TRANSLATING PARTITIONED CELLULAR AUTOMATA INTO CLASSICAL TYPE CELLULAR AUTOMATA

VICTOR POUPET

Laboratoire d'Informatique Fondamentale (LIF), UMR 6166 CNRS, Université de Provence
CMI, 39 rue Joliot-Curie, 13453 Marseille Cedex 13
E-mail address: victor.poupet@lif.univ-mrs.fr
URL: http://www.lif.univ-mrs.fr/ vpoupet/

Abstract. Partitioned cellular automata are a variant of cellular automata that was defined in order to make it very simple to create complex automata having strong properties such as number conservation and reversibility (which are often difficult to obtain on cellular automata). In this article we show how a partitioned cellular automaton can be translated into a regular cellular automaton in such a way that these properties are conserved.

1. Introduction

A number-conserving cellular automaton is a cellular automaton such that all states of cells are represented by integers and that the sum of all states in any finite configuration (the quiescent state corresponds to the value 0) is unchanged after one transition of the automaton. Number-conservation can be seen as a modelization of the physical law of conservation of mass and energy. Thus number conserving cellular automata can be used to model complex physical phenomena, like fluid dynamics [1] or highway traffic flow [6].

However, it appears that designing number-conserving cellular automata with complex transition rules is very difficult. To solve this problem a new kind of cellular automata has been studied by Morita et al. [3] that use a *partitioned* space. It is then possible to construct complex two-dimensional number-conserving (and reversible) PCA, like for example some that simulate any reversible two-counter machine [4].

Although in a number-conserving partitioned cellular automaton the total weight of a configuration is conserved after each transition, each cell has multiple parts and cannot therefore be regarded as a usual type of CA. In this paper we will show how it is possible to translate any number-conserving partitioned cellular automaton into a classical type number-conserving cellular automaton and thus prove the equivalence of the two models.

1.1. Definitions

Definition 1.1 (Cellular Automaton)**.** A *cellular automaton* (CA) is a quadruple $\mathcal{A} = (d, \mathcal{Q}, V, f)$ where

- $d \in \mathbb{N}$ is the dimension of the automaton;
- $\mathcal{Q}$ is a finite set called *set of states*;
- $V = \{v_1, \ldots, V_{|V|}\} \subseteq \mathbb{Z}^d$ is a finite set called *neighborhood*;
- $f : \mathcal{Q}^{|V|} \to \mathcal{Q}$ is the *local transition function*.

Key words and phrases: Partitioned cellular automata, number-conservation, reversibility.

Submitted to JAC (Symposium on Cellular Automata Journées Automates Cellulaires)

For a given automaton $\mathcal{A}$, we call *configuration* of $\mathcal{A}$ any function $\mathfrak{C}$ from $\mathbb{Z}^d$ into $\mathcal{Q}$. The set of all configurations is therefore $\mathcal{Q}^{\mathbb{Z}^2}$. From the local function f we can define a global function F

$$\begin{array}{rcccl} F & : & \mathcal{Q}^{\mathbb{Z}^d} & \to & \mathcal{Q}^{\mathbb{Z}^d} \\ & & \mathfrak{C} & \mapsto & \mathfrak{C}' \quad | \quad \forall x \in \mathbb{Z}^d, \mathfrak{C}'(x) = f(\mathfrak{C}(x+v_1), \ldots, \mathfrak{C}(x+v_{|V|})) \end{array}$$

Elements of $\mathbb{Z}^d$ are called *cells*. Given a configuration $\mathfrak{C}$, we will say that a cell c is in state q if $\mathfrak{C}(c) = q$. If we distinguish a *quiescent* state q_0 such that $f(q_0, \ldots, q_0) = q_0$, we will call *finite* configuration any configuration for which only a finite number of cells is not in the quiescent state. If $\mathfrak{C}$ is a finite configuration, so is $F(\mathfrak{C})$.

Cellular automata will be seen as dynamical systems. If the CA is in the configuration $\mathfrak{C}$ at some time, we will say that it is in the configuration $F(\mathfrak{C})$ at the next time. We can therefore define the *evolution* of a CA from a configuration. This evolution is completely determined by $\mathfrak{C}$.

