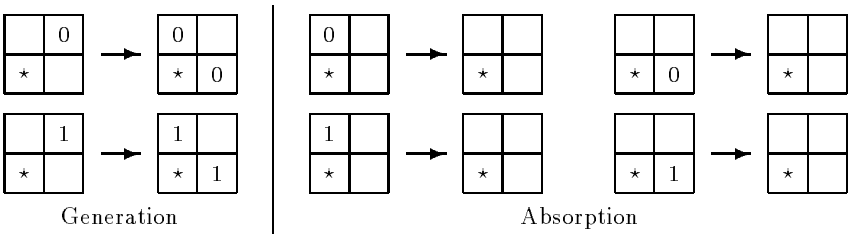

Fig. 17. added rules for e_u^-.

If a signal comes in front of the star, it is doubled and if it comes laterally, it is destroyed. Putting a star on the side of every garbage lines destroys unwanted signals.

Theorem 16. *Any CA can be simulated by the PA* P_u^-.

A path from one a side of a star to the front as in Fig. 18 is enough to make a clock (or signals every k unit of time). The length of the path will be the time between signals. Those clocks just have to be put at the needing entries to provide ongoing endless lines of constants.

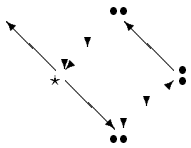

Fig. 18. clock.

The BBM model can also be embedded with a new symbol for creating and erasing signals.

6 Conclusion

Since there are uniform simulations between PA and CA (theorems 11 and 13), the following theorems hold are directly deduced from Corollary 15 and Theorem 16 respectively:

Theorem 17. *There is a reversible CA that is able to simulate strongly any reversible PA and any reversible CA.*

Theorem 18. *There is a unreversible CA able to simulate any PA and any CA.*

With the construction of Sect. 3 and the BBM model, these CA have respectively 16 ($\{\_, \bullet\}^4$) and 81 ($\{\_, \bullet, \star\}^4$) states. Both use Moore's neighborhood (8 closest cells).

These results can easily be translated to d-dimensional PA and CA with $2 \leq d$. Only the results of sections 2 and 3 (reversibility of PA and all simulations between CA and PA and between R-CA and R-CA) and their demonstrations are still valid in dimension 1.

Corollary 19. *There are reversible PA and CA of dimension $d+1$ able to simulate any PA or CA of dimension d.*

Proof. Toffoli proved in [12] that a CA of dimension d can be simulated by one reversible of dimension $d+1$. Since there are CA and PA of dimension $d+1$ able to simulate any reversible of the same dimension. □

Partitioning automata are models of massively parallel architectures as powerful as cellular automata. They are quite simple to design and handle.

It is proved that there are PA (P_u, BBM model) that are reversible and able to simulate any reversible PA or CA and one (P_u^-) that is not reversible but able to simulate any PA or CA. There are also CA that verify those properties.

Now the problem is: what about the other PA and CA of the same dimension? One thing is sure, if any reversible can simulate any PA or CA then P_u and the BBM also can by transitivity.

Constants can be set such that P_u make a single transition of any PA or CA, but what happens with unbounded iteration?

References

1. J. O. Durand-Lose. Partitioning automata, cellular automata, simulation and reversibility. Technical Report 95-01, LIP, ENS Lyon, 46 allée d'Italie, 69 364 LYON 7, 1995.
2. E. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21, 3/4:219–253, 1982.
3. J. Kari. Reversibility of 2D cellular automata is undecidable. *Physica D*, 45:379–385, 1990.
4. N. Margolus. *Physics and Computation.* PhD thesis, MIT, 1988.

5. E. Moore. Machine models of self-reproduction. In *Proceeding of Symposium on Applied Mathematics*, volume 14, pages 17–33, 1962.
6. K. Morita. A simple construction method of a reversible finite automaton out of Fredkin gates, and its related problem. *Transactions of the* IEICE, E 73(6):978–984, June 1990.
7. K. Morita. Computation-universality of one-dimensional one-way reversible cellular automata. *Information Processing Letters*, 42:325–329, 1992.
8. K. Morita and S. Ueno. Computation-universal models of two-dimensional 16-state reversible automata. IEICE *Transactions on Informations and Systems*, E75-D(1):141–147, January 1992.
9. 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.
10. D. Richardson. Tessellations with local transformations. *Journal of Computer and System Sciences*, 6:373–388, 1972.
11. T. Serizawa. Three-state Neumann neighbor cellular automata capable of constructing self-reproducing machines. *Systems and Computers in Japan*, 18(4):33–40, 1987.
12. T. Toffoli. Computation and construction universality of reversible cellular automata. *Journal of Computer and System Sciences*, 15:213–231, 1977.
13. T. Toffoli. Reversible computing. Technical Report MIT/LCS/TM-151, MIT Laboratory for Computer Science, 1980.
14. T. Toffoli and N. Margolus. *Cellular Automata Machine - A New Environment for Modeling.* MIT press, Cambridge, MA, 1987.
15. T. Toffoli and N. Margolus. Invertible cellular automata: A review. *Physica D*, 45:229–253, 1990.
16. S. Wolfram. *Theory and Applications of Cellular Automata.* World Scientific, 1986.