HAL
open science

Four states are enough!

Nicolas Ollinger, Gaétan Richard

To cite this version:

Nicolas Ollinger, Gaétan Richard. Four states are enough!. Theoretical Computer Science, 2011, 412 (1-2), pp.22-32. 10.1016/j.tcs.2010.08.018 . hal-00469841v2

HAL Id: hal-00469841

https://hal.science/hal-00469841v2

Submitted on 17 Aug 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Four states are enough!

N. Ollinger[a], G. Richard[a,b]

[a] *Laboratoire d'informatique fondamentale de Marseille (LIF), Aix-Marseille Université, CNRS, 39 rue Joliot-Curie, 13 013 Marseille, France*

[b] *Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen (Greyc), Université de Caen Basse-Normandie, CNRS, Campus Côte de Nacre, Boulevard du Maréchal Juin, 14 000 Caen, France*

Abstract

This paper presents a 1D intrinsically universal cellular automaton with four states for a first neighbors neighborhood, improving on the previous lower bound and getting nearer to the Turing universality bound. Intrinsic universality is discussed. Construction and proof rely on a combination of bulking techniques with programming using particles and collisions.

Key words: Cellular automata, intrinsic universality, bulking, particles and collisions

1. Introduction

Universal machines are, in some way, the simplest type of complex machines with respect to computational aspects: the sum of all possible behaviors. Universality is also a convenient tool in computation as a way to transform data, that can be further manipulated by the machine, into code. Therefore, computation universality is one of the basic ingredients of self-reproducing cellular automata first introduced by von Neumann [1] and in the subsequent works of Codd [2] and others[1] to achieve construction universality. Since then, universality has been studied on its own in the case of two-dimensional and one-dimensional cellular automata. For a detailed study, see the book chapter [3].

In the 60s and 70s, universality was mainly studied for high-dimensional cellular automata (2D, 3D). In this context, it seems natural to achieve universality by taking inspiration from real-world computers by simulating components of boolean circuits. Wires, gates, clocks, fan-out, signal crossing, *etc* are embedded into the configuration space of some local rule. Using these components, under the assumption that the family of elements is powerful enough, one obtains a universal cellular automaton under every reasonable hypothesis: from boolean circuits, one can wire finite state machines and memory to simulate sequential machines like Turing machines; or, one can wire finite state machines encoding the local rule of a cellular automaton and put infinitely many copies of that machine on a regular lattice, using wires to connect and synchronize the grid of automata to simulate the behavior of the encoded cellular automaton. This way, Banks [4, 5] was able to construct very small universal 2D cellular automata.

In the 70s and 80s, the study of cellular automata shifted to the one-dimensional space, motivated by the formal study of parallel algorithmics and formal languages recognition. In 1D, boolean circuits are no more a natural tool, but, as a configuration looks like a biinfinite tape, simulation of sequential machines like Turing machines is straightforward and provides the basis for a notion of computational (that is Turing) universality. This approach was developed by Smith III [6]. A major difficulty with Turing universality is the lack of a formal precise and general definition. The problem arises from two sources. First, a good commonly accepted formal definition of universality for Turing machines does not seem to exist. Second,

Email addresses: `nicolas.ollinger@lif.univ-mrs.fr` (N. Ollinger), `gaetan.richard@info.unicaen.fr` (G. Richard)

[1] although some later constructions are not universal

encoding finitely described Turing machine configurations into infinite configurations, giving a reasonable halting condition, and a decoding of the result, are delicate tasks. For a discussion on this formalization problem, see the study by Durand and Róka [7] of the universality of Conway's *Game of Life* [8].

In 1D, one can also consider simulating the cells of a simulated cellular automaton by blocks of cells of a simulator cellular automaton, leading to a notion of intrinsic universality. This notion, that coincides with the notion of boolean circuit universality in the case of 2D cellular automata [7, 3], was first pointed out in the one-dimensional case by Banks [4, 5] in the conclusion of his 2D construction, then rediscovered by Albert and Čulik II [9]. An attempt of a formal definition was given in Durand and Róka [7]. Whereas intrinsic universality implies Turing universality, one can prove that the converse is false, see Ollinger [10]. The main topic of this paper is to prove that four states are enough to achieve intrinsic universality in 1D with first neighbors neighborhood.

The paper continues as follows. In section 2, proper definitions of cellular automata and two definitions of intrinsic universality are proposed together with the main structural results. In section 3, the construction of small universal cellular automata is discussed. In section 4, the details of the construction of a 4 states intrinsically universal cellular automaton are given. In section 5, the more difficult question of non universality is considered.

2. Definitions and Properties

A d-dimensional *cellular automaton* is a tuple (d, S, N, f) where S is the finite set of states, $N \subseteq_{\text{finite}} \mathbb{Z}^d$ is the finite *neighborhood*, and $f : S^N \to S$ is the *local rule* of the cellular automaton. A *configuration* $c \in S^{\mathbb{Z}^d}$ is a coloring of the space $\mathbb{Z}^d$ by S. The *global function* $G : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ maps a configuration c to its image $G(c)$ by applying the local rule synchronously and uniformly according to N, *i.e.*, for all $z \in \mathbb{Z}^d$, $G(c)(z) = f(c_{|z+N})$. The set of configurations $S^{\mathbb{Z}^d}$ is endowed with the Cantor topology, *i.e.*, the product topology over $\mathbb{Z}^d$ of the discrete topology on S. This topology is metric, compact, and perfect. Under this topology, continuity corresponds to locality, as clopen sets correspond to sets of all configurations having a finite pattern in a given finite set, *i.e.*, if $C \subseteq S^{\mathbb{Z}^d}$ is a clopen, there exists $M \subseteq_{\text{finite}} \mathbb{Z}^d$ such that for all $c \in C$, $\left\{c' \in S^{\mathbb{Z}^d} \middle| c_{|M} = c'_{|M}\right\} \subseteq C$. Adding invariance by translation, one can enforce uniformity and characterize cellular automata. The *translation*, or *shift*, over S with translation vector $p \in \mathbb{Z}^d$, is the map $\sigma_p : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$, satisfying, for all $c \in S^{\mathbb{Z}^d}$ and $z \in \mathbb{Z}^d$, $\sigma_p(c)(z + p) = c(z)$.

Theorem 1 (Hedlund [11], Richardson [12])**.** *A map is the global function of a cellular automaton if and only if it is a continuous map commuting with translations.*

This theorem allows us to manipulate cellular automata by their global function, composing them, inverting bijective ones, taking cartesian products, *etc* being sure that the result is still the global function of a cellular automaton. For a proper study of cellular automata and their properties, one can read the survey of Kari [13].

