Master 1 d'Informatique Fondamentale

ENS DE LYON

State-Conserving Cellular Automata

Étienne Miquey
<etienne.miquey@ens-lyon.fr>

Internship effectuated in Turku during the summer 2011,
under the supervision of
Timo Jolivet & Jarkko Kari

Turun yliopisto
Matematiikan laitos

Acknowledgments

I shall first warmly thank Jarkko for having given me the opportunity to do my internship with him, and for the enthusiasm he has always shown for my modest work, although it was just a little stone in regards to the mountain he did in theory of cellular automata. I am also really grateful to Timo who accepted to co-supervise my internship in June, introduced me to cellular automata theory from the basis, and even managed to make me appreciate the beauty of space-time diagrams.

At last, I would like to add a thanking to the numerous people, whether the were Greek, Finnish or French, who shared the office with me, and always took the time to answer to my question, no matter if it was about maths, english or Tikz.

Contents

1 Introduction

1.1 Environment

I did this internship in the mathematics department of the University of Turku, in Finland, under the supervision of Jarkko KARI and Timo JOLIVET, who was there in June. Jarkko is actually the only local researcher - plus his PhD students - to really work on cellular automata. But there were some invited people in August working on related topics, namely Alexis BALLIER, Pierre GUILLON and Pascal VANIER. I tried to make the most of this great work environment to improve my knowledge of the topic, but the work I did was totally independent of what they were doing.

In the beginning of June, I had the opportunity to attend the *Unconventional Computation* conference, which gave me a first glance over the topic of cellular automata and discrete dynamical systems. The first part of my internship has been to get me familiar to the cellular automaton theory in general, and more especially to the state-conservation property, the center of my internship. I tried after that to study this particular class of automaton as other ones had already been studied, number-conserving automata among others. All results from section 3 to 6 are mine, otherwise I explicitly mentioned the source.

1.2 Scientific aspect

Cellular automata (CA for short) are among the oldest models of computation. They were first studied by VON NEUMANN and ULAM in the 60ies, while they were looking for mathematical objects describing the behavior of self-reproducing biological systems. A CA consists of a regular grid (in any finite dimension) of cells, each in one of a finite number of states. Each cell is updated by a local function applied to a neighborhood, usually containing the cell itself. CA are then a dynamical system in which both time and space are discrete.

One of the most famous CA is the well-known CONWAY's game of life. In this CA, at time t a cell is dead or alive, and at time $t+1$ it will die, born or stay in its previous state in function of the number of neighbors alive he had at time t. CA possess a lot a physical properties - e.g. they are massively parallel, and all interactions are local -, therefore they are used in many topics to modelize some physical systems. Among many other examples, they are used as models for forest fires, road traffic, the collapse of a sandpit, the propagation of a rumor into a social net, some chemical reactions and even the generation of seashell patterns.

Despite it could seem to be a simple model of computation, it turns out that most of the basic properties - namely surjectivity, nilpotency, etc - are undecidable for dimension 2 and greater, and some are even undecidable for 1-dimensional CA. Actually, a lot of questions still remain open, which, in addition to its numerous applications, makes of it a interesting current topic of research.

As CA are often used for physical modeling, the conservation of some quantities, such as entropy or momentum among others, is a very natural question. Classes of CA conserving one of this quantities have been studied, like for instance number-conserving automata, which can be interpreted as a model for particle interactions [6]. Although it is a subclass with more constrains, it still presents interesting properties, and its universality has been proved [5].

A great importance is also usually attached to the reversibility, on the one hand because it enables to modelize reversible physical phenomenæ, and on the other hand because it would be really energy efficient, due to Landauers principle's. That is why we will especially pay attention to the preservation of reversibility in the Section 5.

The aim of that internship was to study an other of these classes, the one of state-conserving cellular automata (SCCA), briefly mentioned in [6]. A state-conserving CA has to suit an even stronger constrain than a number-conserving one, but the class of SCCA is still a non-trivial class. For 1-dimensional SCCA, I proved an equivalence between definitions of state-conservation, and developed an algorithm to generate some state-conserving CA. Then I paid attention to the universality of SCCA, that is to know whether SCCA have the same computational power than Turing machines or general CA. I focused more especially to the problem of building a Turing-universal SCCA with the smallest neighborhood possible. Finally I used some of the techniques I exposed for the universality question so as to reduce some decision problems to SCCA.

2 Cellular Automata

2.1 Definitions

All along this report, we will only consider one-dimensional objects. Therefore, we will not precise it any longer, but all results only hold for one-dimensional cellular automata. We will denote a semi-infinite sequence of 0 to the left (resp. to the right) by ${}^{\infty}0$ (resp. 0^{∞}).

Definition 2.1. Let S be a finite set. Elements of S will be called *states*. A *configuration* of a 1-dimensional CA is a function

$$c : \mathbb{Z} \to S$$

that assigns a state to each cell. We will often write c_i for $c(i)$ to denote the state of cell i.
A configuration c is said to be *finite* if 0 is quiescent (see definition 2) and if there exists some $r \in \mathbb{Z}$ s.t.

$$\forall n \in \mathbb{Z}, |n| > r \Rightarrow c(n) = 0.$$

A configuration c is said to be *periodic* if there exists a period $r \in \mathbb{Z}$ (we will denote it by $p(c)$) s.t.

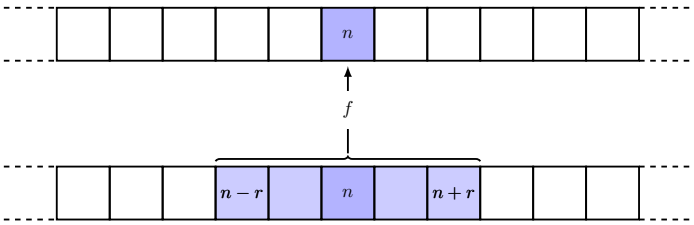
$$\forall n \in \mathbb{Z}, c(n + r) = c(n).$$

Definition 2.2. A *neighborhood vector* of size m is a tuple $\vec{n} = (n_1, \ldots, n_m)$ of distinct integers, so that a cell n has neighbors $n + n_i$ for $i = 1, \ldots, m$. Most of the time, we will use a set of m consecutive cells as neighborhood, centered on the main cell, and we will define the *radius* such that the neighborhood of radius r of a cell n is $\{n - r, \ldots, n, \ldots, n + r\}$.

The configuration of a cellular automaton with state S and size m neighborhood will be updated by a the *local rule* f, which is a function

$$f : S^m \to S$$

A state s is said to be *quiescent* if $f(s, \ldots, s) = s$. The new state of a cell that has neighborhood $s_1, \ldots, s_m$ will be $f(s_1, \ldots, s_m)$.



In a cellular automaton, the same local rule is applied simultaneously to each cell, transforming the whole configuration c in a new one c'. The transformation $c \mapsto c'$ is called the *global function* $G : S^{\mathbb{Z}} \to S^{\mathbb{Z}}$. Finally, a 1-dimensional *cellular automaton* is given by $\mathcal{A} = (S, \vec{n}, f)$. We will denote by $\mathcal{CA}(q, m)$ the set of all CA that have states $S = \{0, \ldots, q - 1\}$ and neighborhood size m, so that $\#\mathcal{CA}(q, m) = q^{q^m}$.

