Discrete Mathematics and Theoretical Computer Science Proceedings **AA (DM-CCG)**, 2001, 145–154

Representing Reversible Cellular Automata with Reversible Block Cellular Automata

Jérôme Durand-Lose

Laboratoire ISSS, Bât. ESSI, BP 145, 06 903 Sophia Antipolis Cedex, France
e-mail: jdurand@unice.fr http://www.i3s.unice.fr/~jdurand

received February 4, 2001, *revised April 20, 2001*, *accepted May 4, 2001*.

Cellular automata are mappings over infinite lattices such that each cell is updated according to the states around it and a unique local function. Block permutations are mappings that generalize a given permutation of blocks (finite arrays of fixed size) to a given partition of the lattice in blocks. We prove that any d-dimensional reversible cellular automaton can be expressed as the composition of $d+1$ block permutations. We built a simulation in linear time of reversible cellular automata by reversible block cellular automata (also known as partitioning CA and CA with the Margolus neighborhood) which is valid for both finite and infinite configurations. This proves a 1990 conjecture by Toffoli and Margolus (*Physica D* 45) improved by Kari in 1996 (*Mathematical System Theory* 29).

Keywords: Cellular automata, reversibility, block cellular automata, partitioning cellular automata.

---

1 Introduction

Cellular automata (CA) provide the most common model for parallel phenomena, computations and architectures. They operate as iterative systems on d-dimensional infinite arrays of *cells* (the underlying space is $\mathbb{Z}^d$). Each cell takes a value from a *finite set of states* $\mathcal{S}$. An iteration of a CA is the synchronous replacement of the state of each cell by the image of the states of the cells around it according to a unique *local function*. The same local function is used for every cell.

A *block* is a (finite) d-dimensional array of states. A *block permutation* (BP) is a generalization of a permutation of blocks, over a regular partition of the lattice $\mathbb{Z}^d$ into blocks. A *reversible block cellular automaton* is the composition of various BP which use the same permutation of blocks e. If e is just a mapping –not necessary a permutation– we speak of *block cellular automaton*. Block cellular automata (BCA) are also known as "CA with the Margolus neighborhood" or "partitioning cellular automata". Let us notice that BCA are not partition*ed* CA as introduced by Morita [Mor95].

Reversible cellular automata (R-CA) are famous for modeling non-dissipative systems as well as for being able to backtrack a phenomenon to its source. Reversibility is also conceived as a way to reduce heating and save energy. We refer the reader to the 1990 article by Toffoli and Margolus [TM90] for a wide survey of the R-CA field (history, aims, uses, decidability…) and a large bibliography (even though it is quite old). In this paper, the authors made the following conjecture about R-CA:

Conjecture 1 [TM90, Conjecture 8.1] *All invertible cellular automata are structurally invertible, i.e., can be (isomorphically) expressed in spacetime as a uniform composition of finite logic primitives.*

A "finite logic primitives" is a representation of a permutation of blocks e. Kari [Kar96] proved Conj. 1 for dimensions 1 and 2. At the end, he conjectures that:

Conjecture 2 [Kar96, Conjecture 5.3] *For every $d \geq 1$, all reversible d-dimensional cellular automata are compositions of block permutations and partial shifts.*

In our definition, the shift is included in the definition of the block permutation.

It should be noted that a reversible CA are quite tricky to design while reversibility is very simple to achieve with BCA. Moreover, reversibility of CA is undecidable in dimension greater than 1 [Kar94] whereas it can be checked easily for BCA. This does not contradict the conjecture since only invertible / reversibility CA are concerned.

In [DL95], Conj. 1 is proved for any dimension d with $2^{d+1}-1$ BP of width $4r$, where r is the greater of the radii of the neighborhood of the CA and of its inverse. Here, Conj. 1 is proved to be true for any dimension d with a lower number of permutations, $d+1$, than proposed by Kari (exponential in d). The construction presented here is done by progressively erasing previous states and adding next states. The $d+1$ partitions are not regularly displayed, their origins are aligned; one goes from a partition to the next one by a constant translation of $(3r, 3r, \dots 3r)$.