A cellular automaton $\mathcal{A}$ is a *subautomaton* of a cellular automaton $\mathcal{B}$, denoted as $\mathcal{A} \sqsubseteq \mathcal{B}$, if there exists an injective map $\varphi : S_{\mathcal{A}} \to S_{\mathcal{B}}$ such that $\overline{\varphi} \circ G_{\mathcal{A}} = G_{\mathcal{B}} \circ \overline{\varphi}$, where $\overline{\varphi}(c) = \varphi \circ c$ for all configurations $c \in S_{\mathcal{A}}^{\mathbb{Z}^d}$. For all $m \in (\mathbb{Z}^+)^d$, $n \in \mathbb{Z}^+$ and $k \in \mathbb{Z}^d$, the $\langle m, n, k\rangle$*-rescaling* of a cellular automaton (d, S, N, f) is the cellular automaton with global function

$$G^{\langle m,n,k\rangle} = b^m \circ \sigma_k \circ G^n \circ b^{-m}$$

where $b^m : S^{\mathbb{Z}^d} \to \left(S^{\prod m}\right)^{\mathbb{Z}^d}$ is the bijective m*-packing map* satisfying, for all $c \in S^{\mathbb{Z}^d}$, $z \in \mathbb{Z}^d$ and $\alpha \in \prod m$, the equation $b^m(c)(z)(\alpha) = c(mz + \alpha)$, b^{-m} is the inverse of b^m, and $\prod m \subseteq \mathbb{Z}^d$ is the m-block satisfying $z \in \prod m$ if and only if for all $i \in \{1, \ldots, d\}$, $0 \leqslant z_i \leqslant m_i$.

The *injective bulking* quasi-order $\leqslant_i$ on cellular automata is defined thanks to subautomaton and rescaling. A cellular automaton $\mathcal{A}$ is simulated by a cellular automaton $\mathcal{B}$, denoted $\mathcal{A} \leqslant_i \mathcal{B}$, if there exists two rescalings $\langle m, n, k\rangle$ and $\langle m', n', k'\rangle$ such that $\mathcal{A}^{\langle m,n,k\rangle} \sqsubseteq \mathcal{B}^{\langle m',n',k'\rangle}$. This relation is a quasi-order with

interesting structural properties, see [10]. It admits a maximal equivalence class that captures the notion of intrinsic universality used in most constructions of the literature. Moreover, the maximal class $\mathcal{U}_i$ admits a stronger characterization.

Definition 1. A cellular automaton $\mathcal{A}$ is *intrinsically universal* with respect to injective bulking if, for all cellular automata $\mathcal{B}$, there exists a rescaling $\langle m, n, k\rangle$ such that $\mathcal{B} \sqsubseteq \mathcal{A}^{\langle m,n,k\rangle}$.

A cellular automaton $\mathcal{A}$ is a *mixautomaton* of a cellular automaton $\mathcal{B}$, denoted as $\mathcal{A} \trianglelefteq \mathcal{B}$, if there exists a map $\phi : S_{\mathcal{A}} \to 2^{S_{\mathcal{B}}}$ with disjoint images, *i.e.*, such that for all $s, s' \in S_{\mathcal{A}}$, $\phi(s) \cap \phi(s') = \emptyset$, such that $\overline{\phi} \circ G_{\mathcal{A}} \supseteq G_{\mathcal{B}} \circ \overline{\phi}$.

The *mixed bulking* quasi-order $\leqslant_m$ on cellular automata is defined thanks to mixautomaton and rescaling. A cellular automaton $\mathcal{A}$ is simulated by a cellular automaton $\mathcal{B}$, denoted $\mathcal{A} \leqslant_m \mathcal{B}$, if there exists two rescalings $\langle m, n, k\rangle$ and $\langle m', n', k'\rangle$ such that $\mathcal{A}^{\langle m,n,k\rangle} \trianglelefteq \mathcal{B}^{\langle m',n',k'\rangle}$. This relation is a quasi-order with interesting structural properties, see Theyssier [14]. As injective bulking is a refinement of mixed bulking, mixed bulking admits a maximal equivalence class that captures the notion of intrinsic universality used in most constructions of the literature. Moreover, the maximal class $\mathcal{U}_m$ admits a stronger characterization.

Definition 2. A cellular automaton $\mathcal{A}$ is *intrinsically universal* with respect to mixed bulking if, for all cellular automaton $\mathcal{B}$, there exists a rescaling $\langle m, n, k\rangle$ such that $\mathcal{B} \trianglelefteq \mathcal{A}^{\langle m,n,k\rangle}$.

Until now, all cellular automata known to be universal for mixed bulking can be shown universal for injective bulking (even if it is sometimes technical and painful).

Open Problem 1. *Does $\mathcal{U}_i = \mathcal{U}_m$?*

For a proper study of bulkings, their motivation and structural properties, see Delorme *et al.* [15, 16]. In the following, we will always consider injective bulking and the class $\mathcal{U}_i$ unless explicitly specified.

A nice property of intrinsic universality is that its formal definition captures constructions of the literature and provides a tool to prove non universality. A first natural question is to discuss decidability. Given a cellular automaton, can we decide if it is universal?

Theorem 2 (Ollinger [17])**.** *The problem of deciding whether an arbitrary cellular automata is intrinsically universal is undecidable.*

This theorem is proved by reducing the nilpotency problem of cellular automata on periodic configurations, obtained by Mazoyer and Rapaport [18]. This reduction works for both definitions of universality. As expected, there is no automatic method to test whether a cellular automaton is intrinsically universal.

3. Constructing Small Universal Cellular Automata

3.1. History

Intrinsic universality is a recursively enumerable property. To prove that a given cellular automaton is universal, it is sufficient to prove that it simulates a fixed intrinsically universal cellular automaton. The undecidability comes from the fact that the size of the blocks needed to simulate one cell can grow unrecursively large with respect to the given cellular automaton.

An intrinsically universal cellular automaton has to simulate cells: an entity computing a local rule and transmitting information to its neighbors. It is a straightforward exercise to construct an intrinsically universal cellular automaton with a small neighborhood and less than twenty states. For a general technique and examples, see [3]. How does one optimize the number of states with respect to a fixed neighborhood?

In dimension 2, with von Neumann neighborhood $\{(0,0),(1,0),(0,1),(-1,0),(0,-1)\}$, encoding boolean circuits, Banks was able to construct a 2 states universal automaton. Notice that finite configurations are encoded into ultimately periodic configurations by intrinsically universal cellular automata.

Theorem 3 (Banks [4, 5])**.** *There exists an intrinsically universal 2D cellular automaton with von Neumann neighborhood and 2 states.*

In dimension 1, a first technique, by Banks, consists of transforming a given universal cellular automaton of dimension 2 by having it simulating one dimensional cellular automata on a torus. The price to pay is either an extended neighborhood (neighbors are far from each other) or an extended set of states.

Corollary 4 (Banks [4, 5])**.** *There exists an intrinsically universal 1D cellular automaton with 2 states and a neighborhood of size 5.*

Even though boolean circuits simulation is difficult, it can still be achieve in dimension 1. By a careful design, we were able in [19] to construct a universal automaton with 6 states and first neighbors neighborhood $\{-1, 0, 1\}$. In this paper, we shall improve this result and give a universal automaton with first neighbors and only 4 states.