Theorem 2.3 (Curtis-Hedlund-Lyndon)**.** *G is a global function of a CA if and only if it is continuous in the product topology of $S^{\mathbb{Z}}$ and commutes with the translation.*

2.2 Representation of a CA

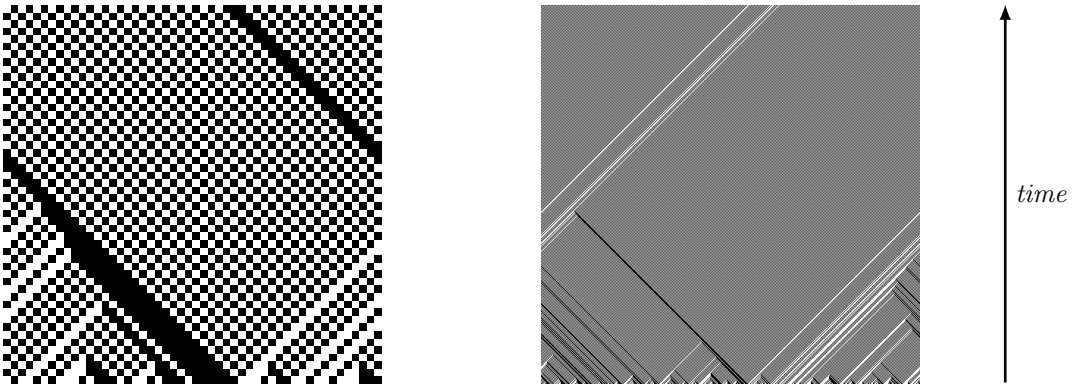
A classical way of representing a CA with states $0, \ldots, q-1$ and neighborhood size m is to use a number in base q and then to convert it in a decimal. Indeed, there are q^m possible neighborhoods, to which the CA associates a number in $[0, q - 1]$. The i-th number in base q will be $f(\overline{i}^{(q)})$. For instance, if $\mathcal{A} \in \mathcal{CA}(2, 3)$ has the following function f:

Neighborhood :	000	001	010	011	100	101	110	111
Value of f:	0	0	0	1	1	1	0	1

then we will denote $\mathcal{A}$ by $\overline{10111000}^{(2)} = 188$.

The *space-time diagram* of a CA is a diagram obtained from the evolution of a configuration under consecutive applications of the CA. Figure 1 gives examples for the automaton #188 (time increasing upwards, 1=■, 0=□.)

An other classical and useful way of representing a cellular automaton is to use a De Bruijn graph.



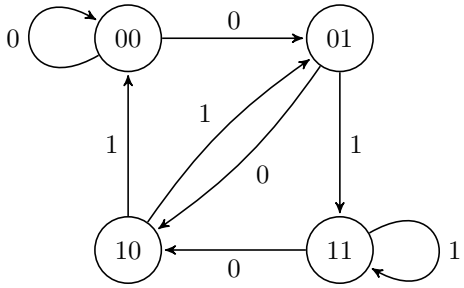
(a) 50 cells, 50 steps (b) 500 cells, 500 steps

Figure 1: Space-time diagram of the automaton #188

Definition 2.4. Given an alphabet S, and an integer m, the *De Bruijn graph* $\mathcal{G} = (V, E)$ over S of width m is the graph of vertices $V = S^{m-1}$, whose edges are the set $E = \{(su, ut) \mid s, t \in S, u \in S^{m-2}\}$. For any word in S^m there is an edge in E, and conversely, so that we can see E as S^m.

The De Bruijn representation of a CA $\mathcal{A} = (S, m, f)$ is the De Bruijn graph over S of width m in which each edge $e \in S^m$ is labeled by $f(e)$. From this representation we can recover entirely G up to a translation, which is does not matter for the properties we will study.

Example 2.5. Here is the De Bruijn graph of the automaton #188.



Basically, a configuration is a bi-infinite path in the graph, and its image is given reading the labels along the path. Furthermore, the De Bruijn graph is the same for any CA in the same class $\mathcal{CA}(q, m)$, only its labeling changes. Thus a lot a properties over CA have an interesting characterization as De Bruijn property, which are often easier to manipulate or to design algorithm, as we will see later .

2.3 Injectivity, surjectivity, reversibility

Definition 2.6. A CA $\mathcal{A}$ is said to be *injective* (resp. *surjective*) if its global function G is.

In other terms, $\mathcal{A}$ injective means that two different configurations have two different images by $\mathcal{A}$, and $\mathcal{A}$ surjective that every configuration has at least one preimage by $\mathcal{A}$.

Considering the space of configurations as a topological space, we have easily that every sequence of configuration has a converging subsequence, which is the compactness of this space. From that, we get many classical results [2], and in particular the following implication.

Proposition 2.7. *If G is injective, then G is also surjective.*

An other interesting property of a cellular automaton is the reversibility. An automaton is reversible if, from any configuration, we can reverse its action to go in the past:

Definition 2.8. A CA is called *reversible* if its function G is bijective and if G^{-1} is also a CA function.

Still using the compactness of the space of configurations and the theorem 2.3 , we show that the inverse function of a CA is actually always a CA function:

Proposition 2.9. *A CA is reversible if and only if it is bijective.*

3 State-Conservation

Intuitively, an automaton is state-conserving if each state that was in the configuration at time t is still in the configuration at time $t+1$, potentially in another cell. We will give in this section some basic properties and give a look at the link with reversibility.

3.1 Definition

Definition 3.1. For each $\alpha \in S$, we denote by $\delta_\alpha : S \to \{0,1\}$ the Dirac function $\delta_\alpha(x) = 1 \Leftrightarrow x = \alpha$. $\mathcal{A} = (S, m, f)$ is a *state-conserving* cellular automaton (SCCA) if and only if for every spatially periodic configuration c,

$$\forall \alpha \in S,\ \sum_{i=0}^{p(c)-1} \delta_\alpha(G(c)_i) = \sum_{i=0}^{p(c)-1} \delta_\alpha(c_i)$$

We set $\mathcal{SC}(q,n) = \{\mathcal{A} \in \mathcal{CA}(q,n) \mid \mathcal{A} \text{ is state conserving}\}$.

We can notice that the property of state-conservation is robust to relabeling. Furthermore, it is sufficient that the equality is true for every $\alpha \in S\backslash\{0\}$.

Lemma 3.2. *If (S, m, f) is an SCCA, then $\forall (s_1, \dots, s_m) \in S^m, f(s_1, \dots, s_m) \in \{s_1, \dots, s_m\}$.*

Proof. Suppose there exists $(s_1, \dots, s_m) \in S^m$ and $\alpha \in S\backslash\{s_1, \dots, s_m\}$ such that $f(s_1, \dots, s_m) = \alpha$. Let $\omega = s_1 \dots s_m$ and $c = {}^\infty s^\infty$. Then we have $\sum_{i=0}^{p(c)-1} \delta_\alpha(G(c)_i) \geq 1$ and $\sum_{i=0}^{p(c)-1} \delta_\alpha(c_i) = 0$, which is absurd. □

Corollary 3.3. *If $\mathcal{A}$ is an SCCA, then all states are quiescent for $\mathcal{A}$.*

We defined previously state-conservation on periodic configurations, but it also possible to express it on finite configurations, or even on any configuration, using a limit. Actually, these three definitions are equivalent :

Proposition 3.4. *Let $\mathcal{A} = (S, m, f)$ be CA and $\alpha \in S\backslash\{0\}$. The following statements are equivalent:*

(a) For every periodic configuration c, $\sum_{i=0}^{p(c)-1} \delta_\alpha(G(c)_i) = \sum_{i=0}^{p(c)-1} \delta_\alpha(c_i)$.

(b) For every finite configuration c, $\sum_{i\in\mathbb{Z}} \delta_\alpha(G(c)_i) = \sum_{i\in\mathbb{Z}} \delta_\alpha(c_i)$.

(c) For every $c \in S^{\mathbb{Z}}$, $\lim_{n\to+\infty} \frac{\mu_n^\alpha(G(c))}{\mu_n^\alpha(c)} = 1$ where $\mu_n^\alpha(c) = \sum_{i=-n}^{n} \delta_\alpha(c_i)$.

Proof. We will prove $(a) \Rightarrow (b) \Rightarrow (c) \Rightarrow (a)$.

Let us assume that there exists c_f a finite configuration s.t. $\sum_{i\in\mathbb{Z}} \delta_\alpha(G(c)_i) \neq \sum_{i\in\mathbb{Z}} \delta_\alpha(c_i)$. As c is finite, there is a $N \in \mathbb{N}$ s.t. for all $n \in \mathbb{Z}$, if $|n| > N$, then $c_n = 0$. Let ω be the word $c_{-N} \cdots c_N$, and c_p the periodic configuration of periodic pattern $0^m \cdot \omega \cdot 0^m$. Then, it is easy to check that

$$\sum_{i=0}^{p(c_p)-1} \delta_\alpha(G(c_p)_i) = \sum_{i\in\mathbb{Z}} \delta_\alpha(G(c)_i) \neq \sum_{i\in\mathbb{Z}} \delta_\alpha(c_i) = \sum_{i=0}^{p(c_p)-1} \delta_\alpha((c_p)_i)$$

Thus, $\neg(b) \Rightarrow \neg(a)$, and $(a) \Rightarrow (b)$, by contraposition.