Remark: In the following, we will only consider two-dimensional CA ($d = 2$).

Definition 1.2 (Number-Conservation)**.** A CA will be said to be *number-conserving* if its states are distinct natural numbers ($\mathcal{Q} \subseteq \mathbb{N}$), the state 0 is quiescent, and for every finite configuration $\mathfrak{C}$ the total weight of $\mathfrak{C}$ (sum of all states) is equal to that of $F(\mathfrak{C})$.

Definition 1.3 (Partitioned Cellular Automaton)**.** A two-dimensional partitioned cellular automaton (PCA) is a four-neighbor two-dimensional CA whose cells are divided into four parts : upper, left, lower and right. The next state of each cell is only determined by the current states of the upper partition of the lower cell, the right partition of the left cell, the lower partition of the upper cell and the left partition of the right cell.

Let us denote by $\mathcal{Q}_p$ the set of all states that a partition of a cell can be into. Then there are $\mathcal{Q}_t = |\mathcal{Q}_p|^4$ different states for each cell. However, the local rules of the automaton can be seen as a function of $\mathcal{Q}_p^4 \to \mathcal{Q}_p^4$. As illustrated by Figure 1. $\mathcal{Q}_p$ will be called the *set of partitioned states* of the PCA.

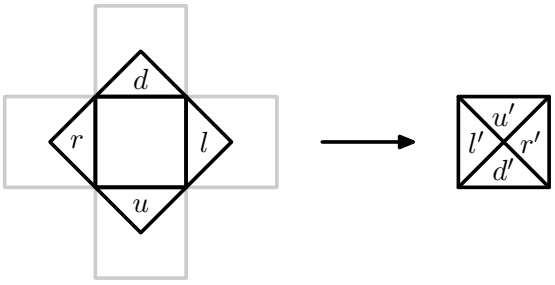

Figure 1: Illustration of the rule $[d, l, u, r] \to [u', r', d', l']$.

It is very easy to see that for PCA global reversibility is equivalent to local reversibility. It is therefore very easy to design reversible automata using this notion.

Note. There are other kinds of PCA. We could for example divide each cell into 5 parts by adding a central partition. It is also possible to consider PCA in other dimensions. However in this paper we will only work with two-dimensional 4-partitioned PCA.

If we want to consider number-conservation for PCA we have to redefine it. In this case, only the partitioned states will be integers. The total weight of a cell is the sum of the four states in its partitions.

The weight of a finite configuration is the sum of all weights of its cells (only a finite number of cells are in a positive state). The automaton is said to be globally number-conservative if for all finite configuration its total weight is conserved after one transition and locally number-conservative if for each rule $(d, l, u, r) \to (u', r', d', l')$ we have $d + l + u + r = u' + r' + d' + l'$.

It is very easy to see that both local and global number-conservation are equivalent for a PCA and hence it is very easy to design a number conservative PCA even when we want to have it perform a complex task.

1.2. Why Translating into Classical Type CA?

We have seen that the notion of PCA is very useful when working with number-conserving or reversible automata as they are very easy to design and that checking both number-conservation and reversibility can be done by a local study of each rule (therefore very simply and quickly).

However the definition of PCA slightly differs from classical CA. Of course it is possible to consider that a PCA is a classical CA and that it has Q_p^4 different states. However, by doing so we see that what we called a number-conserving PCA is not necessarily a number conserving CA because we gave the same weight to different states (for example the states $(1,0,0,0)$ and $(0,0,0,1)$ have the same weight). In other words, even if the PCA are a subclass of CA, the definition we gave of number-conservation in this specific class (which is the definition that makes it easy to design number conserving PCA) is not the same as the usual number-conservation.

Here we will show how a NCPCA can be translated into a classical NCCA without increasing the number of states.

2. Main Theorem

The rest of the article will be devoted to proving the following theorem:

Theorem 2.1. *Given a PCA $\mathcal{A}_p$ of partitioned states $\mathcal{Q}$, there exists a CA $\mathcal{A}_r$ of states $\mathcal{Q}$ working on the neighborhood V_m^{10} such that $\mathcal{A}_r$ mimics the behavior of $\mathcal{A}_p$ (in a very natural sense that we will explain later). Moreover, if ACA_p is number conserving or reversible (or both) then so is $\mathcal{A}_r$.*

In the following, we will consider a PCA of partitioned states $\mathcal{Q}$ and describe how the regular CA described in the theorem works.