In the realm of Turing universality, the smallest universal automaton in dimension 1 has 2 states:

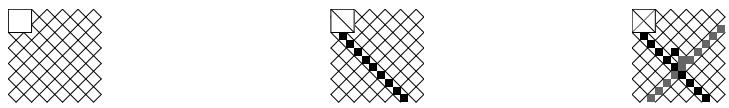
Theorem 5 (Cook [20])**.** *Rule 110 is Turing universal.*

The construction makes heavy use of particles and collisions to simulate special variation of tag systems, see Richard [21] for a formal proof using bidimensional tools for particles and collisions (as used for the 4 states automaton). Unfortunately, a careful study of the set of collisions used show that, contrary to what is claimed by Wolfram [22], the construction cannot be used to prove intrinsic universality of rule 110.

Open Problem 2. *Is rule 110 intrinsically universal?*

3.2. Particles and collisions

Many of previous constructions can be described using the analogy of particles and collisions to encode and compute information. To define these objects, we use the approach developed in [23] considering space-time diagrams as tilings of $S^{\mathbb{Z}^2}$ with local constraints. Therefore, we need to introduce some concepts from discrete geometry: a *coloring* $\mathcal{C}$ is an application from a subset $Sup(\mathcal{C}) \subset \mathbb{Z}^2$ to S. Such coloring is *finite* if $Sup(\mathcal{C})$ is finite. Three natural operations on colorings are *translation* of a coloring $\mathcal{C}$ by a vector $u \in \mathbb{Z}^2$ defined by $(u \cdot \mathcal{C})(z + u) = \mathcal{C}(z)$; *disjoint union* of two colorings $\mathcal{C}$ and $\mathcal{C}'$ with $\mathrm{Sup}(\mathcal{C}) \cap \mathrm{Sup}(\mathcal{C}') = \emptyset$ defined by $\mathcal{C} \oplus \mathcal{C}'(z) = \mathcal{C}(z)(\text{resp. } \mathcal{C}'(z))$ for all $z \in \mathrm{Sup}(\mathcal{C}')(\text{resp. } \mathrm{Sup}(\mathcal{C}'))$; at last, *restriction* of a coloring $\mathcal{C}$ to $D \subset \mathbb{Z}^2$ is denoted by $\mathcal{C}_{|D}$.



(a) A background (b) A particle (c) A collision

Figure 1: Base elements of constructions

Backgrounds (see Fig. 1a) are triplets $\mathfrak{B} = (\mathcal{C}, u, v)$ where: $\mathcal{C}$ is a finite coloring and u, v two non-collinear vectors ensuring that $\bigoplus_{i,j \in \mathbb{Z}^2} (iu + jv) \cdot \mathcal{C}$ is a space-time diagram (this space-time diagram is often also referred as $\mathfrak{B}$ when no confusion is possible). *Particles* (see Fig. 1b) are quadruplets $\mathfrak{P} = (\mathcal{C}, u, \mathfrak{B}, \mathfrak{B}')$ where $\mathcal{C}$ is a finite coloring, u a vector, $\mathfrak{B}_l$ and $\mathfrak{B}_r$ two backgrounds ensuring that $\mathcal{I} = \bigoplus_{k \in \mathbb{Z}} ku \cdot \mathcal{C}$ separates the plane in two 4-connected domains L and R (L being the left-one according to u) so that $\mathfrak{B}_{|L} \oplus \mathcal{I} \oplus \mathfrak{B}'_{|R}$ is a space-time diagram, where $\mathfrak{B}_{|L}$ denotes the restriction of the coloring induced by $\mathfrak{B}$ on L. A *collision* (see Fig. 1c) is a pair $(\mathcal{C}, L)$ where $\mathcal{C}$ is a finite coloring, L is a finite sequence of n particles $\mathfrak{P}_i = (\mathfrak{B}_i, \mathcal{C}_i, u_i, \mathfrak{B}'_i)$, satisfying the following conditions:

- consecutive particles on the list agree on their common background (for all $i \in \mathbb{Z}_n$, $\mathfrak{B}'_i = \mathfrak{B}_{i+1}$);
- particles and finite perturbation form a star ($\mathcal{I} = \mathcal{C} \oplus \bigoplus_{i \in \mathbb{Z}_n, k \in \mathbb{N}} ku_i \cdot \mathcal{C}_i$ cuts the plane in n 4-connected zones and for all $i \in \mathbb{Z}_n$, $\mathcal{C} \oplus \bigoplus_{k \in \mathbb{N}} (ku_i \cdot \mathcal{C}_i \oplus ku_{i+1} \cdot \mathcal{C}_{i+1})$ cuts the plane in two 4-connected zones. Let P_i be the one right of $\mathfrak{P}_i$);

- $\mathfrak{C} = \mathcal{I} \oplus \bigoplus_i \mathfrak{B}_{i|P_i}$ is a space-time diagram.

To describe easily complex space-time diagrams, one idea is to symbolize particles as lines and collisions as points, giving birth to a planar map called catenation scheme as the one in upper-left corner of Fig. 3. Formally, a *catenation scheme* is a planar map whose vertices are labeled by collisions and edges by particles which are coherent with regards to collisions. Since catenation schemes are symbolic representations, it is not clear that there exist associated space-time diagrams. In fact, to go back from a catenation scheme to a "real" space-time diagram, one must give explicit relative positions of collisions as, for example, by giving the number of repetitions for each particle (edge) of the scheme. Such a set of integers is called *affectation* and is *valid* if the resulting object is a space-time diagram. The main result given for catenation schemes is that set of valid affectations can be computed from finite catenation schemes.

Theorem 6 ([23])**.** *Given a finite catenation scheme, the set of valid affectations is a computable semi-linear set.*

In this paper, the constructed automaton is based on particles and collisions and thus, heavily relies on the methodology used for catenation scheme. However, since the particles and collisions are explicitly constructed, they are very small and posses many good combinatorial properties. Therefore we do not need the whole power of catenation scheme and can often give simpler arguments for our specific case. The rest of the paper is devoted to construct the automaton and prove it is intrinsically universal. This is done in two steps: first, we give the automaton and show, using catenation scheme, that it can somehow "simulate" the local behavior of any automaton. Then, we show how to assemble those local simulations into a global one.

4. Constructing a Small Universal Cellular Automaton

Using particles and collisions, we shall give in this section the construction of a 1D universal cellular automaton with first neighbors and only 4 states. The proof is divided in two parts: first, we give the general scheme of the simulation and then we study the details of the construction.

Theorem 7. *There exists an intrinsically universal 1D cellular automaton for mixed bulking with first neighbors neighborhood and 4 states.*

4.1. Building an elementary block

To build a universal cellular automaton, we design a widget able to simulate any transition of a cellular automaton using an *elementary block*. Before going on with this block, one point to notice is that it is sufficient to simulate one-way cellular automata to be intrinsically universal. Therefore, our elementary block is constructed such that it takes three inputs (one transition function and two arguments) and outputs three values (one transition function and two "copies" of the result).