Assume (b). Consider $c \in S^{\mathbb{Z}}$, and for all $n \in \mathbb{N}$, define $\beta^n \in S^{\mathbb{Z}}$ such that $\beta_i^n = \begin{cases} c_i & \text{if } i \in [-n, n] \\ 0 & \text{otherwise} \end{cases}$, so that

$$\mu_n^\alpha(c) = \sum_{i=-n}^{n} \delta_\alpha(c_i) \overset{\text{def}}{=} \sum_{i\in\mathbb{Z}} \delta_\alpha(\beta_i^n) \overset{\text{(b)}}{=} \sum_{i\in\mathbb{Z}} \delta_\alpha(G(\beta^n)_i) \tag{1}$$

Clearly, $G(\beta^n)_i$ and $G(c)_i$ only can be different for i s.t. $|i| > n - m$, and so $|\mu_n^\alpha(G(c)) - \mu_n^\alpha(c)| < 2m$.

- If $\mu_n^\alpha(c) \underset{n\to+\infty}{\longrightarrow} +\infty$, as $\frac{\mu_n^\alpha(c)-m}{\mu_n^\alpha(c)} \leq \frac{\mu_n^\alpha(G(c))}{\mu_n^\alpha(c)} \leq \frac{\mu_n^\alpha(c)+m}{\mu_n^\alpha(c)}$, we get $\frac{\mu_n^\alpha(G(c))}{\mu_n^\alpha(c)} \underset{n\to+\infty}{\longrightarrow} 1$.

- Otherwise, there exists $N_0 \in \mathbb{N}$ such that $\forall n \geq N_0, \delta_\alpha(c_{-n}) = \delta_\alpha(c_n) = 0$. In particular, for all i s.t. $N_0 + m \leq |i| \leq N_0 + 2m = N$, $c_i \neq \alpha$. Using the Lemma 3.2, we get that for all i s.t. $N - m \leq |i| \leq N$, $\delta_\alpha(G(c)_i) = \delta_\alpha(G(\beta^N)_i) = 0$, and as $\forall i > N, \delta_\alpha(G(c)_i) = 0$, for all $n > N$:

$$\mu_n^\alpha(G(c)) = \sum_{i\in\mathbb{Z}} \delta_\alpha(G(c)_i) = \sum_{i=-N}^{N} \delta_\alpha(G(c)_i) = \sum_{i=-N}^{N} \delta_\alpha(G(\beta^N)_i) \overset{(a)}{=} \mu_N^\alpha(c) = \mu_n^\alpha(c)$$

Hence $\frac{\mu_n^\alpha(f(c))}{\mu_n^\alpha(c)} \underset{n\to+\infty}{\longrightarrow} 1$.

Let us assume (c). Let c be a spatially periodic configuration. From the periodicity, we get that there exists $A, B \in \mathbb{N}$ s.t. $\forall k \in \mathbb{Z}, \sum_{i=0}^{p(c)-1} \delta_\alpha(f(c)_{k+i}) = A$ and $\sum_{i=0}^{p(c)-1} \delta_\alpha(c_{k+i}) = B$. Thus, for all $n \in \mathbb{N}$, we have $\mu_{np(c)}^\alpha(f(c)) = 2nA + f(c)_0$ and $\mu_{np(c)}^\alpha(c) = 2nB + c_0$. So that $\frac{\mu_{np(c)}^\alpha(f(c))}{\mu_{np(c)}^\alpha(c)} \to \frac{A}{B}$ Applying (c), we get $\frac{A}{B} = 1$, which implies (a).

□

3.2 Reversibility

A usual question in the study of a subclass of cellular automata is to know its relation with the reversibility. We will show here that reversible SCCA have state-conserving reverse automata, and that the subclasses of state-conserving CA and reversible CA have a non-empty intersection and are not included one in the other.

Proposition 3.5. *Let $\mathcal{A}$ be an SCCA. If $\mathcal{A}$ reversible, then $\mathcal{A}^{-1}$ is also state-conserving.*

Proof. This is trivial using the characterization (c) of Proposition 3.4. Let $c \in S^{\mathbb{Z}}$, $c' = G^{-1}(c)$, so that $c = G(c')$. Easily, we have

$$\lim_{n\to+\infty} \frac{\mu_n^\alpha(G^{-1}(c))}{\mu_n^\alpha(c)} = \lim_{n\to+\infty} \frac{\mu_n^\alpha(c')}{\mu_n^\alpha(G(c'))} = 1.$$

□

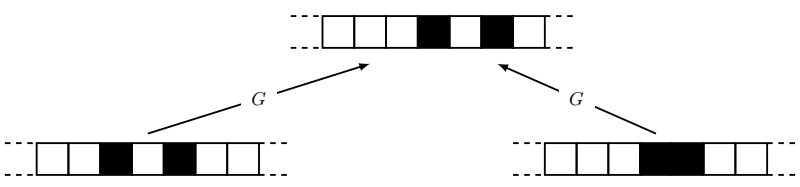
In dimension 1, the reversibility of a cellular automaton is decidable. From propositions 2.7 and 2.9, we have that an automaton is reversible if and only if it is injective, what could be classicaly characterized in terms of De Bruijn graphs.

Proposition 3.6. *$\mathcal{A}$ is injective if and only if in its De Bruijn representation, two different bi-infinite paths always have different labelings.*

We build the pair graph (V_p, E_p) of the De Bruijn graph (V, E), made of vertices $(a, b) \in V \times V$, in which there is an edge $(a, b) \to (a', b')$ each time (a, a') and (b, b') are in E and have the same label. One can easily verify that there exists two two-way infinite paths in the De Bruijn if and only if the pair graph has a cycle going through a vertex out of the diagonal $\{(a, a) \mid a \in V\}$. This existence can for instance be checked with an adapted depth-first search algorithm.

It appears that there exists both reversible and non-reversible non-trivial SCCA. Thereafter are examples of one of each.

Example 3.7. The automaton #188 is an SCCA, but is not reversible. Indeed, both configurations $^\infty 01010^\infty$ and $^\infty 01100^\infty$ will have $^\infty 01010^\infty$ for image. Actually, this automaton is not even surjective, since the very same configuration $^\infty 0110^\infty$ does not have any preimage by G.



Example 3.8. The smallest non-trivial reversible SCCA has 3 states and a neighborhood of size 4. One of the states is a marker (here □). Each time it occurs in a configuration, if the two following cells are ■■ or ■■, they will be swapped (figure 2). Otherwise, we apply the identity. This automaton is trivially an SCCA, and is also reversible.

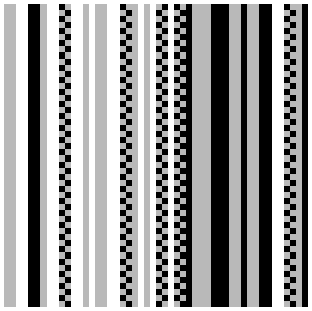


Figure 2: Automaton #4343921833628794389100565499841758197 85

4 Decidability

In this section, we will show that the state-conservation property is decidable, and give effectivily an efficient algorithm to enumerate a class $\mathcal{SC}(q,n)$, using the De Bruijn representation.

4.1 Necessary and sufficient condition

The following gives a necessary and sufficient condition for a CA to be state-conserving, which implies that the state-conserving property is decidable, since the condition is algorithmically checkable.

