Universidad de Chile
Facultad de ciencias físicas y matemáticas
Departamento de ingeniería matemática

Cellular automata as a model of parallel complexities

Tesis para optar al título de grado de doctor en ciencias de la ingeniería, mención modelación matemática

Pierre-Etienne Meunier

Profesor guía
Ivan Rapaport

Profesor co-guía
Jacques-Olivier Lachaud

Miembros de la comisión
Eric Goles
Martin Matamala
Pedro Paulo Balbi De Oliveira
Guillaume Theyssier
Nicolas Schabanel
Damien Woods

Abstract

In this thesis, we investigate the links between questions on dynamical systems, in particular cellular automata, and the theory of complexity. We compare the classical notion of *reduction*, used to define the usual complexity "classes", to the notion of *simulation* in cellular automata. The idea of simulation is more topological, and better understood, than reductions and completeness; moreover, for some definitions of a simulation, *universal* cellular automata (that is, cellular automata able to simulate any other) are known.

After a short introduction to cellular automata in chapter 1, we present in chapter 2, a novel approach on the links between cellular automata and another model of parallelism: *circuits*. The previous works on this topic focused mainly either on showing that a particular cellular automaton had a complex dynamics, or on showing that very strong hypotheses on the rule of a cellular automaton yielded simple dynamics. Our approach is dual to these, as it tries to characterize the classes of cellular automata described by an hypothesis on the algorithmic complexity of predicting their dynamics. We show results on the cellular automata with a constant number of dependencies, and on the cellular automata whose local rule is monotonic and planar. The definitions and comparison results presented in this chapter allow to speak of complexity classes in terms of simulation instead of reduction, that is, to see *computing* systems as *dynamical* systems.

The rest of this thesis (chapters 3 and 4) focuses on a different notion of complexity: *communication complexity*. We generalize the previous works to a more general notion of problem, and show a very generic method to prove that a cellular automaton is not intrinsically universal; that is, that a given cellular automaton does not simulate any other. This contrasts with the situation in the Turing world, where the notion of universality and reduction is too general to allow for such a technique. Moreover, the framework of communication complexity allowed us to construct simple protocols, solving problems of high Turing complexity (PSPACE-complete, Π_0^1-complete), where the classical notions of reductions and completeness would not tell much about intrinsic universality. Using these tools, we present the first proof that a large number of elementary cellular automata are not universal for non-trivial reasons.

Finally, we compare two different definitions of simulations, namely the simulation by *sub-system* and by *factor*. While the first one is best understood, and universal cellular automata are known, the second one remains quite mysterious, and the existence of a universal cellular automaton is a long-standing open problem of the field. We show how to generalize our work, using non-determinism and randomization, to the simulation by factor, and conjecture that this generalization is required, which opens a new and unexpected connection between this problem and well-known results of communication complexity. We conclude with a proof showing that the long-term behavior of cellular automata intrinsically universal by sub-system simulations is much less constrained than for factor simulation, thus showing that the requirements of factor simulation may be too high for an intrinsically universal cellular automaton to exist.

to Georges

Acknowledgments

A PhD. thesis is a scientific adventure, it is also, and maybe most importantly, a human adventure. I am deeply indebted to all the people I have met during these three years passed between France, Chile and Brazil, and who allowed this experience to escape the field of a simple technical experience. First of all, I would like to thank my advisors, Guillaume Theyssier and Ivan Rapaport, who have always been there to help me go through my doubts and my difficulties, be they scientific or administrative.

Guillaume has introduced me to the beautiful theories described and used in this work. His enthusiasm and his intuitions, always flowing on a perfectly measured amount of formalism and mathematical rigor, will remain a great source of inspiration to me. His openness to other forms of knowledge than just mathematics, or even science, is a quality that few people possess to this extent. His teachings on the nature of research, of collaboration, of many aspects of the academic world, were invaluable.

On the Chilean side, Ivan, with his deep insights on the aesthetic value of mathematics, with his critical views on the general direction of a work, was the right complement to Guillaume. He has taught me to keep in mind the final goal, the general direction of my work, even in the moments of profound technical difficulties. I will recall our first discussions about competition in science; his ideas on the relative futility of purely technical achievements in mathematics have been, and will be, a constant guide to my approach of science.

Jacques-Olivier Lachaud is responsible for the existence of this work, for the time I have enjoyed to do it, for the resolution of most of the complex and absurd aspects of our tedious french bureaucracy that I have faced during these three years. Scientific research is an activity where competition, stress and material preoccupations do not belong. Thanks to Jacques-Olivier, I have been able to taste these peaceful moments where the only concern is the current conjecture. I would like to thank him for his advice, for his teaching of academic diplomacy, for his constant support.

I received a precious help in my life in Chile from Eric Goles. His collaboration on designing protocols, and the inspiration he gave me on many aspects of this work, have been as priceless as his expertise of the good places of Santiago.

Thanks to Nicolas Schabanel and Damien Woods, who took the time to proofread this manuscript, and pointed many mistakes and typos. Their constructive remarks and encouragements were of great help, and a warm welcome into the academic world. Thanks to the other members of my jury, Martin Matamala and Pedro Paulo Balbi De Oliveira, for their interest in my work, and the time they have taken to come to my defense.

I would also like to use this page to thank all my colleagues of the LAMA, and in particular the LIMD team: Florian Hatat, Tom Hirschowitz, Pierre Hyvernat, Xavier Provençal, Christophe Raffalli, Laurent Vuillon. I will miss our conversations, our projects, our blackboards, from mathematics to economics, from music to typography, from beers to coffees. The kind of places

this laboratory belongs to is too rare in this world, both for the exceptional quality of life it offers, and the excellence of its members, as scientists and as friends.

During my stays in Chile, I have also had the opportunity to meet brilliant colleagues, and good friends. Raimundo Briceño, with whom I wrote an article; Jairo Navarrete, who first taught me salsa. Clara Fittipaldi, Luis Fernando, Natalia Ruíz, Oscar Agudelo Rico, Natacha Astromujoff, Karina Vilches, who taught me spanish, the latin way of life, football, and have given me uninterrupted *buena onda* since the first day I got to know them.

While in Chile, aside from working on the circuits lower bounds for problems on cellular automata, I also came to know my wife, Elisa. I hope to be able some day to give her the same constant patience, support, and loving attention, that she has given me during these years. I know of no equivalent to her resistance to despair, her tenacity and lucidity. Our constant discussions on all kinds of fields have been, and are to me, a permanent source of hope, knowledge and happiness. Thanks to her.

Finally, I would like to thank my family, my parents and my brothers, all the friends who shared my life during these three years, bringing their humanity, friendship, help and conviviality. I would like to thank in particular Andréanne, Rodrigo, Stephanie, Andrés, Javier Alejandro, Raphaël, Rafael, Lucía, Alexis. Thanks to Itamar, Neuza, Rosângela, Alcino, Rita, for their kind hospitality in Brazil.

Table of contents

Introduction

The intended goal of this manuscript is to build bridges between two definitions of complexity. One of them, called the *algorithmic* complexity, is well-known to any computer scientist as the difficulty of performing some task, such as sorting or optimizing the outcome of some system. The other one, etymologically closer from the word "complexity", is about what happens when many parts of a system are interacting together. Just as cells in a living body, producers and consumers in some non-planned economies, or mathematicians exchanging ideas to prove theorems.

On the algorithmic side, the main objects that we are going to use are two models of computations, one called *communication protocols*, and the other one *circuits*. Communication protocols are found everywhere in our world, they are the basic stone of almost any human collaboration and achievement. The definition we are going to use of communication reflects exactly this idea of collaboration. Our other model, circuits, are basically combinations of logical gates, put together with electrical wires carrying binary values. They are ubiquitous in our everyday life; they are how computers compute, how cell phones make calls, yet the most basic questions about them remain widely open: how to build the most efficient circuits computing a given function? How to prove that some function does not have a circuit of a given size? For all but the most basic computations, the question of whether they can be computed by a very small circuit is still open.

On the other hand, our main object of study, cellular automata, is a prototype of our second definition of complexity. What "does" a cellular automaton is *exactly* this definition: making simple agents evolve with interaction with a small neighborhood. The theory of cellular automata is related to other fields of mathematics, such as dynamical systems, symbolic dynamics, and topology. Several uses of cellular automata have been suggested, ranging from the simple application of them as a *model* of other biological or physical phenomena, to the more general study in the theory of computation.

By their intrinsically parallel nature, cellular automata are a good candidate of such complex functions. Since a cellular automaton performs the same operations, at the same time, on each unit of its memory, it seems difficult to imagine that these operations could be further parallelized. A key argument to this approach is the existence of the notion of simulation between cellular automata. In the more general theory of computing, a common way of measuring the complexity of an algorithm is by *reducing* it to as many problems as possible. To reduce, here, means to use it to solve another problem, by transforming the input for the other problem, to an input our algorithm understands. For instance, the problem of finding the largest group of persons in a society, such that each person of the group knows the entire group, can be *reduced* to the problem of deciding which objects among a given collection, each with a certain volume and value, must be carried in a knapsack, to maximize the total value of the knapsack. It is not really easy to see how the reduction works, and, unfortunately, experience on the reductions we know shows that this is most often the case. The notion of *simulation* in cellular automata, however, provides a geometric means for embedding the dynamics of a cellular automaton into one another. Moreover, this transformation can even be *local*, in the sense that each small part of the input is transformed in the same way. In this work, we define a hierarchy of complexity, using only simple reductions of this kind.

In this work, we investigate both the possibility of using cellular automata as a prototype of such complex functions, and the use of provably complex functions to lower bound the complexity of arbitrary cellular automata.

Motivations and origins of this work

The model of cellular automata was first devised by John von Neumann [1], in the 1950s. The field of discrete dynamical systems, and of symbolic dynamics with it, really emerged with the works of Curtis, Lyndon, Hedlund and others in the 1960s (see for instance [2]). At the same time, the idea of algorithmic complexity was given more and more attention. It is difficult to trace the idea back exactly, but the idea of reasonable computation time is generally attributed to Cobham [3]. The Cook-Levin theorem ([4] and [5]) emphasized the importance of this study.

At the time of writing this thesis, about forty years later, the latest advances in the theory show that exponential-time complete functions are not computable by constant-depth circuits with logical and "sum modulo 6" gates, of unbounded arity (this is a result of 2011 by Williams [6]). This illustrates our relatively poor understanding of algorithmic complexity. However, this situation has motivated great research; indeed, this theory is the first one, of all mathematical theories, to have hardness results about proofs in the theory. More specifically, Razborov and Rudich defined *proofs* of complexity in [7], as programs deciding whether a function given as input has high algorithmic complexity. They showed that such programs cannot be fast and give the correct answer for a non-negligible number of functions. That is, if we find a proof that some function is complex, the argument is likely to be neither simple, nor easily generalizable.

On the side of cellular automata, important progresses have been made. Fundamental results in symbolic dynamics, such as the Moore-Myhill theorem [8] have been proven. The works of Kari, building upon the theory of tilings, especially on the aperiodic tilesets of Berger or Robinson, proved the undecidability of many properties of cellular automata. These results contributed to develop the computational side of cellular automata. Finally, an elegant notion of computational equivalence, called *bulking*, developed by several authors, is now well understood, and has been the source of a rich theory. We can cite the reference papers of Delorme, Mazoyer, Ollinger and Theyssier, [9] and [10], but these notions date back to the works of Banks [11]. We detail this theory in chapter 1.

As an attempt to lower bound the complexity of computations, and especially of parallel ones, Yao introduced in 1979 the idea of *communication complexity* [12]. Communication complexity extends and generalizes the wider area of *information theory*, started by the works of

Shannon (for instance in [13]). The idea is to view any computation as a series of communication between the parts of the computing system. In this model, two players need to collaborate to compute a given function. Each of them has an arbitrary computational power, but only knows a half of the input. We review in chapter 3 the fundamental results of this theory, as well as the connection between this model and cellular automata, first found by Dürr, Rapaport and Theyssier in [14]. One of the interesting aspects of communication complexity, that we used in a part of this manuscript, is the possibility to compute the actual communication complexity of a function by an actual algorithm (and thus computing experimental values).

Cellular automata and parallel complexity

Razborov and Rudich's result about natural proofs gave us one of the starting questions of this work: since cellular automata are very easy to simulate on a computer, even for quite large input sizes, would it be possible to get an experimental lower bound on the circuit complexity of predicting their evolution after a few steps? That is, would it be possible to compute an actual lower bound of the depth of circuits necessary needed to predict the first fifteen or twenty steps of the evolution?

This question remains open, but our chapter 2 shows links between cellular automata and circuits. The main question of that chapter is to infer dynamics of a cellular automaton based on hypotheses on its computational complexity. After showing how to extend the notion of simulation to circuits, we examine in detail the most basic class of circuits that may be used to predict the evolution of cellular automata: circuits of constant depth. Also, a major circuit-related question is *uniformity*. Having a family of circuits computing some functions for different input sizes does not necessarily mean that this function is computable. Indeed, an unbounded number of uncomputable "hints" may be hidden in circuit families. We show, for the small complexity classes that we studied, how the classical patch to this problem (that is, generating the families by an algorithm on a Turing machine) becomes unnecessary for cellular automata.

Finally, we present a generalization of a classical result about planar circuits with monotone gates to multi-valued logic, and show how cellular automata with the same hypotheses can be studied. For general monotonic cellular automata, we show how to construct an intrinsically universal cellular automaton, with a similar hypothesis of monotonicity. Then, for a restriction of the neighborhood of these automata, similar to the hypothesis of planarity for circuits, we show several reasons to believe that the complexity of these objects is quite small. Our study also opens the way to a generalization of these results, the proof of which is actually quite simple in the boolean case, to multi-valued logic.

This line of research is the complexity side of a more algebraic theory called *clone theory*, originated by Post, and more particularly his classification of boolean functions in [15].

Lower-bounding the communications in cellular automata

A natural question of complex systems, is related to the "flow" of information between their parts. The study of this question of information in cellular automata is fascinating, and a large part of this manuscript is devoted to it. Our main results show that, for a single cellular automaton to reproduce a large diversity of possible behaviors, it *must* have a high communication complexity; in other words, its dynamics must be a complicated combination of *each* element of its initial configuration.

In the general theory of computing, reductions are usually defined to prove *completeness* of a problem. If C is a complexity class, a problem is said to be C-complete for reduction R, if it can be used to solve any problem of C, modulo a transformation of the input by an algorithm of R. In cellular automata, the same notion exists for simulations. In this work, we will mainly focus on two simulations: the *sub-automaton* relation, written $\sqsubseteq$, and the *factor* relation, written $\trianglelefteq$. Although requiring a *simulation* between two problems instead of a *reduction* seems a stronger condition, a fascinating fact is that there are complete cellular automata, at least for reduction $\sqsubseteq$, even for the class of all cellular automata. These last automata are called *intrinsically universal* cellular automata.

We show in chapter 3, a general method, using communication complexity, to prove (for a number of classes of cellular automata) that a given cellular automaton cannot be intrinsically universal. Using the possibilities of simulations and experiments offered by the framework of communication complexity, and by cellular automata, we then apply this method to a set of cellular automata in which the existence of a universal cellular automaton is still an open problem: the *elementary* cellular automata. Our main results in this chapter are compatibility results: the general approach is to ask a question (we also say "problem") about the dynamics of cellular automata, and try to solve it using a communication protocol. For some cellular automata, the question will be easy to answer; more precisely, it will not require the precise knowledge of all the parts of the configuration to be answered, and a simple protocol will work. On others, each precise detail will be needed, and answering the question will need many communications. We show that there is a number of problems for which the existence of a simple communication protocol for some automaton $\mathcal{A}$ implies that $\mathcal{A}$ cannot be intrinsically universal.

Moreover, if the problems we study also have a computational interest, that is, if there is something to be said about their algorithmic complexity, then these results, along with the existence of an intrinsically universal cellular automaton, provides an example object for which many questions we can ask about its dynamics have high algorithmic complexity.

Finally, in chapter 4, we present a generalization of this approach to the objects and questions of symbolic dynamics. We first show how to use two other definitions of communication complexity, using non-determinism and randomization, to solve the new problems. The general question of this chapter is related to the *limit set* of a cellular automaton, that is, the set of patterns a cellular automaton can generate after an arbitrary time. Our main result shows that, although the communication complexity of the limit set increases with simulation $\trianglelefteq$ (that is, if F simulates G with relation $\trianglelefteq$, then F's limit set is more complex that G's), it is not the case for simulation $\sqsubseteq$.

The proof of compatibility for simulation $\trianglelefteq$ requires a generalization of communication complexity: we need to introduce either randomization, or non-determinism, and the theorem even seems to be false for deterministic communication complexity. This fact gives a new and original view on the difference between the two simulations, while the main open problem of this field remains: is there an intrinsically universal cellular automaton for $\trianglelefteq$?

1 Cellular automata: definitions

In this chapter, we introduce formally our main objects of study: cellular automata, and give a few insights on the current state of knowledge of this field. Our point of view will be divided between the discrete nature of these objects, with definitions and questions seeing cellular automata as *computing machines*, and the more classical point of view of dynamical systems and symbolic dynamics. This duality is intentional; it is one of the main justifications of our work, and many of our results relate dynamic properties with computational ones.

1.1 First definitions

A dynamical system is a couple (X,F) where X is a compact space and F a continuous function of $X \to X$. The main questions of dynamical systems are related to the iterations F^t of the map, hence the name “dynamical”. Cellular automata are a special kind of dynamical systems, initially introduced by John Von Neumann in a book [1] as a model of “self-reproducing” machines. This first definition looked more like the definition of a machine:

Definition 1.1.1 A cellular automaton is a 4-uple $\mathcal{A} = (\mathfrak{C}_\mathcal{A}, N_\mathcal{A}, Q_\mathcal{A}, \delta_\mathcal{A})$, where :

- $\mathfrak{C}_\mathcal{A} = \mathbb{Z}^d$ is the set of cells of cellular automaton, and d is called its *dimension*.
- $N_\mathcal{A}$ is a tuple of vectors, defining the *neighborhood* of $\mathcal{A}$.
- $Q_\mathcal{A}$ is a finite set called the *alphabet*, whose elements are called the *states*.
- $f_\mathcal{A} : Q^{N_\mathcal{A}} \to Q$ is the *local transition function*.

A few years after this definition, the link with dynamical systems was found by Gustav Arnold Hedlund in [2]. This is one of the roots of the ideas of this manuscript: if a complex object can be given an equivalent definition in terms of simple mathematical objects, it allows to reason on abstract objects without too much consideration for the problems raised by its discrete nature.

For a given cellular automaton $\mathcal{A}$, a *configuration* for $\mathcal{A}$ is any element of $Q^{\mathfrak{C}_\mathcal{A}}$, that is, an assignment of a state of Q to all cells of $\mathfrak{C}$. Given a configuration c and an element i of $\mathfrak{C}_\mathcal{A}$, we denote by c_i the state assigned to cell i of c. Now, the *global function* of $\mathcal{A}$ is *the* function $F_\mathcal{A}$ from $\mathfrak{C}_\mathcal{A}^Q \to \mathfrak{C}\mathcal{A}^Q$ such that $F(c)[i] = f\big(c[i+n_1], \dots, c\big[i+n_{|N_\mathcal{A}|}\big]\big)$, where the n_i are the elements of $N_\mathcal{A}$.

However, with this formalism, several local rules can represent the same global function. Indeed, let f be a cellular automaton with neighborhood $\{1, \dots, n\}$, and let f' be a new local function defined by $f'(c[1], \dots, c[n+m]) = f(c[1], \dots, c[n])$. It defines the same global function as f, only with a larger neighborhood. To avoid this problem, and useless formalism, we will only consider in this work the *canonical* neighborhood of cellular automata, i.e. the representation with the smallest neighborhood defining the same global function.

When working with cellular automata, we rarely see actual configurations. Instead, we work with *finite* parts of them, that we call hereafter *finite configurations*. Since we do not have all the information required to compute the whole evolution, the rule can only be applied a finite number of times. The application of a rule *as long as possible* means that we apply the global function to a configuration until its size has reached a size smaller than the size of the neighborhood. We denote this by f^*:

$$f^*(x) = f^{\left\lfloor \frac{|x|-1}{2r} \right\rfloor}(x)$$

When working with cellular automata, we will often use the graphical representation of figure 1.1, called a *space-time diagram*: we simply draw each of the configurations $c, F(c)$, $F^2(c), \ldots, F^n(c)$, as a horizontal series of squares representing the cells (the time goes from the bottom up).

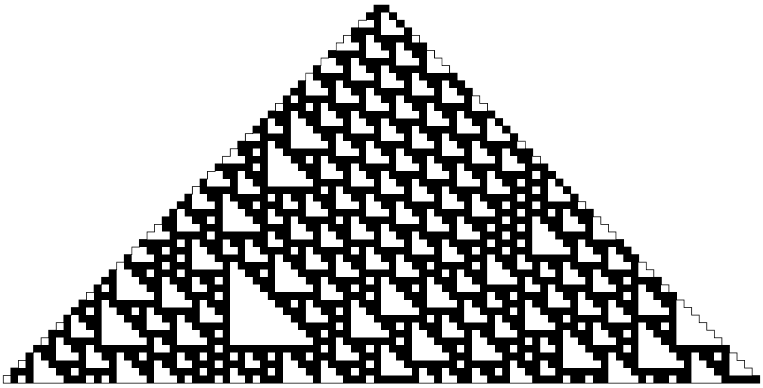


Figure 1.1 - Space-time diagram of elementary rule 110

1.1.1 Information locality and uniformity

We can already remark, from the definition of cellular automata that we gave in definition , two important properties of cellular automata, that distinguish them from other well-known models of computation such as Turing machines or boolean circuits. These properties are their *uniformity*, meaning that the definition of the computation does not depend on the position in the configurations, nor on the size of the input. And the information locality, meaning that information is subject to constraints on its travel speed.

