How to simulate Turing machines by invertible one-dimensional cellular automata

Jean-Christophe Dubacq*
Département de Mathématiques et d'Informatique,
École Normale Supérieure de Lyon,
46, allée d'Italie, 69364 Lyon Cedex 07, France

December 4, 1995

Abstract

The issue of testing invertibility of cellular automata has been often discussed. Constructing invertible automata is very useful for simulating invertible dynamical systems, based on local rules. The computation universality of cellular automata has long been positively resolved, and by showing that any cellular automaton could be simulated by an invertible one having a superior dimension, Toffoli proved that invertible cellular automaton of dimension $d \geq 2$ were computation-universal. Morita proved that any invertible Turing Machine could be simulated by a one-dimensional invertible cellular automaton, which proved computation-universality of invertible cellular automata. This article shows how to simulate any Turing Machine by an invertible cellular automaton with no loss of time and gives, as a corollary, an easier proof of this result.

Keywords: cellular automata, universality, invertibility.

1 Introduction

A cellular automaton can be seen as a simple array of cells, each cell interacting with a defined set of neighbors, and changing of state according to an internal table in a synchronous way (*i.e.* each cell considers the *old* states of the other cells to compute its *new* state). One of the best known example of cellular automaton is Conway's game of Life.

We define and use a subclass of cellular automata that can be easily used for handling simulations of small local processes. These *partitioned* cellular automata are a convenient tool for describing some processes of cellular computations. In this subclass, we consider that there are several flows of information that go through each cell. The set of neighbors defines the direction and all the characteristics of that information flow, that is quantified by a product of finite sets. All that can be done by a cell is to melt the different flows, creating new ones or deleting some others, according to a deterministic table of

*2,rue Lebaudy, 78710 Rosny/Seine, France ; email: jcdubacq@ens.ens-lyon.fr

transition. The description of these automata allows easier handling than the one of a random cellular automaton.

The purpose of the second part of this paper is to prove the following result :

Theorem 1 *Any 1-tape Turing Machine can be simulated without loss of time by an invertible partitionned cellular automaton.*

Morita, in [3], proved that *invertible* Turing Machines could be simulated by invertible cellular automata, and he also presented a proof that Turing Machines could be simulated by invertible ones, but the transformation of a TM into an invertible one needs a quadratic time to run. Our construction is independant of Morita's result.

The idea of the proof is to have each cell simulating one tape cell of the Turing machine, while conveying to the left an history of all the transformations performed by the head.

2 Definitions and properties

In this section, we shall define a subclass of cellular automata that are called *partitionned* cellular automata. These automata are very simple to use in simulation of local processes, and easy to define. Their formal definition allows primary results that are quite interesting.

2.1 Partitioned cellular automata.

Partitioned cellular automata (PCA) are formally defined as quadruplets $(d, \mathcal{S}, \mathcal{N}, f)$ where:

- d is the *dimension* of the PCA;
- $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \ldots \times \mathcal{S}_r$ where $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_r$ are finite sets of state;
- $r \in \mathbb{N}$ is the number of neighbors;
- $\mathcal{N} = \{z_1, z_2, \ldots, z_r\}$ is a finite subset of $\mathbb{Z}^d$ (the *neighbourhood* of the PCA);
- f is the *transition function* defined from $\mathcal{S}^{\mathcal{N}} \to \mathcal{S}$ with the following conditions:
  - f is the composition of two functions $\Phi \circ \mathcal{G}_{\mathcal{N}}$;
  - Φ is a function from $\mathcal{S}$ into $\mathcal{S}$;
  - $\mathcal{G}_{\mathcal{N}}$ is the function defined from $\mathcal{S}^{\mathcal{N}}$ into $\mathcal{S}$ by

$$\mathcal{G}_{\mathcal{N}}(s_1, s_2, \ldots, s_r) = (s_{z_1}^{(1)}, s_{z_2}^{(2)}, \ldots, s_{z_r}^{(r)})$$

    where $s_{z_i}^{(j)}$ is the j^{th} component of the i^{th} neighbor.