3. Preliminaries

3.1. Main Idea

First, we will explain how to convert configurations of $\mathcal{A}_p$ into configurations of $\mathcal{A}_r$ in such a way that no information is lost so that from such an image configuration the CA $\mathcal{A}_r$ can easily mimic the behavior of $\mathcal{A}_p$. Then we will show how we can ensure that $\mathcal{A}_r$ has the same properties (number-conservation and reversibility) than $\mathcal{A}_p$. These properties will be trivially conserved on valid configurations (configurations that are images of configurations of $\mathcal{A}_p$ by the transformation mentioned above) but not necessarily on invalid ones. In this case we will show that the CA $\mathcal{A}_r$ can detect the irregularities and "freeze" its behavior so that nothing happens that can break number-conservation or reversibility: after all, the identity CA is both conservative and reversible.

3.2. Vocabulary

To avoid confusions between the original PCA and the new CA we will clearly distinguish the vocabulary between the two.

From now on we will call *square cell* a cell of the original PCA. A partition will be one portion of a square cell (a square cell has 4 partitions which are naturally the upper, right, lower and left partitions).

Most of the time, the word *cell* will refer to a cell of the new CA. This automaton works as classical CA do, but on a Moore neighborhood of radius 10 (we will see later why such a neighborhood is needed).

We will call *states* the elements of $\mathcal{Q}$ (the partitioned states of $\mathcal{A}_p$ and the states of the cells in $\mathcal{A}_r$).

The state 0 will be called *blank* state. All other states will be *colored* states. The blank state is assumed to be quiescent. A blank cell is a cell whose state is 0, and a colored cell is a cell whose state is not 0.

If we consider a given partition p in the PCA, there are some partitions that play a particular role from its point of view. The first of these particular partitions is the one that is in front of it (the lower partition of the upper square cell in the case of an upper partition for example). This partition will be called p's *facing* partition. The three other partitions in the same square cell as p will be called p's *brother* partitions.

Later on, we will establish a parallelism between certain patterns of the new CA (called *blocks*) and the partitions of the original one, which will lead to the use of the terms *facing blocks* and *brother blocks* whose meaning will be straightforward.

3.3. Switch and Mix: the Key to Understanding PCA

There is an important property in the way PCA local rules are defined. As shown by Figure 1, the transition is such that four partitions (the *outter* ones shown on the left part of the figure) entirely define the next states in four other partitions (the *inner* ones). But there is more than meets the eye...

Indeed, transitions of a PCA can be split in two *virtual* steps that are purely local transformations: the *switch* and the *mix*. During the switch, all partitions exchange their state with their facing partition. Once this is done, all brother partitions in a square cell "mix" their states to produce the result of the local transition rule.

These two steps are virtual because they happen both in one single step of the automaton. It might seem useless to consider an intermediate step, but it gives the automaton an important property: pure locality. Both steps can now be expressed as local transformations of some parts of the configuration: the switch transforms the pairs of facing partitions whereas the mix transforms the four brother cells inside of a square cell, and for both of these transformations the *affecting* area is the same as the *affected* area[1].

This property is reminiscent of the kind of cellular automata considered by N. Margolus to build small Turing-universal machines [2].

3.4. From the Configurations of $\mathcal{A}_p$ to Those of $\mathcal{A}_r$

Starting from a configuration $\mathfrak{C}$ of $\mathcal{A}_p$, we obtain the configuration $\tau(\mathfrak{C})$ of $\mathcal{A}_r$ by transforming each square-cell into a 4×4 pattern of cells of $\mathcal{A}_r$ as illustrated by Figures 2 and 3.

We will call *valid* a configuration of $\mathcal{A}_r$ that is the image by τ of some configuration of $\mathcal{A}_p$ and *invalid* a configuration that is not.