To provide a good view of the automaton, all needed materials are depicted in Fig. 2 and Fig. 3. The first figure gives the local transition function and particles, the second one gives the catenation scheme along with all interesting extracts of corresponding space-time diagram. The rest of this section is devoted to describe and explain the contents of these two figures. The local transition function is fully depicted on the top of Fig. 2. In fact, it is not a real transition rule since some cases (denoted by interrogation marks) are not used and thus can take arbitrary values.

Even though giving the rules of our cellular automata would be sufficient, we will explain the intuition behind our local transition function as an aid to understanding the automaton. The first part of the explaination is devoted presenting background and particles used in the construction. Those structures are depicted in the corresponding part of Fig. 2. For each structure, we give a meaningful extract of space time, its name along with used local transition function cases. The formal definition can be trivially extracted from those diagrams.

One requirement of our cellular automaton is that, contrary to other known constructions, it does not use a uniform background but a bi-colored check-board (B). With this new approach, we need two states

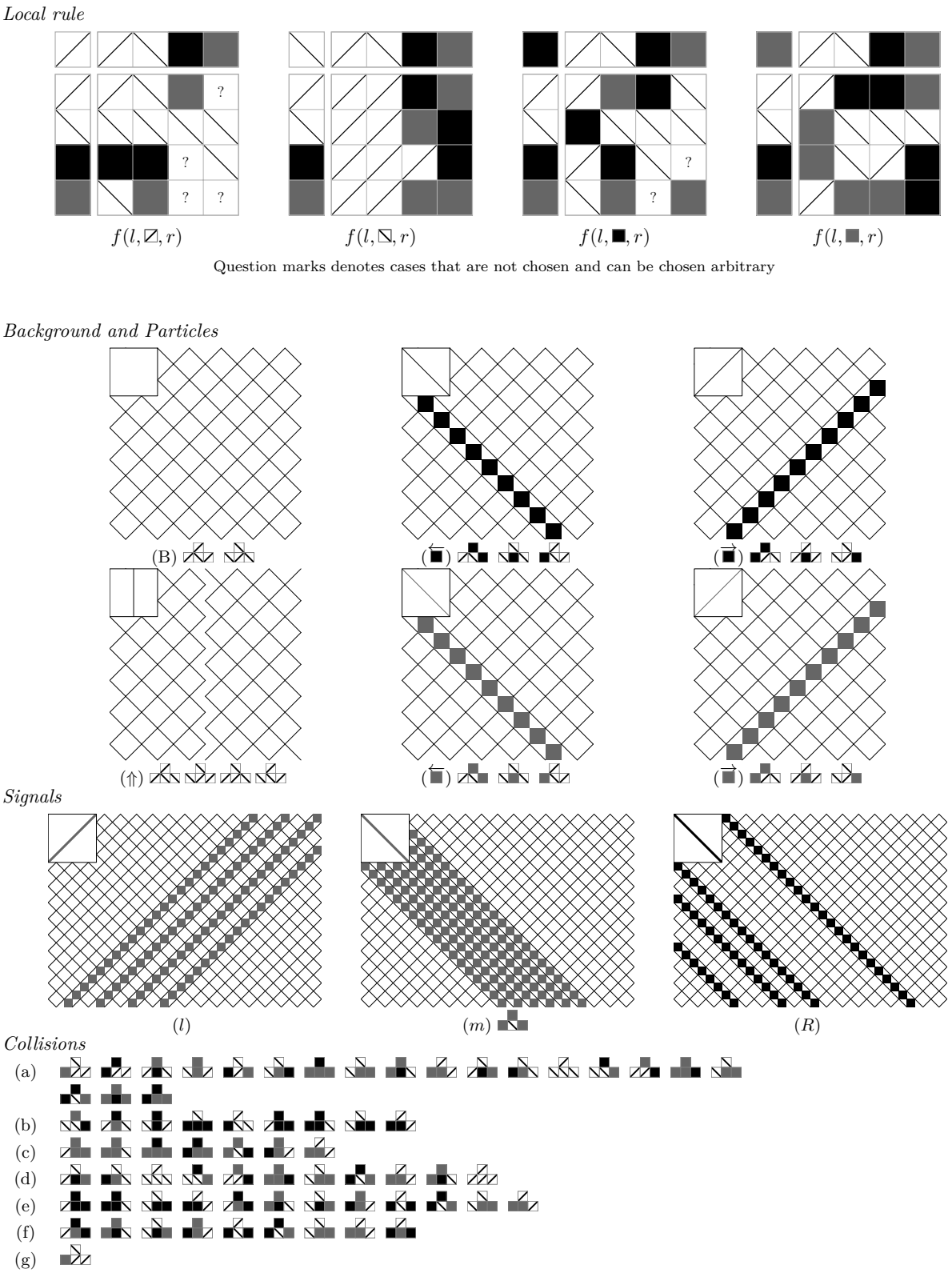


Figure 2: Local rule and particles

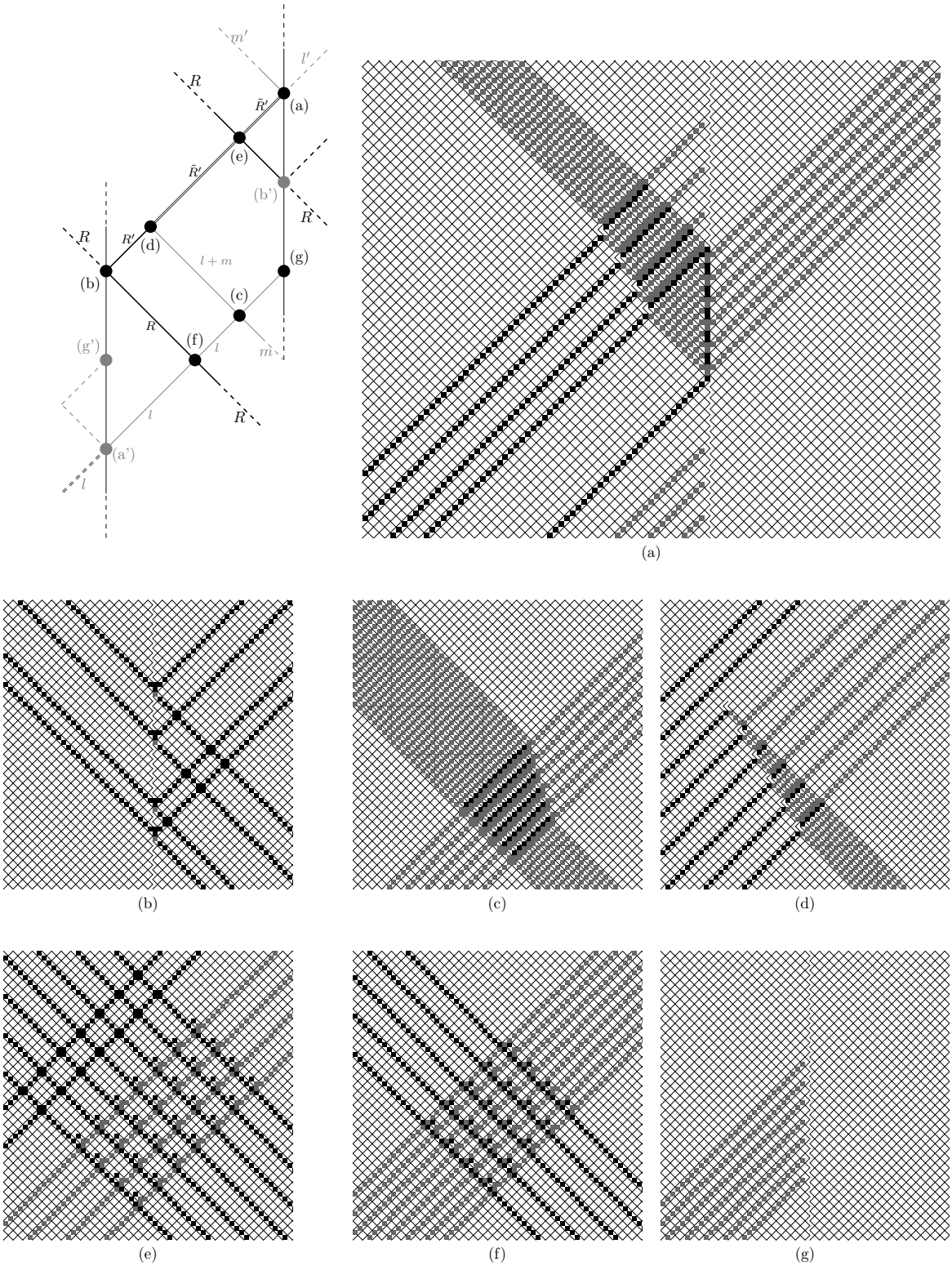


Figure 3: Constructed block and collisions

(instead of one) for the background but this allows us to have a greater range of particles since each remaining state can lead to two distinct particles according to its alignment within the background. Thus, with the two remaining states, we can construct four different particles ($\overleftarrow{\blacksquare}$, $\overrightarrow{\blacksquare}$, $\overleftarrow{\blacksquare}$, $\overrightarrow{\blacksquare}$). Furthermore, since the background has two different phases, one can construct an additional particle ($\Uparrow$) by taking advantage of the gap between the two phases. Those are the main particles used in the construction.

To encode information, the basic idea is to use groups of parallel particles that we call *signals*, depicted in the corresponding section of Fig. 2. In these groups, information can be encoded in two different ways: either by the number and type of the particles used or by relative position between those particles. Here, we use both approaches. As noted earlier, to simulate a one-way cellular automaton, we need to encode three main types of information: the *left state* (i.e., state of the left neighbor) is encoded in unary by the number of (regularly spaced) $\overrightarrow{\blacksquare}$ particles (signal l). In a similar manner, the *center state* is encoded in unary by the number of (regularly spaced) $\overleftarrow{\blacksquare}$ particles (signal m). The *local transition function* is encoded as an array of integers, with each integer encoded by spaces between a pair of particles $\overleftarrow{\blacksquare}$ (signal R): the j-th element of the array corresponds to the space between the j-th and the $j+1$-th particle of the signal. To be exact, the space is two plus the sum of numbers of $\boxslash$ between two consecutive $\blacksquare$ (for the previously mentioned states). This way of counting may seem a little obscure but is chosen to give nice formulae in the end of this section. In the rest of the paper, names of symbols are also used when speaking of encoded values.