All definitions and proofs in this paper can be read without any previous knowledge of the subject. The paper is structured as follows. The definitions of cellular automata, block partitions, reversibility and simulation are given in Section 2. In Section 3, for any reversible CA A, we exhibit $d+1$ BP such that the global function of A corresponds to their composition.

During the simulation, some cells have to store their previous and next states. This is achieved by embedding the states in $(\mathcal{S} \cup \{\perp\})^2$ ($\mathcal{S}$ is the set of states of the simulated R-CA). All permutations of blocks are compatible, *i.e.*, the useful parts of their domains and ranges do not overlap.

In Sect. 4, we collapse the set of states of the BP from $(\mathcal{S} \cup \{\perp\})^2$ to $\mathcal{S} \cup \mathcal{S}^2$ and prove that all the local functions of the BP are compatible. This allows to iterate the composition of the BP and to built the corresponding BCA. This yields a simulation where encoding and decoding functions are identities, what we call a representation.

It should be noted that Kari also used $\mathcal{S} \cup \mathcal{S}^2$ as the set of states for intermediate configurations.

2 Definitions

Let d be a natural number. Cellular automata and block permutations define mappings over d-dimensional infinite arrays over a finite set of *states* $\mathcal{S}$. Let $\mathcal{C} = \mathcal{S}^{\mathbb{Z}^d}$ be the set of all configurations. For any configuration c and any subset E of $\mathbb{Z}^d$, $c_{|E}$ is the restriction of c to E.

The set of integers $\{i, i+1, i+2, \dots j\}$ is denoted $[\![i, j]\!]$. For any $x \in \mathbb{Z}^d$, σ_x is the shift by x (*i.e.* $\forall c \in \mathcal{C}, \forall i \in \mathbb{Z}^d, (\sigma_x(c))_i = c_{i+x}$).

2.1 Cellular automaton

A *cellular automaton* (CA) A is defined by $(d, \mathcal{S}, r, f)$, where the *radius* r is a natural number and the *local function* f maps $\mathcal{S}^{(2r+1)^d}$ into $\mathcal{S}$. The *global function* of A $\mathcal{G}_A$ maps configurations into configurations as follows:

$$\forall c \in \mathcal{C},\ \forall i \in \mathbb{Z}^d,\quad (\mathcal{G}_A(c))_i \;=\; f(c_{|i+[\![-r,r]\!]^d}) \enspace .$$

The next state of a cell depends only on the states of the cells which are at distance at most r. All cells are updated simultaneously and no global variable is used.

2.2 Block permutation

A *block permutation* (BP) is defined by $(d, \mathcal{S}, w, o, e)$ where the *width* w is a natural number. We define the *volume* V to be the following subset of $\mathbb{Z}^d$: $V = [\![0, w-1]\!]^d$. The coordinate o belongs to V. We call *block* a mapping from V to $\mathcal{S}$, or, equivalently, an array of states whose underlying lattice is V. The block function e is a permutation of the blocks, $e : \mathcal{S}^V \to \mathcal{S}^V$.

The *block permutation* T is the following mapping over $\mathcal{C}$: for any $c \in \mathcal{C}$, for any $i \in \mathbb{Z}^d$, let $a = i \operatorname{div} w$ and $b = i \bmod w$ ($a \in \mathbb{Z}^d$ and $b \in [\![0, w-1]\!]$) so that $i = a\,.\,w + b$, then $T(c)_i = e(c_{|a\,.\,w+V})_b$. In other words, the configuration is partitioned into regularly displayed blocks, then each block is replaced by its image by the permutation of blocks e as in Fig. 1.

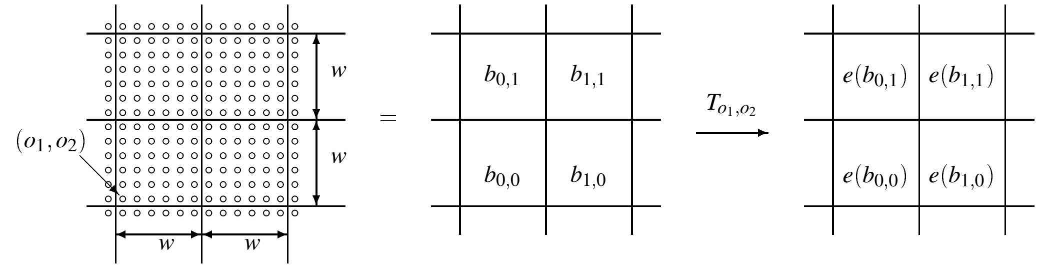

Fig. 1: T_{o_1,o_2}, the block permutation of width w and origin (o_1, o_2).