The precise statement of what this means is the object of the Curtis-Hedlund-Lyndon theorem, which proves an equivalence between definition and a definition of cellular automata as a topological object. Our first definitions concern the configuration spaces:

Definition 1.1.2 The *Cantor topology* on space $X = Q^{\mathfrak{C}}$ is the product topology over $\mathfrak{C}$ of the discrete topology over Q. A natural basis of open sets for this topology are *cylinder sets*, that is, for each $i \in \mathfrak{C}$ and each word $a \in Q^*$, the following sets, that we write $[a]_i$:

$$[a]_i = \left\{x \in Q^{\mathfrak{C}} \mid x_i = a[0], \dots, x[i+|a|-1] = a[|a|-1]\right\}$$

The same topology can be defined by the following distance, called the *Cantor distance*:

$$d_{\text{Cantor}}(x,y) = \sum_{i \in \mathfrak{C}} \frac{\delta(x[i],y[i])}{2^{d(0,i)}}$$

According to definition 1.1.1, since $\mathfrak{C} = \mathbb{Z}^d$ for some integer d, it is always a countable metric space, so this definition always makes sense.

Also, when $\mathfrak{C} = \mathbb{Z}$, we call the *shift* transformation, and we write σ, the function of $Q^{\mathfrak{C}} \to Q^{\mathfrak{C}}$ defined by $\sigma(c)[i] = c[i+1]$. Adopting the vocabulary of symbolic dynamics, we can now restate our first definition of cellular automata, or at least of the space of the configurations:

Definition 1.1.3 Let Q be a finite set. A *subshift*, or *shift space*, is a closed subset of $Q^{\mathbb{Z}}$ (for the Cantor topology), stable under σ. A special case of subshift is the *full shift*, that is, the whole space $Q^{\mathbb{Z}}$.

An equivalent definition of subshifts can be given as sets of the configurations that avoid a given collection of finite patterns $\mathcal{F} \subseteq Q^*$, that is, sets of the form $\left\{x \in Q^{\mathbb{Z}} \mid x[i], x[i+1], \dots, x[j] \notin \mathcal{F}\right\}$. In this definition, the complementary of $\mathcal{F}$ in Q^* is called the *language* of subshift S, and is written $\mathcal{L}(S)$.

From our definition of the global function of a cellular automaton, we can already state the following easy properties:

- If $\mathcal{A}$ is a cellular automaton, then for any $i \in \mathbb{Z}$:

  $$F_{\mathcal{A}}(\sigma(c))[i] = f_{\mathcal{A}}(i+N_1+1, \dots, i+N_n+1) = F_{\mathcal{A}}(c)[i+1]$$

  Thus, $F_{\mathcal{A}} \circ \sigma = \sigma \circ F_{\mathcal{A}}$.
- If $\mathcal{A}$ is a cellular automaton, then for any $c \in Q^{\mathfrak{C}}$, and any neighborhood (in the topological sense) $V(F(c))$ of $F(c)$, there is a neighborhood $V(c)$ of c such that $F(V(c)) \subseteq V(F(c))$: for instance if $V(c) = \{x \in V(F(c)) \mid x[i+N_j] = c[i+N_j], i \in pos(C), N_j \in N_{\mathcal{A}}\}$, where C is the smallest cylinder in which $V(F(c))$ is included, and $pos(C) \subseteq \mathfrak{C}_{\mathcal{A}}$ is the set of those positions for which all the points of C have the same value. This means that $F_{\mathcal{A}}$ is *continuous* for the Cantor topology.

Now, the more surprising result of [2] is that the converse also holds:

Theorem 1.1.1 *(theorem 3.1 of [2])*A function from a shift space to itself is the global function of a cellular automaton if and only if it is continuous for the Cantor topology and commutes with σ.

We will need also to distinguish two particular types of functions of this kind:

Definition 1.1.4 A *factor map* φ between two dynamical systems (X,F) and (Y,G) is a continuous function of $X \to Y$ such that $\varphi \circ F = G \circ \varphi$.

By Theorem 1.1.1, if φ is a factor map, then it can be defined as the global function of a cellular automaton. In the case where this cellular automaton has radius 0, we say that φ is a *coloring*.

Subshifts are sets of infinite elements of $Q^{\mathbb{Z}}$, for some finite set Q, what we have called *configurations* in the beginning of this chapter. To measure their complexity, we need a notion of language, that is, of sets of finite words, associated to a subshift:

Definition 1.1.5 The language of a subshift S is the set of all finite patterns appearing in the elements of S:

$$\mathcal{L}(S) = \{s[i], \ldots, s[i+n] \mid s \in S, i \in \mathbb{Z}, n \in \mathbb{N}\}$$

Moreover, in this work, we will mainly consider a few special cases of subshifts:

Definition 1.1.6 A *subshift of finite type*, also called an SFT in the sequel, is a subshift whose language is finite.

Definition 1.1.7 A *sofic subshift* is a subshift whose language is rational.

1.2 Simulations

One of the most elegant ideas of computability theory is the concept of acceptable programming systems, and universal machines. This was the core idea behind the Von Neumann architecture. However, the special properties of cellular automata, most remarkably their spatial and temporal uniformity, allow for a stronger type of simulation, a *syntactic* one, preserving not only the computational power, as with Turing machines, but also a large number of dynamical properties. This idea was already present, although implicitly, in the notion of "self-reproducing machines" of Von Neumann's book. The precise formalization that we are going to define now was first done in the restricted context of two-dimensional cellular automata by Banks in [11], before being generalized by Mazoyer and Rapaport in [16], and then further extended by Ollinger in [17], then by Theyssier in [18]

This new definition of universality is based on the idea of simulation by *bulking*, a local transformation of the rule of a cellular automata. Its key ideas are a notion of *rescaling* and different relations of *embedding* of cellular automata into one another.

1.2.1 Rescaling

The ingredients of the rescaling are the following: temporal rescaling, that is, skipping a constant number of time steps at each step; spatial "packing", that is, taking blocks of a constant number of cells as only one cell, and a translation (the shift). Formally, let F be the global function of a one-dimensional cellular automaton on alphabet Q. For any $m \geqslant 1$, we define the following bijective packing map $b_m : Q^{\mathbb{Z}} \to (Q^m)^{\mathbb{Z}}$ by:

$$\forall z \in \mathbb{Z},\ b_m(c)[z] = (c[mz] \ldots c[mz+m-1])$$

We call the *rescaling* of F by parameters m, t, z, and we write $F^{\langle m,t,z \rangle}$, the following function:

$$F^{\langle m,t,z \rangle} = b_m \circ \sigma^z \circ F^t \circ b_m^{-1}$$

By theorem 1.1.1, this is still the global function of a cellular automaton, since all of these four functions are continuous, and their composition clearly commutes with σ.
Now, how can we say that a cellular automaton simulates another one ? Obviously, this rescaling operation does not change fundamentally the dynamics of cellular automata; the only real distortion is the skipping of a fixed number of steps, different in the simulator and the simulated automaton. Now, we need a notion to say that a cellular automaton "embeds" another one. In the sections 1.2.2 and 1.2.3, we show two different ways of defining such a relation; the first one, called the *sub-automaton* relation, injects an automaton into a part of the simulator automaton. The other one, called the *factor* relation, projects the states of the simulator onto states of the simulated automaton.

Definition 1.2.1 Let F and G be two cellular automata, and $\mathcal{R}$ a preorder relation. We say that F simulates G by $\mathcal{R}$, and we write $G \preccurlyeq_{\mathcal{R}} F$, if:

$$\exists m_F,\ t_F,\ z_F,\ m_G,\ t_G,\ z_G,\ G^{\langle m_G,t_G,z_G \rangle}\ \mathcal{R}\ F^{\langle m_F,t_F,z_F \rangle}$$

For both relations, we consider the question of the existence of *intrinsically universal* cellular automata:

Definition 1.2.2 A cellular automaton F is *intrinsically universal,* or simply *universal,* for some set S of cellular automata, and relation $\mathcal{R}$, if:

$$\forall G \in S,\ G \preccurlyeq_{\mathcal{R}} F$$

1.2.2 The sub-automaton relation

Our first simulation relation "embeds" a cellular automaton into another one, in the sense that only a part of the configurations of the simulator are used by the simulation.

Definition 1.2.3 Let F and G two one-dimensional cellular automata, we say that F is a *sub-automaton* of G, and we write $F \sqsubseteq G$, if and only if:

$$\exists\ \varphi : Q_F \to Q_G,\ \varphi \text{ is one-to-one and } G \circ \overline{\varphi} = \overline{\varphi} \circ F$$

In this case, we say that φ is the map *induced* by $\sqsubseteq$.

There are several constructions of universal cellular automata using this relation, see [10] for a complete account, or [17] and [19] for complete constructions of universal cellular automaton for this relation.

1.2.3 The coloring relation

The other simulation, that we study in this work is called the factor relation, and comes from symbolic dynamics. The idea behind sub-automata was to "hide" parts of the configuration space, by simulating simpler automata only on a subset of the configurations. In the factor relation, this is forbidden. More precisely, any configuration of the simulator should simulate some configuration of the factor automaton:

Definition 1.2.4 Let F and G two one-dimensional cellular automata, we say that F is a *factor* of G, and we write $F \trianglelefteq G$, if and only if:

$$\exists\ \psi : Q_G \to Q_F,\ \psi \text{ is onto and } F \circ \overline{\psi} = \overline{\psi} \circ G$$

In this case, we say that ψ is the map *induced* by $\trianglelefteq$.

However natural this definition may seem, the existence of a $\trianglelefteq$-universal automaton is still unknown:

Open problem 1.2.1 *(from [18])* Is there a cellular automaton $\trianglelefteq$-universal ?

1.3 Particular classes of cellular automata

We consider here several restrictions, or "classes", of the general framework of cellular automata, with specific properties. A first restriction that we may consider is the class of *elementary* cellular automata, of radius 1 and alphabet $\{0,1\}$, often referred to by their numbering:

Definition 1.3.1 The local rule called *elementary rule* n is the rule defined on alphabet $\{0,1\}$ by:

$$f(c[0], c[1], c[2]) = \frac{n}{2^{4c[0]+2c[1]+c[2]}} \bmod 2$$

Another class of useful cellular automata is those cellular automata with a *spreading state*, that is, a state that spreads everywhere if it appears once in the configuration:

Definition 1.3.2 Let F be a cellular automaton with alphabet Q, radius r and local rule f. A state $q \in Q$ is called a *spreading state* for F if for all $x \in Q^{2r+1}$, $f(x) = q$ whenever q appears in x.

Remark that there can be only one spreading state in a given cellular automaton, and that this notion extends to any dimension d by considering a neighborhood of dimension d instead of Q^{2r+1}.

Understanding the link between the local definition, and the global properties of cellular automata is one of the key problems, and their properties as functions, and as dynamical systems, give a convenient framework to start this study. In particular, several classes have been investigated:

Definition 1.3.3 A *surjective* cellular automaton is a cellular automaton whose global function $F : Q^{\mathbb{Z}} \to Q^{\mathbb{Z}}$ is surjective, that is,

$$\forall\, y \in Q^{\mathbb{Z}}, \exists\, x \in Q^{\mathbb{Z}}, F(x) = y$$

From now on, we will denote the set of surjective cellular automata by Surj.

This class, containing at the same time simple automata such as the shift, and very complex ones such as elementary rule 30, has been extensively studied. One of the most notable results about it, the *Moore-Myhill theorem*, gives an alternative characterization of class Surj:

Theorem 1.3.1 *(from [8] and [20])* A cellular automaton F, of any dimension, is surjective if and only if it is locally injective, that is,

$$\forall x, y \in Q_F^{\mathfrak{C}_F}, \left|\{x[i] \neq y[i] \mid i \in \mathfrak{C}_F\}\right| \text{ is finite} \Rightarrow F(x) \neq F(y)$$

Many results about this class have been proven by Hedlund [2], among which the following results about the cardinality of the antecedents of a configuration, by a surjective rule:

Theorem 1.3.2 Let F be a surjective cellular automaton of radius r and alphabet Q. Then for any configuration $x \in Q^*$:

$$\left|F^{-1}(x)\right| = |Q|^r$$

Moreover, in one dimension, another surprising result is that surjectivity is decidable. This result, due to Amoroso and Patt, first appeared in [21]:

Theorem 1.3.3 *(from [21])* The set of onto one-dimensional cellular automata is recursive, as well as the set of one-to-one cellular automata.

The disparity of this class has raised intriguing open problems. We present some of them here:

Open problem 1.3.1 *(from [10])* Is there a universal cellular automaton for class Surj ?

Another interesting open problem about cellular automata is known as the *dense periodic orbits conjecture*:

Open problem 1.3.2 *(from [22])* Let F be a surjective cellular automaton. Is the set of points x such that the sequence $(F^t(x))_t$ is periodic dense in $Q^{\mathbb{Z}}$?

An interesting subset of Surj is the class of *reversible* cellular automata, that is, those automata whose global function is onto *and* one-to-one, like the shift, for instance. This class is better understood, and a universal cellular automaton is even known, as proved in [23]. See also [24] for recent results.

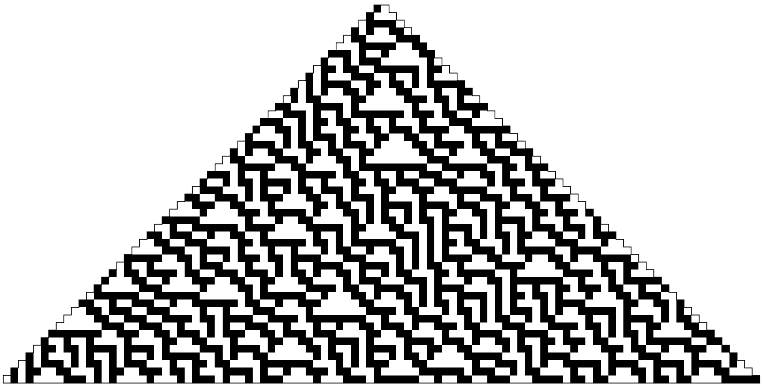


Figure 1.2 - Space-time diagram of elementary rule 30

Several properties that have been studied come from the theory of dynamical systems. A detailed account of this theory, applied to cellular automata, can be found in [25]. We restate some of these definitions here:

Definition 1.3.4 Let X be a metric space, and F a function of $X \to X$. F is said to be *equicontinuous* at point $x_0 \in X$ if:

$$\forall \epsilon > 0,\ \exists \delta > 0, \forall x \in X, \forall t > 0,\ d(x, x_0) \leqslant \delta \Rightarrow d\Big(F^t(x), F^t(x_0)\Big) \leqslant \epsilon$$

Another class that we will need in this manuscript is the class of linear cellular automata. A complete discussion of what this means, with a variety of applications and examples, can be found in [26]. In the present work, we define it as follows:

Definition 1.3.5 A *semigroup* $(S, \oplus)$ is an algebraic structure with a set S an an associative law $\oplus$.

Definition 1.3.6 A *linear* cellular automaton is a cellular automaton whose alphabet is a semigroup $(Q, \oplus)$, and whose global functions verifies:

$$F(x \overline{\oplus} y) = F(x) \overline{\oplus} F(y)$$

Where $x \overline{\oplus} y$ denotes the uniform extension of $\oplus$ to words or configurations:

$$(x \overline{\oplus} y)[i] = (x[i] \oplus y[i])$$

Interesting properties of these cellular automata include the existence of fractal structures, as studied in [27], or [28].

1.3.1 Decidability

The decidability of the properties of a system is an important question in the study of its computational power. One of the first examples of undecidable behaviors of a machine was the halting problem, shown undecidable by Turing in [29], by one of the first applications of Cantor's diagonal argument to computing. In cellular automata, a number of properties have been shown undecidable.

The first property that we will present here is the periodicity of cellular automata. The following result, due to Culik, Pachl and Yu in [30] in dimension at least two, and to Kari [31] for dimension one, shows that there is no general way to detect whether a cellular automaton given by its local rule is nilpotent:

Definition 1.3.7 A cellular automaton F is *nilpotent* if there is a time step t and a state q_0 of the alphabet such that $\forall\, x \in Q^{\mathfrak{C}_F}, \forall\, i \in \mathfrak{C}_F$, $F^t(x)[i] = q_0$.

Theorem 1.3.4 The set of local rules of cellular automata representing a *nilpotent* cellular automaton is recursively enumerable but not recursive.

Another variant of this kind of simple behavior after a few steps is the *periodicity* of a cellular automaton. A periodic cellular automaton, of period t, will always return to its initial configuration after t steps, independently of the what the initial configuration was:

Theorem 1.3.5 The set of local rules of cellular automata representing a *periodic* cellular automaton, that is, a cellular automaton F such that $F^t(x) = x$, is recursively enumerable but not recursive.

2 Circuits and cellular automata

As soon as computer scientists began to understand the limitations of sequential computation that the Turing model provided, with the Cook-Levin theorem ([4] and [5]), the idea of parallelism began to raise interest in the community. At this time, probably building upon the experience gained in making computing hardware, the model of boolean circuits emerged as a new paradigm to solve the hard questions of the recent algorithmic science. Unfortunately, it was understood soon thereafter what new problems this model would cause. Indeed, if the P $=$ NP problem was already identified as the limit of our understanding of algorithms, as soon as NC, the equivalent for circuits of P in sequential computation as the "reasonable class of computation", had been formulated, a new question was asked: is P $=$ NC? In other words, would it be possible to further accelerate exponentially all the computations we know to be reasonable?

This chapter offers an overview of this field, and shows how our model of parallel computation, cellular automata, compares to this classical one. To our knowledge, the main two approaches that have been taken to study this question are the one of Cook in [32] and Neary and Woods in [33], where the authors use the dynamics of cellular automata to lower-bound their complexity, and the approach of e.g. Moore in [26], where the author upper-bounds the complexity of cellular automata, by providing circuits to predict them. These approaches both study the implications of hypotheses on the dynamics of cellular automata on their algorithmic complexity. The first one to find completeness results, the other one to find upper bounds on their complexity.

Our approach is complementary to this one, and our main question is: what do hypotheses on the algorithmic complexity of cellular automata mean to their dynamics? After reviewing the definitions and classical results of this field, we will mainly focus on two particular classes, the *monotonic planar* cellular automata, and the automata with a number of dependencies bounded by a constant.

2.1 Circuits as graphs of algorithms

2.1.1 Definitions and examples

Definition 2.1.1 A circuit is a 4-uple $C = (Q, \mathcal{G}, V, E)$, with Q a finite set (the *alphabet*), (V, E) a directed acyclic graph, the *underlying graph*, $\mathcal{G}$ a finite set of functions of $Q^* \to Q$, called the *gates*. The vertices of V are labeled as follows:

- Each vertex of ingoing arity (or *fan-in*) $d \geqslant 0$ is labeled by a function of $Q^{Q^d} \cap \mathcal{G}$, that is, by some function of $\mathcal{G}$ with d inputs.
- The vertices with ingoing arity $d = 0$ are called the *inputs* of the circuit, and are labeled by some $i \in \mathbb{N}$.

Some of the vertices of outgoing arity (or *fan-out*) 0 are called the *output vertices*. There must be at least one output vertex, and, if there are N of them, they are numbered 1 through N.

Finally, the maximal length of a path of graph (V,E) is called the *depth* of C, and is written $d(C)$.

We now need to define what we mean by "evaluating a circuit" or that a circuit "computes a function". The evaluation procedure of a circuit is as follows:

Algorithm 2.1.1 The process of computing a function with a circuit is as follows: on input $x = x_1, \dots, x_n$, we compute a *valuation* of the circuit, that is, we construct a function $\mathrm{Val} : E \rightarrow Q$, such that:

- For each input vertex v, labeled by i, we set $\mathrm{Val}(v) = x_i$.
- Each other vertex v, with label g, such that Val has been constructed for all its predecessor vertices (the $v_1,\dots,v_d$ with an edge $(v_i,v) \in E$), receives value $\mathrm{Val}(v) = g(\mathrm{Val}(v_1),\dots,\mathrm{Val}(v_d))$.

If there are N output vertices $v_1,\dots,v_N$, the final value of the circuit is $(\mathrm{Val}(v_1),\dots,\mathrm{Val}(v_N))$. In the sequel, we will often write $C(x)$ to mean "the final value of the circuit after its evaluation on input x".

The problem of computing $C(x)$ is called the *circuit value problem of* C *on input* x.

Sometimes, depending on the circuit, the function Val can be computed faster than the canonical procedure described by Algorithm 2.1.1. For instance, when the fan-out of all the vertices of a circuit C is at most one (and hence the underlying graph of C is a forest), then the two children of each node can be evaluated in parallel.

In the sequel, unless explicitly mentioned, we will only consider the case of finite Qs. See [34] for an introduction to the existing generalizations to larger value sets. The first type of circuits that will interest us are the *boolean circuits*, where $Q = \{0,1\}$. Our first result shows what kind of gates we need:

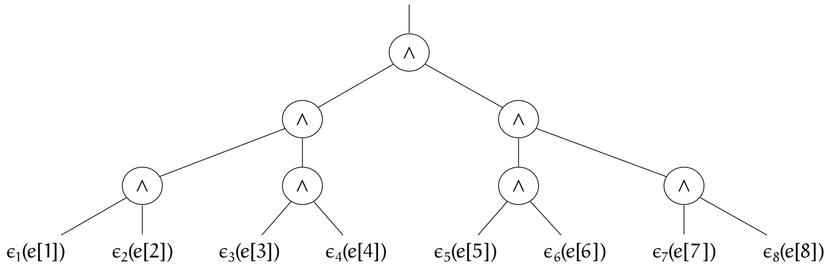


Figure 2.1 - The circuit of a conjunction

Proposition 2.1.1 The gate set $G = \{\wedge,\vee,\neg\}$ is *complete* for functions of $\{0,1\}^n \rightarrow \{0,1\}$, that is, any function of $\{0,1\}^n \rightarrow \{0,1\}$ is exprimable by a circuit with gates from G, of depth $n + \log n + 2$.