During computation, other kind of signals appear: First, a mirror image of signal R (R') which encodes the same information using $\overrightarrow{\blacksquare}$ particles (going in the opposite direction). At last, an altered version of R', which we denotes as $\tilde{R}'$, appears. In this altered version, some of the leftmost $\overrightarrow{\blacksquare}$ particles are replaced by $\overrightarrow{\blacksquare}$.

With such construction, two main types of problem can occur. The first one is that a small shift of one particle can change the behavior of the collision. The second one is that two particles which are coming closer can interact with each others. For those reasons, one is often expected to give a set of exact positions for every particle and show that those positions do not present any of the previous problem. In our case, one very interesting property is that our construction cannot have the first problem due to the fact that particles can only be placed in one way relative to the background. Thus we can state the following lemma:

Lemma 8. *When two of the particles (or signals) described above collide, the occurring collision is always the same.*

Proof. As studied by J. P Crutchfield *et al.* [24], the number of ways two particles (or signals) can collide depends on the relative position between those two particles. Possible relative positions takes into account repetition vectors of those particles and constraints induced by backgrounds. In our case, repetition vectors are either $(1,1)$, $(0,2)$ or $(-1,1)$ which suggests the possibility of two distinct collisions. However, the background forbids one of those possibility leaving only one possible case. □

With this very strong property, we can forget about exact position of particles and focus on symbolic behavior. Let us go and construct the dynamic. The scheme of elementary block is depicted in Fig. 3 (along with extracts of all induced collisions). The block is made the following way: left and right borders are delimited by $\Uparrow$ particle. At the bottom, we have a left state signal l, a rule R and a center state signal m. The left state signal is going through the rule (collision f) and then collides with center state signal (collision c). This collision outputs an unused copy of the left signal to the right. This signal is erased when encountering the right border (collision g). Collision c also sends a signal encoding the sum of left and center state to the left. This new signal is encountering (collision d) the mirror copy R' (created when R crossed the left border during collision b). During collision d, the number of particle of R' altered[2] is exactly the encoded value (i.e., the sum of left and center signals). The signal embedding the sum is destroyed by the collision and the altered rule $\tilde{R}$ proceed to the right. After crossing another rule R (collision e), the altered rule $\tilde{R}$ collides with the right border and produces at the same time a new center state signal and a left state signal (located right of the border) in collision a.

[2]Last alteration is an erasing rather than a change of the particle but this does not change the behavior

Additional cases needed for each collision in the local transition function are made explicit at the bottom of Fig. 3. This scheme gives us a symbolic block which somehow "computes" the rule. Now, let us prove that this symbolic behavior does really correspond to a valid space-time diagram and look at the details of the computed function. The following proposition deals with the first problem by ensuring that, under reasonable conditions, the scheme of the symbolic block is a valid space-time diagram. Moreover, it gives some additional results on regularity of this block.