The *block permutation of origin* o, T_o is $\sigma_o \circ T \circ \sigma_{-o}$. It is the same as before but the partition is shifted by o. An example of a two-dimensional block permutation is given in Fig. 1.

We call *reversible block cellular automaton* (R-BCA) the composition of various BP with the same volume w and permutation e. If e is a mapping –not necessarily a permutation– of blocks, the composition is referred as just a block cellular automaton (BCA). The reversible term is explained in the next subsection.

2.3 Reversibility

Both cellular automata, block CA and block permutations define synchronous and massively parallel mappings $\mathcal{G}$ over $\mathcal{S}^{\mathbb{Z}^d}$.

An automaton A is *reversible* (or *invertible*) if and only if $\mathcal{G}_A$ is bijective and there exists another automaton B such that $\mathcal{G}_B = \mathcal{G}_A^{-1}$. The automaton B is called the *inverse* of A. Reversible CA are denoted R-CA.

Amoroso and Patt [AP72] provided an algorithm to check whether a 1-dimensional cellular automaton is reversible or not. Kari [Kar94] proved that the reversibility of CA is undecidable in greater dimensions.

By construction, BP are reversible; one simply uses the inverse permutation on the same partition to get the inverse block partition. By composition, it is easy to prove that a block CA is reversible if and only if its block function e is a permutation.

2.4 Simulation

Since we are dealing with iterative systems, we use the following definition.

For any two functions $f : F \to F$ and $g : G \to G$, we say that *g simulates f in linear time* τ if there exist two encoding functions $\alpha : F \to G$ and $\beta : G \to F$, space and time inexpensive compared to f and g, such that:

$$\forall x \in F,\ \forall n \in \mathbb{N}, \qquad f^n(x) \ = \ \beta \circ g^{\tau n} \circ \alpha(x) \ . \tag{1}$$

This corresponds to the commuting diagram of Fig. 2. The function g can be used instead of f for iterating.

$$\forall n \in \mathbb{N},\ 0 \leq n, \qquad \begin{tikzcd} F \arrow[r, "f^n"] \arrow[d, "\alpha"'] & F \\ G \arrow[r, "g^{\tau n}"] & G \arrow[u, "\beta"'] \end{tikzcd}$$

Fig. 2: g simulates f in linear time τ.

For $n = 0$, we get $\forall x, \beta \circ \alpha(x) = x$. If both f and g are invertible and g simulates f, by the uniqueness of predecessors, (1) still holds for n negative.

An automaton simulates another if and only if its global function simulates the global function of the other. We speak of a *representation* if F is included in G, α is the identity and β is undefined out of F. The domain of g does not have to be included in F, but $g^\tau(F)$ must be included in F.

If factor τ is 1 then the simulation is in *real time*.

3 Simulation of a R-CA by a composition of BP

Let $A = (\mathcal{S}, d, r, f)$ be a reversible cellular automaton. In this section, we prove that it can be simulated by the composition of $d+1$ block permutations.

The inverse CA of A, A^{-1}, can be effectively computed: it is easy to built the composition of two CA and check whether it is the identity. Since there are countably many CA, it is possible to test every CA until the inverse is found. The algorithm stops in finite time; but in any dimension greater than one its complexity cannot be bounded by any computable function because of the undecidability of reversibility.

We consider that the radius r of A is large enough for both A and A^{-1}. Let $w = 3r(d+1)$ be the common width of all the BP.

3.1 Block partitions

The following sets are finite sub-arrays of $\mathbb{Z}^d$. They are used to locate previous and next states during the iterations. We denote $\overline{r}$ the vector $(r, r, \ldots r)$ of $\mathbb{Z}^d$. For every λ in $[\![0, d+1]\!]$, let

$$\begin{aligned} E^{\mathsf{P}}_\lambda &= \bigcup_{\lambda \leq \mu < d+1} \left(3\mu\overline{r} + [\![r, 3(d+1)r - r - 1]\!]^d \right) \quad \textit{and} \\ E^{\mathsf{N}}_\lambda &= \bigcup_{0 \leq \mu < \lambda} \left(3\mu\overline{r} + [\![r, 3(d+1)r - r - 1]\!]^d \right) \ . \end{aligned}$$

These sets are supposed to be completed by and close under all $\pm 3(d{+}1)r$ shifts in every direction. This will never be indicated to ease the presentation.