Proposition 4.1 ([6])**.** *$\mathcal{A} = (S, m, f)$ is state-conserving if and only if, for all $\alpha, x_1, \dots, x_n \in S$,*

$$\delta_\alpha(f(x_1,\dots,x_n)) = \delta_\alpha(x_1) + \sum_{k=1}^{m-1} \delta_\alpha(f(\underbrace{0,\dots,0}_{n-k},x_2,\dots,x_{k+1})) - \delta_\alpha(f(\underbrace{0,\dots,0}_{n-k},x_1,\dots,x_k))$$

This condition has the advantage of being generalisable for greater dimensions, but it would be particularily inefficient to generate one of the sets $\mathcal{SC}(q,n)$. Indeed it would cost $O(q^{q^{q^m}})$ tests (one per α), each of them requiring to compute a sum and $2m$ call to f.

Nevertheless, using De Bruijn graphs, we can design a better algorithm, adapting an idea developed in [1] for number-conserving CA. Let us consider $\mathcal{A} \in \mathcal{CA}(q,m)$. Let $v : Q \to \mathbb{Z}^q$ be the function associating to i the i^{th} elementary vector $\vec{u}_i$. For any edge e, let us define the $letter(e)$ as the first letter of e seen as a vector of $\mathbb{Z}^q$ and the weight of e as $\mu(e) = v(letter(e)) - v(label(e))$. For instance, if $q = 3$, we have $\mu(01 \xrightarrow{1} 12) = (1,0,0) - (0,1,0) = (1,-1,0)$. We define the weight of a path in the usual way, as the sum of the weight of its edges.

Proposition 4.2. *Let $\mathcal{A}$ be a CA and $\mathcal{G} = (V, E)$ be its De Bruijn representation. Then, $\mathcal{A}$ is state-conserving if and only if every cycle in $\mathcal{G}$ has weight $(0,\dots,0)$.*

Proof. Let us suppose that $\mathcal{A}$ is not state-conserving. From the definition, there exists a periodic configuration c and $\alpha \in S$ such that $\sum_{i=0}^{p(c)-1} \delta_\alpha(G(c)_i) \neq \sum_{i=0}^{p(c)-1} \delta_\alpha(c_i)$. Like every other periodic configuration, c corresponds to a cycle in G. Along this cycle, we will read exactly $\sum_{i=0}^{p(c)-1} \delta_\alpha(c_i)$ time α as letter, and $\sum_{i=0}^{p(c)-1} \delta_\alpha(G(c)_i)$ as a label. So that the α^{th} composant of the weight of the cycle is nonzero.

Conversly, if we have a cycle that has its k^{th} composant different from zero. Then, if ω is the word read along the cycle, with $c = \omega^*$, we have $\sum_{i=0}^{p(c)-1} \delta_k(G(c)_i) \neq \sum_{i=0}^{p(c)-1} \delta_k(c_i)$, and so $\mathcal{A}$ is not state-conserving. □

4.2 Generating state-conserving rules

If $\mathcal{G} = (V, E, \mu)$ is a weighted graph, we said that it admits $\nu : V \to \mathbb{Z}^q$ as a *potential function* if and only if

$$\forall (s,t) \in E,\ \nu(t) = \nu(s) + \mu((s,t)). \tag{2}$$

One can easily check that the following holds :

Proposition 4.3. *A strongly connected weighted graph admits a potential function if and only if all its cycles have weight 0.*

To list $\mathcal{SC}(q,m)$, we will enumerate $\mathcal{CA}(q,m)$ by labelling the edges, and backtrack as soon as a labelling is not suitable for the existence of a potential function. The efficiency of the algorithm depends of the way we go through the graph to label it. Indeed, each time we reach a vertex which already has a potential value, it constrains us for the label value, and avoid a lot of possibilities. We begin by sorting edges so as to encounter the maximum of constrains as soon as possible, by doing a depth-first search in which, at each step, we check first if any of the neighbors has already been reached. It gives us a sorted list $[e_0, \dots, e_{n-1}]$ of edges. Then we call the algorithm 1 on SCCA(0).

Algorithm 1 SCCA(i)

Require: A list $[e_0, \dots, e_{n-1}]$ of the edges
1: **if** i=n **:**
2: add the current rule to the list, and return
3: **else**
4: /* $e_i \equiv s_i \to t_i$ */
5: **if** $\nu(e_i)$ is already defined **:**
6: **if** it is possible to define $label(e_i)$ such that $\nu(t_i) = \nu(s_i) + \mu(e_i)$ **:**
7: SCCA$(i+1)$
8: **end if**
9: **else**
10: **for each** $\alpha \in S$ **:**
11: label(e_i):=α
12: $\nu(t_i) := \vec{u}_\alpha + \nu(s_i)$
13: SCCA(i+1)
14: **end for**
15: erase the definition of $\nu(t_i)$ and return
16: **end if**
17: **end if**

Figures 3 and 4 presents some result obtained with this program. Some of them were already known by A.Moreira[1], new results are in blue. Some larger samples of automata and the source code of the program[2] are available at `http://perso.ens-lyon.fr/etienne.miquey/utu/`

5 Universality

The goal of this section is to show that SCCA have the computational power of Turing machine and general CA. First, we will expose an almost optimal simulation of a Turing machine, then a simulation that preserves reversibility, and finally a simulation of general CA.

5.1 Turing machine simulation

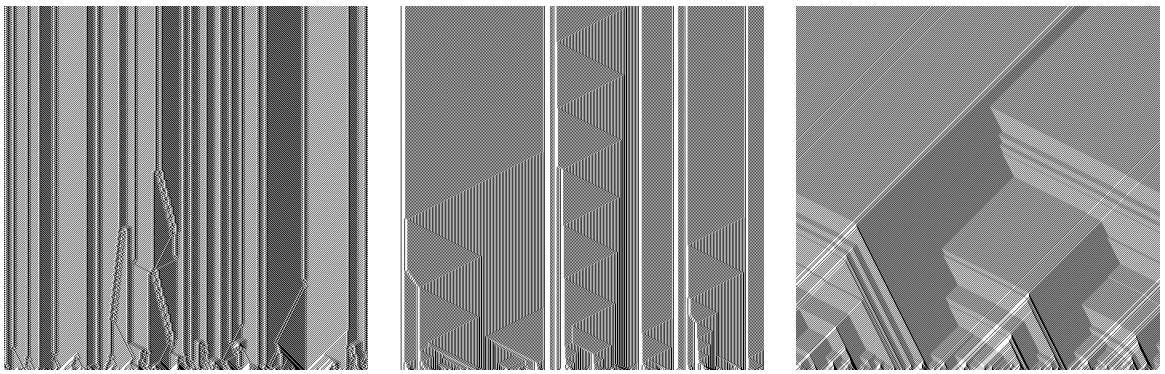
Definition 5.1. A *Turing machine* (TM) is a tuple $\mathcal{M} = (Q, \Sigma, q_0, \mathcal{F}, \delta)$ where :

- Q is the set of states
- Σ is the tape alphabet
- $q_0 \in Q$ is the initial state
- $\mathcal{F} \subset$ is the set of accepting states
- $\delta : Q \times \Sigma \to Q \times \Sigma \times \{\triangleleft, \triangleright\}$ is the transition function

[1] *See his webpage* `http://www.dim.uchile.cl/~anmoreir/ncca/mywork.html`
[2] *Its effective complexity can only be observed experimentaly, use the verbose option.*

q \ n	2	3	4	5	6	7
2	2	5	22	428	133184	1571814309
3	2	15	6312	?	?	?
4	2	89	241121850	?	?	?
5	2	843	?	?	?	?
6	2	11645	?	?	?	?
7	2	227895	?	?	?	?
8	2	6285809	?	?	?	?

Figure 3: Cardinal of $\mathcal{SC}(q, n)$



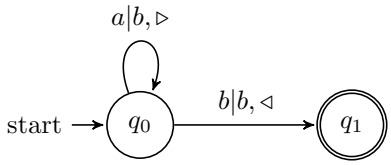
(a) $\mathcal{A}_1 \in \mathcal{SC}(2, 5)$ (b) $\mathcal{A}_2 \in \mathcal{SC}(2, 6)$ (c) $\mathcal{A}_3 \in \mathcal{SC}(3, 4)$

Figure 4: Some SCCA

The TM reads a symbol of Σ on its tape (that we will suppose bi-infinite), and then, according to δ, writes on the cell and moves to the right or to the left.