Proposition 9. *For any encoded rule R such that spaces between particles are even numbers larger than four, for any reasonable encoded value in left and middle state signal (i.e. both not null and their values are such that their sum is less than the number of integers encoded in R) the scheme of the symbolic block corresponds to a valid space-time diagram. Moreover, for a fixed R, the size of the space-time diagram does not depend on the encoded states and those blocks can tile the plane.*

Proof. First of all, the previous lemma ensures us that there is only one type of collision for each pair of particles; The non null condition ensures that all particles exist. The proof that collisions are valid can be directly deduced from extracts given in Fig. 3. Due to the fact that constraints are local, periodicity considerations are sufficient to prove the validity of the collisions occurring. In fact, it is sufficient to check the validity of one repetition of each periodic portion and the joint between different portion. The case of collisions c, e, f and g is easily dealt with since the perturbation inside the collision is periodic: adding one (or more) particles is the same as increasing the size of the periodic portion of the collision. Collision b requires space between particles $\overleftarrow{\blacksquare}$ to be greater than four. For collision d, the constraint is just that there are at least one particles $\overrightarrow{\blacksquare}$ left (i.e. sum of left and right values is less that the number of integers encoded in R). The last significant point is in collision a: since the vertical portion in the collision has periodicity $(0, 4)$, it requires that inter-space in $\tilde{R}$ being even (this inter-space is of course the same as in R). If all these constraints are respected (which is the case for our proposition) then the resulting catenation scheme can be implemented as a valid space-time diagram. That is, there exists a space-time diagram where particles and collisions are positioned in the same way as in the catenation scheme.

Now let us fix R, this implies that encoded values in states signals are bounded. Therefore, all signals are of bounded size and, up to increasing the size of the cell, they can be considered as objects with negligible size. With this, it is sufficient to take relative positions of signal as in the scheme of Fig. 3 to achieve same size blocks. More formally, if we chose for origin the collision a' and take s a value large enough relative to the size of collisions and signals; the positions of particles are chosen to ensure collisions around the following positions: f at (s, s), c at $(1.5s, 1.5s)$, g at $(2s, 2s)$, b at $(0, 2s)$, d at $(.5s, 2.5s)$, e at $(1.5s, 3.5s)$ and a at $(2s, 4s)$. Thus, it is possible to achieve a symbolic block with "approximate" positions. By the fact that particles always interact in the same way, such block correspond to a valid space-time diagram.

The last point is to prove that such space-time diagrams, associated to elementary blocks, tile the plane: that is, that the small approximations that we have left occurs in the previous paragraph can not add and become large. Before that point, the first remark is that symbolic blocks already tile the plane if we respect the exacts values. Now if we look in details at Fig. 3, borders are left untouched by all collisions and rule signals are only shifted by a constant when encountering a border. This implies that crossings of rule signals and borders (collisions b) form a regular grid on the plane. Finally, the position of all other collisions only depends on a small number of neighbor points on this grid. It is directly visible for e and a. Positions of collisions f, g and c also depend on a' but its position is fixed by the previous case. At last, collision d depends on c which was already treated. □

With this result, we have an elementary block able to do simple computation according to a rule R and which can tile the plane. Before combining those elementary blocks in the next section, we must first study exactly which function is computed by our block. For the same periodicity reason as previously, it is sufficient to look at what happens in the case of collision a in Fig. 3.

Lemma 10. *Let $R(i)_{1 \leq i \leq N}$ be the array of N integers encoded in R and x_l (resp. x_m) be the one encoded in l (resp. m); then the block leaves R unchanged and outputs m' and l' with encoded values respectively $R(x_l + x_m) + x_l + x_m - (N + 1)$ and $R(x_l + x_m)/2$.* □

Thus we can construct a block parametrized by some rule R which is represented as a sequence of integers. As long as all elements of this sequence are even integers greater that four. The block behaves as denoted in the previous lemma for any pair of input encoding values x_l,x_m such that the sum $x_m + x_l$ is less than the number of integer in the sequence R. Note that we do not care for the moment if the input condition is preserved on the outputs.

This block computation is like a cell in a cellular automaton except that it sends different values for left and right states. This difference prevents us from giving a direct simulation and requires quite additional work to get rid of this problem and be able to iterate the use of elementary blocks. A better way to overcome this problem would be to alter the automaton or use unused cases in the local transition function to ensure equality of outputs. However, for the moment, we do not manage to do such a thing. Therefore an alternative (and quite combinatorial) solution is presented in the following to use this elementary blocks in intrinsic simulation.

4.2. Combining blocks

To simulate a cellular automaton using elementary blocks, the idea is to iterate them as depicted in Fig. 5 and choose a rule which emulates the behavior of the cellular automaton, the states being encoded into inputs and outputs of the blocks. Although we can make a wide range of possible rules for the blocks, restrictions on the form of the outputs is too tight for us to directly encode the transition rule inside it (or more precisely, we do not currently know how to do this).

To overcome this difficulty, we choose to group elementary blocks by two in the way depicted in Fig. 4: we use the right output of the first one as the input for the second one. In this way, we have a new basic block which has three inputs x_l,x_m, $x_{\tilde{m}}$ and three outputs $x_{l'}$, $x_{m'}$ and $x_{\tilde{m}'}$. The relation between inputs and outputs is still fully determined by the rule R in the following way: denoting as l^0 the intermediate value, we have that $x_{m'} = R(x_l + x_m) + x_l + x_m(N+1)$, $x_{l^0} = R(x_l + x_m)/2$, $x_{\tilde{m}'} = R(x_{l^0} + x_{\tilde{m}}) + x_{l^0} + x_{\tilde{m}}(N+1)$ and lastly $x_{l'} = R(x_{l^0} + x_{\tilde{m}})/2$.

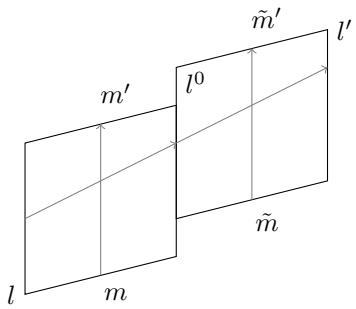


Figure 4: Symbolic chaining of elementary blocks (encoding of state is omitted)

At this point, we want this double blocks to "emulate" the local transition function of any cellular automaton. In fact, since there exists intrinsically universal one-way cellular automata, we can restrict ourselves without loss of generality to the case of one-way cellular automata $(1, S, \{0, 1\}, f)$ with $f : S^2 \to S$.

As we only need two inputs, we choose to search the encoding in which the values of $\tilde{m}$ are all equal to a fixed constant c. To maintain this choice throughout the iterations of blocks, we must have that $c = x_{\tilde{m}'} = x_{\tilde{m}}$ that is $R(x_{l^0} + x_{\tilde{m}}) + x_{l^0} + x_{\tilde{m}}(N+1) = \tilde{m}$ which can be restated as $R(x_{l^0} + c) = (N+1) - x_{l^0}$. Now, we must find two encodings ξ_l and ξ_m from $\mathbb{N}$ to Q such that if $\xi_l(x_l) = q$ and $\xi_m(x_m) = q'$ then the block satisfies $\xi_l(l') = \xi_m(m') = f(q, q')$.

Those two encodings are based on a vision of integers as written in binary. First, let us assimilate the set S as strictly positives integers $\{1, 2, 3, \ldots, |S|\}$. Thus, if we denote as k the value $\lceil \log(|S| + 1) \rceil$ any $s \in S$ can be written as the binary word $s_1 s_2 \ldots s_k$.