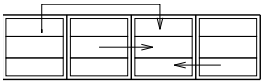

Figure 1: The set of neighbors of $\mathcal{A}$ and the action of $\mathcal{G}_{\{-2,-1,+1\}}$

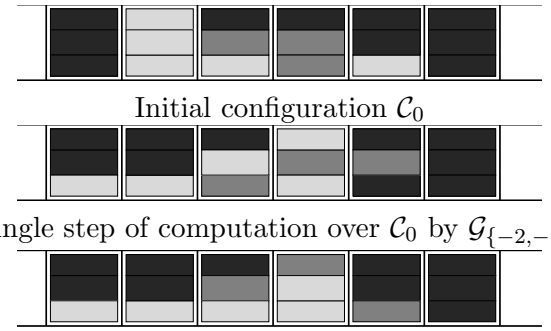

Figure 2: Applying $\mathcal{A}$ to a configuration

As an exemple we give the action of the following PCA:

$$\mathcal{A} = (1, \{0,1,2\} \times \{0,1,2\} \times \{0,1,2\}, \{-2,-1,+1\}, \text{sorting} \circ \mathcal{G}_{\{-2,-1,+1\}})$$

where the function *sorting* is the function that takes a list of three numbers as input and gives the same list in increasing order as output.

A PCA is useful for moving around information (as quantified by the various sets of states). Information will move in straight line if Φ is the identity function or could be melted locally by the action of Φ.

The difference between partitionned cellular automata and cellular automata is that the local transition function is always defined as being the composition of $\mathcal{G}_{\mathcal{N}}$ and of another fonction, and that the set of states is really defined as being a product of sets.

2.2 Invertibility

Lemma 1 *A partitioned cellular automaton $\mathcal{A}$ is invertible iff its function Φ is a bijection.*

Proof. First we prove that if Φ is not a bijection, then $\mathcal{A}$ is not invertible. We exhibate two distincts configurations that have the same image. As Φ is not bijective, it is also not injective (as it goes from a finite set into the same finite set). Hence, there does exist two states of the PCA that have the same image. Let's denote them by $(\sigma_1, \sigma_2, \ldots, \sigma_r)$ and $(\theta_1, \theta_2, \ldots, \theta_r)$. Then, let's take two configurations that are identical except for the neighbors of a cell c in which the i^{th} component of the i^{th} neighbor is σ_i in the configuration C_1 and θ_i in the configuration C_2. If we apply $\mathcal{A}$ on these two

configurations, it is clear that $\mathcal{G}_{\mathcal{N}}$ lets all the cells identical except the cell c, and that Φ applied to c lets all the cells identical (as both configurations get the same state for cell c). Hence, we get a configuration that has two ancestors, and $\mathcal{A}$ is not invertible.

Now, we shall prove that if Φ is a bijection, the PCA $\mathcal{A}$ is invertible. We simply construct $\mathcal{A}^{-1}$. Let's denote the state of a cell c by s_c. Let $\mathcal{A}^{-1} = (d, \mathcal{S}, \mathcal{N}, g)$ with g a function that can be written as $\mathcal{G}_{-\mathcal{N}} \circ \widetilde{\Phi^{-1}}$ where $\widetilde{\Phi^{-1}}$ is the extension of Φ^{-1} to $\mathcal{S}^{\mathcal{N}}$:

$$\widetilde{\Phi^{-1}}(s_{c_1}, s_{c_2}, \ldots, s_{c_r}) = (\Phi^{-1}(s_{c_1}), \Phi^{-1}(s_{c_2}), \ldots, \Phi^{-1}(s_{c_r})))$$