Proof. We can clearly represent the 0 constant function by a formula of depth 1. Else, for any $e \in \{0,1\}^n$ such that $f(e) = 1$, we build the expression $\epsilon_1(e[1]) \wedge \ldots \wedge \epsilon_n(e[n])$, where $\epsilon_i(x) = x$ if $e[i] = 1$, and $\neg\ x$ else. This conjunction can be written as a circuit of depth $\log n + 2$ (see figure 2.1), because the conjunction requires a depth $\log n + 1$, and negating some of the variables depth 1. Now, we can plug all these wires to a tree similar to the one on figure 2.1, but with $\vee$ gates instead of $\wedge$s. Since there are at most 2^n inputs, this requires a tree of depth n. □

Let us remark that in the proof of Proposition 2.1.1, the circuits we built were of depth $n + \log n + 2$, and of size in $O(n2^n)$. And changing the gate set, as long as the new one is also complete, can only change the complexity up to a constant factor. And since any computational problem can be encoded as a boolean function (more details about this can be found in the proof of the Cook-Levin theorem [4]), the main question of the whole theory of algorithms can be therefore reformulated as "under what conditions can we design better circuits?". The answer is "not often", and this result is classically known as the *Shannon effect*:

Theorem 2.1.2 *(from [35])* Most functions of $\{0,1\}^n \rightarrow \{0,1\}$ cannot be computed by circuits of size in $o(2^n)$.

Proof. This can be proved by a simple counting argument: there are 2^{2^n} distinct boolean functions of $\{0,1\}^n \rightarrow \{0,1\}$. How many circuits of size $p(n)$ are there? If we can connect any two gates with zero, one or two connections (since the underlying graph of a circuit is directed), there are $3^{p(n)^2}$ different possible ways of connecting a circuit.

We also need to decide what gates the vertices will be labeled by. Each vertex might be labeled by a gate from $\mathcal{G}$, a variable among $\{x_1, \ldots, x_n\}$, or a constant 0 or 1. Hence, there are $(n + |\mathcal{G}| + 2)^{p(n)}$ possible labelings. In total, there are at most $3^{p(n)^2}(n + 2 + |\mathcal{G}|)^{p(n)}$ circuits of size $p(n)$, which is negligible compared to the number of boolean functions, as soon as $p(n) \in o(2^n)$. □

This theorem allows to define the first notion of algorithmic complexity of this manuscript, namely, *circuit complexity*. Contrarily to Turing machines, one circuit only operates on inputs of a given fixed size. To solve algorithmic problems with circuits, we need to define circuits for all possible input sizes:

Definition 2.1.2 A *circuit family* is a sequence $(C_n)_n$ of circuits. Moreover, if f is a function of $\{0,1\}^* \rightarrow \{0,1\}$, we say that $(C_n)_n$ computes f if $\forall n \in \mathbb{N}$, $x \in \{0,1\}^n$, $C_n(x) = f(x)$. We call function $n \mapsto d(C_n)$ the *circuit complexity* of $(C_n)_n$.

However, this is not a very realistic model of computation. For instance, according to our definitions, the circuit family $(C_n)_n$ defined for all n and $x \in \{0,1\}^n$ by "$C_n(x) = 1 \Leftrightarrow \varphi_n$halts on input n", where φ_n is the nth Turing machine, is a legal circuit family, of depth and size 1, but it computes an uncomputable function. A common restriction imposed on circuit families to avoid this situation is *uniformity*:

Definition 2.1.3 Let $\mathcal{C}$ be a Turing complexity class. A $\mathcal{C}$-*uniform* circuit family is a circuit family such that there is a Turing machine $\varphi \in \mathcal{C}$ writing, when given the unary representation of n as input, an encoding of C_n on its output tape.

Only now can we define a notion of circuit complexity that can be compared to Turing complexity. The surprising fact about Theorem 2.1.2 is that although we know that most functions are "hard", no one has been able to prove yet that a boolean function could not be computed by a uniform family of circuits of, say, polynomial size.

The same problem is also open for parallel time. We do not know a proof that a boolean function cannot be computed by a uniform family of *polylogarithmic* depth. The name "parallel time" is accurate here, since boolean circuits are really a model of parallel computing. This is because if we want evaluate a circuit C, as described by Algorithm 2.1.1, we can do so in "rounds", where in each round we compute $\mathrm{Val}(\nu)$ for all the vertices ν where this is possible, then we will be done evaluating the whole circuit in exactly $d(C)$ rounds. An immediate corollary of the Shannon effect is that most functions also require a linear depth to be computed, because 2^n gates do not fit in much less than $O(n)$ rounds.

Understanding this problem means finding what, in a computation, must be done sequentially. The answer is known as Spira's theorem for formulas, that is, circuits in which the underlying graph is a tree:

Theorem 2.1.3 *(Spira's theorem)* Any boolean formula F is equivalent to a formula F' of depth at most $4 \cdot \log(|F|)$.

Proof. By induction on $|F|$. It is true for $|F| = 1$. For $|F| \geqslant 2$, let G be the minimal subformula of F of size greater than $t/2$. Since $|G| \geqslant 2$, G must have one or two direct subformulas, of size at most $t/2$ by minimality of $|G|$. Hence, by induction hypothesis, G can be written as a formula of depth $1 + 4 \cdot \log(t/2)$.

If $G = F$, the proof is finished. Else, let us write F_0 (respectively F_1) the formula where G has been replaced by 0 (respectively 1) in F. Since F_0 and F_1 are of size at most $t/2$, they are equivalent, by induction hypothesis, to formulas F_0' and F_1', of depth $4 \cdot \log(t/2)$.

Finally, F is clearly equivalent to $(G \wedge F_1') \vee (\neg\, G \wedge F_0')$, which is a formula of depth $3 + 1 + 4 \cdot \log(t/2)$, that is, $4 \cdot \log t$. □

As for the general situation, we do not know. Even the latest progresses, such as William's result [6], do not bring much intuition on the proof techniques needed to solve these problems. Even worse, a paper of 1994, by Razoborov and Rudich [7] introduced the notion of *natural proof*, which are algorithms recognizing functions with no polynomial-size circuit family. Their proof shows that, under commonly believed hypotheses, for such an algorithm to exist, it must either be applicable to a very small number of functions, or its running time must be unpracticable (more precisely, an exponential in the size of the truth table of the input function, which is already of size 2^n for input size n).

Finally, let us state a few more definitions that we will need in our study.

Definition 2.1.4 The complexity class P is the set of all languages over some finite alphabet that are recognized by a Turing machine running in time polynomial in the size of the input.

Definition 2.1.5 The complexity class LOGSPACE is the set of all languages over some finite alphabet that are recognized by a Turing machine running in space logarithmic in the size of the input.

Definition 2.1.6 The complexity class NC is the set of all languages over a finite alphabet Q that are recognizable by a LOGSPACE-uniform family of circuits with one output, of polynomial size, and polylogarithmic depth, that is, $O(\log^k n)$ for some k. "Recognizable" here means that there are a distinguished state $q_\top \in Q$ such that a word is recognized if and only if the circuit outputs $q_\top$.

In this chapter, we try to show why we think that cellular automata, as an inherently parallel computational model, may be an appropriate model for the study of these questions.

2.2 Circuits and cellular automata

The first problem we need to deal with, in this adaptation of cellular automata to boolean circuitry, is the problem of input and output representation. Circuits compute on *bits*, and cellular automata on *states*. This problem is far from being trivial, as we will see.

2.2.1 Of states and bits

However, choosing an alphabet Q different from $\{0,1\}$ for our gates to operate on does not change drastically the complexity as we have defined it in section 2.1.1, as shown by the following propositions:

Lemma 2.2.1 Let C be a circuit over alphabet Q and gates $\mathcal{G}$ with arity at most d. Then, for any complete gate set $\mathcal{G}'$ on an alphabet Q' such that $|Q'| \geqslant |Q|$, there is a circuit C' and a one-to-one application $\varphi : Q \rightarrow Q'$ such that:

- for any $x \in Q^n$, $C' \circ \varphi = \varphi \circ C$.
- there are two positive integer constants α and β, depending only on $|Q|$ and $\mathcal{G}$, such that $d(C') \leqslant \beta d(C)$ and $|C'| \leqslant \alpha |C|$.

Proof. φ can be any one-to-one application of $Q \rightarrow Q'$. We simply replace any gate in C by a small circuit with gates of $\mathcal{G}'$, which is always possible since, for any function $g \in \mathcal{G}$, the function defined on $\varphi(Q)^d$ by $g'(x_1, \ldots, x_d) = \varphi^{-1}(g(\varphi(x_1), \ldots, \varphi(x_d)))$ is clearly a function of $(Q')^d \rightarrow Q'$, and thus there is a circuit of depth $O(d \log |Q'|)$ and size $O\left(2^{d \log |Q'|}\right)$ computing it.

Then, we can show by an easy induction that at each step the evaluation process described by algorithm 2.1.1, each wire carries value $\varphi(q)$, if the corresponding wire in C carried q. □

Conversely, we can do almost the same construction, with a somewhat more complicated encoding, to solve the case where the alphabet of the new circuit is smaller:

Lemma 2.2.2 Let C be a circuit over alphabet Q and gates $\mathcal{G}$, with N output gates $C_1, \ldots, C_N$. For any $q \in Q$, we write $\langle q \rangle$ an encoding of q by words of $\{0,1\}^{\lceil \log_2 |Q| \rceil}$. Then we can design a circuit C', on alphabet $\{0,1\}$, with gates in $\{\wedge, \vee, \neg\}$, and $N' = \lceil \log_2 |Q| \rceil \cdot N$ output gates $(C'_i)_i$, such that:

- for any $x \in Q^n$, and any $1 \leqslant i \leqslant N$, $\langle C_i(x) \rangle = C_{q \cdot i}'(\langle x \rangle), \ldots, C_{q \cdot i + q - 1}'(\langle x \rangle)$, with $q = \lceil \log_2 |Q| \rceil$.
- there are two positive integer constants α and β, depending only on $|Q|$ and $\mathcal{G}$, such that for all $i \leqslant N$, $d(C_i) \leqslant \beta d(C)$ and $|C_i| \leqslant \alpha |C|$.

Proof. Since $\{\wedge, \vee, \neg\}$ is complete for $\{0,1\}$, as proved in proposition 2.1.1, each gate of $\mathcal{G}$ of arity d can be replaced by $\lceil \log |Q| \rceil$ circuits computing functions of $\{0,1\}^{d \lceil \log |Q| \rceil} \rightarrow \{0,1\}$, which we can choose of size $\alpha = O\left(2^{d \log |Q|}\right)$ and depth $\beta = O(d \log |Q|)$, again by Proposition 2.1.1.

A simple induction on the steps of Algorithm 2.1.1 shows that this new circuit computes the same function as C. □

Proposition 2.2.3 Let C be a circuit over alphabet Q and gate set $\mathcal{G}$. For any other alphabet Q' with at least two states, any gate set $\mathcal{G}'$ complete for Q', there is an integer $c \leqslant 1 + \log_2 |Q'|$, a one-to-one encoding function $E : Q \rightarrow Q'^c$ and a circuit C', such that:

- for any output vertex C_i of C, $\langle C_i(x) \rangle = C_{m \cdot i}'(\langle x \rangle), \ldots, C_{m \cdot i + m - 1}'(\langle x \rangle)$, with $m = \lceil \log_2 |Q'| \rceil$.
- there are two constants α and β, depending only on $|Q|$ and $|Q'|$, such that $|C'| \leqslant \alpha |C|$ and $d(C') \leqslant \beta d(C)$.

Proof. If $|Q'| \geqslant |Q|$, then it is Lemma 2.2.1, with $c = 1$. Else, we can transform C into a boolean circuit with gates $\{\wedge, \vee, \neg\}$, by Lemma 2.2.2, and then to a circuit over Q' by Lemma 2.2.1 again. □

By the way, this proposition shows how a gate set can be complete for an alphabet: indeed, in this proof, we have used the completeness of $\mathcal{G}'$ essentially for computing boolean circuits and functions of $Q^{\log_2 |Q|} \rightarrow Q$. Since Proposition 2.1.1 shows that only three different gates are necessary to compute any boolean circuit, and that there are only a finite number of functions of $Q^{\log_2|Q|} \to Q$, this proves the following corollary:

Corollary 2.2.4 For each finite alphabet Q, there is a gate set of finite size complete for Q.

2.2.2 Computational problems about cellular automata

Defining a computational problem from cellular automata is quite easy. We will see in chapter 3 several ways of defining this, with different computational complexities. For the moment, let us focus on the following problem:

Definition 2.2.1 Let F be a cellular automaton over state set Q_F. The problem $\textsc{Pred}_F : Q_F^* \to Q_F$ is defined by:

$$\textsc{Pred}_F(x) = F^*(x)[1]$$

Where $F^*(x)[1]$ means the first letter of word $F^*(x)$, as defined in section 1.1.1.

We will see why this problem is P-complete later, along with a stronger construction. For the moment, what interests us in this problem is the behavior of its circuit complexity with respect to sub-automaton simulation and bulking, as defined in Definition 1.2.3. To define precisely what a "higher complexity" means, we need the following relation, already defined in a first version in [36], finally corrected by [37]:

Definition 2.2.2 Let f and g be two functions of $\mathbb{N} \to \mathbb{N}$. We write that $f \prec g$ if there are non-constant affine functions (that is, of the form $x \mapsto ax+b$ for some constants a and b) α, β, γ, δ such that:

$$\alpha \circ f \circ \beta \leqslant \gamma \circ g \circ \delta$$

It is not difficult to see that this relation is a pre-order relation, that is, reflexive and transitive.

Proposition 2.2.5 Let F be a cellular automaton. If $\textsc{Pred}_F$ has a circuit family $(C_n)_n$ on some alphabet Q and gate set $\mathcal{G}$ complete for Q, then for any bulking parameters m, t and z, there is a circuit family $(C'_n)_n$ computing $\textsc{Pred}_{F^{\langle m,t,z \rangle}}$, such that:

$$d(C'_n) \prec d(C_n) \quad \text{and} \quad d(C_n) \prec d(C'_n)$$

$$|C'_n| \prec |C'_n| \quad \text{and} \quad |C_n| \prec |C'_n|$$

Where $\prec$ is the relation of Definition 2.2.2.

Proof. We prove this by showing that each ingredient of the simulation preserves the complexity. Since the alphabet is changed several times during this proof (and the gate set with it), as stated by Proposition 2.2.3, we need to be careful about alphabet problems. However, $\mathcal{G}$ is complete for Q, this is not a huge problem:

- First, if there is a circuit family $(C_n)_n$ for $\textsc{Pred}_F$, then we can transform it into a circuit family for $\textsc{Pred}_{F^{\langle m,0,0\rangle}}$ by simply taking, for each n, m copies of $C_{m\lceil n/m\rceil}$, with their output vertices renumbered. The output of this circuit is thus an element of $\left(Q_{C_n}\right)^m$, which is exactly what we need.

  Conversely, if there is a circuit family $(C_n)_n$ for $\textsc{Pred}_{F^{m,0,0}}$, then we can easily transform it into a circuit family for $\textsc{Pred}_F$: for each input size n, we consider $C_{\lfloor n/m\rfloor}$, and add a circuit C' to simulate the evolution of F for the $n \bmod m$ remaining steps, from the value of $\textsc{Pred}_{F^{\langle m,0,0\rangle}}$ computed by $C_{\lfloor n/m\rfloor}$. This can clearly be done by a circuit of size and depth depending only on m and $|Q|$, by Proposition 2.2.3.
- Now, if we have a circuit family $(C_n)_n$ to compute $\textsc{Pred}_F$, then $\textsc{Pred}_{F^{\langle 0,t,0\rangle}}$ is clearly computed by family $(C_{n\cdot t})_n$. However, the converse needs a more complicated circuitry, since we need to compute the evolution of F even on configurations of size n such that $(n-1)$ is not a multiple of t.

  In order to compute the evolution of F for configurations of size n, we need to iterate F $t_0(n) = \frac{n-1}{2r}$ times, where r is F's radius. Thus, if we have a circuit family to compute $\textsc{Pred}_{F^{\langle 0,t,0\rangle}}$, we can simply copy it $2r\cdot(t_0(n) \bmod t)+1$ times, to compute the result of $t\cdot\left\lfloor\frac{t_0(n)}{t}\right\rfloor$ iterations of F. Then, we can simply connect the $O(2r\cdot(t_0(n) \bmod t)+1)$ output vertices of the resulting circuit to another circuit iterating F for the $t_0(n) \bmod t$ remaining iterations. This is again possible with a circuit of depth and size depending only on t and $|Q|$, by Proposition 2.2.3.
- As for the composition with the shift, this can be done easily by adding $2\left\lfloor\frac{n-1}{r}\right\rfloor$ ignored inputs to the circuit of size n, with r the radius of F. This is because the composition of F with the shift is a cellular automaton of radius $r+1$, where the $r-1$ left cells are ignored by the local rule.

  Conversely, to compute $\textsc{Pred}_F$ with circuits for $\textsc{Pred}_{\sigma\circ F}$, we can simply, for each input size, feed constants states to the ignored inputs. This is still a valid circuit, and the value of the constants does not matter. □

Proposition 2.2.6 If F and G are two cellular automata such that $F \sqsubseteq G$, and there is a circuit family $(C_n)_n$, with alphabet Q and gates set $\mathcal{G}$ complete for Q, computing $\textsc{Pred}_G$, then there is a circuit family $(C'_n)_n$ computing $\textsc{Pred}_F$, such that for all n, $d(C'_n) \in O(d(C_n))$, and $|C'_n| \in O(|C_n|)$.

Proof. For each n, we only need to add a circuit of constant depth and size to compute φ for each input, then φ^{-1} for each output vertex of C_n, if φ is the one-to-one application induced by $\sqsubseteq$. This can be done, by completeness of $\mathcal{G}$ and Proposition 2.2.3. □

Altogether, we have proved that the circuit size and depth of predicting cellular automata was conserved by simulation. That is, anticipating on the terminology of chapter 4, that for any non-decreasing function $f : \mathbb{N} \to \mathbb{N}$, the set of cellular automata F such that there is a circuit family of depth in $O(f)$ computing $\textsc{Pred}_F$ is an *ideal* of the bulking pre-order.

2.3 Monotonic cellular automata

Monotonicity has long been a subject of interest, both in dynamical systems and complexity theory. Several complex processes are amenable to formal analysis thanks to their being monotonic. This is the case for instance, of many biochemical processes such as gene activation networks, or of the "easy" cases of Ising model dynamics, such as bootstrap percolation.

In complexity theory, the importance of monotonic computations is even higher, as *the* most important open problem of the theory, that is, complexity lower bounds, becomes feasible for monotonic functions. The following result, proved by Raz and Widgerson in [38], lower bounds the depth of monotonic circuits that would be needed to decide if a graph (V, E) has a *perfect matching*, which is a decomposition of V into a set of disjoint pairs (v_0, v_1) of E.

Theorem 2.3.1 *(from [38])* Any monotonic circuit computing the perfect matching function on n-vertex graphs requires $\Omega(n)$ depth.

Here "monotonic circuit" means a boolean circuit with all its gates monotonic, with the following definition for a monotonic function:

Definition 2.3.1 Let d be an integer, Q a finite alphabet with a partial order relation $\leqslant$. A monotonic function $f : Q^d \to Q$ is a function such that:

$$\forall x_1, \dots, x_d, y_1, \dots, y_d, y_1 \geqslant x_1, \dots, y_d \geqslant x_d \Rightarrow f(y_1, \dots, y_d) \geqslant f(x_1, \dots, x_d)$$

It is an easy exercise to show that the functions computable with monotonic circuits are exactly the monotonic functions. In the boolean binary case, a monotonic circuit is a circuit whose gates are in $\{\wedge, \vee\}$. To avoid any kind of confusion, this is the right place to state Allgood's second principle (the first one is to be found in [34]):

Allgood's second principle Let C be a circuit of size n. By applying De Morgan's laws at each gate of C that are followed by a $\neg$ gate, Allgood thinks he gets an equivalent circuit of size n where the negations only appear at the inputs.

Allgood's wrong. To see this, consider the following circuit:

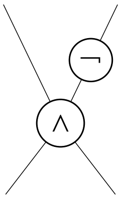


Figure 2.2 - Allgood's construction

To perform his transformation, Allgood would have to duplicate the $\wedge$ gate, and to perform the same operation again at each node of each level of the formula. Although the depth and the computed function really do not change, the size of his circuits, that he once thought in $O(n)$, would rather be of the order of 2^n !

A more reasonable way of simulating a non-monotonic circuit with a monotonic one involves duplication of the inputs:

Algorithm 2.3.1 Each wire is represented by two wires (a, b). The $\wedge$ gates are transformed into two gates ($a \wedge a'$, $b \vee b'$), the $\vee$ into ($a \vee a'$, $b \wedge b'$), and the negations by (b, a). If each input bit x_i is transformed into $(x_i, \neg\ x_i)$, one can check that Algorithm 2.1.1 computes at each step the same values as before, along with the negation of each one.

2.3.1 Planarity

A possible solution, to avoid the solution of Algorithm 2.3.1, is to restrict the form of the underlying graph to a graph where these simulations of non-monotonic gates by wire crossing are impossible: a planar graph. The following result shows that monotonic circuits with a planar underlying graph have an easier circuit value problem:

Theorem 2.3.2 *(from [39])* Any boolean monotonic planar circuit of polynomial size computes the same function as a circuit of polynomial size and polylogarithmic depth.

Now, what does this mean for our model of parallel computing, cellular automata? If we consider cellular automata of arbitrary radius, the same kind of cheating is possible: indeed, the construction of Algorithm crucially used the fact that in any order on Q, for any $q_0, q_1, q_2, q_3 \in Q$, whenever $q_0 < q_1$ and $q_2 < q_3$, the pairs (q_0, q_3) and (q_1, q_2) are incomparable.