In the point of view of cellular automata, the set $\mathscr{T}$ of TM tapes could be seen as $\Sigma^{\mathbb{N}} \times (Q \times \Sigma) \times \Sigma^{\mathbb{N}}$, or even as $(\overline{Q} \times \Sigma)^{\mathbb{Z}}$ where $\overline{Q} = Q \cup \{\emptyset\}$, and δ as a local function from $\mathscr{T}$ to itself[3].

Example 5.2. The following machine, starting on a cell of the tape, will go to the right changing every a to b until reading a b.



To prove the Turing-universality of SCCA, we would like to be able to simulate every TM by an SCCA, with a neighborhood size as small as possible. We will do it in two steps. First, we will show how to encode any TM into an ordinary cellular automata, and then how to turn this automata into a state-conserving one.

Before going any further, we need to introduce the notions of simulation and mapping. A simulation of a system $\mathcal{S}_1$ by a system $\mathcal{S}_2$ is a construction which enables to do computation of $\mathcal{S}_1$ with $\mathcal{S}_2$. We will say that it uses a mapping when a block of the input $\mathcal{S}_1$ is encoded by a block of input for $\mathcal{S}_2$.

Definition 5.3. For any finite sets A, B, we call *mapping* a function $\chi : A^{\mathbb{Z}} \to B^{\mathbb{Z}}$ if there exists $m = 2r + 1 \in \mathbb{N}$ and $X : S^m \to Q \times \Sigma$ such that :

- $\forall c \in S^{\mathbb{Z}}, \forall i \in \mathbb{Z}, \exists j_i \in \mathbb{Z} \ / \ \chi(c)_i = X([c_{j_i} - r, c_{j_i} + r])$
- $(j_i)_{i \in \mathbb{Z}}$ is an increasing sequence.

[3] *Of course, a function of this shape is not, in general, the transition of a TM.*

We denote by π the function $i \mapsto j_i$.

Definition 5.4. We will call *simulation* any couple (Φ, ε_ϕ) such that

1. for every TM $\mathcal{M}$, $\Phi(\mathcal{M}) = \mathcal{A}_\mathcal{M}$ is a CA
2. ε_ϕ is a function from $\mathscr{T}$ to $S^\mathbb{Z}$, such that there exists χ verifying $\chi \circ \varepsilon_\phi = \mathrm{id}$
3. if we denote by $\xrightarrow{\mathcal{M}}$ (resp. $\xrightarrow{\mathcal{A}}$) a step in $\mathcal{M}$ (resp. $\mathcal{A}$), for any tape $t \in \mathscr{T}$, the following diagram commutes:

$$\begin{array}{ccc} t & \xrightarrow{\mathcal{M}} & \mathcal{M}(t) \\ \varepsilon_\phi \downarrow & & \downarrow \varepsilon_\phi \\ c & \xrightarrow{\mathcal{A}} & G(c) \end{array}$$

We say that a simulation is *minimal* if, given the lengths $|Q|, |\Sigma|$ there is no other simulation using a smaller radius. We say that a simulation uses a mapping if χ is a mapping.

If we have a simulation at our disposal, then it means that we are able to compute with $\mathcal{A}$ what $\mathcal{M}$ would do :

$$t \xrightarrow{\varepsilon_\phi} c \xrightarrow{\mathcal{A}} G(c) \xrightarrow{\chi} \mathcal{M}(t)$$

We shall note that ε_ϕ is necessarily injective, and χ surjective. Moreover, given the sets Q and Σ, they are a finite number of Turing machines, and thus for every $\mathcal{M}$, the radius of $\Phi(\mathcal{M})$ only depends on the lengths $|Q|$ and $|\Sigma|$.

The idea we will use to simulate a TM is quite common, and lays on a method first developed by Lindgren and Nordahl [4]. Cells of the TM would be represented each one by a cell in the CA, separated by markers, plus one cell standing for the head with the state of the machine inside. One step in the machine will be divided into two in the CA, distinguished by the state of the markers:

• : we change the state of the read cell and place the head (in the suitable state) on the side it will move;

↔ : we move the head next to the next cell

At the end, when the machine reaches a final state, we just apply the identity. We shall notice that this construction only requires a radius 1. Rather than detailing the definition of the appropriate (Φ, ε_ϕ), which is not really difficult, we will illustrate it on the figure 5 for the machine of the example 5.2.

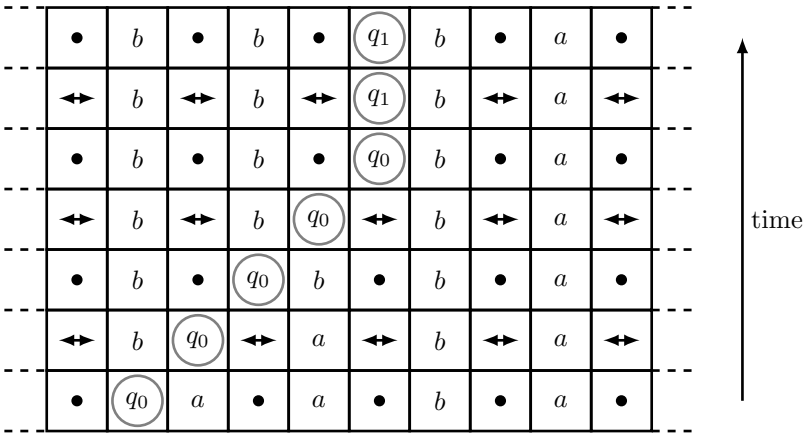
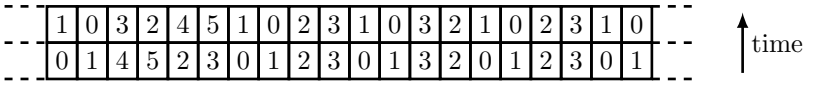


Figure 5: Simulation of a Turing machine

To turn that kind of CA into an SCCA, we will encode each letter as block of cells containing a permutation of a set. The markers are going to be encoded over $\{0,1\}$ (01 for •, and 10 for ↔ for instance). We will use permutation of $\{2, \ldots, n+1\}$, with n s.t. $n! > |\Sigma|$ to encode tape letters, and permutation of $\{n+2, n+p+1\}$, with $p! > |Q|$ for the states of the machine. In the previous example, $a \mapsto 23$, $b \mapsto 32$, $q_0 \mapsto 45$, $q_1 \mapsto 54$ would work, and the two first lines of the figure 5 would be :



To ensure that the automaton is state-conserving, we apply the identity on every block of cells which does not correspond to a valid encoding, for instance if there are two heads of a TM side by side, or if the block is not between two markers of the same kind. Therefore, it requires to have a radius r such that $r \geq n + p + 1$, for the last cell of the head to see the whole block on its side until the head of the marker, and S has to be $\{0, \dots, n + p + 1\}$. As it is locally state-conserving on any block between two markers, one can easily check that the automaton is an SCCA.

We could also have used permutations to encode directly couples of $(\overline{Q} \times \Sigma) \bigcup (\Sigma \times \overline{Q})$, in which case n would have had to be greater than $2|\overline{Q}||\Sigma|$, and radius to be $r = n + 1$ to see the whole block. This is in general slightly better as soon as $|Q|$ and $|\Sigma|$ grow.

Theorem 5.5. *If $\mathcal{M} = (Q, \Sigma, q_0, \mathcal{F}, \delta)$ is a Turing machine, n is such that $n! > 2|\overline{Q}||\Sigma|$, then there exists $\mathcal{A} \in \mathcal{SC}(n + 2, 2n + 1)$ that simulates $\mathcal{M}$.*

Then we have a construction with a radius sub-logarithmic in the alphabets length. The next proposition show that this is not possible to have a constant radius, and that in fact, the simulation we exposed has the best order of magnitude.