Hence, the i^{th} component of a cell c is $(\Phi^{-1}(s'_{c-z_i}))^{(i)}$, where s'_c is the state of the cell c after having applied $\mathcal{A}$ to the initial configuration. We can expand it into:

$$(\Phi^{-1}(\Phi(\mathcal{G}_{\mathcal{N}}(s_{c-z_i+z_1}, s_{c-z_i+z_2}, \ldots, s_{c-z_i+z_r})))^{(i)}$$

i.e. $(\mathcal{G}_{\mathcal{N}}(s_{c-z_i+z_1}, s_{c-z_i+z_2}, \ldots, s_{c-z_i+z_r}))^{(i)}$, *i.e.* $s^{(i)}_{c-z_i+z_i}$, *i.e.* $s^{(i)}_c$. As that computation stands for any cell c and any $i, 1 \leq i \leq r$, the automata $\mathcal{A}$ is invertible and admits $\mathcal{A}^{-1}$ for inverse. □

Let us note that the inverse for the PCA we built is not exactly a PCA since the melting is done *before* the translation, but it is not fundamentally different (and it is still a CA).

2.3 Consequences

We have just built a subclass of CA that is very easily described, and for which invertibility is easily decidable. This section tries to explain why the transformation doesn't preserve invertibility. As Kari showed in [1] that in general case, invertibility of 2D cellular automata is undecidable, there is a difference between the two classes. CA can obviously simulate PCA, as the definition of these can be seen as just a definition of a subset. PCA can simulate CA as follows: just make each cell send the whole information to all the neighbors, and then let each cell apply the transition function of the CA to a cell (as every cell will know the states of the neighbors). But the simulation does not preserve bijectivity (because just a few states of the cells correspond to the simulation). Thus it is not possible to prove that invertibility of the PCA is equivalent to the invertibility of the corresponding CA. In fact, we build a mapping from one set of configurations into another set of configurations that is a bijection in the case of the simulation of a PCA by a CA (hence decidability of invertibility is kept) and that is not surjective in the case of the simulation of a generic CA by a PCA.

3 Simulation of Turing Machines by one-dimensional invertible partitioned cellular automata

We are interested in the possibility of simulating processes with partitioned cellular automata, especially the simulation of Turing Machines. This resolves the problem of

universality of invertible cellular automata, as invertible partitioned cellular automata are a subset of invertible cellular automata and simulation of a Turing Machine allows the computation of any computable function. Morita already analysed the relations between universality and cellular automata in [3], [4] and [2], but this study brings a new approach to the problem.

3.1 Deterministic Turing Machine

The Turing Machine that we shall consider in this section can be defined as a triplet $T = (\mathcal{F}, \mathcal{Q}, \tau)$ where:

- $\mathcal{F}$ is a finite set of symbols (the set of characters of the Turing Machine) in which we can distinguish a *blank* symbol, denoted by 0,
- $\mathcal{Q}$ is a finite set of *states* of the R/W head, in which we can distinguish an *acceptance halt-state* q_Y,
- τ is a function from $\mathcal{F} \times \mathcal{Q} \to \mathcal{F} \times \mathcal{Q} \times \{\text{left}, \text{right}\}$, the *transition function* of the Turing Machine T.

A configuration of a tape is given by a mapping from $\mathbb{Z}$ into $\mathcal{F}$. A configuration of a Turing Machine is the configuration of a tape and the position and internal state of the head. For each computation step, the R/W head that is initiated to a starting position makes a transition according to the function τ depending on the value of the cell and the value of the state of the head, rewrites a new value in the cell, and then moves whether to the right or to the left (according to τ).

A valid configuration is a configuration that can be deduced from an initial mapping from $\mathbb{Z}$ into $\mathcal{F}$ with a finite number of symbols different from 0 and the head at the left end of the tape in a finite number of steps.

3.2 Definition of $\mathcal{A}_\tau$

Being given a Turing Machine T, we shall consider the following PCA $\mathcal{A}_\tau$:

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

$$\Sigma = \mathcal{Q} \cup \{\natural\}$$

$$\mathcal{A}_\tau = (1, \mathcal{F} \times \Sigma \times \Sigma \times (\mathcal{F} \times \Sigma \times \{\text{left}, \text{right}\}), \mathcal{N}, \Phi \circ \mathcal{G}_\mathcal{N}\}$$

We shall denote the state of a cell by (a, b, c, d) and partially define Φ in the following way:

- If $b \neq \natural, c = \natural, d = (0, \natural, \text{left})$, then, if we suppose $(a', b', \delta) = \tau(a, b)$:

$$\begin{array}{rcll} \Phi(a, b, \natural, d) & = & (a', b', \natural, (a, b, \text{left})) & \text{if } \delta = \text{left} \\ & = & (a', \natural, b', (a, b, \text{left})) & \text{if } \delta = \text{right} \end{array}$$

- If $b = \natural, c \neq \natural, d = (0, \natural, \text{left})$, then, if we suppose $(a', c', \delta) = \tau(a, c)$:

$$\begin{array}{rcll} \Phi(a, \natural, c, d) & = & (a', c', \natural, (a, c, \text{right})) & \text{if } \delta = \text{left} \\ & = & (a', \natural, c', (a, c, \text{right})) & \text{if } \delta = \text{right} \end{array}$$

- If $b = c = \natural$, then

$$\Phi(a, \natural, \natural, d) = (a, \natural, \natural, d)$$

We shall prove that Φ restricted to the already-defined states is injective. Hence, we will be able to extend Φ to a bijective function by choosing any image not already used for the remaining undefined states.

Remark that the number of states of the $\mathcal{A}$ is $2f^2(q+1)^3$, where f is the size of the set of characters of T and q is the number of states of T.

Lemma 2 *The restriction of Φ to the previous configurations is injective.*

Proof. Injectivity of Φ in a same group of states is trivial, as d is always $(0, \natural, \text{left})$. We shall just check that there can't be any confusion between two different groups. The two first groups (b or c different from $\natural$) are obviously distinct because of the last component of the images. Moreover, no element of these two groups can be seen as an element of the third one, as the second element of the last component is necessarily different from $\natural$ in the first two groups and is $\natural$ in the third. □

Corollary 1 *Φ can be extended into a bijective function. Hence, $\mathcal{A}_\tau$ is an invertible one-dimensional cellular automaton.*

3.3 Simulation of T by $\mathcal{A}_\tau$

Valid configurations of $\mathcal{A}_\tau$ are defined as configurations of the form $(0, \natural, \natural, (0, \natural, \text{left}))$ (blank cells) everywhere but in a finite number of cells, where it can be of the form $(\gamma, \natural, \natural, (0, \natural, \text{left}))$, with $\gamma \in \mathcal{F}$ (clean cells) or $(\gamma, \natural, \natural, \kappa)$, with $\gamma \in \mathcal{F}$ and $\kappa \in \mathcal{F} \times \Sigma \times \{left, right\}$ (used cells). Moreover, there must be exactly one cell in the line that has the form $(\gamma, a, \natural, (0, \natural, \text{left}))$ with $a \neq \natural$ and $\gamma \in \mathcal{F}$, for which all cells on the right are either blank or clean. The set of these configurations defines the *Turing-valid configurations*.

Lemma 3 *There exists an injection from the set of configurations of T into the set of Turing-valid configurations of $\mathcal{A}_\tau$*

Proof. Let C be a configuration of T. We build a Turing-valid configuration of $\mathcal{A}_\tau$ as follows:

- $\forall j \neq 1$, $\mathcal{C}_C(j) = (C(j), \natural, \natural, (0, \natural, \text{left}))$;
- $\mathcal{C}_C(1) = (C(1), q_0, \natural, (0, \natural, \text{left}))$, where q_0 is the initial state of the Turing Machine T.