We can generalize this to the following lemma:

Lemma 2.3.3 Let Q be a finite set, totally ordered by $<$, and $f : Q^n \to Q$ be any function. If there are m incomparable n-uples $u_1, \ldots, u_m$ of Q^n, then there is a monotonic function g such that for each i, $g(u_i) = f(u_i)$.

Proof. Simply set $g(u_i) = f(u_i)$ for each i, this does not contradict the monotonicity of g, since the (u_i) are incomparable. Then for each i, and all n-uple $x > u_i$, we set $g(x) = \max Q$. Similarly, for all $x < u_i$, we set $g(x) = \min Q$. This is possible because there is no n-uple x such that $u_i < x < u_j$, since the (u_i) are incomparable. □

And then use it to prove that monotonic local rules are as powerful as any other rule, as long as their radius is not too restricted:

Proposition 2.3.4 Let F be a cellular automaton of radius one with alphabet Q. There is a cellular automaton G, with alphabet $\{0,1\}$, and a monotonic local rule, such that $F \preccurlyeq_{\sqsubseteq} G$.

Proof. Let $Q = \{q_0, \dots, q_{n-1}\}$, with $q_0 < q_1 < \dots < q_{n-1}$. There are n different blocks of length $n+3$ of the form $u_i = 110^{i+1}10^{n-i}$ (for $0 \leqslant i < n$), and they are pairwise incomparable. We can thus encode the states of Q by $n = |Q|$ blocks of this form. To encode F of radius one, we need G to have radius $2n+5$ (so that each cell "sees" the whole encoding of the neighborhood). The idea is to preserve the sub-blocks of the form 110. To do this, we set:

$$\forall a \in \{0,1\}^{2n+3}_{,}\ b \in \{0,1\}^{2n+5}_{,} G(a110b) = 0$$

$$\forall a \in \{0,1\}^{2n+4}_{,}\ b \in \{0,1\}^{2n+4}_{,} G(a110b) = 1$$

$$\forall a \in \{0,1\}^{2n+5}_{,}\ b \in \{0,1\}^{2n+3}_{,} G(a110b) = 1$$

For any other input encoding the neighborhood $(a,b,c) \in Q^3$, G computes the encoding of $F(abc)$. If a is encoded by block u_i, b by u_j, c by u_k and $F(abc)$ by u_l for some i,j,k,l, then we set, for any $x \in \{0,1\}^{n-l-1}$ and any $y \in \{0,1\}^{l+2}$:

$$G(x110^{i+1}0^{n-i}110^{j+1}0^{n-j}110^{k+1}0^{n-i}y) = 1$$

In any other input $x \in \{0,1\}^{4n+11}$, falling in neither of the two cases, we set $G(x) = 0$. Finally, by Lemma 2.3.3, we can complete the definition of G so that it remains monotonic. It is clear that $F \preccurlyeq_{\sqsubseteq} G$, since this proof essentially shows that $F \sqsubseteq G^{\langle n+3,1,0 \rangle}$. □

Corollary 2.3.5 There is an intrinsically universal cellular automaton with a monotonic local rule.

2.3.2 The dynamics of monotonic automata

This notion of monotonicity mirrors the notion in circuits, and the simulation techniques from circuits easily generalize to this context, as Proposition 2.3.4 shows. Is is thus natural to ask what the dynamics of cellular automata with a monotonic local function, and a restricted radius, would become. The natural example to start with is the case of automata depending only of two neighbors:

Definition 2.3.2 A *planar monotonic* cellular automaton is a cellular automaton with neighborhood $\{0,1\}$, such that there is an order relation on its alphabet making its local function monotonic.

The first result we are going to show gives an example both of what kind of methods may be used to analyze monotonic planar cellular automata, and of the possible behaviors of these cellular automata.

Proposition 2.3.6 There are only two surjective planar monotonic cellular automata: Id and σ.

Proof. A convenient method to study planar monotonic cellular automata is the following representation of the lattice of Q^2 ordered by $<$:

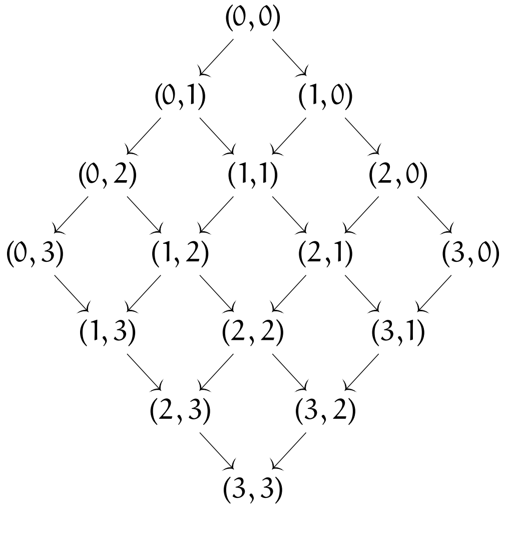


Figure 2.3 - The lattice of $(Q^2,<)$

A planar monotonic rule affects a value to each vertex of the lattice, such that the sequence of values induced by any path is non-decreasing. Let us call 0 the minimal state, $n = |Q|$ the maximal one, and we must prove that one of the following two cases holds:

- For each $q \in Q$, q appears on each vertex of line $(q,0),\dots,(q,n-1)$.
- For each $q \in Q$, q appears on each vertex of column $(0,q),\dots,(n-1,q)$.

State 0 cannot appear at both $(0,1)$ and $(1,0)$, because a configuration x with only zeros and x' with only zeros except at one position where it has a one, would have the same image, and this would contradict the Moore-Myhill theorem (Theorem 1.3.1). Thus, since all the paths in the lattice must be non-decreasing, 0 can appear only on the first line, or on the first column of the diagram (that is, on one of the two paths $(0,0),\dots,(0,n-1)$ and $(0,0),\dots,(n-1,0)$). By the same argument, $n-1$ must appear on the last line or last column. Moreover, since 0 and $n-1$ cannot appear on the same vertex, either they appear on the first and the last line, or on the first and the last column. Without loss of generality, let us assume, like on the following figure, that they both appear on a line.

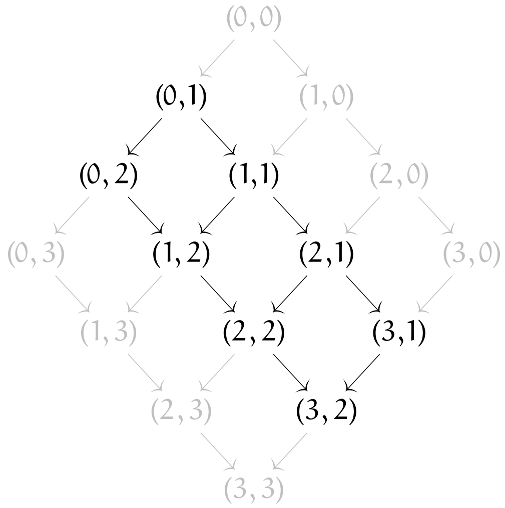


Figure 2.4 - The lattice of $(Q^2_, <)$

Now, since F is onto, for any $q \in \{1,\dots,n-2\}$, the two letters word $0q$ must have an antecedent by F. Therefore, since the only antecedents for 0 end with state 0 itself, an antecedent of $0q$ must be of the form $q_0 0 q_1$, with $q_0, q_1 \in Q$, and thus, the cardinal of $\{F(0q_1) \mid q_1 \in Q\}$ must be at least $q-2$. This shows that all the vertices of column $(0,0),\dots,(0,n-1)$ must have different values.

The same argument applies to column $(n-1,0),\dots,(n-1,n-1)$. Since the paths in the lattice are all non-decreasing, for each $q \in Q$, all the states between $(0,q)$ and $(n-1,q)$ must be equal, and this completes the proof. □

This proposition also holds for the more general case of monotonic cellular automata with arbitrary radius. The main difference is that the lattice is much more complicated and its drawing is not of much help. However, we can generalize our arguments:

Proposition 2.3.7 Let F be a surjective monotonic cellular automata of radius r. Then $F \in \left\{\sigma^{-r}, \sigma^{-r+1}, \dots, \sigma^{r}\right\}$.

Proof. Let again number the states from 0 through $n-1$. Then:

- By the same arguments as in the proof of proposition 2.3.6, at least one of the configurations with only one 1, that is, $F(10^{2r}), F(010^{2r-1}), \dots, F(0^{2r}1)$, say the ith one, must be non-zero. This means that all the antecedents of 0 must be of the form $q_0 \dots q^{2r}$, with $q_i \neq 0$, by monotonicity of F.

We now need a way to place the same kind of constraints on antecedents of 0 by F, that we placed with the antecedents of $0q$ in the proof of proposition 2.3.6:

- Consider the antecedents by F of the configurations of the form $0^{i-1}q0^{2r-i}$, for all $q \in Q$. Since all the antecedents of 0 must have a 0 at the ith position, this means that the $F^{-1}(\{0^{i-1}q0^{2r-i}\})$ (for $q \in Q$) are all of the form ${*}^r0^iq_00^{2r-i}{*}^r$, with $*$ meaning "any state of Q". Thus, the $\{F(0^{i-1}q_{00}^{2r-i}|q_0 \in Q\}$ are all different, and, since there are $|Q|$ of them.
- Again, the same argument applies to the configurations of the form $(n-1)^{i-1}q(n-1)^{2r-i}$: all the images of these configurations (for all $q \in Q$) are different.

Altogether, we have proved that $F(0^{i-1}q0^{2r-i}) = F\left((n-1)^{i-1}q(n-1)^{2r-i}\right) = q$. Thus, for any x between these two configurations (that is, a x such that $x[i] = q$), $F(x) = q$. □

The proof of Theorem 2.3.2 proceeds by enumerating the gates, from left to right in a planar embedding of the circuit's underlying graph, and considering "blocks" of consecutive 1s in this enumeration. A key argument of the proof is that a block can never split. Unfortunately, for multi-valued logic, this argument does not work. In section , we give conjectures on why is it so. However, an equivalent notion can still be found:

Definition 2.3.3 Let Q be a finite set with a total order relation $\leqslant$. A *local maximum* in a configuration $x \in Q^{\mathbb{Z}}$ is a position i in x such that $x[i-1] \leqslant x[i] \geqslant x[i+1]$, and $x[i-1] \neq x[i]$ or $x[i] \neq x[i+1]$ (or both).

A *local minimum* is the same, with the order relation reversed.

This definition allows us to state the following lemma:

Lemma 2.3.8 Let F be a planar monotonic cellular automaton. If there is a finite number of local extrema in some configuration x for F, then there cannot be more in $F(x)$.

Moreover, if there is a leftmost extremum in x, and it is a local maximum (respectively a local minimum), then the left extremum in $F(x)$ is also a local maximum (respectively a local minimum).

Proof. There is no great intuition behind this proof. We just consider all the cases for the orders between the states of a configurations, compute the relations on the states of its image induced by the monotonicity of F, and check that it works.

First consider the case of an isolated local maximum. Let x_i^t be the cell at position i after t iteration of the rule. If we assume that, at time t, $x_{i-2}^t \leqslant x_{i-1}^t \leqslant x_i^t \geqslant x_{i+1}^t \geqslant x_{i+2}^t$, then at time $t+1$:

$$\text{Either} \quad x_{i-2}^{t+1} \leqslant x_{i-1}^{t+1} \leqslant x_i^{t+1} \geqslant x_{i+1}^{t+1} \quad \text{or} \quad x_{i-2}^{t+1} \leqslant x_{i-1}^{t+1} \geqslant x_i^{t+1} \geqslant x_{i+1}^{t+1}$$

Because the pairs $\left(x_{i-1}^t, x_i^t\right)$ and $\left(x_i^t, x_{i+1}^t\right)$ are incomparable or equal, but since $\geqslant$ is a *total* order relation, their images under F's local rule must be comparable.

Assume now that the local extrema are not isolated. In this case, we split the configuration at points where two consecutive relations between states are equal: either $a \leqslant b \leqslant c$, or $a \geqslant b \geqslant c$. Between these zones, the relations are alternated, and each state is either a local maximum, or a local minimum.

Let i be a position such that $x_i^t \leqslant x_{i+1}^t \leqslant x_{i+2}^t > x_{i+3}^t$ (a symmetric argument holds for $x_i^t \geqslant x_{i+1}^t \geqslant x_{i+2}^t$). Then we already know that $x_i^{t+1} \leqslant x_{i+1}^{t+1}$. Know let j be the first position greater than i such that:

1. Either $x_j^t \leqslant x_{j+1}^t \leqslant x_{j+2}^t$.
2. Or $x_j^t \geqslant x_{j+1}^t \geqslant x_{j+2}^t$.

But in either case, since x_{i+1}^t is not a local extremum at step t, the relations between two consecutive states between position $i+1$ and j are undetermined. But this undetermined portion has only one more cell, and since this changes the parity of its length, there cannot be more alternations, and hence, no more local extrema.

Moreover, if there is an alternation after position i, then it must be a local maximum (a local minimum in the symmetric argument). Therefore, the leftmost extremum remains a maximum after one step if it was a maximum, and a minimum if it was a minimum. □

This allows us to define a notion of "energy" in these automata. The problem is that such a definition, counting the local extrema, would be obviously decreasing on finite configurations (and thus Lemma 2.3.8 would be useless), and infinite on non-trivial infinite configurations. Considering periodic configurations solves this problem:

Definition 2.3.4 Let Q be a finite set, and x a p-periodic configuration of $Q^{\mathbb{Z}}$, for some integer p. We call the *energy* of x, and we write $E(x)$, the following quantity:

$$E(x) = \frac{|\{i \in \{0, \dots, p-1\} \mid i \text{ is a local extremum}\}|}{p}$$

In words, the energy of a configuration is the number of local extrema in a period, divided by the length of the period.

Lemma 2.3.9 Let F be a monotonic planar cellular automaton over alphabet Q, and $x \in Q^{\mathbb{Z}}$. Then $E(F(x)) \leqslant E(x)$.

Proof. This is an easy corollary of Lemma 2.3.8.

□

This does not yet allow to conclude, but at least allows us to suspect that these cellular automata do not have a great simulation power:

Conjecture 2.3.1 There is no monotonic planar intrinsically universal cellular automaton. For example, no monotonic planar cellular automaton can simulate the cartesian product of σ and σ^{-1}.

By the way, the proofs that a monotonic planar circuit of polynomial size is equivalent to a circuit of polylogarithmic depth (see [39] for instance) do not apply when the alphabet is not boolean. To our knowledge, this problem has not been solved yet:

Open problem 2.3.2 Let C be a monotonic planar circuit of polynomial size on some alphabet Q of cardinality at least 3. Is there an circuit computing the same function in polylogarithmic depth? In logarithmic depth?

2.4 Bounds on the circuit complexity of automata

In this section, we present several research directions showing how to approach complexity questions, and classical complexity classes, with cellular automata. This idea is still at the stage of a research plan, as even the basic results of complexity theory, as seen in the last section about monotonicity, are yet to be adapted. However, several clues show that this may be a pertinent approach: as we will see, the circuit families computing cellular automata are naturally uniform (see Definition 2.1.3), and the objects may be studied from a topological point of view. Moreover, we will see in chapters 3 and 4 several ways to lower bound the “simulation power” of cellular automata by elegant arguments.

This section is focused on the simplest class of NC circuits, namely, NC^0, the uniform families of circuits of constant depth. The cellular automata that can be predicted by circuits of these families obviously have a constant number of inputs, at each step. The converse also holds:

Definition 2.4.1 Let F by a cellular automaton with alphabet Q and radius r. A dependency of F at step t is a position $-rt \leqslant i \leqslant rt$ such that there is a configuration $x \in Q^{2rt+1}$ and two states $q, q' \in Q$, such that:

$$F^t(x_{-rt}...x_{i-1}qx_{i+1}...x_{rt}) \neq F^t(x_{-rt}...x_{i-1}q'x_{i+1}...x_{rt})$$

Moreover, we call $D_t(F)$ the set of dependencies of F at step t, and we call a *witness* of a position i, a configuration x of Q^{2rt+1} such that there are q and q' such that $F^t(x_{-rt}...x_{i-1}qx_{i+1}...x_{rt}) \neq F^t(x_{-rt}...x_{i-1}q'x_{i+1}...x_{rt})$.

Our first result shows that it is undecidable whether a given cellular automaton has a bounded number of dependencies:

Proposition 2.4.1 Let F be a cellular automaton and m an integer. The problem of deciding whether $|D_t(F)| \leqslant m$ for all t is undecidable.

Proof. If it were decidable, we could use an algorithm for it to decide the nilpotency problem, which was shown undecidable by Kari [40]. To show this, from any cellular automaton G, we construct a cellular automaton depending on at most m cells if and only if G is nilpotent.

Let G, be a cellular automaton with alphabet Q, radius r. Take some $q \in Q$ such that $g(q^{2r+1}) = q$ (if there is no such q, then G is not nilpotent). We construct a cellular automaton F by adding another layer to G, with alphabet $\{0,1\}$, radius r, whose behavior is the following:

- if all the neighbors of a cell, on the G component, are in state q, then become 0.
- else become 1.

If there is some t_0 such that $\forall t \geqslant t_0, \forall x,\ G^t(x) = q$, then for any $t \geqslant t_0$, $D_t(F) = \emptyset$. Else, there are configurations where F computes the "or" of all the new layer. To see this, let t_0 be any integer, and x be such that $G^{t_0}(x) \neq q$. Since $g(q^m) = q$, there must be, at each step t between 0 and t_0, a subword of the $F^t(x)$ with the qs at most r cells from each other. Let us take a maximal such subword. If its size is, at some step t_1, larger than m, then the new layer is computing the "or" of at least $m+1$ cells, and $F^{t_0-t_1}$ depends on more than m cells. Else, we can take a maximal subword, and duplicate it several times with exactly $r+1$ cells in state q between the copies. In this case, there can be no collision between the copies, and the new layer is computing the "or" of the whole configuration. □

Now, our interest for this class of cellular automata with a bounded number of dependencies comes from the following result, linking a classical complexity class, NC^0, with these cellular automata:

Proposition 2.4.2 Let F be a cellular automaton of radius r. Then PRED_F has a family of circuits in NC^0 if and only if there is an integer m such that for all t, $|D_t(F)| \leqslant m$.

Proof. As we said, the $\Rightarrow$ direction is obvious. Now, if for all t, $|D_t(F)| \leqslant m$, then we only need to construct an algorithm generating a family, and using only a logarithmic amount of space in the size of the input.

We prove, by induction on the size of the configurations, that there is an algorithm, running in logarithmic space, and computing, on input t written in unary, a circuit for PRED_F on configurations of size $2rt+1$, where the (at most m) inputs are labeled by their position on the configuration.

- For configurations of size $2r+1$, we can enumerate all the functions of $Q^{2r+1} \to Q$, and take the one that corresponds to F's local rule. We just have to check this, for each function, on all the configurations of size $2r+1$.
- Assume we have a procedure working in logarithmic space to find a circuit C_t for size $2rt+1$. Then for configurations of size $2rt+2r+1$, we can build a representation of a circuit C with at most $m+2r+1$ inputs, computing the composition of C_t and F's local rule.

  Then, we enumerate again all circuits with m inputs, and verify each time, on all the parts of the configuration F^{t+1} might depend on (that is, all the words of size $2r+1$ around the positions of C_t's inputs), if the enumerated circuit computes the correct function. This involves circuit evaluation, but since the circuits are all of constant size, with at most m integers represented in binary, this evaluation can clearly be done in logarithmic space.

  Once the correct circuit has been found, we use the logarithmic work space to compute the new positions of the inputs and label the input nodes. Since F^{t+1} is a cellular automaton and depends on at most m cells, there must be a circuit with m inputs computing it, and this circuit is equivalent to the composition of F^t with F's local rule. □

Now, our conjecture about those automata with a constant number of dependencies is the following:

Conjecture 2.4.1 Let F be a cellular automaton. If there is an integer m, such that for all t, $|D_t(F)| \leqslant m$, then the dependencies are not too far from lines in the space time diagram. More precisely, there are at most $2m$ real constants $\lambda_1, \ldots, \lambda_m$, $\epsilon_1, \ldots, \epsilon_m$, such that:

$$\forall t,\ D_t(F) \subseteq \bigcup_i \{\lambda_i t - \epsilon_i, \ldots, \lambda_i t + \epsilon_i\}$$

It is quite easy to see why this conjecture works for $m = 0$. For $m = 1$, this not really difficult either:

Proposition 2.4.3 Let F be a cellular automaton such that for all t, $|D_t(F)| \leqslant 1$. Then Conjecture 2.4.1 is true for F.

Proof. Let Q be F's alphabet, and r its radius. For $x \in Q^{2r+1}$, there is an index $-r \leqslant i \leqslant r$ such that $F(x) = g(x[i])$. Hence, $F^t = g^t(x[i \cdot t])$ □
In the case of $m = 2$, we can do almost the same:

Proposition 2.4.4 Let F be a cellular automaton such that for all t, $|D_t(F)| \leqslant 2$. Then Conjecture 2.4.1 is true for F.

Proof. Let us call i_t and j_t the two dependencies of F at time t. There are two cases:

- Either the two dependencies can be arbitrary far from each other. In this case, for all N, there is a time τ_0 at which the two dependencies are N cells from each other. We will choose a useful value for N later. For now, for all t, let i_t be the position of F's leftmost dependency at time t, and j_t be the position of its other dependency.

  By hypothesis, for all t, there is a function $g_t : Q^2 \to Q$, such that for all x, $F^t(x) = g_t(x[i_t], x[j_t])$. Since there are $|Q|^{2|Q|}$ functions of $Q^2 \to Q$, by the pigeonholes principle, between times τ_0 and $\tau_0 + |Q|^{|Q|^2} + 1$, at least two must be equal. Thus, let t_0 and t_1 two integers such that $\tau_0 \leqslant t_0 < t_1 \leqslant \tau_0 + |Q|^{|Q|^2} + 1$.