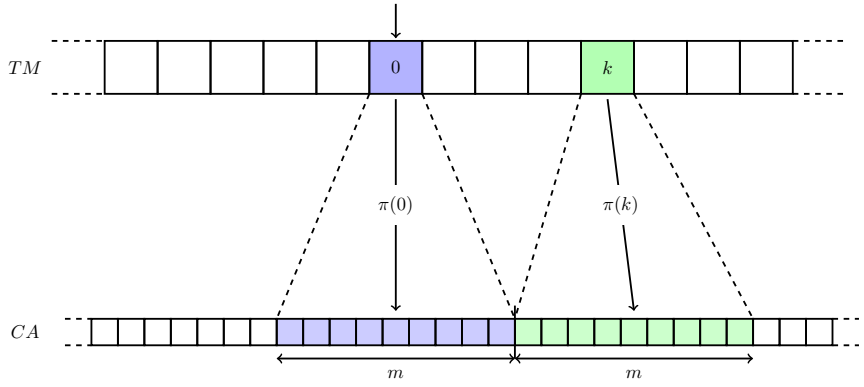
Proposition 5.6. *Assume that we have a simulation (Φ, ε_ϕ) using a mapping and such that for every Turing machine $\mathcal{M}$, $\phi(\mathcal{M})$ is an SCCA. Let Q and Σ be some fixed alphabets, and r be the common radius of all $\phi(Q, \Sigma, q_0, \mathcal{F}, \delta)$. Then we have that $(2r + 1)! = O(|Q||\Sigma|)$. Furthermore, if Φ is minimal then the mapping verifies $|\pi(1) - \pi(0)| \geq 2r + 1$.*

Sketch of the proof. Let us denote Σ by $\{\gamma, \alpha_1, \dots, \alpha_{|\Sigma|-1}\}$ and Q by $\{q_0, \dots, q_{|Q|-1}\}$. Consider a TM that, in any state q_j, will :

- move left each time it reads γ
- change α_i to α_{i+1} if $i < |\Sigma| - 1$ and move right
- change $\alpha_{|\Sigma|-1}$ to α_1, and q_j to q_{j+1} (or stop if $j = |Q| - 1$).

Let this machine start with the cell 0 containing α_1, and 1 containing γ. Then, for every q_j, the TM heads will be $|\Sigma| - 1$ times on cell 0, each time reading a different letter. So that except the cell 0, everything on the tape is in the same state . As the diagram of the Definition 5.4 commutes, this is also the case in the image of the tape i.e. all the changes in the CA have to take place in the windows of 0: $[\pi(0) - r, \pi(0) + r]$. Which means that $|Q|(|\Sigma| - 1)$ couples (q, α) can be encoded in that window, using the same CA states $s_1, \dots, s_m$ (otherwise, it would not be state-conserving). Easily, we have that the biggest number of possibilities of encoding with states $s_1, \dots, s_m$ over m cells is to have $\forall i \neq j, s_i \neq s_j$, and that number is $(m!)$. Hence $m! > |Q|(|\Sigma| - 1)$.

Define $k = \min\{n \in \mathbb{N} \mid \pi(n) - r > \pi(0) + r\}$, the first cell in $\mathcal{M}$ such that its window in $\mathcal{A}$ does not over-cross the one of 0. We will assume here that $\pi(k) - r = \pi(0) + r + 1$, otherwise, the proof is basically the same, except that we have to consider $k - 1$ and that the calculations are slightly more tricky.



For the reasons we explained previously, both windows have to be state-conserving in themselves. So that we have at most $(m!)^2$ possibilities of encoding over that space in $\mathcal{A}$. But that space potentially has to encode every possibilities of the states of cell 0 to k in the TM, with one head moving between this cell, that is $|Q||\Sigma|^{k+1}$. So that we finally get $m! > |Q|^{\frac{1}{2}}|\Sigma|^{\frac{k+1}{2}}$. As we want m to be minimal for any $|Q|$

and $|\Sigma|$, it has to be better than the one we gave for the Proposition 5.6, so that in general we necessarily have:

$$|Q|^{\frac{1}{2}}|\Sigma|^{\frac{k+1}{2}} \leq 2|\overline{Q}||\Sigma| \qquad \Longleftrightarrow \qquad |\Sigma|^{\frac{k-1}{2}} \leq 2|\overline{Q}|^{\frac{1}{2}}$$

Therefore $k = 1$, which means that there is not any information over-crossing in the TM, as in the construction we presented above. □

Thus our construction has an optimal order of magnitude, under the hypothesis that the simulation uses a mapping. Getting a smaller would require another concept of simulation. Nevertheless, this is not totally satisfying, because it does not preserve the reversibility. Indeed, if $\mathcal{M}$ is a reversible TM, and $\mathcal{A}_\mathcal{M}$ the automaton we obtain, $\mathcal{A}_\mathcal{M}$ is reversible on valid configurations (that is $\{\varepsilon_\phi(t) \mid t \in \mathscr{T}\}$), but in general is not reversible on any configuration.

5.2 Reversible simulation

We will here describe an other method of simulation preserving the reversibility on any configuration. This construction is inspired from work of Morita [7], using first a partitioned cellular automaton.

Definition 5.7. A *partitioned cellular automaton* (PCA) is a CA of local function f such that $r = 1$ and there exists L, C, R some finite sets such that :

- $S = L \times C \times R$
- $f : R \times C \times L \to L \times C \times R$

The idea is that the neighborhood is made of the center part of a cell, the right part of the left cell, and the left part of the right cell, so that at a step, one part (left, center, right) of a cell is only used once to compute the next configuration. Which leads to following lemma :

Lemma 5.8. *[7] If $\mathcal{A}$ is a PCA, then $\mathcal{A}$ is globally reversible if and only if it is locally reversible.*

In other terms, to have $\mathcal{A}$ reversible, it is sufficient to define f as a bijection, which is quite easy. Indeed, f is defined between two sets of same cardinal, so that we only have to define it bijectively on the subset that interests us, and then to extend it.

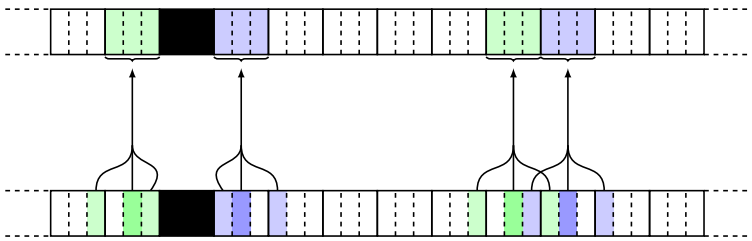
Briefly, we recall here the construction given in [7] to simulate a TM. We use an other definition of TMs, which is equivalent to the one we gave, in which head moves and tape writing are separated in two steps, and $\mathcal{F} = \{q_f\}$. Formally, we add a new symbol $\triangledown$, and for all $(q, a) \in Q \times \Sigma$, either $\delta(q, a) = q', b, \triangledown$, either $\delta(q, a) = q', a, \triangleleft/\triangleright$.

In our PCA, we have $L = R = Q \cup \{\bullet, \star\}$ and $C = L \times \Sigma$. For a given machine $\mathcal{M}$, f is defined as follow :

a) $f(\bullet, (q, a), \bullet) = (\bullet, (q', b), \bullet)$ if $\delta(q, a) = q', b, \triangledown$
b) $f(\bullet, (q, a), \bullet) = (q', (\bullet, a), \bullet)$ if $\delta(q, a) = q', a, \triangleleft$
c) $\forall q \in Q,\ f(\bullet, (\bullet, a), q) = (\bullet, (q, a), \bullet)$
d) $\forall q \in Q,\ f(q, (\bullet, a), \bullet) = (\bullet, (q, a), \bullet)$
e) $f(\bullet, (\star, a), \star) = (\bullet, (q_0, a), \bullet)$
f) $f(\bullet, (q_f, a), \bullet) = (\bullet, (\star, a), \star)$
g) $f(\star, (\bullet, a), \bullet) = (\star, (\bullet, a), \star)$
h) $f(\bullet, (\star, a), \bullet) = (\bullet, (\star, a), \bullet)$

Rules (a-b) describe the computation of $\mathcal{M}$, (c-d) manage the head moves, and (e-h) handle reversibility after the machine halts and before it begins.