Lemma 3 *These sets verify the symmetry $E^{\mathsf{N}}_{\lambda} = -3rd - E^{\mathsf{P}}_{d+1-\lambda}$ for $0 \leq \lambda \leq d{+}1$; and the equalities: $E^{\mathsf{P}}_{d+1} = E^{\mathsf{N}}_{0} = \emptyset$ and $E^{\mathsf{P}}_{0} = E^{\mathsf{N}}_{d+1} = \mathbb{Z}^d$.*

Proof The symmetry and the equality with $\emptyset$ are obvious.
Let us prove that $E^{\mathsf{N}}_{d+1} = \mathbb{Z}^d$, the last equality follows by symmetry. Let x be any element of the underlying lattice $\mathbb{Z}^d$. The $d{+}1$ sets (of $\mathbb{Z}$) $3\lambda r + [\![-r, r{-}1]\!]$ (for $\lambda \in [\![0, d]\!]$) are non-empty and disjoint. Since x has d coordinates, there exists λ_0 such that none of the coordinates of x belongs to $3\lambda_0 r + [\![-r, r{-}1]\!]$. This means that all the coordinates of x belong to $3\lambda_0 r + [\![r, 3(d{+}1)r{-}r{-}1]\!]$ –remember that E^{N}_{λ} is closed by $3(d{+}1)r$ shifts– thus $x \in E^{\mathsf{N}}_{d+1}$. ∎

The BP use the set of states $(\mathcal{S} \cup \{\perp\})^2$ to store the previous configuration in the first component and the next one in the second component. The state $\perp$ stand for the missing parts.

Let us define the following configurations:

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

The states are completed by $\perp$ everywhere they are not in of the restrictions; $\perp$ also denotes the configuration where all cells are in state $\perp$. Lemma 3 implies that: $\mathcal{E}_0(c) = (c, \perp)$ and $\mathcal{E}_{d+1}(c) = (\perp, \mathcal{G}(c))$.

Let B_λ be a BP of width $3(d{+}1)r$ and origin $3\lambda r$. The width of B_λ matches the length of the shift closure of the sets E^{P}_{λ} and E^{N}_{λ}. The partition of blocks e_λ is defined so that B_λ maps reversibly $\mathcal{E}_\lambda(c)$ into $\mathcal{E}_{\lambda+1}(c)$. The aim is to get the commuting diagram of Fig. 3.

$$\begin{tikzcd}
c \arrow[rrrrr, "\mathcal{G}_A"] \arrow[d] & & & & & \mathcal{G}_A(c) \\
(c,\perp) = \mathcal{E}_0(c) \arrow[r, "B_0"] & \mathcal{E}_1(c) \arrow[r, "B_1"] & \mathcal{E}_2(c) \arrow[r, dotted] & \mathcal{E}_d(c) \arrow[r, "B_d"] & \mathcal{E}_{d+1}(c) = (\perp, \mathcal{G}_A(c)) \arrow[ur]
\end{tikzcd}$$

Fig. 3: Simulation commuting diagram.

The BP B_λ add next states and erase previous states. Let us prove that there is enough data in $\mathcal{E}_\lambda(c)$ to compute the added next states, and then that the erasing of previous states is done reversibly.