  We claim that, for $t_2 = t_1 + (t_1 - t_0)$:

  - $g_{t_2} = g_{t_1}$
  - $i_{t_2} = i_{t_1} + \left(i_{t_1} - i_{t_0}\right)$ and $j_{t_2} = j_{t_1} + \left(j_{t_1} - j_{t_0}\right)$

  Indeed, for all $x \in Q^{2rt_2+1}$:

  $$F^{t_2}(x) = g_{t_1}\left(F^{\delta_t}(x[i_1 - r\delta_t]...x[i_1 + r\delta_t]), F^{\delta_t}(x[j_1 - r\delta_t]...x[j_1 + r\delta_t])\right)$$

  $$F^{t_2}(x) = g_{t_0}\left(F^{\delta_t}(x[i_0 - r\delta_t]...x[i_0 + r\delta_t]), F^{\delta_t}(x[j_0 - r\delta_t]...x[j_0 + r\delta_t])\right)$$

  Hence, since we have assumed that i_1 and j_1 were far away from each other (we should choose $N > 4r|Q|^{|Q|^2}$), this means that $i_2 - i_1 = i_1 - i_0$, and, similarly, $j_2 - j_1 = j_1 - j_0$.
- Or there is some constant d such that for all t, $|i_t - j_t| \leqslant d$, and in this case, we use a similar argument on the block of size d between i_t and j_t.

  Indeed, for each t, there is a function $g_t : Q^d \to Q$ and an index i_t such that for all $x \in Q^{2rt+1}$, $F^t(x) = g_t(x[i_t], ..., x[i_t + d - 1])$. Again, since there is a finite number of such functions, in the $|Q|^{|Q|^d}$ first elements of sequence $(g_t)_t$, at least two of them must be equal.

  By the same argument, we conclude that the sequence $(i_t)_t$ is ultimately periodic. □

3 Communication complexity and intrinsic universality

Many processes in interacting systems, whether computational, social, biological... may be viewed as series of *communication* between the parts of the system. This is the case, for instance, in economics: economists, such as Hayek [41] or Stiglitz [42], sometimes think of their systems as games of communications between their actors, and prove results with hypothesis such as the communication of information among the "actors". This is the case, for instance, in social networks, that are actually very few things more than communication networks. This was also the case, for instance, in section 2.3, where adding an hypothesis on the locality of communications changed drastically our computational power.

The formalism developed by Yao in 1979 (see [12]), in order to study the communications inside a system, proved really pertinent in many fields of complexity theory. In this model, a system must compute some function of an input distributed between its parts (often called *players* in the sequel). Since each player knows only his part of the input, they need to communicate in order to compute the function. The *communication complexity* of the function is then the total quantity of information that needs to be communicated between the players to compute the function, without any assumption on their computational power.

3.1 Computation as transmission of information

In his seminal paper, Yao introduced this model with two players, giving to each of them arbitrary computational power.

3.1.1 Definitions and first properties

Let X and Y be two finite sets, and $\varphi : X{\times}Y \rightarrow \{0,1\}$ a decision problem on space $X{\times}Y$. Two players, Alice and Bob, need to compute φ for some input $(x,y) \in X{\times}Y$. The problem is that Alice knows only x, and Bob only y. In order to perform this task, they may thus need to communicate using a *protocol,* chosen beforehand, and depending only on φ. This protocol must specify, at each stage, whether the protocol is over, and the result has been found, or else, who speaks, what he or she says, as a function of his or her input, and of the previous stages of the protocol, as this is the only information available to the players.

In order to measure the amount of communication needed to compute φ, we follow a more formal definition, and represent the protocols as binary trees, like the one on figure 3.1. In these "protocol trees", each internal node v is labeled by either a function $a_v : X \rightarrow \{l,r\}$, or a function $b_v : Y \rightarrow \{l,r\}$, and each leaf is labeled by either 0 or 1. Finally, we say that the protocol *computes* function φ correctly if for all $(x,y) \in X{\times}Y$, the leaf reached from the root by turning left at each node v, if $a_v(x) = l$ (or $b_v(y) = l$), and right else, is labeled by $\varphi(x,y)$.

For the players, this means that each node v labeled by an a_v corresponds to a round where Alice speaks, and each node labeled by b_v corresponds to one where Bob speaks. At each node, the player who speaks may say only one bit, but he is allowed to speak several times consecutively. Now what interests us in this formalism, is the number of communicated bits *in the worst case*, that is, the maximum length of a path from the root to a leaf, or the tree height:

Definition 3.1.1 *(from [43])* For a function $\varphi : X{\times}Y \to \{0,1\}$, the (deterministic) communication complexity of φ, also called its *cost* and written $D(\varphi)$, is the minimum over all protocols $\mathcal{P}$ computing φ, of the height of $\mathcal{P}$.

There is a very simple protocol, that works for any function: Bob sends all of his input to Alice, she computes and outputs the result alone, and we are done. But in many cases, we can be more clever than this. For instance, to compute the sum of all the bits in the input, Alice and Bob may only communicate $O(\log n)$ bits: Alice may sum all the bits in her input, send the result to Bob, and then Bob adds this sum the sums of the bits in his input, and outputs the answer.

In many cases though, there is no better protocol than the trivial one. To prove such lower bounds, Yao considered the *matrix* M_φ of function φ, with its rows indexed by X, and its columns indexed by Y, and defined by $M_\varphi(i,j) = \varphi(i,j)$, and proved the following theorem:

Theorem 3.1.1 *(from [43])* Any deterministic protocol of cost c, computing function φ, induces a partition of M_φ into 2^c monochromatic rectangles, that is, sets $A{\times}B$ such that $A \subseteq X$ and $B \subseteq Y$.

Proof. We prove by induction on the height of the protocol tree that for any node v, the set R_v of inputs that reach v is a rectangle:

- At the root, this is clearly the case since $R_r = X{\times}Y$.
- Let $R_v = A_v{\times}B_v$ be the rectangle corresponding to some internal node v, labeled by a_v. Then the inputs reaching v_l and v_r, the two sons of v, are respectively $R_{v_r} = \left(A_v \cap a_v^{-1}(\{l\})\right){\times} B_v$ and $R_{v_l} = \left(R_v \cap a_v^{-1}(\{r\})\right){\times}B_v$, which are again rectangles. □

Thanks to this theorem, it becomes possible to lower bound the communication complexity of a given function. For instance, consider the equality function, in which Alice and Bob each receive an n-bit string, and they must decide whether these strings are actually the same. In the sequel, we will often call this problem EQ. The matrix of this function is thus the identity matrix. Proving a lower bound on the communication complexity of a function requires finding a lower bound on the number of monochromatic rectangles in its matrix. To show this, we introduce the following definition of *fooling sets*. The idea is that no two elements of a fooling set can be in the same monochromatic rectangle:

Definition 3.1.2 *(from [43] and [44])* A *fooling set* $S \subseteq X{\times}Y$ for a function φ is a set of inputs for φ such that for any two pairs (x_0,y_0) and (x_1,y_1) in S, either (x_0,y_1) or (x_1,y_0) is not in S.

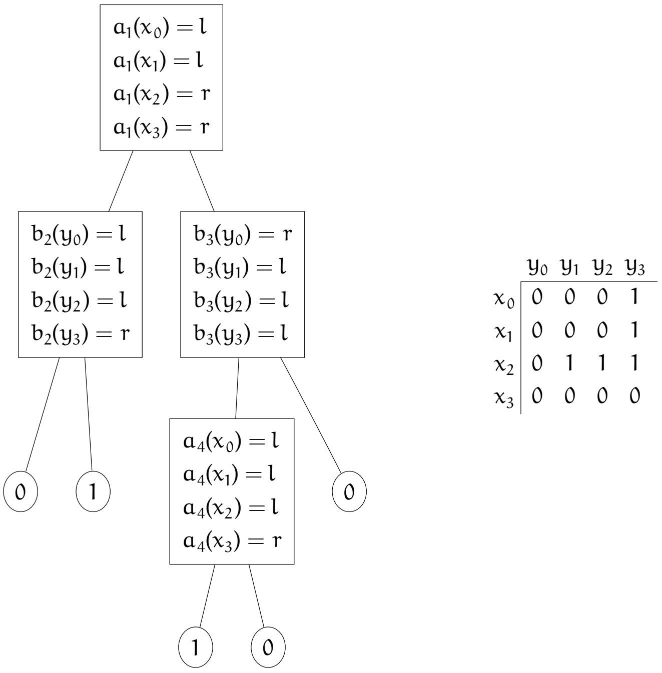
Consider again the example of the EQ problem, whose matrix is the identity. The set $S = \{(x,x) \mid x \in \{0,1\}^n\}$ is a fooling set of size 2^n, since for any two different elements (x,x) and (y,y) of S, both (x,y) and (y,x) are colored 0, while (x,x) and (y,y) are colored 1 in EQ's matrix.

Finally, we mention another way of proving lower-bounds on the communication complexity of a function:

Theorem 3.1.2 Let φ a function of $X \times Y \to Z$. Then $\log_2(\mathrm{rank} M_f) \leqslant D(f)$, when the rank is taken over the field of reals.

Proof. Let R be a monochromatic rectangle in matrix M_φ, then we define the matrix M_R as $M_R(i,j) = 1$ if $(i,j) \in R$, and 0 else. For a partition of M_φ into monochromatic rectangles $(R_i)_i$, $M_\varphi = \sum_i M_{R_i}$. Since $\mathrm{rank}(M_\varphi) \leqslant \sum_i \mathrm{rank}\big(M_{R_i}\big)$, and the M_{R_i} are all of rank 1, the result follows. □

However, it is not known whether this bound is tight or not. This is called the "logrank conjecture":



$$\begin{array}{c|cccc} & y_0 & y_1 & y_2 & y_3 \\ \hline x_0 & 0 & 0 & 0 & 1 \\ x_1 & 0 & 0 & 0 & 1 \\ x_2 & 0 & 1 & 1 & 1 \\ x_3 & 0 & 0 & 0 & 0 \end{array}$$

Figure 3.1 - A protocol tree, along with the matrix of the computed function

Conjecture 3.1.1 For any function $\varphi : X \times Y \to Z$, $D(\varphi) \in O(\log_2(\mathrm{rank} M_\varphi))$.

3.1.2 Relations with classical complexity

The idea of communication complexity initially originated from the study of parallel computing. But as we said in the introduction of this chapter, many processes, in any computational task, can be viewed as a protocol for communicating different parts of an input. In fact, the idea had already been present, in sequential computing, before. The most notable occurrence, to our knowledge, is to be found in the proof of the following theorem, by Alan Cobham:

Theorem 3.1.3 *(from [3])* The set PAL of palindromes on an alphabet Σ requires $\Omega(n^2)$ time on a Turing machine with one tape and one head.

The idea of this theorem is to draw a line between two halves of the used space, and to watch the sequence of states the machine is in when the head crosses this line. These sequences are called *crossing sequences*. This idea has much to do with communication complexity, and has been generalized by Juraj Hromkovič and Georg Schnitger to any Turing machine. In the following theorem, the input is split in the middle between Alice and Bob: Alice gets the left $n_A = \lfloor n/2 \rfloor$ bits, and Bob gets the right $n_B = \lceil n/2 \rceil$ bits, of the input. The communication problem is now well defined, with function $\varphi_{\mathcal{L},n} : \{0,1\}^{n_A} \times \{0,1\}^{n_B} \to \{0,1\}$ defined by $\varphi_{\mathcal{L},n}(x,y) = \chi_{\mathcal{L},n}(\overline{xy})$, the characteristic function of $\mathcal{L}$.

Theorem 3.1.4 *(from [45])* Let $\mathcal{L} \subseteq \{0,1\}$. For any Turing machine $\mathcal{M}$ with one tape and one head recognizing $\mathcal{L}$, let $T_{\mathcal{M}}(n)$ be its running time on inputs of size n. We have:

$$T_{\mathcal{M}}(n) \in \Omega\Big(D(\mathcal{L})^2\Big)$$

As a natural model of parallel computing, boolean circuits have even more to do with communication complexity. Indeed, the following two results show how a gate in a circuit, or in a formula, can be seen as a communication step between two sub-circuits.

Proposition 3.1.5 For any function $\varphi : \{0,1\}^n \to \{0,1\}$, any input splitting σ, and any binary circuit C_φ computing φ, we have:

$$|C_\varphi| \in \Omega(D(\varphi \circ \sigma))$$

Proof. The proof is done by constructing a protocol computing φ, with cost $|C_\varphi|$, if C_φ is a circuit for φ. This is done by coloring all the gates in the circuit by either $\mathcal{A}$ or $\mathcal{B}$, depending on which players knows the output of the gate. The players then follow the circuit from the leaves to the root, using the following procedure:

- The value of each leaf, that is, of each node of level 0, is known either by Alice or by Bob, depending the repartition of the inputs.

- Then, assume we have a coloring of all the gates at level l. Then we can color each gate of level $l+1$ either by $\mathcal{A}$ if both of its inputs have color $\mathcal{A}$, by $\mathcal{B}$ if both of its inputs have color $\mathcal{B}$. If the colors differ, either Alice communicates the value of the gate's input she knows, and the gate gets colored in $\mathcal{B}$, or Bob does it, and the gate gets colored in $\mathcal{A}$. □

Unfortunately, this technique does not allow to prove lower bounds on circuit size larger than n for inputs of size n, which, as we have seen in section 2.1.1, is not much better than proving that the computed function depends on all its input. However, what is notable in this proposition is that we did not need to make any hypothesis on the particular gate set used by the circuit.

3.1.3 Splitting the input

The framework of communication we have introduced only considers functions working on Cartesian products of two sets. Studying traditional computational problems within it requires to decide how to split the input between the players. Theorem 3.1.4, for instance, required that the input be split in the middle into two contiguous parts of equal length, with Alice getting the $n/2$ first bits, and Bob the $n/2$ last ones. In proposition 3.1.5, however, the splitting was freer, but since we were proving a lower bound, the sensible choice was the one maximizing the communication complexity.

There is a discussion of the topic in chapter 7 of [43], where the propositions of splitting are either the worst partition over all sets, as in proposition 3.1.5, or the one minimizing the communication complexity, with the additional constraint that the sets be of equal size. In our case, since most of the systems we are going to study in this chapter are cellular automata, which are σ-commuting functions, the natural way of splitting the input is into a left and a right part, choosing the size of these so as to maximize the communication complexity. Indeed, the parts need to be contiguous in order for the complexity to be compatible with the simulations we have defined in 1.2. Moreover, we need to allow to cut the configuration at any position, since it would be easy, by simply applying shifts, to transform any complex automaton into a trivial one. We may thus define, for alphabet Q, any integer n, and $i \leqslant n$, the i-concatenation function $\mathcal{C}_i : Q^i \times Q^{n-i} \to Q^n$. We can now properly define what we mean by the communication complexity of a computational problem:

Definition 3.1.3 Let φ_n an application of $Q^n \to Z$. The communication complexity of φ, hereafter denoted as $cc(\varphi_n)$, is defined by:

$$cc(\varphi_n) = \max_{i \leqslant n} D(\varphi_n \circ \mathcal{C}_i)$$

On some problems, this question is non-trivial. For instance, we will see in section 3.3.1 the example of a cellular automaton with two distinct maxima in its communication complexity. Before that, the following lemma will be quite useful in the understanding of the proofs below:

Lemma 3.1.6 Let $\mathcal{L}$ be a rational language. Then $\mathrm{cc}(\chi_{\mathcal{L}}) \in O(1)$.

Proof. Since there is a finite automaton recognizing $\mathcal{L}$, then for any splitting of the input, Alice begins to run the automaton on the left part of the input, she transmits her last state to Bob, who resumes the recognizing and answers. □

3.2 Communication complexity of cellular automata

The concepts of universality is a fundamental idea in the theory of computing, almost since the beginning of this theory. Its latest extension is the concept of *completeness* in complexity theory. However, proving lower bounds and necessary conditions for complexity is usually a hard task. In this chapter, we generalize the approach started in [14] and [18], to prove necessary conditions for intrinsic universality, using communication complexity. This also allows us to develop another point of view on the concepts of bulking (see [9] and [10]). The idea is the same as completeness for complexity classes: if we study a particular dynamical aspect of a cellular automaton $\mathcal{A}$, then this dynamical aspect must be at least as hard to analyze as the same aspect on the cellular automata we claim $\mathcal{A}$ simulates.

The difference with classical complexity theory is that our objects of study are now, almost by definition, dynamical systems, and thus we may use results of this field of mathematics to help our understanding of these objects. Another difference is that the experimental approach has been present in dynamical systems almost since the beginning, and communication complexity also opens the way to a rich playground of experimentation.

3.2.1 Communication problems

Cellular automata have always been studied under a variety of points of view and questions, whether dynamical, topological, or computational. As observed by [18], some of these properties are "compatible" with the simulations we defined in section 1.2, which means that if property $\mathfrak{P}$ is easy to decide on automaton $\mathcal{A}$, then it must also be easy to decide on automaton $\mathcal{B} \preccurlyeq \mathcal{A}$. In this section, we are going to define what a "property" means in terms of communication problems, and show some properties that are compatible with our notions of simulation:

Definition 3.2.1 Let $\mathcal{A}$ be the set of all cellular automata. A *problem* on $\mathcal{A}$ is a family of functions $(\mathfrak{P}_F)_{F\in\mathcal{A}}$ indexed by $\mathcal{A}$.

3.2.1.1 The prediction problem

We have already seen a communication problem in chapter 2, under the name Pred. Let us recall its definition: This problem is quite natural: it is the problem of predicting the future of a given configuration for the rule under consideration.

Definition 3.2.2 Let F be a cellular automaton over state set Q_F. The problem $\textsc{Pred}_F$ is defined by:

$$\forall x \in Q_F^*,\ \textsc{Pred}_F(x) = F^*(x)[1]$$

Where $F^*(x)[1]$ means the first letter of word $F^*(x)$, as defined in section 1.1.1.

As explained in section 3.1.3, it is easy to convert this decision problem into a communication problem $\varphi_i : Q^i{\times}Q^{n-i}$, where $\varphi_i(x,y) = \textsc{Pred}_F(xy)$. We can then define the communication complexity of $\textsc{Pred}_F$ as $cc(\textsc{Pred}_F) = \max_{0\leqslant i<n} D(\varphi_i)$.

There are classes of cellular automata for which this problem is easy, as proved in [36]. For instance, the class of linear automata (see definition 1.3.6). This class has been often deemed "chaotic" for its topological properties. To our knowledge, the approach closest to ours is to be found in a paper by Chris Moore [26], where the author constructs smart algebraic formulas to predict cellular automata with several distinct algebraic structures. Our approach allows for much simpler proofs with less hypotheses:

Proposition 3.2.1 Let F be a rule on an alphabet Q_F with a particular state 0, $\oplus$ an operator on Q_F, such that $\forall a \in Q_F$, $0 + a = a + 0 = a$, and $F(a \oplus b) = F(a) \oplus F(b)$. Then:

$$cc(\textsc{Pred}_F) \in O(1)$$

Proof. For any $a \in Q_F^i$, any $b \in Q_F^{n-i}$. It is enough for Alice to compute $\textsc{Pred}_F(a0...0)$, for Bob to compute $\textsc{Pred}_F(0...0b)$, and then within $O(1)$ bits of communication, to compute the final result $\textsc{Pred}_F(a0...0) \oplus \textsc{Pred}_F(0...0b)$. □

If F is equicontinuous, this problem is also easy. Indeed, if F is equicontinuous, it has only a constant number of dependencies, as shown by Sablik in [46], and thus Alice and Bob only need to communicate a constant number of cells to compute the result.

3.2.1.2 The temporal invasion problem

Problem TInv is the problem of deciding whether, in a periodic configuration p_u, changing a finite portion of the configuration from its original value to x affects the long-term behavior of the automaton, that is, if the configuration will be forever different from the original one.

Definition 3.2.3 Let Q an alphabet, and u a finite word of Q^* of length n. We denote by p_u the configuration defined by $\forall i \in \mathbb{Z}$, $p_u[i] = u[i \bmod n]$

We denote by $p_u(x)$ the configuration define by $p_u(x)[i] = x$ if $0 \leqslant i < |x|$, and $p_u(x)[i] = p_u[i]$ else.

Definition 3.2.4 Let F be a cellular automaton over state set Q_F, and u some word of Q^*. The problem $\textsc{TInv}_{F,u}$ is defined by:

$$\textsc{TInv}_F(x) = 1 \quad \text{if and only if} \qquad \forall t \in \mathbb{N},\ F^t(p_u(x)) \neq F^t(p_u)$$

In the corresponding communication problem, the input is only x, since u, and thus p_u, are known by both players. From theorem 1.3.1, we can already state the following result:

Proposition 3.2.2 Let F be a surjective cellular automaton over state set Q. Then:

$$\forall u \in Q^*, \ cc(\textsc{TInv}_{F,u}) \in O(1)$$

Proof. From theorem 1.3.1, since p_u and $p_u(x)$ differ only on a finite number of positions, then for all t, if $F^t(p_u) \neq F^t(p_u(x))$ then $F^{t+1}(p_u) \neq F^{t+1}(p_u(x))$. Thus it is enough for Alice and Bob to check if their part of the configuration differs from the corresponding positions in p_u, and this information can be communicated within only one bit of information. □

3.2.1.3 The spatial invasion problem

In the previous problem, the differences may stay or not in the configuration. However, this does not say anything about the *shape* of the changed zone. For instance, it may be the case, as in the shift automaton, that the changed zone stay the same forever. Or, as when the automaton has a spreading state, for instance, that the entire configuration be invaded. This is exactly the question we ask in problem $\textsc{SInv}$: for a given word u, does the width of the zone changed between $F^t(p_u)$ and $F^t(p_u(x))$ extend infinitely as t grows ?