Then, we choose to work with integers represented by binary elements of size $(4+8k+3)$ (that is between 0 and $2^{4+8k+3} - 1$). Informally, those binary representations can be understood as follows: the first 3 bits

(*header*) serve to recognize the kind of signal encountered; each of the k blocks of 8 bits (*digits*) is used to encode one bit of the state; lastly the remaining 4 bits are used to ensure that constraints on parity are respected (*footer*). With this representation, let $s = s_1 s_2 \dots s_k \in S$, we define $\xi_l(x) = s$ if and only if x is of the form 0101 $(\perp\perp s_i 0 \perp\perp 00)_{1 \leq i \leq k} \perp\perp\perp$ where $\perp$ denotes an arbitrary bit. The same way, we define $\xi_m(x) = s$ if and only if x is on the form 0000 $(\perp\perp 00 \perp\perp s_i 0)_{1 \leq i \leq k} \perp\perp\perp$.

The last point is to prove that we can choose the function R such that, for any pair (x_l, x_m) such that $\xi_l(x_l) = s$ and $\xi_m(x_m) = s'$, we have $\xi_l(x_{l'}) = \xi_m(x_{m'}) = f(s, s')$. For this, let us assume the following arbitrary restrictions: we choose to set the number N (which is the domain of R) to $2^{4+8k+3} - 1$ and the value $c = x_{\tilde{m}'} = x_{\tilde{m}} = 2^{4+8k+2}$ as depicted in Tab. 1.

Let us now study the form of values present inside the block. Those values are depicted in Tab. 1. The first remark is that in the sum $(x_l + x_m)$; it is possible to retrieve both the value of s and s' in an unambiguous way. Thus the result of our output can take the value $f(s, s') = t_1 t_2 \dots t_k$. The choice of the value $R(x_l, x_m)$ can be divided in two parts: we chose it to be on the form $01XX\ (YYXXYYXX)100$; X bits will serve to set the value of $x_{m'}$ and Y bits the one of $x_{l'}$. Since $x_{m'} = R(x_l + x_m) + x_l + x_m - (N+1)$, we can see that the X bits allows us to set the corresponding digits of $x_{m'}$ ensuring that $\xi_m(x_{m'}) = t$.

For the Y part, if we set it as indicated in the table, then we can see that the sum $(x_{l_0} + x_{\tilde{m}})$ begins with 11 as the two first bits and thus is ensured to be distinct from any valid $(x_l + x_m)$ sum. Therefore, it is possible to choose $R(x_{l_0} + x_{\tilde{m}})$ as the value indicated previously to respect the constraint $x_{\tilde{m}} = x_{\tilde{m}'}$. Moreover with our previous choice, the output l' does satisfy $\xi_l(x_{l'}) = t$.

As a final note, it is important to note that neither of our encoded values can be zero (for x_m, one must recall that s cannot be zero).

	header	(	digits			)	footer	
N	1111	(11	11	11	11	)	111	
x_l	0101	($\perp\perp$	$s_i 0$	$\perp\perp$	00	)	$\perp\perp\perp$	
x_m	0000	($\perp\perp$	00	$\perp\perp$	$s'_i 0$	)	$\perp\perp\perp$	
$x_{\tilde{m}}$	1000	(00	00	00	00	)	000	
$x_l + x_m$	$01xx$	($\perp\perp$	$s_i \perp$	$\perp\perp$	$s'_i \perp$	)	$\perp\perp\perp$	where $xx = 01$ or 10
$R(x_l + x_m)$	$10 \bullet\bullet$	($\bar{t}_i 1$	$\bullet\bullet$	11	$\bullet\bullet$	)	100	
$x_{m'}$	0000	($\perp\perp$	00	$\perp\perp$	$t_i 0$	)	$\perp\perp\perp$	
x_{l^0}	$010\perp$	($\perp\bar{t}_i$	$1\perp$	$\perp 1$	$1\perp$	)	$\perp 10$	
$N - x_{l^0} + 1$	$101\perp$	($\perp t_i$	$0\perp$	$\perp 0$	$0\perp$	)	$\perp 10$	
$x_{l'}$	0101	($\perp\perp$	$t_i 0$	$\perp\perp$	00	)	$\perp\perp\perp$	

$\perp$ denotes arbitrary value and $\bullet$ value to be fixed.

Table 1: Encoding of states inside values

Here, one can notice that we use a mixed bulking since we have some "junk" which we cannot control but end up not interfering with the simulation. One relevant problem is to know whether we can get rid of this junk and thus achieve injective bulking. In this case, we conjecture that such achievement would be possible by using the following scheme: do one step of computation then take some time to get rid of the junk. However, we have not pursed this reflexion any further since the workload to achieve this result seems too high with respect to the interest of the result.

Thanks to the previous encoding, we can now simulate any cellular automaton in the following way: we input blocks with the rule corresponding to the automaton and containing an encoding of each state of the initial configuration. As each block simulates one cell, we can "see" the evolution of the simulated cellular automaton in the space time diagram.

However, the blocks are not aligned horizontally but are organized in a diagonal way (see for example Fig. 5). This problem can be worked around with a well-known construction. The idea is the following: instead of directly simulating an initial automaton $(1, S, \{-1, 0\}, f)$ we use the automaton $(1, S' = \{B\} \cup$

$S_{\rightarrow} \cup S_{\uparrow} \cup S_{\nearrow\!\!\!\!\nearrow}, f')$ defined by for all $s, s' \in S$:

$$\begin{array}{l} f'(s_{\rightarrow}, B) = s_{\rightarrow} \\ f'(B, s_{\uparrow}) = s_{\uparrow} \\ f'(s_{\nearrow\!\!\!\!\nearrow}, B) = s_{\rightarrow} \\ f'(B, s_{\nearrow\!\!\!\!\nearrow}) = s_{\uparrow} \\ f'(s_{\rightarrow}, s'_{\uparrow}) = f(s, s')_{\nearrow\!\!\!\!\nearrow} \\ f'(x, y) = B \text{ otherwise} \end{array}$$

Informally, the intermediate automaton runs the following way: states are replaced by signals going either vertically ($\uparrow$) or diagonally ($\rightarrow$). When both signals meet, we make one step of computation and output two copies of the result (one going up and one going to the upper-right). This way, it is possible to simulate the initial automaton with any wanted scaling and correct the slope by scaling until we find a block at the same level (see Fig. 5). Therefore, one cell s of the initial cellular automaton can be simulated with cells on the forms $s_{\nearrow\!\!\!\!\nearrow} B^m$ for any fixed $m \in \mathbb{N}$.