Lemma 4 *For any λ in $[\![0, d]\!]$, there is enough information in $\mathcal{E}_\lambda(c)$ to compute $\mathcal{E}_{\lambda+1}(c)$.*

Proof The next states added belong to:

$$\Delta_\lambda \ = \ E^{\mathsf{N}}_{\lambda+1} \setminus E^{\mathsf{N}}_{\lambda} \ = \ \left(3\lambda\overline{r} + [\![r, 3(d{+}1)r{-}r{-}1]\!]^d \right) \ \setminus \bigcup_{0 \leq \mu < \lambda} \left(3\mu\overline{r} + [\![r, 3(d{+}1)r{-}r{-}1]\!]^d \right) \ .$$

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

For any μ in $[\![0, \lambda-1]\!]$, since $x \notin 3\mu\bar{r} + [\![r, 3(d+1)r-r-1]\!]^d$, there is some index j_μ such that $x_{j_\mu} \notin 3\mu r + [\![r, 3(d+1)r-r-1]\!]$. So x_{j_μ} is in $3\mu r + [\![-r, r-1]\!]$ (remember that all is $3(d+1)r$ periodic). Since the sets $3\mu r + [\![-r, r-1]\!]$ are disjoint, all j_μ must be different and there are λ of them.

Let y be any cell needed to compute x, y belongs to $x + [\![-r, r]\!]$, then, for all μ in $[\![0, d-1]\!]$, y_{j_μ} must be in $3\mu r + [\![-2r, 2r-1]\!]$. By contradiction, let us assume that there exists such a y which does not belong to E^{P}_λ then for all $\nu \in [\![\lambda, d+1]\!]$, there exists some k_ν such that y_{k_ν} does not belong to $\nu r + [\![r, 3(d+1)r-r-1]\!]$, or equivalently, $y_{k_\nu} \in 3\nu r + [\![-r, r-1]\!]$. Since the sets $3\nu r + [\![-r, r-1]\!]$ are disjoint, all the k_ν must be different and there are $d+1-\lambda$ of them.

Altogether, there are $d+1$ j_μ and k_ν for d values so there exist μ_0 and ν_0 such that $j_{\mu_0} = k_{\nu_0}$. Then the intersection of $3\mu_0 r + [\![-2r, 2r-1]\!]$ and $3\nu_0 r + [\![-r, r-1]\!]$ is not empty. This means that $\mu_0 = \nu_0$, but by construction, $\mu_0 < \nu_0$.

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

Using the symmetry between E^{N} and E^{P}, the previous states erased can be computed from the next states in $\mathcal{E}_{\lambda+1}(c)$. This means that the corresponding blocks of $\mathcal{E}_\lambda(c)$ and $\mathcal{E}_{\lambda+1}(c)$ in the partition of B_λ can be uniquely determined one from the other. The partial function e_λ can be completed so that e_λ is a permutation.

The two BP for dimension 1 are given in Fig. 4. This corresponds to the construction of Kari [Kar96].

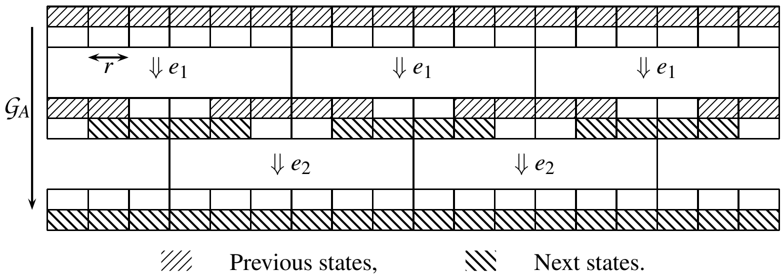

Fig. 4: The two steps in dimension 1.

In the 2-dimensional case, the partitions are given in Fig. 5. The three BP are detailed in Fig. 6. It does not correspond to the construction of Kari any more.

4 Collapsing the states and the permutations

We collapse the states on $\mathcal{S} \cup \mathcal{S}^2$ to make iterations of the simulating R-BCA possible and to get a representation. We also show that the partial definitions of functions e_λ are compatible so that they can be merged into a unique e to define a reversible block cellular automaton.

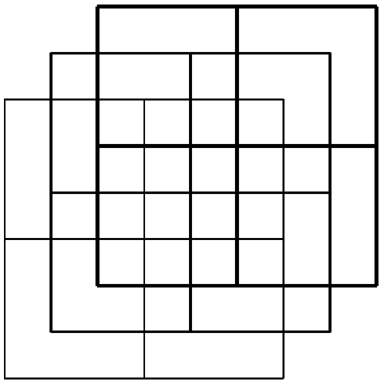

Fig. 5: The three partitions in dimension 2.