The formal definition of the "width" of a zone is as follows:

Definition 3.2.5 Let F be a cellular automaton over state set Q, and u a finite word of Q^*. We denote by $\delta_l(t)$ (respectively $\delta_r(t)$) the leftmost (respectively rightmost) difference between $F^t(p_u(x))$ and $F^t(p_u)$, that is:

$$\delta_l(t) = \min\Big\{i \mid F^t(p_u(x))[i] \neq F^t(p_u)[i]\Big\}$$

$$\delta_r(t) = \max\Big\{i \mid F^t(p_u(x))[i] \neq F^t(p_u)[i]\Big\}$$

Then:

$$\textsc{SInv}_{F,u}(x) = 1 \quad \text{if and only if} \quad \lim_{t\to\infty} |\delta_r - \delta_l| = \infty$$

Again, there is a class of cellular automata for which this problem is easy: the class of positively expansive cellular automata:

Proposition 3.2.3 Let F be a positively expansive cellular automaton over alphabet Q, then for any $u \in Q^*$, there is a protocol in $O(1)$ communicated bits for $\textsc{SInv}_{F,u}$.

Proof. According to a classical result by P. Kůrka in [47], there is a positive constant α such that $\delta_l(t) \leqslant -\alpha t$ and $\delta_r(t) \geqslant \alpha t$. Therefore, Alice and Bob only have to check that $p_u(x) \neq p_u$, which can be done with one bit of communication, since each player can check his own configuration separately. □

For equicontinuous cellular automata, this problem is not hard either:

Proposition 3.2.4 Let F be an equicontinuous cellular automaton. Then for all $u \in Q_F$, $D\left(\textsc{SInv}_F^u\right) \in O(1)$

Proof. This is almost the definition of equicontinuity. □

3.2.1.4 The cycle-length problem

In this last problem, we consider spatially periodic configurations. It is easy to see that the sequence c, $F(c)$, $F^2(c)$... becomes periodic after at most $|Q|^n$ steps, if Q is the alphabet of cellular automaton F, and n the length of the spatial period of configuration c. This problem is the problem of deciding, for a given uniform bound k on the period length, if the length of the period, on the input configuration, exceeds k. Formally:

Definition 3.2.6 Let F a cellular automaton, operating on alphabet Q, and c a periodic configuration, for F. The *length of the period* of F on c is:

$$\lambda(F, c) = \min\left\{p \mid \exists t_0, \forall t \geqslant t_0,\ F^{t+p}(c) = F^t(c)\right\}$$

Definition 3.2.7 Let F be a cellular automaton, and k an integer. Problem $\textsc{Cycle}_F^k$ is defined by:

$$\textsc{Cycle}_F^k(c) = 1 \quad \text{if and only if} \quad \lambda(F, c) \geqslant k$$

Again, there is a natural class of cellular automata for which this problem is simple, the *reversible* ones. Here is why:

Proposition 3.2.5 Let F be a reversible cellular automaton, and k an integer. Then there is a protocol for $\textsc{Cycle}_F^k$ in $O(1)$ communicated bits.

Proof. Since F is reversible, and ultimately periodic on configuration c, it is in fact periodic. To see this, let $p = \lambda(F, c)$, and t_0 the smallest integer such that $F^{t_0}(c) = F^{t_0+p}(c)$. Assume that $t_0 >$

0. Then, $F^{t_0-1}(c) \neq F^{t_0-1+p}$ by minimality of t_0, but $F(F^{t_0-1}(c)) = F(F^{t_0-1+p})$, which contradicts the fact that F is reversible.

Let us call r the radius of F. Thus, if the players send to each other the left kr cells, and the right kr cells of their respective configuration, they can compute the evolution of the automaton for k steps, and check at each step if they have already seen the configuration. □

3.2.2 Necessary conditions for universality

Now, our method is the following: for each problem $\mathfrak{P}$ that we have defined in section 3.2.1, we are going to show that if $\mathcal{A} \preccurlyeq \mathcal{B}$, then $cc(\mathfrak{P}_{\mathcal{A}}) \prec cc(\mathfrak{P}_{\mathcal{B}})$ (with $\prec$ the relation defined in Definition 2.2.2). Then, constructing a cellular automaton hard for problem $\mathfrak{P}$ (that is, with its communication complexity in $\Omega(n)$) will be enough to show that for a cellular automaton $\mathcal{A}$ to be universal, it must verify $cc(\mathfrak{P}_{\mathcal{A}}) \notin o(n)$. This is because if some function $f : \mathbb{Z} \to \mathbb{Z}$ is in $\Omega(n)$, and $g : \mathbb{Z} \to \mathbb{Z}$ is such that $f \prec g$, then $g \notin o(n)$.

3.2.2.1 Compatibility with simulations

For problem $\textsc{Pred}$, we have the following:

Proposition 3.2.6 If $F \preccurlyeq G$, then $cc(\textsc{Pred}_F) \prec cc(\textsc{Pred}_G)$.

Proof. We consider each of the ingredients involved in relation $\preccurlyeq$:

- Sub-automaton: if $F \sqsubseteq G$, then each valid protocol to compute $\textsc{Pred}_G$ is also a valid protocol to compute $\textsc{Pred}_F$, because this relation only involves a local renaming of the states by Alice and Bob.
- Iteration of the rule: We have $cc(\textsc{Pred}_{F^t}) \in \theta(cc(\textsc{Pred}_F))$, since we can use a protocol $\mathcal{P}$ for $\textsc{Pred}_{F^t}$ to solve $\textsc{Pred}_F$. If F has radius r, then F^t is an automaton of radius rt. This means that on a configuration x, Alice and Bob can use $\mathcal{P}$, at most rt times, to predict the $F^{\lfloor |x|/(rt) \rfloor}(x)$, that has at most rt cells, and then output the answer by computing the remaining iterations on this result.

  The other direction is simpler, as Alice and Bob can simply apply the same protocol. However, F^t has radius rt; therefore, if the size of the input is not a multiple of t, Alice and Bob may compute too many iterations of the rule. The solution is to forget the $|x| \bmod (rt)$ rightmost states of the configuration, and then apply the protocol for $\textsc{Pred}_F$.
- Packing: Let F be a cellular automaton, and n be fixed. We can solve $\textsc{Pred}_{F^{\langle m,1,0\rangle}}$ using a protocol for $\textsc{Pred}_F$: indeed, what a protocol for $\textsc{Pred}_{F^{\langle m,1,0\rangle}}$ computes is the value of a block of m cells, thus applying m times a protocol for $\textsc{Pred}_F$ yields the result. Therefore:

$$cc(\textsc{Pred}_{F^{\langle m,1,0\rangle}})(n) \leqslant m\,cc(\textsc{Pred}_F)(n)$$

To see why the converse holds, we need to build a protocol for $\textsc{Pred}_F$ from a protocol for $\textsc{Pred}_{F^{\langle m,1,0\rangle}}$. The problem is that in an arbitrary splitting of the input for F, Alice and Bob may not know the central block of m cells. But at most m states need to be communicated, before applying the protocol for $\textsc{Pred}_{F^{\langle m,1,0\rangle}}$, for Alice and Bob to know this block.

$$cc(\textsc{Pred}_F)(n) \leqslant cc(\textsc{Pred}_{F^{\langle m,1,0\rangle}})(\lceil n/m \rceil) + m\log(|Q_F|) + O(1)$$

- $\textsc{Shift}$: This operation only affects the splitting of the input, and the radius of the rule. Since we consider the worst-case splitting in both F and G, this operation does not change the communication complexity of $\textsc{Pred}$: a protocol for size n for $F^{\langle 1,1,0\rangle}$ will work for size $n+2z$ for $F^{\langle 1,1,z\rangle}$. □

Now, for problem $\textsc{TInv}$, the same result also holds:

Proposition 3.2.7 If $F \preccurlyeq G$, and $u_F \in Q_F^*$, then there exists $u_G \in Q_G^*$ such that $cc\left(\textsc{TInv}_F^{u_F}\right) \prec cc\left(\textsc{TInv}_G^{u_G}\right)$.

Proof. First, thanks to the properties of simulation $\preccurlyeq$, if c is a periodic configuration of Q_F, then there is a corresponding periodic configuration of Q_G on which G simulates the behavior of F.

- $\textsc{Packings}$: By the same arguments as in proposition 3.2.6, for any m, t, z, then for word u_F, defined as the concatenation of m times word u_F, $\textsc{TInv}_{F^{\langle m,t,z\rangle}}^{u_F}$ if and only if $\textsc{TInv}_F^{u_F}$. Hence, the two problems have the same communication complexity, up to a constant term to compute the packed input from the unpacked one (as in the proof of proposition 3.2.6).
- $\textsc{Sub-automaton}$: If $F \sqsubseteq G$, if we have a protocol for any $u \in Q_G$, we can use it to decide for $\textsc{Inv}_F^v$, for all $v \in Q_G$. Here is how: in the first step of the protocol, Alice and Bob both convert their respective configuration into a configuration of Q_G. This conversion gives a unique result, and requires no communication. Then, they use the protocol for G to decide the result. Since relation $\preccurlyeq$ corresponds to a one-to-one map from configurations of Q_G to the configurations of Q_G, the invasion happens in F if and only if it happens in the *corresponding* configuration of G, so this protocol computes the correct result. □

We can again prove the same result for $\textsc{SInv}$:

Proposition 3.2.8 If $F \preccurlyeq G$, and $u_F \in Q_F^*$, then there exists $u_G \in Q_G^*$ such that $cc\left(\textsc{SInv}_F^{u_F}\right) \prec cc\left(\textsc{SInv}_G^{u_G}\right)$.

Proof. For the same reasons as in proposition 3.2.7, if c is a periodic configuration of Q_F, then the corresponding configuration of Q_G for $\sqsubseteq$ is also periodic, so that

- $\textsc{Packings}$: By the same arguments as in proposition 3.2.7, for any $u \in Q_F$, $\textsc{SInv}_{F^{\langle m,t,z\rangle}}^{u_F}$ if and only if $\textsc{SInv}_F^{u_F}$, and therefore, $cc\left(\textsc{SInv}_{F^{\langle m,t,z\rangle}}^{u_F}\right) = cc\left(\textsc{SInv}_F^{u_F}\right)$.

- Sub-automaton: By the exact same argument as in proposition 3.2.7, since if $F \sqsubseteq G$, the sub-automaton relation induces a one-to-one map φ from configurations for F to configurations for G. Thus, the two functions δ_l and δ_r are exactly the same in F and in the restriction of G to the configurations of $\varphi(\mathbb{Z}^{Q_F})$, and the method we used in proposition 3.2.7 to get a protocol for $\textsc{SInv}_F^u$ from a protocol for $\textsc{SInv}_G^v$ is still usable here. □

Finally, for our problem Cycle, this result also holds:

Proposition 3.2.9 If $F \preccurlyeq G$ then $cc\left(\textsc{Cycle}_F^k\right) \prec cc\left(\textsc{Cycle}_G^{k'}\right)$.

Proof. The effect of rescaling transformations on cyclic orbits of periodic configurations is to change the spatial period length, as well as the temporal cycle length. More precisely:

- If $F \sqsubseteq G$ then for any k, $cc\left(\textsc{Cycle}_F^k\right) \prec cc\left(\textsc{Cycle}_G^k\right)$.
- For any k,
$$cc\left(\textsc{Cycle}_F^k\right) \prec cc\left(\textsc{Cycle}_{F^{\langle m,1,0\rangle}}^k\right) \quad \text{and} \quad cc\left(\textsc{Cycle}_{F^{\langle m,1,0\rangle}}^k\right) \prec cc\left(\textsc{Cycle}_F^k\right)$$
- For any t and any k:
$$cc\left(\textsc{Cycle}_{F^{\langle 1,t,0\rangle}}^k\right) \prec cc\left(\textsc{Cycle}_F^{kt}\right)$$
- For any t and any k such that $k \bmod t = 0$:
$$cc\left(\textsc{Cycle}_F^k\right) \prec cc\left(\textsc{Cycle}_{F^{\langle 1,t,0\rangle}}^{k/t}\right)$$

Altogether, this proves the proposition □

3.2.2.2 Hardness and orthogonality of the basic problems

A natural question that can be asked about these definitions is why we need more than one problem. As noted in section 3.2.1, each problem makes a different class of cellular automata easy. To formalize this notion, we need to prove, for each couple of basic problems, the existence of a cellular automaton hard for one problem, and easy for the other one. “Hard” here means with maximal communication complexity, while “easy” means in $O(1)$.

We first construct a cellular automaton Φ easy for Pred, (that is, with its communication complexity in $O(1)$), and hard for all the other problems (that is, in $\Omega(n)$).

The idea of the construction is to embed an equality test (more precisely, a palindrome test) launching signals that invade the configuration if the test fails, while keeping the Pred problem easy. Deciding if a word is a palindrome is the same as deciding if the two players have the same configuration, which, according to the example we gave for definition 3.1.2, requires $\Omega(n)$ bits of communication.

To do this, we will use two components, one with signals moving quickly out of the way, and the other one with tests, launching the signals on the other component while staying unaltered:

1. The first layer has alphabet $\Gamma_1 = \left\{\overrightarrow{0}, \overrightarrow{1}, \overleftarrow{0}, \overleftarrow{1}, \top, \emptyset_1, K_1\right\}$. The $\overleftarrow{x}$ shift to the left, and the $\overrightarrow{x}$ to the right. The K_1 state is spreading, and if there is no K_1 in its neighborhood, the $\top$ state stays unaltered.
2. The second layer is an automaton on alphabet $\Gamma_2 = \left\{s, \emptyset_2, \rightarrow, \leftarrow, K_2\right\}$, where the $\rightarrow$ and $\leftarrow$ shift to the right and to the left, respectively, K_2 is spreading, and $\emptyset_2$ is quiescent. Also, state s represents a "seed" for the signals, meaning that when it appears, it disappears on the next step, launching a $\leftarrow$ on its left, and a $\rightarrow$ on its right. Moreover, $\rightarrow$ and $\leftarrow$ signals can cross: if a cell has a $\rightarrow$ on its left, and a $\leftarrow$ on its right, it becomes an s.

We need to add a few rules to make sure that degenerate configurations are simple for both problems, and this is the purpose of states K_1 and K_2: if one of them appears on a layer, then it makes the other one appear, and they propagate on the whole configuration, erasing everything. This happens:

- When a $\overrightarrow{x}$ state appears on the left of a $\overleftarrow{x}$, on the first component.
- Or when a $\overleftarrow{x}$ appears in the same cell as a $\leftarrow$ (or a $\overrightarrow{x}$ in the same cell as a $\rightarrow$). This ensures that signals $\rightarrow$ and $\leftarrow$, on the second component, can never cross.

Finally, in order to perform an equality test, we add the rule that if a $\top$ state has a $\overrightarrow{a}$ on its left, a $\overleftarrow{b}$ on its right, where $a \neq b$, then an s appears on the next step on the second layer.

Proposition 3.2.10 Automaton Φ is such that:

1. $cc(\textsc{Pred}_\Phi) \in O(1)$.
2. there is a word $u \in Q_\Phi$ such that $cc\left(\textsc{SInv}^u_\Phi\right) \in \Omega(n)$ and $cc\left(\textsc{TInv}^u_\Phi\right) \in \Omega(n)$.
3. For all integers k, $cc\left(\textsc{Cycle}^k_\Phi\right) \in \Omega(n)$.

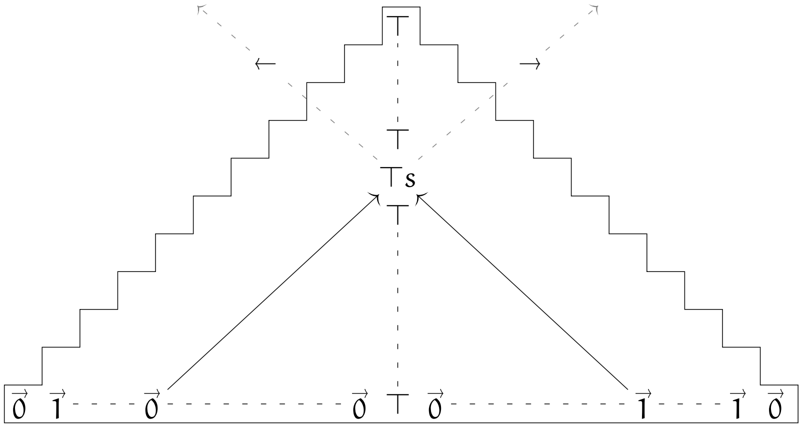

$\leftarrow$
$\rightarrow$
$\top$
$\top$
$\top s$
$\top$
$\overrightarrow{0}$ $\overrightarrow{1}$ $\overrightarrow{0}$ $\overrightarrow{0}$ $\top$ $\overrightarrow{0}$ $\overrightarrow{1}$ $\overrightarrow{1}$ $\overrightarrow{0}$

Figure 3.2 - Rule Φ

Proof.

1. A protocol for $\textsc{Pred}_\Phi$ needs to predict the value of both layers. First, if the configuration is not well-formed, that is, a left part with only $\overrightarrow{x}$ states, and a right part with only $\overleftarrow{x}$ states,

separated by a $\top$ if both parts are not empty, then a K_1 and a K_2 state will appear, and thus the result will be state (K_1, K_2). This can be checked locally by Alice and Bob, and transmitted within a constant number of communicated bits; indeed, these configuration is easily described by a regular expression, and Lemma 3.1.6. Else:

  - On the first layer, the result will always be the result of a shift, or, in the case where the central cell is a $\top$, also a $\top$. This can be easily checked within constant communication.
  - On the second layer, there are four (possibly overlapping) possibilities:
    - If the leftmost state of Alice's differs from the rightmost state of Bob's, and the central cell is a $\top$ state, then the result is $\top$ on the first component, and an s on the second one.
    - If the central cell is not a $\top$, but there is a $\top$ somewhere else in the configuration, and the corresponding word is not a palindrome, the top state is a $\rightarrow$.
    - If the initial configuration contains an s or a $\rightarrow$ in its leftmost cell, and an s or a $\leftarrow$ in its rightmost cell, an s is generated at the top of the triangle. If only one side has an s or an arrow, then an arrow arrives at the top.
    - In all other cases, the result is a $\emptyset_2$.

  All of these cases can be checked by Alice and Bob using only local information, thus the amount of communication needed is constant.

2. To show that Φ is hard for $\textsc{SInv}$ and $\textsc{TInv}$, it is enough to find a word u and a fooling set for $\textsc{SInv}_\Phi^u$ and $\textsc{TInv}_\Phi^u$ of size S with $\log|S| \in \Omega(n)$:

$$S = \left\{\left(\overrightarrow{x_1}, \emptyset_2\right)\ldots\left(\overrightarrow{x_n}, \emptyset_2\right)\top\left(\overleftarrow{x_n}, \emptyset_2\right)\ldots\left(\overleftarrow{x_1}, \emptyset_2\right) \mid \forall i,\ x_i \in \{0,1\}\right\}$$

3. If the initial configuration is incorrect, then the period is 1. Else, if the configuration has no $\top$ state, then the dynamics is either a shift (if there are signals), or the identity. In all these cases, this condition can be checked within $O(1)$ bits of communication.

   Finally, if the configuration is well-formed, and there is a $\top$ state, the same fooling set as in case 2 can be used. □

Now, we are going to construct a cellular automaton X easy for $\textsc{Cycle}$, and hard for $\textsc{TInv}$ and $\textsc{SInv}$:

Proposition 3.2.11 There is a cellular automaton X, and a word u such that:

1. $\mathrm{cc}\left(\textsc{TInv}_X^u\right) \in \Omega(n)$, $\mathrm{cc}\left(\textsc{SInv}_X^u\right) \in \Omega(n)$ and $\mathrm{cc}(\textsc{Pred}_X) \in \Omega(n)$.
2. For all k, $\mathrm{cc}\left(\textsc{Cycle}_X^k\right) \in O(1)$.

Proof. The idea is to reuse the construction of Φ, by simply adding the following two rules:

- On the second component, when two signals should cross, they disappear instead. Formally, when a cell sees two signals in colliding directions in its neighborhood, it becomes a $\emptyset_2$ state. Thus, no signals in opposite directions can coexist for more than n steps, and all configurations that are not shifts are of period 1. This can still be checked, by the same argument, in $O(1)$ bits of communication.

- On the first component, we add a state $\perp$, such that when a test fails, it sends signals on the second component, but also transforms $\top$ into $\perp$. These $\perp$ states have the same behavior as $\top$s, except that they do not launch signals on test failures, and they can never return to state $\top$. This makes the Pred problem also difficult, since it amounts to the problem of deciding whether a test has ever failed. □

Now that we know how to make cellular automata that have their Pred and Cycle problems hard, we can easily trivialize their SInv and TInv problems:

Proposition 3.2.12 There is a cellular automaton Ψ, and a word u such that:

1. for all k, $\mathrm{cc}\left(\mathrm{Cycle}^{k}_{\Psi}\right) \in \Omega(n)$.
2. $\mathrm{cc}(\mathrm{Pred}_{\Psi}) \in \Omega(n)$.
3. $\mathrm{cc}\left(\mathrm{SInv}^{u}_{\Psi}\right) \in O(1)$.
4. $\mathrm{cc}\left(\mathrm{TInv}^{u}_{\Psi}\right) \in O(1)$.

Proof. We need to make two modifications from Φ:

- The first one is to add two states to the second component, $\rightharpoonup$ and $\leftharpoonup$, that are launched each times the equality test perform by $\top$ succeeds (that is, when state s does not appear). This makes SInv and TInv trivial, while keeping the Cycle^{k} problem hard for all k.
- The other modification is the same as in proposition 3.2.11. We add a state $\perp$ to the first component, that launches only $\rightharpoonup$ and $\leftharpoonup$ signals, whatever the result of the test.