[1] This property is reminiscent of the kind of cellular automata considered by N. Margolus to build small Turing-universal machines [2]. In fact PCA are highly related to the Margolus model as one kind can easily be translated into the other by simple geometric operations on the configurations and local transition rules.

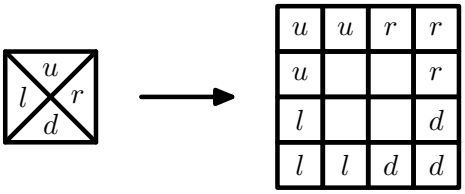

Figure 2: How to translate a square cell into a portion of configuration of $\mathcal{A}_r$.

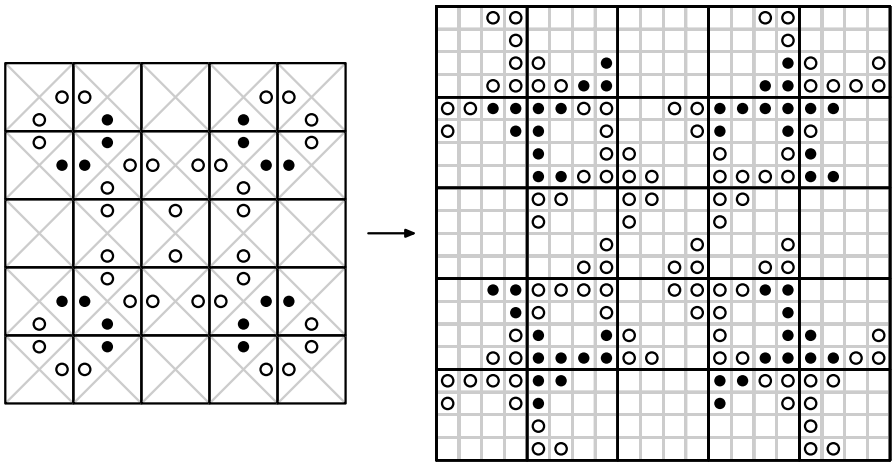

Figure 3: Example of conversion of a configuration of a PCA into a configuration of our new automaton. The original PCA has 3 different states : white, black and blank.

4. Evolution of $\mathcal{A}_r$

The evolution of a cell in $\mathcal{A}_r$ will happen in 3 steps. The cell will always start assuming that it is in a valid configuration and try to apply the rule of $\mathcal{A}_p$ on this valid configuration. However, if it finds that it is not in a valid configuration it will stop its behavior to make sure that no irreversible or non-conserving action is made (the behavior of $\mathcal{A}_p$ needs not be mimicked if the configuration is invalid).

During the first step the cell will try to find the block in which it lies and check that this block is correctly formed. Then, as it will know the orientation of its block, the second step is to look at its facing block and check that it is also well formed. If it is then it will apply the *switch* step with its facing block. The third and last step is then to look at its brother blocks and check that they are well formed and that they have made the switch step. If everything is correct, the blocks will apply the *mix* step, which completes the transition.

All of these 3 steps will in fact take place in one single transition of the CA.

4.1. Finding the Block

A valid configuration of $\mathcal{A}_r$ looks like what is shown in Figure 4.

In such a configuration, a colored cell can easily determine its block by looking at its immediate neighborhood: if for each of the four quadrants (up-right, up-left, down-left and down-right) it looks at its 3 corresponding neighbors (up, up-right and right for the up-right quadrant for example), there should be exactly one quadrant in which it has exactly one blank neighbor and two colored neighbors in the same state as itself (the correct quadrant is shown for some cells in the figure). When the block is found, the orientation of the block gives the position of the partition that it corresponds to in the pre-image of the configuration (upper, left, lower or right). From there the cell knows where its facing block and its brother blocks should be.

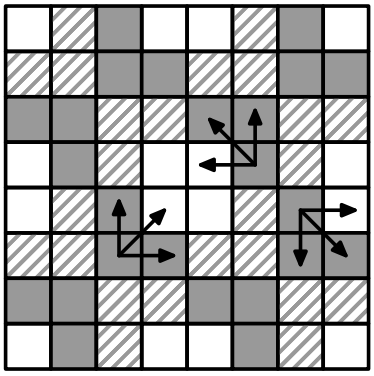

Figure 4: A valid configuration.