To turn it into an SCCA, we encode each complete cell of the PCA as a permutation block, and separate every blocks with markers. We define the radius so each cell of a block has in its neighborhood the two whole neighbor blocks, and give the exact same rule as in the PCA for any valid configuration. Then, we only have to define the rules over wrong configurations, with regards to the local bijectivity in order to preserve the Lemma 5.8. If a configuration c is not valid, that is to say for all $t \in \mathscr{T}$, $\varepsilon_\phi(t) \neq c$, then it contains several heads or $\star$ in its neighborhood (which does not really matter, it handled by the extension of f to a bijection of $L \times C \times R$), or c contains a bad block. A block is considered bad as soon as it is not a permutation block between two markers. If a block is bad, then we just apply the identity to it. If a cell has a bad right (resp. left) neighbor, it uses its own left-cell (resp. right-cell) instead of the right (resp. left) one of the bad cell :



Proposition 5.9. *If $\mathcal{M}$ is a reversible TM, then the corresponding automaton $\mathcal{A}_\mathcal{M}$ is a reversible SCCA.*

Proof. There are two points to check. Firstly that $\mathcal{A}_\mathcal{M}$ is effectively state-conserving, but as a block is either a permutation, either a wrong one preserved by the identity, this is obvious. Secondly that the automaton is reversible. If the configuration is valid, this is the exact result given in [7]. If it is not, we only have to see if there is any ambiguity around bad cells. But as each part of cell is still used once and only once, the Lemma 5.8 still holds, and as the Turing rule is reversible, there is always exactly one image and one preimage. □

It is easy to check that we have indeed defined a valid simulation, that is with a way of going from a TM tape to a CA configuration, and the way back. If we take a universal Turing machine as input of our simulation, we will obviously have a CA Turing-universal. As a consequence, and due to the existence of universal reversible Turing machines, we get the following theorem :

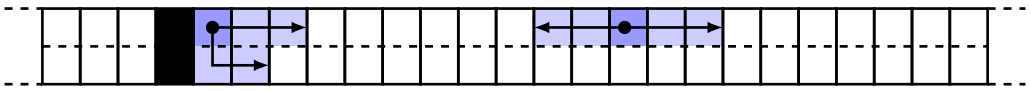
Theorem 5.10. *There exists a reversible Turing-universal SCCA.*

5.3 Intrinsically universal automaton

In this section, we will show how to simulate any CA by an SCCA in such a way that the simulation preserves reversibility. Thanks to this, we will be able to derive many usual results for general CA. We can easily adapt the definition 5.4 to define the simulation of a CA by another CA.

Let us denote by $\mathcal{A}$ the automaton we want to simulate, and by $\mathcal{A}_{\mathcal{SC}}$ the one we will build. The idea is still the same, a cell is encoded by a block, and blocks are separated by markers. In that way, we have the same complexity that we used to have for TM simulation, and actually we can slightly adapt the proof of the Proposition 5.6 to show that there is no really better way of doing it using a mapping.

As usual, the problem is to conserve reversibility when there are bad cells. For that, we will use a double-layer automaton, in which each cell is a couple, standing for both layers. The upper layer is ruled by the same rule than $\mathcal{A}$, and the lower one, by the symmetric one (left neighbors become the right ones, and conversely). On valid configuration, we only pay attention to the upper one, setting at the beginning the second one to configuration $0*$ and letting the layer evolve. Obviously, in that case, if $\mathcal{A}$ is reversible, so is $\mathcal{A}_{\mathcal{SC}}$. Now, when there is a bad cell, as usual we apply to it the identity, and the its neighbors use the low layer. The right neighbor of a bad cell will use its usual right neighbors, but its left neighbor is now its low part, the second left neighbor is the low part of its right neighbor, and so on:



Proposition 5.11. *If a CA $\mathcal{A}$ is reversible, so is its corresponding SCCA $\mathcal{A}_{\mathcal{SC}}$.*

Proof. Let c be any configuration for $\mathcal{A}_{\mathcal{SC}}$. Let us pick a valid connected component (that is without bad cell). There are three cases:

1. The component is the whole configuration, obviously it has exactly one preimage by $\mathcal{A}_{\mathcal{SC}}$.
2. The component is semi-infinite. Then, we can consider the two layers as one valid configuration who would have been fold down.
3. The component is folded. Then, we can consider the two layers as a ring, and so the component behaves exactly as a valid periodic configuration.

Hence c has exactly one preimage by $\mathcal{A}_{\mathcal{SC}}$. □

Definition 5.12. An automaton is said to be *intrinsically universal* if it is able to simulate any other CA.

As our construction clearly is a simulation in the sense of the Definition 5.4, it is obvious that if a CA is intrinsically universal, so is its simulation by an SCCA. Using the existence of an intrinsically universal CA [8], we get the same result for SCCA :

Theorem 5.13. *There exists an intrinsically universal SCCA.*

6 Undecidable problems

6.1 1-dimensional SCCA

One of the motivation of the previous constructions is to be easily able to reduce a lot of undecidability problem to SCCA. An immediate corollary of our last simulation, that preserves intrinsically universality, is the undecidability of this property :

Theorem 6.1. *The intrinsic universality problem is undecidable over SCCA.*

Proof. This problem is undecidable for CA in general [8]. As our construction preserves intrinsic universality, this is also true for SCCA. □

We will give here two other examples of decision problem for reversible CA that are reducible to reversible SCCA, namely periodicity and immortality problems.

A CA is said to be *periodic* if $\forall c \in S^{\mathbb{Z}}, \exists n \in \mathbb{N}$ *s.t.* $G^n(c) = c$. This problem is undecidable for reversible CA [3], and we can prove the same thing for SCCA. We should note previously that in cellular automata periodicity and uniform periodicity are equivalent. Indeed, let $\mathcal{A}$ be a periodic CA and suppose that for all $n \geq 1$, there exists $c_n \in S^{\mathbb{Z}}$ s.t. $G^n(c_n) \neq c_n$. Each c_n has a finite segment p_n which is mapped in n steps into a state which is different from the center of p_n. If a configuration c contains every p_n, then c is not periodic for $\mathcal{A}$, which is absurd.

Theorem 6.2. *It is undecidable whether a reversible SCCA is periodic.*

Proof. Let $\mathcal{A}$ be a reversible CA, $\mathcal{A}_{\mathcal{SC}}$ the reversible automaton obtained with the simulation (Φ, ε_ϕ) of section 5.3. We have to show that $\mathcal{A}_{\mathcal{SC}}$ is periodic if and only if $\mathcal{A}$ is. If there exists a configuration c which is not periodic for $\mathcal{A}$, clearly $\varepsilon_\phi(c)$ is not periodic for $\mathcal{A}_{\mathcal{SC}}$. Conversely, if $\mathcal{A}$ is periodic, $\mathcal{A}_{\mathcal{SC}}$ is also periodic on every valid configurations.

For the other ones, the reasoning is very similar to the proof we did for the Proposition5.11. We consider once more the connected components, each of them being periodic. Since periodicity implies uniform periodicity, there exists in $\mathcal{A}$ a common period n to every connected component, and thus any configuration is periodic. □

Let $\mathcal{H}$ be a set of halting states. A configuration c is said halting if there exists $i \in \mathbb{Z}$ s.t. $c_i \in \mathcal{H}$. A CA is said to be *mortal* if for all $c \in S^{\mathbb{Z}}$, there exists $n \in \mathbb{N}$ s.t. $G^n(c)$ is halting. It is also undecidable whether a reversible CA is immortal[3]. Clearly, as states are conserved all along the execution, this problem is trivial for a SCCA. Nonetheless, if we change a little the problem to have $\mathcal{H}'$ a finite set of finite patterns, and said that an automaton is *pattern-mortal* if there exists a segment $[a, b] \subset \mathbb{Z}$ s.t. $c_{[a,b]} \in \mathcal{H}'$, this problem turns to be undecidable:

Theorem 6.3. *The pattern-mortality problem is undecidable for reversible SCCA.*

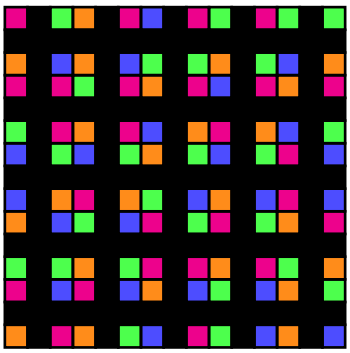
Proof. Let $\mathcal{A}$ (of global function G) be a reversible CA, $\mathcal{H}$ be a set of states of $\mathcal{A}$, and $\mathcal{A}_{\mathcal{SC}}$ the reversible automaton (of function $G_{\mathcal{SC}}$) obtained with the simulation (Φ, ε_ϕ) of section 5.3. For every pattern h of $\mathcal{H}$, we denote by $\overline{h}$ its corresponding pattern for $\mathcal{A}_{\mathcal{SC}}$ determined by ε_ϕ. If a cell is encoded over a block of size b, we define $\mathcal{B}$ as the finite set of bad blocks of size b, which are those that could not be found into a valid configuration. Finally we set $\mathcal{H}' = \{\overline{h} \mid h \in \mathcal{H}\} \cup \mathcal{B}$ and claim that $\mathcal{A}$ is mortal if and only if $\mathcal{A}_{\mathcal{SC}}$ is pattern-mortal.

Indeed, if $\mathcal{A}$ is immortal, there exists a configuration c such that for all $i \in \mathbb{Z}$, for all $n \in \mathbb{N}$, $G^n(c)_i \notin \mathcal{H}$. Then for all $i \in \mathbb{Z}$, for all $n \in \mathbb{N}$, $G^n_{\mathcal{SC}}(\varepsilon_\phi(c))_{[i,i+b]} \notin \mathcal{H}'$. Conversely, assume $\mathcal{A}$ is mortal. Let c be a configuration for $\mathcal{A}_{\mathcal{SC}}$. Either it is a valid configuration, and then its preimage is mortal for $\mathcal{A}$, and c is pattern-halting for $\mathcal{A}_{\mathcal{SC}}$. Either it is not, and then there exists an $i \in \mathbb{Z}$ such that $c_{[i,i+b]} \in \mathcal{B}$. In both cases, c is pattern-halting for $\mathcal{A}_{\mathcal{SC}}$, which proves that $\mathcal{A}_{\mathcal{SC}}$ is pattern-mortal. □

6.2 2-dimensional SCCA

We will not give here formal definitions, but most of the time, all the definitions we gave before can be extended to d-dimensional CA, replacing everywhere $\mathbb{Z}$ by $\mathbb{Z}^d$. If we only focus on 2-dimensional CA, adapting works that have been done for number-conserving cellular automata (the definition is the same, without using δ_α), we can show that the proposition 3.4 still holds. In addition, we still have the decidability of the state-conserving problem, with a result analogous to the proposition 4.1. Nevertheless, De Bruijn graph does not exist for 2-CA, we have no efficient algorithm *a priori*.

Clearly, we can simulate any 2-CA by a 2-SCCA, with the same kind of ideas we used previously. We encode each cell by a permutation block of a rectangular shape (to be able to tile the plane), and separate blocks by a border of marker. For instance, to encode a 2-CA until 24 states, we will use ■ as marker, and ■ ■ ■ ■ as states. The encoding of a configuration will look like this :



We know that reversibility is undecidable for 2-CA, and we would like to reduced this problem for 2-SCCA :

Conjecture 6.4. *Reversibility is undecidable for 2-SCCA.*

A simulation preserving the reversibility would be sufficient to prove that reversibility is still undecidable for 2-SCCA. We sketch here a construction similar to the one we exposed for 1-CA, but we do not know whether it is suitable (see Question 5).

Let A be a 2-CA, f its local function. We will use here a two-layer 2-dimensional cellular automaton, whose upper layer will contain $\mathcal{A}$, and the lower one will be used to handle bad neighborhood. So, each cell will contain a couple of states -one for each layer-, and every cell will as usual be encoded into a rectangular block of cell in the SCCA, whereas we separate blocks by a border of marker. The upper layer will be ruled by f, and the lower-one by the rule obtained applying a central symmetry to f. As soon as a block encounters a bad block in its neighborhood, it uses the lower layer instead, and conversely (see Figure 6).

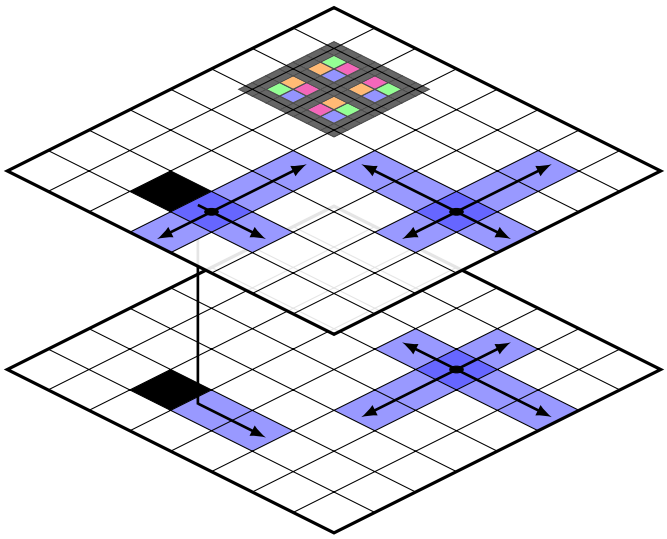


Figure 6: Simulation of a 2-CA into an SCCA

Obviously, if $\mathcal{A}$ is not reversible, neither $\mathcal{A}_{SC}$ is. But the converse remains a question :

Question 6.5. *Is $\mathcal{A}_{SC}$ reversible if $\mathcal{A}$ is ?*

7 Conclusion

We made here a first study of state-conserving cellular automata, and deal with some of their basic properties. Then we gave a characterization of state-conservation over De Bruijn graph, and implemented an efficient algorithm to list the classes $\mathcal{SC}(q, n)$. We also designed simulation preserving the reversibility to enhanced the universality of state-conserving cellular automata, and illustrate their utility by some few examples of reduction of decision problem. This might be a good first basis for any further work on this class of automaton in the future. Moreover, in addition to the question 6.5, one can wonder if the automaton $\mathcal{A}_1$ and $\mathcal{A}_2$ on Figure 4 are Turing universal, because they seem to behave like the rule 110 (which was proved universal by COOK), with a propagation of signals.

For a more personal aspect, this internship has enabled me to discover the theory of cellular automata starting from scratch. In that sense, it was particularly interesting, since it made me get familiar to CA thanks to some lectures note from a course Jarkko gives inT urku and some few papers. And it has above all given me basics knowledges in a new topic, with its related points of view and questions.

References

[1] Bastien Le Gloannec, *Around kari's traffic cellular automaton for the density classification task*, 2009.

[2] Jarkko Kari, *Theory of cellular automata: A survey*, Theoretical Computer Science **334** (2005), no. 1-3, 3 – 33.

[3] Jarkko Kari and Nicolas Ollinger, *Periodicity and Immortality in Reversible Computing*, Additional material available on the web at http://www.lif.univ-mrs.fr/ñollinge/rec/gnirut/.

[4] K Lindgren and M Nordahl, *Universal computation in simple one dimensional cellular automata*, Complex Systems (1990), no. 4.

[5] A. Moreira, *Universality and Decidability of Number-Conserving Cellular Automata*, ArXiv Nonlinear Sciences e-prints (2003).

[6] A. Moreira, N. Boccara, and E. Goles, *On Conservative and Monotone One-dimensional Cellular Automata and Their Particle Representation*, ArXiv Nonlinear Sciences e-prints (2003).

[7] Kenichi Morita, *Computation-universality of one-dimensional one-way reversible cellular automata*, Inf. Process. Lett. **42** (1992), 325–329.

[8] Nicolas Ollinger, *The intrinsic universality problem of one-dimensional cellular automata*, STACS 2003 (Helmut Alt and Michel Habib, eds.), Lecture Notes in Computer Science, vol. 2607, Springer Berlin / Heidelberg, 2003, pp. 632–641.