This makes the SInv and TInv problems trivial, because they now amount to decide whether the configuration has a $\top$ or $\perp$ state initially, and whether they will ever launch spreading states. The other two problems are shown to be in $\Omega(n)$ with the same fooling set as in proposition 3.2.10. □

The last two automata we need to complete our proof of mutual "orthogonality" are examples for SInv and TInv:

Proposition 3.2.13 There is a cellular automaton Υ such that:

1. $\exists u,\ \mathrm{SInv}^{u}_{\Upsilon} \in \Omega(n)$
2. $\forall u,\ \mathrm{TInv}^{u}_{\Upsilon} \in O(1)$

Proof. We use again essentially the same construction as for Ψ, in the proof of proposition 3.2.12, but this time, instead of launching signals $\rightharpoonup$ and $\leftharpoonup$, we launch only one type of signals, but on the left side only when the test succeeds, and on both sides else. With the same argument as in the proof of that proposition, it easy to see that this automaton has communication complexity in $O(1)$ for TInv, because even if no spreading state is raised, either the automaton is a shift, or there is a $\top$ state, launching signals at each step.

The same fooling set as for the proof of proposition 3.2.10 can be used once more here. On these configurations, signals are launched on both sides if and only if at least one test fails. Else, the configuration is never invaded. □

Proposition 3.2.14 There is a cellular automaton Υ' such that:

1. $\exists u$, $\text{TInv}^{u}_{\Upsilon'} \in \Omega(n)$
2. $\forall u$, $\text{SInv}^{u}_{\Upsilon'} \in O(1)$

Proof. We only need one component, similar to the first component of Φ, with an alphabet $\Gamma = \left\{\overrightarrow{0}, \overrightarrow{1}, \overleftarrow{0}, \overleftarrow{1}, \top, \bot, \emptyset\right\}$. The $\overleftarrow{x}$ shift to the left, and the $\overrightarrow{x}$ to the right. The idea is that $\emptyset$ is almost spreading, i.e. it spreads to any other state, except $\top$ and $\bot$.

This way, no invasion can ever occur. A fooling set for this problem, with word $u = \emptyset$, is for instance $\left\{\overrightarrow{x_1} \dots \overrightarrow{x_n} \top \overleftarrow{x_n} \dots \overleftarrow{x_1}\right\}$. □

3.2.2.3 Necessary conditions for universality

As a consequence of all the propositions of section 3.2.2.2, we get the following theorem:

Theorem 3.2.15 Let F be an intrinsically universal cellular automaton. Then:

$$cc(\text{Pred}_F) \notin o(n)$$

$$\forall k,\ cc\left(\text{Cycle}^{k}_{F}\right) \notin o(n)$$

$$\exists\, u \in Q_F^*,\ cc\left(\text{SInv}^{u}_{F}\right) \notin o(n)$$

$$\exists\, u \in Q_F^*,\ cc\left(\text{TInv}^{u}_{F}\right) \notin o(n)$$

And none of these conditions is implied by any logical combination of the other ones.

Proof. This is because of the properties of relation $\prec$: if $f \prec g$, and $f \in \Omega(n)$, then $g \notin o(f)$. □

Choosing communication problems to study intrinsic universality is a hard question. The first problem to be studied under this approach was the Pred problem, in [14]. The approach was later generalized by [36] and [48]. Another attempt was made in [49], but quite unrelated to simulations, and thus to intrinsic universality. As noted in section 3.2.2.2, another concern, in the choice of communication problems, is the utility of the new problem as a tool to prove non-universality. This is why we proved results of what we called "orthogonality", in order to verify that our four problems are really useful to our approach.

Surprisingly though, the logical combination of two orthogonal problems $\mathcal{P}$ and $\mathcal{Q}$ can sometimes make up a new problem $\mathcal{R}$, orthogonal to $\mathcal{P}$ and $\mathcal{Q}$. We first proved this result, in [48] with Raimundo Briceño, and we restate this result here. The example problem is the following one:

Definition 3.2.8 Let F be a cellular automaton, and u a word over its alphabet. The problem $\mathrm{CI{\scriptstyle NV}}_F^u$ is defined by:

$$\mathrm{CI{\scriptstyle NV}}_F^u = \mathrm{TI{\scriptstyle NV}}_F^u \wedge \neg\ \mathrm{SI{\scriptstyle NV}}_F^u$$

Proposition 3.2.16 There is a cellular automaton F, and a word u over the alphabet of F, such that:

$$cc\left(\mathrm{CI{\scriptstyle NV}}_F^u\right) \in \Omega(n)$$

Proof. This is a simple corollary of proposition 3.2.13. Indeed, if $\mathrm{CI{\scriptstyle NV}}_F^u$ was simple (not in $\Omega(n)$) for all automata and all u, we could use a protocol not in $\Omega(n)$ for $\mathrm{TI{\scriptstyle NV}}_F^u$ to construct a protocol not in $\Omega(n)$ for $\mathrm{SI{\scriptstyle NV}}_F^u$, for all F and all u, and thus in particular for automaton Υ described in proposition 3.2.13. This is because $\mathrm{SI{\scriptstyle NV}}_F^u = \mathrm{TI{\scriptstyle NV}}_F^u \wedge \neg\ \mathrm{CI{\scriptstyle NV}}_F^u$. □

This method of proof can be further applied to prove, not constructively, that the three problems SInv, TInv and CInv must be orthogonal, since any problem is a logical combination of the other two:

Proposition 3.2.17 For any two problems $\mathcal{P}$ and $\mathcal{Q}$ among {SInv,TInv,CInv}, there is a cellular automaton F such that:

- $\exists u,\ cc\left(\mathcal{P}_F^u\right) \in \Omega(n)$
- $\forall u,\ cc\left(\mathcal{Q}_F^u\right) \in O(1)$

This definition also gives an interesting connection with a famous open problem called the *direct sum* problem in [43]. In this problem, the players are requested to solve two unrelated communication problems on two distinct inputs. That is, Alice receives two inputs x_f and x_g, Bob receives y_f and y_g, and they need to compute (f,g). The question is whether it would be possible to find *unrelated* functions f and g for which a protocol for (f,g) would use less communication than the sum of the complexities of the best protocols for f and for g.

Open problem 3.2.1 Let $f : X{\times}Y \to Z$ and $g : X'{\times}Y' \to Z'$ be two independent functions. Also, let $h : (X{\times}X'){\times}(Y{\times}Y') \to (Z{\times}Z')$ be the cartesian product of f and g, that is, $h((x,x'),(y,y')) = (f(x,x'),g(y,y'))$.

$$\text{Is it possible that } \; D(h) < D(f)+D(g) \; ?$$

3.3 Examples

In this section, we show examples constructions of the use of communication complexity with cellular automata.

3.3.1 An example of non-trivial input splitting

Finally, let us conclude the presentation of this problem with the example of an input splitting problem with two maxima:

Proposition 3.3.1 There is a cellular automaton F and two indexes $i < j$ such that $cc(\textsc{Pred}_F) = D(\textsc{Pred}_F \circ \mathcal{C}_i) = D(\textsc{Pred}_F \circ \mathcal{C}_j)$, and for any $i < k < j$, $D(\textsc{Pred}_F \circ \mathcal{C}_k) < cc(\textsc{Pred}_F)$

Proof. The first cellular automaton we build achieves only one maximum, but it demonstrates the idea of the "guidance" system that we use in the final construction. Let $Q = \{a,b,\top_a,\top_b,\bot,1,2,K\}$, and F the automaton with alphabet Q, defined by:

$$F(1,2,x) = x, F(2,x,1) = 1, F(x,1,2) = 2$$

$$F(x,y,z) = x \;\text{ when }\; y \in \{a,b\} \;\text{ and }\; z \in \{a,b\}$$

$$F(x,T,y) = T_x, F(T_x,y,x) = T$$

$$F(x,y,z) = K \;\text{ otherwise}$$

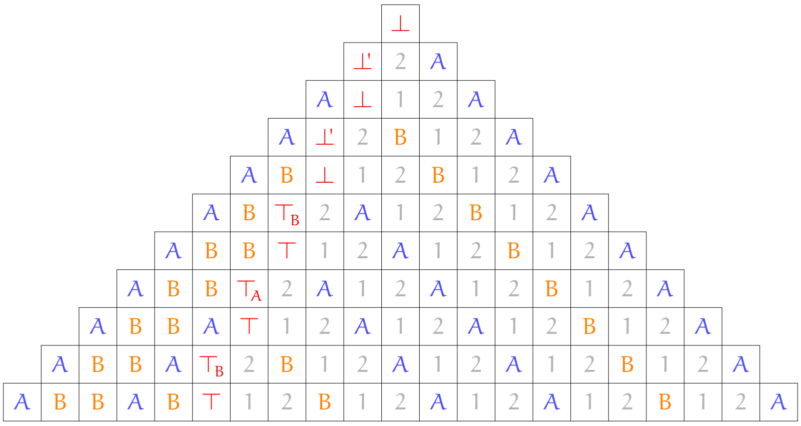

⊥
⊥' 2 A
A ⊥ 1 2 A
A ⊥' 2 B 1 2 A
A B ⊥ 1 2 B 1 2 A
A B $\top_B$ 2 A 1 2 B 1 2 A
A B B ⊤ 1 2 A 1 2 B 1 2 A
A B B $\top_A$ 2 A 1 2 A 1 2 B 1 2 A
A B B A ⊤ 1 2 A 1 2 A 1 2 B 1 2 A
A B B A $\top_B$ 2 B 1 2 A 1 2 A 1 2 B 1 2 A
A B B A B ⊤ 1 2 B 1 2 A 1 2 A 1 2 B 1 2 A

Figure 3.3

Now, let us build a symmetric version of this automaton. The easiest way to do this would be to consider a cartesian product of F an a "reverted" version F. However, this would not prove anything, as the communication complexity would stay high between the two maxima. Our construction must thus ensure that we can only solve one of the two versions of the problem: either the one with the ⊤s shifting rightwise, or leftwise. To achieve this, we use more or less the same guidance system as before, although with two more states to make it reversible. All the transitions "types" (modulo A/B symmetry) not yielding K appear on figures 3.4 and 3.5.

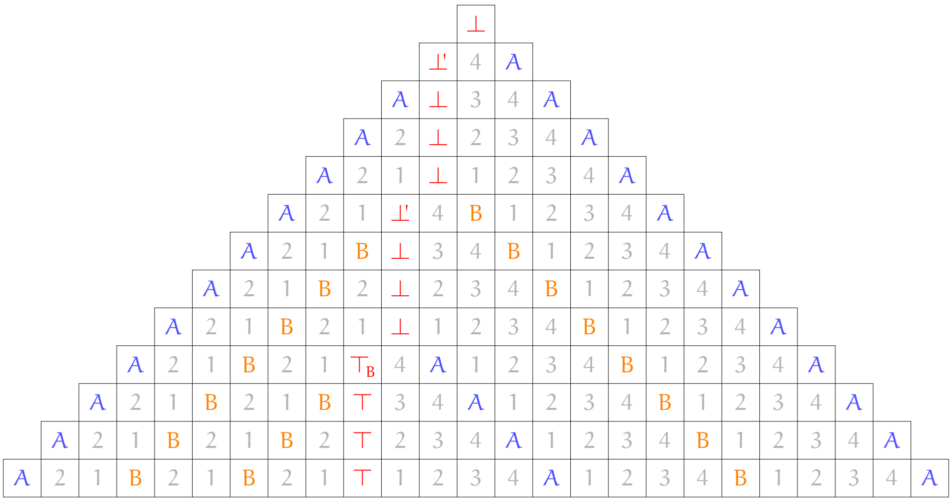

⊥
⊥' 4 A
A ⊥ 3 4 A
A 2 ⊥ 2 3 4 A
A 2 1 ⊥ 1 2 3 4 A
A 2 1 ⊥' 4 B 1 2 3 4 A
A 2 1 B ⊥ 3 4 B 1 2 3 4 A
A 2 1 B 2 ⊥ 2 3 4 B 1 2 3 4 A
A 2 1 B 2 1 ⊥ 1 2 3 4 B 1 2 3 4 A
A 2 1 B 2 1 $\top_B$ 4 A 1 2 3 4 B 1 2 3 4 A
A 2 1 B 2 1 B ⊤ 3 4 A 1 2 3 4 B 1 2 3 4 A
A 2 1 B 2 1 B 2 ⊤ 2 3 4 A 1 2 3 4 B 1 2 3 4 A
A 2 1 B 2 1 B 2 1 ⊤ 1 2 3 4 A 1 2 3 4 B 1 2 3 4 A

Figure 3.4

Now, the interesting part of this rule is that, contrarily to the one on figure 3.3, we can make it symmetric. Moreover, the two sets of configurations, those with the ⊤s shifting to the right, and those with the ⊤s shifting to the left, and where K never appears, are disjoint. If we try to make the two sides progress at the same speed, then a K appears, erasing all the configuration. The analysis of the communication complexity is simple: if no K appears, then there must be at most one of $\{\top, \top_A, \top_B, \bot\}$ (let us call them the *test states* from now on), at any step. If it is not at position $3(n-1)/8$ or $5(n-1)/8$, for a configuration of size n, then the dynamics is essentially a shift, and either Alice or Bob can predict it with no more communications. Else, the problem of predicting whether the final state will be in $\{\top, \top_A, \top_B\}$, or in $\{\bot, \bot'\}$ amounts to computing an instance of Eq on alphabet $\{A, B\}$, and has thus communication complexity $\Omega(n)$.

Since we can describe all the sets of configurations by regular expressions, knowing in which case we are can be done in $O(1)$ bits of communication by Lemma 3.1.6. □

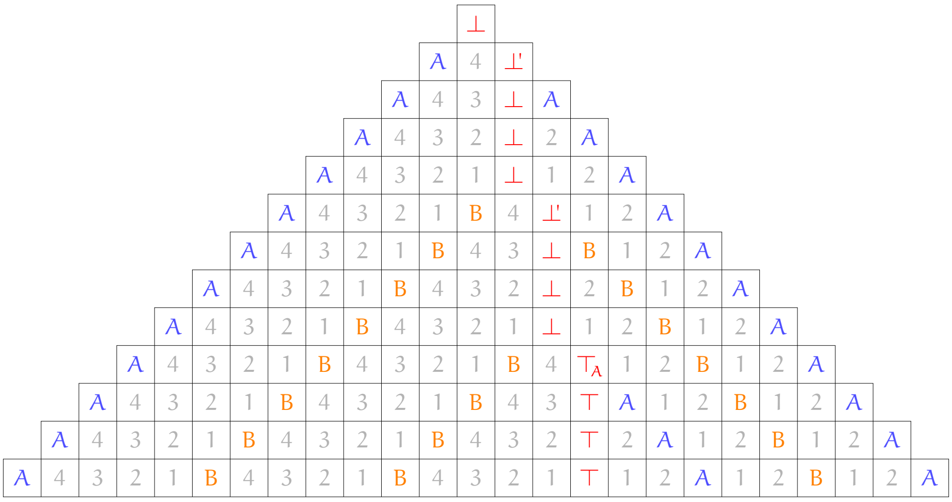

⊥
A 4 ⊥'
A 4 3 ⊥ A
A 4 3 2 ⊥ 2 A
A 4 3 2 1 ⊥ 1 2 A
A 4 3 2 1 B 4 ⊥' 1 2 A
A 4 3 2 1 B 4 3 ⊥ B 1 2 A
A 4 3 2 1 B 4 3 2 ⊥ 2 B 1 2 A
A 4 3 2 1 B 4 3 2 1 ⊥ 1 2 B 1 2 A
A 4 3 2 1 B 4 3 2 1 B 4 $\top_A$ 1 2 B 1 2 A
A 4 3 2 1 B 4 3 2 1 B 4 3 ⊤ A 1 2 B 1 2 A
A 4 3 2 1 B 4 3 2 1 B 4 3 2 ⊤ 2 A 1 2 B 1 2 A
A 4 3 2 1 B 4 3 2 1 B 4 3 2 1 ⊤ 1 2 A 1 2 B 1 2 A

Figure 3.5

3.4 The classical complexity of our problems

We now show how our approach may allow for a finer analysis of cellular automata than classical complexity, by constructing example automata with a quite low communication complexity (and therefore non-universal), but a maximal classical complexity in their respective classes.

3.4.1 Pred is p-complete

It is a well-known result that any intrinsically universal cellular automaton is p-complete for Pred. In this section, we prove a somewhat stronger result, by constructing a cellular automaton that, while remaining p-complete, has a communication complexity in $O(n^{1/k})$, with k arbitrarily large.

Proposition 3.4.1 Let k be an integer, then there is a cellular automaton F such that the language of configurations x on which $\textsc{Pred}_F(x) = 1$ is P-complete, and $cc(\textsc{Pred}_F) \in O(n^{1/k})$.

Proof. Let $\mathcal{M}$ be a Turing machine. We construct a cellular automaton F simulating $\mathcal{M}$ slowly but still in polynomial time: it takes n^k steps of F to simulate n steps of $\mathcal{M}$. Choosing a P-complete $\mathcal{M}$ in the first place yields the P-completeness of $\textsc{Pred}_F$.

First, it is easy to construct a cellular automaton simulationg $\mathcal{M}$ in real time. We encode each symbol of the tape alphabet of the Turing machine by a state of the cellular automaton, and add a “layer” for the head, with alphabet $\{\rightarrow, \leftarrow\} \cup Q_{\mathcal{M}}$, if $Q_{\mathcal{M}}$ was the set of states of the Turing machine. The rules are easy to infer from the transitions of $\mathcal{M}$, with the addition that we need to add a spreading state, erasing all the configuration whenever a $\rightarrow$ is found next to a $\leftarrow$, without a head cell between them.

We then add a new layer to slow down the simulation: it consists of a single particle, moving left and right inside a marked region of the configuration. More precisely, it goes right until it reaches the end of the marked region, then adds a marked cell at the end, starts to move left to reach the other end, adds a marked cell, goes right again, etc. Also, to ensure that there is always one particle, we use the same construction with states $\rightarrow$ and $\leftarrow$. Clearly, for any cell in finite marked region, seeing n traversals of the particle takes $\Omega(n^2)$ steps. Then, the idea is to authorize the Turing head to move only when it sees the particle in its neighborhood. This ensures that n steps of the Turing machine take n^2 steps of the cellular automaton to simulate.

We can iterate this construction by adding another layer with a second particle controlling the moves of the first one, etc. Finally, after adding k particles, the automaton simulates $\mathcal{M}$, slowed down by a factor n^k. We can thus give a protocol in $O(n^{1/k})$ bits for $\textsc{Pred}_F$:

- Either the configuration is improperly encoded (i.e. there is a $\rightarrow$ next to a $\leftarrow$ on any of the layers, or several heads), and then the automaton will generate a spreading state at some point; Alice and Bob can simply answer that, and this is always easy for them to detect, no matter how the splitting of the input is.
- Or there is a layer without particle, or no initial “marked zone”, then nothing can happen. Alice and Bob know the answer in constant time.
- Or the configuration is well-formed, and then the result of $\textsc{Pred}_F$ only depends on $2n^{1/k}$ cells around the central cell. Alice can thus simply communicate these cells to Bob, who outputs the answer. □

This proposition poses a natural open problem: how simple can a P-complete cellular automaton be ? Is the bound of proposition 3.4.1 tight ?

Open problem 3.4.1 Is there a P-complete cellular automaton F with $cc(\textsc{Pred}_F) \in o(n^{1/k})$ for any integer k ? and in $O(\log n)$?

3.4.2 The invasion problems are Π_1^0-complete

Proposition 3.4.2

1. For any cellular automaton F and any word u, $\textsc{SInv}_F^u \in \Pi_1^0$.
2. There is a cellular automaton F and a word u such that $\textsc{SInv}_F^u$ is Π_1^0 complete, and yet $cc\left(\textsc{SInv}_F^u\right) \in O(\log n)$

Proof.

1. Let F and u as in the proposition, and consider problem $\textsc{SInv}_F^u$. For an input $x_1...x_n$, we use the notations $\delta_l(t)$ and $\delta_r(t)$ for the leftmost and rightmost differences at time t between the orbit of p_u and the orbit of $p_u(x_1...x_n)$ (see definition 3.2.5).

   We need to prove the following lemma:

   Lemma 3.4.3 There is a recursive function β such that for any n, any input $x_1...x_n$ and any $\Delta \geqslant 0$:

   $$\exists t,\ \delta_r(t) - \delta_l(t) \geqslant \Delta \Leftrightarrow \exists t,\ t \leqslant \beta(\Delta) \wedge \delta_r(t) - \delta_l(t) \geqslant \Delta$$

   Proof. First, the orbit of p_u is ultimately periodic: there are t_0 and p_0 such that for any $t \geqslant t_0$, it holds that $F^t(p_u) = F^{t+p_0}(p_u)$. Given an input $x_1...x_n$ for the problem, let us write $w(t)$ the word of length $\delta_r(t) - \delta_l(t)$ starting at position $\delta_l(t)$ in configuration $F^t(p_u(x_1...x_n))$. Assume that $|w(t)|$ is bounded, then it is ultimately periodic; let us call its period p_1. But now, $\delta_l(t) \bmod |u|$ is also ultimately periodic, since the orbit of p_u is, and $|w(t)|$ is bounded. If we call its period p_2, then the system is ultimately $\mathrm{ppcm}(p_0, p_1, p_2)$-periodic.

   But if we assume $\forall t,\ |w(t)| \leqslant \Delta$, simulating the evolution for $|Q|^{|w(t)|}|u|p_0$ steps is enough to pass through all the possible states of the system before returning to a state already seen. □

   Now, by this lemma, the predicate $\exists\ t \leqslant \beta(\Delta),\ \delta_r(t) - \delta_l(t) \geqslant \Delta$ is recursive, and thus this point follows from the following characterization of $\textsc{SInv}$:

   $$\textsc{SInv}_F^u = \forall \Delta \geqslant 0,\ \exists\ t \leqslant \beta(\Delta),\ \delta_r(t) - \delta_l(t) \geqslant \Delta$$

2. It is sufficient to simulate a two-counters machine, according to a classical result by Minsky [50]. Here is how to do it: our alphabet is $\{A, M, B, 0, \emptyset\} \times (Q \cup \{\rightarrow, \leftarrow, \emptyset, \kappa\})$, where κ is a spreading state and $\emptyset$ is quiescent. We say that a portion of the configuration is *valid* when it is of the form $0^*A^+MB^+0^*$ on the first layer of the cartesian product, and $\rightarrow^+ q \leftarrow^+$ on the second one.