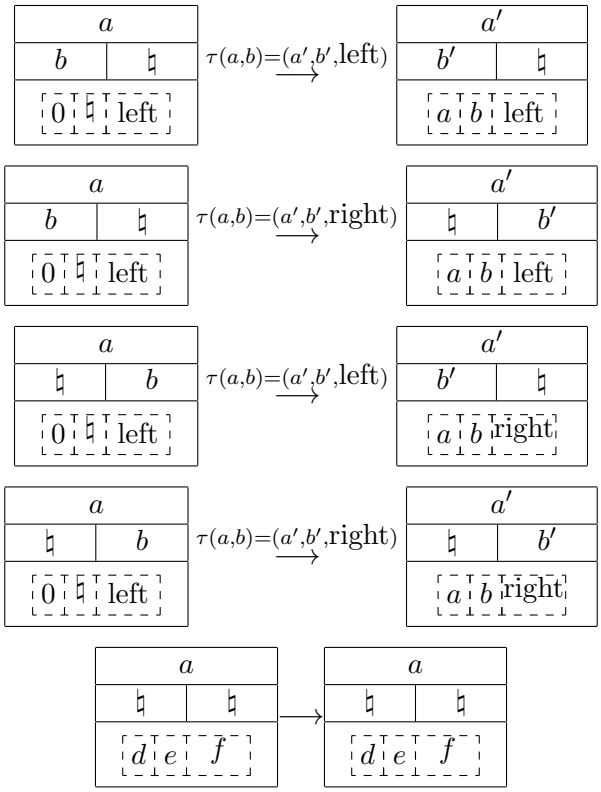

Figure 3: Transition rules for $\mathcal{A}_\tau$

The configuration we get for $\mathcal{A}_\tau$ is clearly Turing-valid. The injectivity results from the mapping of C into $\mathcal{C}_C$. □

Now, we shall prove the main theorem:

Theoreme 1 *The automaton $\mathcal{A}_\tau$ simulates the computation of T on any configuration C of T with no loss of time.*

Proof. We just have to show a function that turns the configuration of $\mathcal{A}_\tau$ obtained from Turing-valid configurations by iterations of $\mathcal{A}_\tau$ into configurations of T obtained by the same number of computation steps. We do this by writing that the line of cells is turned into the tape of T by keeping just the first field of it (*i.e.* the *character* of the cell). The R/W head is also in the only state that is different from $\natural$ in the second or third fields. We'll just write that the position of the R/W head is the abciss minus one of a cell from which the second field is different from $\natural$, or the abciss plus one of a cell from which the third field is different from $\natural$. In fact, we have to guarantee that there's only one cell for which the second or the third field is different from $\natural$ to prove the validity of that transformation.

We can prove by induction the following property: there exists one and only one cell (with abciss x_0) for which the second or third field is different from $\natural$, and $\forall x > x_0$, the fourth field is $(0, \natural, \text{left})$. It's true for the Turing-valid configurations. Let's suppose that it is true for the n^{th} iteration. The R/W head is on the cell x_0 (symbol different from $\natural$). After the iteration of $\mathcal{G}_\mathcal{N}$, the R/W head is either one cell on left, or one cell on right. The shift on left of the fourth fields that are different from $(0, \natural, \text{left})$ implies that they are at most in cell $x_0 - 2$; the R/W head being in position $x_0 - 1$ or $x_0 + 1$, the fourth component is $(0, \natural, \text{left})$ and the function Φ ensures that there is only one R/W head after the application of Φ. Hence, the condition is true for the $(n+1)^{th}$ application of $\mathcal{A}_\tau$ because the configuration is still Turing-valid.

The last check is for the correctness of the simulation of the Turing Machine T. The move of the R/W head is correct, as given by the function τ. The tape is correctly simulated, because the cells change only if the R/W head is operating on them, and, if it operates, the transition is the one that would be done by the Turing Machine (according to the definition of Φ). Hence, for any instant t, there is an exact correspondance between the tape of T and the configuration of the cellular automaton. □

From the preceding, we can deduce the following corollary:

Corollary 2 *There exist universal invertible one-dimensional cellular automata.*

Acknowledgements

I would like to thank the people that have directed my work, Jacques Mazoyer and Bruno Durand, of the Laboratoire d'informatique du Parallélisme of the École Normale Supérieure de Lyon.

References

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

[2] K. Morita. Any irreversible cellular automaton can be simulated by a reversible. Technical report, IEICE, 1992.

[3] K. Morita and M. Harao. Computation universality of one-dimensional reversible (injective). *Transactions of the IEICE*, E-72(6):758–762, June 1989.

[4] K. Morita and S. Ueno. Computation-universal models of two-dimensional 16-state reversible automata. *IEICE Trans. Inf. and Syst.*, E75–D(1):141–147, January 1992.