By looking one step farther (radius 2), the cell can determine if the other cells on its block can determine correctly the block they are in (and come to the same conclusion as itself). The block is considered *well formed* if there was exactly one possible block, that the cells in this block all have the same state and that all cells in the block consider themselves as part of this block.

If the considered cell is blank, the situation is slightly more complex. A blank cell can be either a cell of a blank block or a *marking blank* cell (one of the blank cells between the blocks). To find its situation, the cell will look at a Moore neighborhood of radius 5 around itself. In this neighborhood, it will try to identify the well formed colored blocks (as defined previously). According to what it finds, it will either conclude that it is in an invalid configuration (if the blocks are not well formed or do not match the ones with the others) or in a valid one and therefore know its own situation.

The only things that the cell needs to identify are its facing and brother blocks. These are all located in a neighborhood of radius 3. Since a neighborhood of radius 2 is necessary for a colored cell to know if its block is well formed, the blank cell needs a radius 5 neighborhood to determine a possible match for its facing and brother blocks.

Only if the blank cell has managed to find its block (and therefore the location of its facing and brother blocks) does it proceed to the next step (otherwise it remains blank).

Note that it is possible that the blank cell sees only blank cells on its radius 3 neighborhood and therefore cannot find any well formed block by looking at its radius 5 neighborhood. In this case it will not know its position but this is not a problem since, if it is in a well formed configuration, all its facing and brother blocks are blank. Since the blank state is quiescent, the cell should stay blank anyway.

All in all, any cell can find its block and make sure that the other cells in its block consider themselves as being part of the same well formed block by looking at a radius 5 neighborhood around itself.

4.2. Switch

The switch happens between a block and its facing block. During the switch a cell of one block takes the state of its facing block. To make sure that the switch is conservative and reversible we have to make sure that both blocks are well formed and that each considers the other as its facing block.

For a given cell, all the cells in its facing block are located in a radius 3 neighborhood. Since a radius 5 neighborhood was needed to find the block of a cell, by looking at a radius 8 neighborhood, a cell can check that the blocks of all the cells it considers as its facing block also consider themselves as being in its facing block.

If this is the case it take the state of these cells (switch). Otherwise, it does nothing.

The switch is a virtual step, which means that the cell does not actually change its state yet. But if the conditions for the switch are met, and that the next step (mix) does not happen, the cell will really take the switched state at the end of the transition.

4.3. Mix

The mix is very similar to the switch. However it is necessary to check a larger area: the cell must check that all its brother blocks are well formed and that all the facing blocks of its brother blocks are too in order to make sure that all the brother blocks have made a successful switch.

The cells in the facing block of a brother block of a given cell are all located in a radius 5 neighborhood of the cell, which means that by looking at a radius 10 Moore neighborhood around itself the cell can check that all its brother blocks have successfully performed their switch and will also perform the mixing step, meaning that it will look at the states of the facing blocks of its brother blocks and will apply the rule of the PCA to determine its own new state.

If an error is spotted while checking before the mix, the cell does not mix with its brother blocks and therefore keeps the switched state. Otherwise, the switched state is mixed and the switch remains a virtual step.

5. Correctness of the Automaton

5.1. Number-Conservation

The number-conservation is guaranteed by the strong verifications that we do all the time. In fact, in most cases the cell keeps its own state.

First of all it is important to notice that modifications of states occur in block transitions, that is to say that a cell only decides to change its state when it knows that it is part of a well formed block and by the definition of a well formed block all cells of the same block consider themselves in the same block (this means that if a cell c considers that c' is in its block then c' considers that c is in its block). The consequence of this is that each time that c will change its state, this change is made as a "block change" and therefore c' will change its state the same way.

This means that a cell that isn't part of a well formed block will never change its state.

Once we have this in mind, we see that a block will only change its state in two occasions : either after a switch with its facing block or after a mix with its brother blocks.

In the first case the block first checks that the facing block is correctly oriented so that this facing block will in turn take the current block's information. There is therefore a real block exchange, no state is lost nor created, there is evidently number-conservation when a switch occurs.