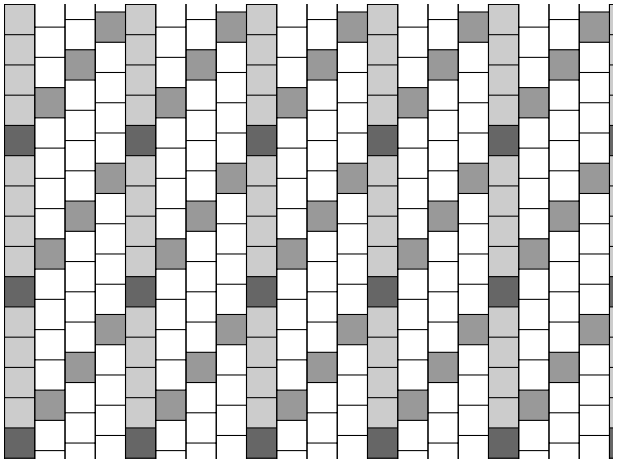


Figure 5: Straightening simulation (light grey squares represents blocks encoding $\uparrow$ states, medium $\rightarrow$ and dark $\nearrow\!\!\!\!\nearrow$ ones.)

5. Conclusion

We have constructed a four states intrinsically universal cellular automaton with first neighbors neighborhood. Thanks to the formal definition, a proof of non universality is possible for given cellular automata. However, due to the undecidability of the property, there is no general method. Still, several invariants can be used, coming from the study of the bulking quasi-orders.

A first interesting problem is the pattern problem. Given a cellular automaton, a finite configuration and a pattern, the question is to decide if the pattern appears in the orbit of the configuration. For Turing universal, thus also for intrinsically universal cellular automata, the problem has maximal complexity. For simpler cellular automata it is decidable.

Theorem 11. *The pattern problem is undecidable for any intrinsically universal cellular automaton.*

A second interesting problem is the verification problem. Given a cellular automaton, a finite pattern and a state, the question is to decide if the state is obtained by iterating the local rule on the pattern until it consists of only one state. A simple simulation permits to test this property in polynomial time. By adapting the classical proof of Ladner [25], one can prove that it is P-complete in general. For simpler cellular automata, the complexity can be lower.

Theorem 12. *The verification problem is P-complete for any intrinsically universal cellular automaton.*

Unfortunately, in the case of rule 110, the pattern problem is undecidable and the verification problem has been shown P-complete by Neary and Woods:

Theorem 13 (Neary and Woods [26])**.** *Rule 110 is P-complete.*

Open Problem 3. *Develop more tools to prove non universality.*

References

[1] J. von Neumann, Theory of Self-Reproducing Automata, University of Illinois Press, Urbana, Ill., 1966, (A. W. Burks, ed.).

[2] E. F. Codd, Cellular Automata, Academic Press, New York, 1968.

[3] N. Ollinger, Universalities in cellular automata, in: G. Rozenberg, T. Baeck, J. Kok (Eds.), Handbook of Natural Computing, Springer, Berlin, (to appear).

[4] E. R. Banks, Universality in cellular automata, in: Symposium on Switching and Automata Theory (Santa Monica, California, 1970), IEEE, 1970, pp. 194–215.

[5] E. R. Banks, Information processing and transmission in cellular automata, Ph.D. thesis, Massachusetts Institute of Technology (1971).

[6] A. R. Smith, III, Simple computation-universal cellular spaces, Journal of the ACM 18 (1971) 339–353.

[7] B. Durand, Z. Róka, The game of life: universality revisited, in: M. Delorme, J. Mazoyer (Eds.), Cellular automata (Saissac, 1996), Kluwer Acad. Publ., Dordrecht, 1999, pp. 51–74.

[8] E. R. Berlekamp, J. H. Conway, R. K. Guy, Winning ways for your mathematical plays. Vol. 2, Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1982, games in particular.

[9] J. Albert, K. Čulik, II, A simple universal cellular automaton and its one-way and totalistic version, Complex Systems 1 (1) (1987) 1–16.

[10] N. Ollinger, Automates cellulaires : structures, Ph.D. thesis, École Normale Supérieure de Lyon (2002).

[11] G. A. Hedlund, Endormorphisms and automorphisms of the shift dynamical system, Mathematical Systems Theory 3 (1969) 320–375.

[12] D. Richardson, Tessellations with local transformations, Journal of Computer and System Sciences 6 (1972) 373–388.

[13] J. Kari, Theory of cellular automata: A survey, Theoretical Computer Science 334 (2005) 3–33.

[14] G. Theyssier, Automates cellulaires : un modèle de complexités, Ph.D. thesis, École Normale Supérieure de Lyon (2005).

[15] M. Delorme, J. Mazoyer, N. Ollinger, G. Theyssier, Bulking I: an abstract theory of bulking, `http://hal.archives-ouvertes.fr/hal-00451732/` (accepted to *Theoretical Computer Science*).

[16] M. Delorme, J. Mazoyer, N. Ollinger, G. Theyssier, Bulking II: Classifications of cellular automata, `http://hal.archives-ouvertes.fr/hal-00451727/` (accepted to *Theoretical Computer Science*).

[17] N. Ollinger, The intrinsic universality problem of one-dimensional cellular automata, in: H. Alt, M. Habib (Eds.), Symposium on Theoretical Aspects of Computer Science (STACS'2003), Vol. 2607 of Lecture Notes in Computer Science, Springer, Berlin, 2003, pp. 632–641.

[18] J. Mazoyer, I. Rapaport, Global fixed point attractors of circular cellular automata and periodic tilings of the plane: undecidability results, Discrete Mathematics 199 (1-3) (1999) 103–122.

[19] N. Ollinger, The quest for small universal cellular automata, in: P. Widmayer, F. Triguero, R. Morales, M. Hennessy, S. Eidenbenz, R. Conejo (Eds.), International Colloquium on Automata, languages and programming (Málaga, Spain, 2002), Vol. 2380 of Lecture Notes in Computer Science, Springer, Berlin, 2002, pp. 318–329.

[20] M. Cook, Universality in elementary cellular automata, Complex Systems 15 (2004) 1–40.

[21] G. Richard, Rule 110: universality and catenations, in: B. Durand (Ed.), Symposium on Cellular Automata Journées Automates Cellulaires (JAC'2008), MCCME Publishing House, Moscow, 2008, pp. 141–160.

[22] S. Wolfram, A new kind of science, Wolfram Media Inc., Champaign, Ilinois, US, United States, 2002.

[23] N. Ollinger, G. Richard, Collisions and their catenations: Ultimately periodic tilings of the plane, in: IFIP-TCS, Vol. 273, Springer Boston, 2008, pp. 229–240.

[24] W. Hordijk, C. R. Shalizi, J. P. Crutchfield, Upper bound on the products of particle interactions in cellular automata, Phys. D 154 (3-4) (2001) 240–258.

[25] R. E. Ladner, The circuit value problem is log space complete for p, SIGACT News 7 (1) (1975) 18–20.

[26] T. Neary, D. Woods, P-completeness of cellular automaton rule 110, in: Proceedings of ICALP 2006, Vol. 4051 of Lecture Notes in Computer Science, Springer, Berlin, 2006, pp. 132–143.