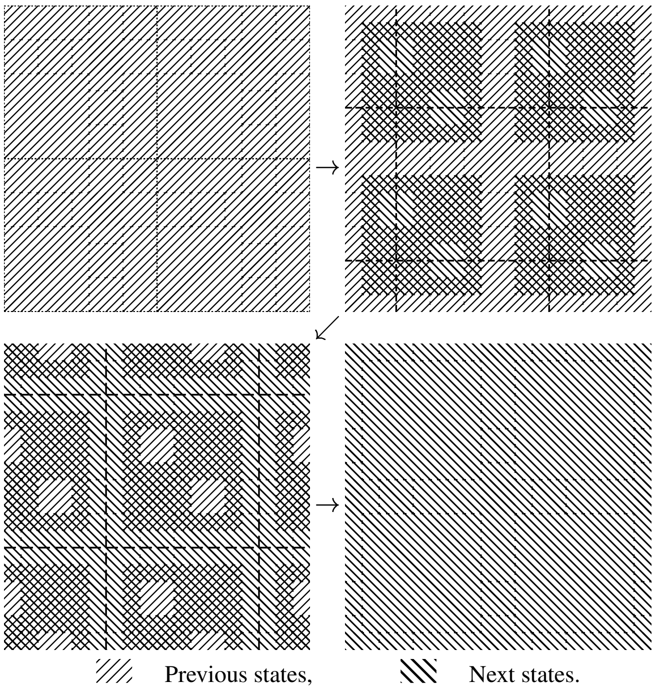

Fig. 6: The 3 steps in dimension 2.

Lemma 5 *The current BP B_λ can be identified by the position of the double states inside the blocks of partition λ.*

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

For i in $[\![0, d]\!]$, let ε_i be the following element of $\mathbb{Z}^d$:

$$\begin{array}{rcl} \varepsilon_0 &=& 3\lambda\overline{r} \ , \\ \varepsilon_1 &=& 3\lambda\overline{r} + (-3r, -3r, -3r, \ldots -3r) \ , \\ \varepsilon_2 &=& 3\lambda\overline{r} + (-3r, -6r, -6r, -6r, \ldots -6r) \ , \\ \varepsilon_i &=& 3\lambda\overline{r} + (-3r, -6r, -9r, \ldots -3(i-1)r, -3ir, \ldots -3ir) \ , \\ \varepsilon_d &=& 3\lambda\overline{r} + (-3r, -6r, -9r, \ldots -3dr) \ . \end{array}$$

Let us prove that the cell at ε_i holds two states only if it belongs to both E^{N}_λ and E^{P}_λ. The equation

$$3\lambda\overline{r} + (-3r, -6r, -9r, \ldots -3(i-1)r, -3ir, \ldots -3ir) \in 3\lambda\overline{r} + [\![r, 3(d+1)r-r-1]\!]^d \subset E^{\mathsf{N}}_\lambda \ \ (\lambda \neq d+1)$$

implies that ε_i belongs to E^{P}_λ for $\lambda \leq d$ –which is always the case. The point ε_i belongs to E^{N}_λ only if $\varepsilon_i \in \bigcup_{0 \leq \mu < \lambda} \left(3\mu\overline{r} + [\![r, 3(d+1)r-r-1]\!]^d\right)$ that is (by Lemma 3 and a shift):

$$(-3r, -6r, \ldots -3(i-1)r, -3ir, \ldots -3ir) \in \bigcup_{-\lambda \leq \mu < 0} \left(3\mu\overline{r} + [\![r, 3(d+1)r-r-1]\!]^d\right) \ .$$

The situation is depicted in Fig. 7. The fact that $-3r$ is a coordinate of ε_i implies that the shifted intervals are not to be considered. Since $-3r$, $-6r$, $-9r$,... and $-3ir$ have to be in the interval, the maximum i such that the equality is true is $\lambda-1$.

Then λ is the maximum i such that ε_i holds two states, plus one. If there is no such i then $\lambda = 0$.