In the second case the block also checks that all three other brother blocks are well formed and correctly oriented and that they have already performed a switch so that they will in turn decide to perform the mix. This ensures that all four apply the rule of the original PCA and thus as the rules of the PCA are number-conserving (this is our hypothesis) the mix operation is also number-conserving.

Therefore, every time there is any kind of change in the states of a block we have guaranteed that this change is compensated by the change of 1 or 3 other blocks (and of course this guarantees the conservation at the level of the cells because every block has exactly 3 cells that have all the same state).

5.2. Reversibility

If $\mathcal{A}_p$ is reversible, then so is $\mathcal{A}_r$: given a configuration $\mathfrak{C}$ of $\mathcal{A}_r$, we can rebuild its predecessor (because it is reversible, the CA is bijective as a consequence of the Moore-Myhill theorem).

The important thing to see is that a well formed block stays well formed as $\mathcal{A}_r$ evolves and that a non well formed block cannot become well formed (with the exception of large areas of blank cells that can be considered as both well formed and not well formed, without any consequence because the blank state is quiescent).

Because of this, one can check the status of a given cell. If it is in no well formed block or that its facing block is not well formed we know that it didn't change its state. If it is part of a well formed block, that its facing block is also well formed but that one of its brother blocks or the facing block of one of its brother blocks is not well formed, then the block has simply switched with its facing block. Lastly, if all the surrounding blocks are correct, the cell (and its whole block) has applied the rule of the reversible PCA $/ACA_p$, which is a reversible local rule, so the pre-image of the cell can be determined by looking at its brother blocks' states.

5.3. Simulation of the Original PCA

In order to be useful in any way, our automaton must be able to simulate the behaviour of the original PCA.

This simulation is very easy to realize as the rules of our automaton have been designed for it.

As we have explained earlier, any configuration of the PCA $\mathcal{A}_p$ can be translated into a valid configuration of the resulting CA $\mathcal{A}_r$. Conversely, every valid configuration of $\mathcal{A}_r$ trivially corresponds to a configuration of the original one (applying the obvious reverse transformation).

Moreover, the rules of the automaton have been designed in such a way that in a valid configuration every cell could find its block, its orientation and therefore its facing and brother blocks, that are the ones that correspond to the facing and brother partitions in the partitioned automaton.

The only times when a cell cannot find its block in a valid configuration is when it is part of a blank block corresponding to a blank partition surrounded by blank partitions in the original PCA. In this case, the cell will do nothing (because it cannot determine whether or not the configuration is valid) which incidentally happens to be what it is supposed to do because the blank state is quiescent (so a blank partition surrounded by other blank partitions should remain blank).

After the blocks have been found, the switch and mix steps will occur correctly. The mix is the step that eventually applies the local rule of $\mathcal{A}_p$ (the switch has no incidence in a valid configuration, it is only used to guarantee conservation in an invalid configuration).

The simulation property can be expressed by using the simple translation function τ from the configurations of $\mathcal{A}_p$ into valid configurations of $\mathcal{A}_r$. For any configuration $\mathfrak{C}$ of $\mathcal{A}_p$, we have

$$F_p(\mathfrak{C}) = \tau^{-1}(F_r(\tau(\mathfrak{C})))$$

where F_p is the global transition rule of $\mathcal{A}_p$ and F_r that of $\mathcal{A}_r$.

6. Comments

6.1. Another Solution

There is a simpler solution if one wants only to convert a number-conserving and/or reversible PCA into a regular CA that conserves these properties.

The solution is to increase the number of states fourfold by adding a *directional layer*: if $\mathcal{Q}$ is the set of partitioned states of the original PCA, we make a CA working on the states $\mathcal{Q} \times \{\uparrow, \leftarrow, \downarrow, \rightarrow\}$.

A square cell of $\mathcal{A}_p$ is here transformed into a 2×2 square of cells, each holding a partitioned state and the arrow indicating the partition it corresponds to. The arrows do not change over time.

The behavior of the automaton is then similar to the one explained in the article but the needed neighborhood is smaller (a radius 2 neighborhood is sufficient because the configurations are more compactly represented and there are less possibilities for errors that have to be checked).