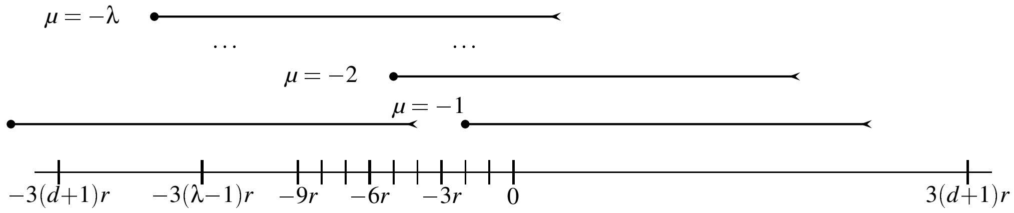

Fig. 7: Graphical representation of the inclusion of ε_i following any direction.

Inside a block of the λ partition, ε_i simplifies to $(-3r, -6r, -9r, \ldots -3(i-1)r, -3ir, \ldots -3ir)$, which is independent of λ. It is enough to know the position of the double states in a block to know which permutation of blocks e_λ to use. ■

Remark 6 *Thanks to the symmetry (Lem. 3), it also holds that the position of double states after the BP indicate which e_λ was used.*

Above Lemma and Remark show that all the partial definitions of the permutations of blocks of the BP are compatible for domains and ranges. They can be grouped in a unique bijective local function and states can be collapsed. Altogether:

Theorem 7 *In any dimension d, any R-CA of radius r can be expressed as a composition of* $d+1$ *BP of width* $3(d+1)r$ *and with the same permutation of blocks e, or as a R-BCA with* $d+1$ *partitions.*

The origins of the partitions are: 0, $3r$, $6r$, $9r$... and $3dr$.

5 Conclusion

Conjectures 1 and 2 are true. It should be noted that even if states in $\mathcal{S}^2$ are used in intermediate configurations during the simulation, it perfectly works since the input and output are restricted to $\mathcal{S}$. Since the BP are compatible and the states are collapsed, the composition can be iterated directly. This defines a reversible block cellular automaton (with $d+1$ BP) which simulates the R-CA in real time.

The fact that the BCA representation can be effectively constructed does not contradict the undecidability of reversibility of CA because the inverse CA is needed for the construction.

The proof of Th. 7 is not as explicit and visible as the one in [DL95]. Nevertheless, the number of BP needed is lowered from $2^{d+1}-1$ to $d+1$. Generation and erasement are done concurrently, not one after the other as in [DL95].

We believe that it is not possible to make a representation with less that $d+1$ BP.

Compare to [DL95], the main drawback is that the volume of the blocks is $(3r(d+1))^d$ instead of $(4r)^d$. The complexity of a BP (or a BCA) is the size of the table of its local function e. It should be noted that if the number of BP is decreasing, the complexity is increasing.

The expression with BP allows one to use reversible circuitry in order to build R-CA. This was done in [DL95] to prove that, for $2 \leq d$, there exists d-dimensional R-CA (based on the the Billiard ball model) able to simulate any d-dimensional R-CA in linear time on infinite configurations. This result was extended in [DL97] to the first dimension.

For more about the relation between R-CA and BCA, we refer to the 1999 article by Kari [Kar99] which refers to an unpublished earlier version of the present article.

References

[AP72] 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.

[DL95] J. Durand-Lose. Reversible cellular automaton able to simulate any other reversible one using partitioning automata. In *LATIN '95*, number 911 in LNCS, pages 230–244. Springer-Verlag, 1995.

[DL97] J. Durand-Lose. Intrinsic universality of a 1-dimensional reversible cellular automaton. In *STACS '97*, number 1200 in LNCS, pages 439–450. Springer-Verlag, 1997.

[Kar94] J. Kari. Reversibility and surjectivity problems of cellular automata. *Journal of Computer and System Sciences*, 48(1):149–182, 1994.

[Kar96] J. Kari. Representation of reversible cellular automata with block permutations. *Mathematical System Theory*, 29:47–61, 1996.

[Kar99] J. Kari. On the circuit depth of structurally reversible cellular automata. *Fundamenta Informaticae*, 38(1–2):93–107, 1999.

[Mor95] K. Morita. Reversible simulation of one-dimensional irreversible cellular automata. *Theoretical Computer Science*, 148:157–163, 1995.

[TM90] T. Toffoli and N. Margolus. Invertible cellular automata: A review. *Physica D*, 45:229–253, 1990.