Fhe four directions are given the weights 0, $|\mathcal{Q}|$, $2|Q|$ and $3|\mathcal{Q}|$ respectively, and the weight of a cell is the sum of its state and the weight of its direction.

This method is simpler to implement but has many disadvantages over the one described in the article, most notably that it requires finite configurations to be translated into non finite ones (but still ultimately periodic) and that it increases the number of states.

6.2. Rotation Invariance

In the whole construction that we have described in this article, all four directions (up, left, down and right) have been considered similarly. Even the translation τ from configurations of $\mathcal{A}_p$ into configurations of $\mathcal{A}_r$ is rotation-invariant.

This means that if the PCA $\mathcal{A}_p$ has a rotation-invariant local rule then our resulting automaton $\mathcal{A}_r$ will also be. This is a good thing when devising simple PCA based on physical principles (both number-conservation and reversibility are often sought after because of physical considerations).

6.3. Other Partitions

In this article we have only considered two-dimensional 4-partitioned cellular automata. However there exist other kinds of partitioned cellular automata.

Important variants include for example one dimensional 2-partitioned and 3-partitioned cellular automata and two-dimensional 5-partitioned cellular automata.

For all of these different forms of PCA it is possible to adapt the construction explained in this article to obtained similar results. The key lies in the correct choice of the translation function τ that will define valid configurations.

Some important points are to be observed when choosing such a function:

- All resulting blocks must comprise the same number of cells. If not, the switch and mix steps are not conservative anymore.
- Marking blank states cannot be omitted. These states are more than mere fillers since they are the only way to determine the orientation of blocks. Markers must also be blank, not only to ensure that finite configurations have a finite total weight but also because the marking state must be quiescent (in case a cell is surrounded by cells in the marking state). In a number-conserving PCA all states are quiescent, but that is not necessarily the case for a reversible PCA.
- The translation pattern must be rotation invariant if rotation invariance is to be conserved by the transformation.
- Blocks must not necessarily be connex but non-connexity will probably increase the size of the required neighborhood in order to perform the necessary checks.

There are some simple translation functions that fit the aforementioned variants of PCA which make the results valid on these too. A smaller translation pattern might also exist for the 4-partitioned cellular automata considered in this article.

7. Conclusion

We have therefore shown how any partitioned cellular automaton can be converted into a regular cellular automaton in such a way that important properties such as number-conservation, reversibility and rotation invariance are conserved in the process.

The number of states is not increased (it is even decreased if we consider that a PCA has $\mathcal{Q}^4$ states where $\mathcal{Q}$ is the set of its partitioned states) but the considered neighborhood is.

The result is mainly interesting as a theoretical equivalence (there is no other real need to translate a PCA into a regular CA) because some constructions are much easier to perform on PCA. For instance, K. Morita was able to devise a PCA with only 3 states that is reversible, number-conserving, rotation invariant and Turing-complete. Our construction gives a regular CA that has all these same properties (although it works on a quite large neighborhood).

References

[1] U. Frisch, B. Hasslacher, and Y. Pomeau: Lattice-Gas Automata for the Navier-Stokes Equation, *Physica* **49D** (1991) 295-322.

[2] N. Margolus: Physics-Like Models of Computation, *Physica* **10D** (1984) 81-85.

[3] K. Morita and K. Imai: Number-Conserving Reversible Cellular Automata and their Computation-Universality, *Proceedings of MFCS'98 Workshop on Cellular Automata*, Brno (1998) 51-68.

[4] K. Morita, Y. Tojima and K. Imai: A Simple Computer Embedded in a Reversible and Number-Conserving Two-Dimensional Cellular Space, *Multiple-Valued Logic*, vol.6 (2001) 483-514.

[5] K. Morita, Y. Tojima, K. Imai and T. Ogiro: Universal Computing in Reversible and Number-Conserving Two-Dimensional Cellular Spaces, *Collision Based Computing*, Springer-Verlag (2002).

[6] K. Nagel and M. Schreckenberg: A Cellular Automaton Model for Freeway Traffic, *Journal of Physics* I, 2 (1992) 2221-2229.