   The dynamics of the automaton is as follows:

   - Whenever the configuration is invalid, which can clearly be checked within $O(1)$ bits of communication (thanks to Lemma 3.1.6), a κ state is generated and spreads.

- Else, on each coding segment, the (necessarily unique) head goes repeatedly from one end of the segment to the other end, and extends the segment at each pass by adding a $\rightarrow$ on its left, a $\leftarrow$ on its right, and a 0 on the data layer. If the extension step is blocked by another segment, then the state κ is generated and spreads.
- Moreover, at each pass on the segment, the head executes one of the basic instruction of the underlying two-counters machine:
  - Testing if a counter is empty, which, in can be done by checking if there is a 0 on the right, or on the left of the M state.
  - Decrementing can be done by replacing the leftmost A, or the rightmost B, by a 0.
  - Incrementing can be done by replacing a 0 by A on the left of the leftmost A, or a 0 by B on the right of the rightmost B.
  - The head can also stop.

Moreover, if any behavior of the head leads to an incoherence (decrementing an empty counter, writing a B instead of an A...), state κ appears and erases all the configuration.

With this definition, and if $u = \emptyset$, the halting problem for the two-counters machine encoded in F reduces to $\textsc{SInv}_F^u$:

$$\textsc{SInv}_F^u(x) = 0 \Leftrightarrow \mathcal{M} \text{ loops or halts}$$

Therefore, by a suitable choice of the machine $\mathcal{M}$ simulated by F, we can make $\textsc{SInv}_F^u$ Π_1^0-complete.

To conclude the proof, we show that $cc\left(\textsc{SInv}_F^u\right) \in O(\log n)$. For a given input w, split between Alice and Bob, the following protocol runs in $O(\log n)$ bits and determines $\textsc{SInv}_F^u$:

- First, Alice and Bob can check locally whether the configuration is correct. If it is, then they know there is only one head. Then Alice can send the state of the counters of her rightmost segment encoding a machine, along with the number of $\emptyset$s between this segment and Bob's part. If this part collides with Bob's leftmost simulated machine, then the configuration is invaded by κs.y This can clearly be done in $O(\log n)$ bits of communication.
- If there are several heads, and they do not halt, then the heads extending the $\rightarrow$ and $\leftarrow$ on the second layer will collide, generating a κ, and thus invading the configuration.
- If the configuration is invalid, or if one of the machines does not loop or halt for itself, then the result is known in $O(1)$ bits. □

We can use the same construction to prove that there is an F and a u for which $\textsc{TInv}_F^u$ is Π_1^0-complete, while remaining of quite low communication complexity:

Proposition 3.4.4

1. For any cellular automaton F and any word u, $\textsc{TInv}_F^u \in \Pi_1^0$.

2. There is a cellular automaton F and a word u such that TInv_F^u is Π_1^0-complete, and yet $cc\left(\mathrm{TInv}_F^u\right) \in O(\log n)$

Proof. This proof is relatively simpler than the proof of proposition 3.4.2.

1. First, for any cellular automaton F of radius r, and any word u, there is a simple Π_1^0 formula corresponding to this problem:

   $$\forall t,\ \exists\ i \leqslant rt,\ F^t(p_u(x))[i] \neq F^t(p_u)[i]$$

2. Now, we can use almost the same construction, also with $u = \emptyset$ with an additional "cleaner" state, launched by the head upon halting, and replacing all the As and Bs, on both sides of the head, by $\emptyset$ on both components, then stopping at the end of the parts of the configuration used by the simulation of the machine.

   This way, on any u not completely blank, if the input contains a head, a κ state will appear, invading the configuration. If the input has no head, the dynamics is trivial. Now, if all the two-counters machines simulated by the input halt, the configuration will become all blank again after they have all halted. Else, it will continue to run forever. □

3.4.3 Cycle is PSPACE-complete

We show a somewhat weaker result for this problem, than for the three previous problems; although there is an F an a k for which Cycle_F^k is indeed PSPACE-complete, we do not know of an automaton achieving this with $cc\left(\mathrm{Cycle}_F^k\right) \in o(n)$:

Proposition 3.4.5

1. For any cellular automaton F and any $k \geqslant 1$, $\mathrm{Cycle}_F^k \in$ PSPACE
2. There is a cellular automaton F and a integer k such that Cycle_F^k is PSPACE-complete.

Proof.

1. Let F and $k \geqslant 1$ be fixed. The length of the cycle reached by iterating F on a periodic initial configuration c can be determined in polynomial space with the algorithm described below. Let n be the period of c. Starting from c, the cycle is reached in less than $|Q|^n$, where Q is F's alphabet:
   1. Compute $c_0 = F^{\alpha^n}(c)$. This can be operated directly on the configuration, without recording the intermediate steps.
   2. Save a copy of configuration c_0 (still of size n), and start to simulate the automaton for k steps on the working copy. At each step, verify if the current configuration is equal to c_0. If so, output 1 and exit. Else, at the end of the k steps, output 0 and exit.

2. To show this, we embed a Turing machine $\mathcal{M}$, deciding a PSPACE-complete problem, into a cyclic configuration for a cellular automaton. $\mathcal{M}$ works in polynomial space, meaning that there is a polynomial $P \in \mathbb{N}[X]$ such that for any $x \in \Gamma^*$, it will never use more than $P(|x|)$ cells.

   This embedding is very similar to the Turing computations that we performed with cellular automata in the proof of proposition 3.4.1, without the moving particle. Then, if we make the accepting state spreading, the configuration will be erased completely upon acception of the input by $\mathcal{M}$, and the cycle length will be 0. Moreover, the accepting state launches a particle, erasing everything behind itself, while rotating. This proves than on valid configurations, $\textsc{Cycle}_F^k(\langle x \rangle) = 1$ if and only if $\mathcal{M}$ accepts x, where $\langle x \rangle$ means "an encoding of x into a configuration for the cellular automaton".

   Now, a polynomial time transducer can easily output a configuration of F. It first outputs a translation of the initial configuration, and then $P(|x|)$ blank states. □

3.5 Experimental theoretical computer science

A particularly interesting part of the study of cellular automata is the experiments it allows for. In this section, we show we took advantage of the ease of enumerating and simulating entire classes of cellular automata, to prove real theorems. This approach, to our knowledge, has not been so widely used yet to *prove* things, but rather to produce *images* of space-time diagrams.

Moreover, many theoretical results about communication complexity are based on provable properties of function matrices, but are not aware of a general study on concrete matrices generated by complex functions. This situation is not extremely different in the Turing world, but this ability has not been largely used yet; to our knowledge, one of the most notable publications in this field has been [51].

The beginning of this research direction was also motivated by Neary and Woods' result about the P-completeness of rule 110. The common belief before this result was that simulating Turing machines with this rule could only be done with an exponential overhead. Since, as we have shown in section 3.4.1, the prediction problem of a universal cellular automaton must be P-complete, this new result could possibly mean that an elementary cellular automaton is universal.

In this section, we have used experiments on the communication complexity of elementary cellular automata to pre-classify them according to their *one-way* communication complexity. This restriction is justified by the following theorem:

Theorem 3.5.1 Let $\mathcal{P}$ be a protocol computing a function f of $X{\times}Y \to Z$, in which all the nodes are labeled by a_is (that is, only Alice speaks). Then the monochromatic rectangles $\mathcal{P}$ induces on matrix f are of the form $X_0{\times}\{y \in Y \mid f(x,y) = z\}$, for some $X_0 \subseteq X$ and $z \in Z$.

Proof. This is shown with an even simpler induction than in the proof of theorem 3.1.1:

- For a protocol when there is no need for communication, it is obviously true.
- With more communication rounds, the first bit Alice says induces a partition of her inputs into two sets $X_0, X_1 \subseteq X$. By induction hypothesis, the end of the protocol partitions each of these two sets again, so that X_0 gets partitioned into a family $(X_{0,i})_i$, and X_1 into a family $(X_{1,i})_i$. In the end, the union of these families defines a partition of $X \times Y$ into sets of the form $X_j \times \{y \in Y \mid f(x,y) = z\}$. □

The basic way of using this theorem is the following: we compute the *exact* one-way communication complexity for small classes of cellular automata by counting, for each input size n, the number of different lines in matrices of $M_{2^n}(\mathbb{Z}_2)$. This restriction is obviously justified by complexity constraints: we could have as well assumed the log-rank conjecture (open problem 3.1.2, but, while it is quite easy to devise an algorithm to count the number of different lines in a matrix, in linear time, this is not the case for matrix rank algorithms (which are all at least quadratic in 2^n).

3.5.1 The elementary cellular automata

In this section, we present a collection of protocols built with the following methodology: we classified the cellular automata according to their complexity for problem Pred (our only problem with a reasonable computation time). Then, we tried to build a small protocol for those cellular automata with low complexity. For some of them, such as rule 94 or 218, the complexity of Pred was high, but there were “walls” in the rule, which made their SInv problem trivial.

Proposition 3.5.2 Rules 15, 51, 60, 90, 105, 108, 128, 136, 150, 160, 170 and 204 are linear, and thus have a prediction protocol in O(1).

Proposition 3.5.3 Rule 76 has a protocol in O(1) for Pred.

Proof. On all configurations after one step, rule 76 behaves like rule 204, because the only difference is on 111, which has no antecedent. Therefore, Alice and Bob need to communicate one bit to compute the first step, and then follow the protocol for rule 204. □

Proposition 3.5.4 Rules 0, 1, 2, 4, 8, 10, 12, 19, 24, 34, 36, 38, 42, 46, 72, 76, 108, 127, 138, 200 have a constant number of dependencies, and thus have a protocol in O(1) for Pred.

Proof.

- Rule 0 is nilpotent.
- Any configuration of the form 0001^n000 is stable under F_1^2, and neither 1001 nor 101 have antecedents by F_1, thus F_1^t at most depends on the seven center cells.

- In rule 2, after one step, there can never be two 1s separated by less than two 0s. And, on these configurations, rule 2 is a shift.
- In rule 4, after one step, the 1s are all separated by at least one 0, and on these configurations, the rule is the identity.
- Rule 8 is nilpotent.
- Rule 10 is a left shift on all the configurations with no three consecutive 1s. Fortunately, these configurations never appear after one step.
- For rule 12, the only configurations after one step have only isolated 1s, on which this rule is the identity.
- In rule 19, after two steps, there are no isolated 0s or 1s, and on these configurations, F_{19}^2 is the identity.
- The only difference between rule 24 and the symmetric of rule 2 is on transition 011, which has no antecedent. The same protocol (reverting the roles of Alice and Bob) can be used, after simulating one step of the rule.
- Rule 34 is a left shift on the configurations with no block of two consecutive 0s, and these blocks do not have antecedents.
- For rule 36, we find out by exhaustive search that the only stable pattern is 00100, and all other patterns of length five become 0 after two steps.
- For rule 38, another exhaustive search shows that on $F_{38}(\{0,1\}^{\mathbb{Z}})$, $F_{38}^2 = \sigma^2$.
- For rule 42, after one step, there are no three consecutive 1s in the configuration, and the rule is a left shift on these configurations.
- Rule 46 is a left shift except on 010, which has no antecedent, and 111, whose antecedents have 010s. Therefore, after two steps, this rule is actually a left shift.
- In rule 72, for any a and b, $F_{72}(a0110b) = 0110$. But 111 does not have antecedents by F_{72}, and 010 does not have antecedents by F_{72}^2. Therefore, the only configurations of $F_{72}^2(\{0,1\}^{\mathbb{Z}})$ are of the form $(\{(0110)0^*_,0^*\})^*$.
- Any block of three cells, except 111, is stable under rule 72. Therefore, this block disappears after one step.
- An exhaustive search on all the blocks of length 7 of rule 108 show that F_{108}^2 is the identity on $F_{108}^2(\{0,1\}^{\mathbb{Z}})$.
- Rule 138 is a left shift, except on 101, which has no antecedent and thus disappears after one iteration.
- In rule 200, any 0 is stable (for any a and b, $F_{200}(a0b) = 0$), and so are the blocks of at least two 1s. Moreover, isolated 1s do not have antecedents. Therefore, the rule depends only on the three central cells. □

Proposition 3.5.5 Rule 5 has a protocol for SINV in $O(1)$ bits.

Proof. For any value of a and b, $F_5(a010b) = 010$, and for any a, b, c, d, $F_5^2(ab000cd) \in \{000, 010\}$. Therefore, for the configuration to be invaded, u should neither contain more than three consecutive 0s, nor less than two consecutive 1s. This is not possible after one iteration of the rule since $F_5(11011) = 000$, and $F_5(110011) = 0000$. □

Proposition 3.5.6 Rule 7 has a protocol for SInv in O(1) bits.

Proof. First notice that for any values of w, x, y and z, $F_7^2(w11xyz) = 11$. Since $F_7(0000) = 11$ and $F_7(0001) = 11$, a periodic word u that would be invaded should have neither blocks of more than two 0s, nor blocks of more than one 1. Thus, it should be described by the regular expression $(\{0,00\}1)^*$. But since $F_7(0010) = 11$, this leaves only one possibility : the pattern should be 01^*. Thus, any perturbation of size n stays at most n bits wide, and no invasion can ever occur. □

Proposition 3.5.7 Rule 13 and 29 have a protocol for SInv in O(1) bits.

Proof. Let us remark that for any values of a and b, $F(a01b) = 01$, for both rules. Thus, if the input is different from the periodic background, and it is to be invaded, then the background has only 1s. But then the last cell that is different from the background in the input is a 0, and this forms a wall. Thus, no invasion can ever occur. □

Proposition 3.5.8 Rule 28 has a protocol for SInv in O(1) bits.

Proof. First remark that since for any values of a and b, $F_{28}(a01b) = 01$. Hence, any periodic background that would be invaded should be uniform (i.e. only 0s or only 1s). But then the left of the configuration is necessarily invaded, and the first 01 or 10 creates a wall. □

Proposition 3.5.9 Rule 78 has a protocol for SInv in O(1) bits.

Proof. It is not hard to see that the configurations where all the 0s are separated, and with no three consecutive 1s are stable under this rule. Now, 111 has no antecedent by rule 78. Thus on the configurations without this pattern, for any value of a and $n \geqslant 2$, $F_{78}(a10^n1) = 10^{n-1}1$. Then, it is not hard to check that on the configurations with no block of more than one consecutive 0 or more than two consecutive 1, rule 78 is the identity. □

Proposition 3.5.10 Rule 140 has a protocol for SInv in O(1) bits.

Proof. For any values of a and b, $F_{140}(a0b) = 0$. □

Proposition 3.5.11 Rule 172 has a constant protocol for invasion

Proof. For an invasion to occur, there should be no two consecutive 0s in the background, since for any values of a and b, $F_{178}(a00b) = 00$. Also remark that 010 has only two antecedents, and they both contain a 00, and also that the only antecedents of 00 contain 00s.

So let us suppose that the background has no couple of 0s. Since no 010 can occur in the configuration after one step of evolution, then either the input does not contain a couple of 0s, and thus the rule behaves like a left shift – there is no invasion, or it has a couple of 0s – and a discussion is needed. We consider the leftmost 00 appearing in the configuration. It is clear that it forms a wall. We shall prove that it invades the configuration rightwise. We only need to check that for all values of a, b and c such that $abc0$ does not contain a wall, we have:

$$F_{172}(abc00) = *00 \qquad \square$$

Proposition 3.5.12 Rule 32 has a protocol for SInv in $O(1)$ bits.

Proof. If u is not of the form $(01)^*$, then F_{32} is uniformly 0 after n steps, with n the input length. Else, if $u = (01)^*$, and $p_u(x) \neq p_u$, then the configuration gets invaded with 0s. □

Proposition 3.5.13 Rule 156 has a protocol for SInv in O(1).

Proof. First notice that if the period u is not uniform, then there are walls (01) around the input x and then x does not invade p_u. Else, if u is uniform, then p_u is invaded, either to the left if $u \in 1^*$, or to the right if $u \in 0^*$. □

Proposition 3.5.14 There is a protocol in $O(1)$ for Pred and SInv for rule 27.

Proof. First notice that $F^2(*111***) = 111$, and $F(*000*) = 111$. Thus, if the orbit of p_u contains a block of three 1s or three 0s, then no invasion can occur. Else, an exhaustive exploration of all configurations of size 6 shows that the only possible configurations that do not generate 111 or 000 are described by the following regular expression :

$$A = (011 + 001)^*$$

Moreover, it is easy to notice that this set of configurations is stable under F, and that $F^2(w_1w_2w_3w_4w_5) = w_5$ for w any subword of a word in A. Thus, if $p_u(x)$ is still in A, no invasion can occur, since F^2 is a left shift. Else, $p_u(x)$ is not in A. In this case at least one block of three 1s occurs somewhere in $p_u(x)$, $F(p_u(x))$ or $F^2(p_u(x))$, propagating to the right of the configuration. There are two cases :

1. Alice has the leftmost one (or Alice and Bob share it, which can be determined within constant communication). In this case, since the rightmost wall shifts to the right at the same speed than the leftmost one, Alice can simply assume that Bob has got any configuration :

since the chunks after his rightmost wall only consist of periodic background pattern, only shifting right, it does not change the result.

2. If Bob has it, and $p_u(x) \neq p_u$ in Alice's part, then Alice's part is shifted to the left, and Bob's leftmost "wall" (blocks of 111) to the right. The background is always invaded here. Else, Bob knows Alice's configuration, since it is the same as in pu : he can still predict everything. □

Proposition 3.5.15 Rule 44 has a protocol for SInv in O(1) bits.

Proof. First remark that $\forall\ a,b \in \{0,1\}$, $F_{44}(a00b) = 00$: 00 is a "wall". Thus, among the eight possible groups of three states, 000, 001, 100 are walls. Now for the remaining blocks:

- $\forall\ a \in \{0,1\}$, $F_{44}(111a) = 00$.
- $\forall\ a,b \in \{0,1\}$, $F_{44}(010ab) = 111$, and thus yields a wall.

The only remaining blocks are $W = \{011,101,110\}$. If the orbit of p_u has no wall, then $p_u \in \{011\}^*$. Else, no invasion is possible. Thus, we can simply remark that for all rotations of 011 (i.e. all words $w \in W$), we have: $F_{44}^2(w00) = 0$. Since 00 is a wall, this means that any x such that $p_u(x) \neq p_u$ will invade the configuration, erasing all the left part of it, from the wall on.

Since the condition $p_u(x) \neq p_u$ can be checked locally, two bits of communication are enough to decide $\text{Inv}_{F_{44}}$. □

Proposition 3.5.16 Rules 23, 50, 77, 178 and 232 have a protocol for Pred in $O(\log n)$ bits.

Proof. They all leave stable either {00,11} or {01,10}, where a "stable" set S means that for any $s \in S$, there is an $s' \in S$ such that for all $w_1, w_2 \in \{0,1\}$, $F(w_1 s w_2) = s'$.

In this case, since the part before the first block has to be either uniform or alternate between 0s and 1s, a protocol for Pred can simply send the length of this part. Since no information can pass through the stable subwords, this is enough to predict the evolution of the configuration. □

Proposition 3.5.17 Rules 40, 130, 162 and 168 have a protocol for SInv in O(1) bits.

Proof. First notice that in all four rules, for any $a,b \in \{0,1\}$, $f(ab0) = 0$. This if a word u has at least one 0, then no word x can invade p_u. Else, the perturbation stays at most $w+1$ bits wide, with w the distance between the leftmost 0 of x and its rightmost one. □

The few next lemmas and definitions study the case of rule 94, which maybe our most complicated example of an elementary cellular automaton simple for SInv. This proof was published in [36]

Definition 3.5.1 We call the configurations of rule 94 *additive* if their language is included in $((00)^{+}(11)^{+})^{*}$.

Now, for *additive* configurations, it is clear that the behavior of rule 94 is exactly the same as the behavior of rule 90, which is bi-permutative. This is because of the following lemma:

Lemma 3.5.18 If x is an additive configuration, then so is $F_{94}(x)$. Moreover, F_{94} is bipermutative on these configurations.

Proof. This is because $F_{94}(00(11)^{n}00) = 11(00)^{n-1}11$ and $F_{94}(11(00)^{n}11) = 11(00)^{n-1}11$ for all $n \geqslant 1$. But F_{94} only differs from rule 90 on transition 010, which never appears here; hence the bipermutativity. □

Lemma 3.5.19 If c is a non-additive configuration which does not contain 010, then 101 appears after a finite time and it is a wall. More precisely, a wall appears after $t+1$ steps at the middle of any occurrence of $10^{2t+1}1$ or $01^{2t+3}0$, for $t \geqslant 0$.

Proof. 101 is a wall because for all a, b, $F_{94}(a101b) = 101$. Moreover, $F_{94}(10^{n}1) = 10^{n-2}1$ for $n \geqslant 2$. □

Lemma 3.5.20 The orbit of a configuration c contains a wall if and only if $F_{94}(c)$ is not additive.

Proof. From lemma 3.5.19, it is enough to show that if c is a configuration where 101 does not appear, then 010 does not appear either in $F_{94}(c)$. To see this, it is enough to check that any word u such that $F_{94}(u) = 010$ must contain 101. □

We can now conclude by giving a protocol for SINV:

Proposition 3.5.21 For any $u \in \{0,1\}^{*}$, $cc\left(\text{SINV}_{94}^{u}\right) \in O(\log n)$.

Proof. If u is such that the orbit of p_u contains a wall, then invasion never occurs. Thus, for the problem to be difficult, from lemma 3.5.20, $F_{94}(p_u)$ must be additive. In this situation, two cases are to be considered, depending on the input $x_1 \ldots x_n$:

- Either $F_{94}(p_u(x_1 \ldots x_n))$ is also additive, and then, by lemma 3.5.18, the configuration is invaded if and only if $F_{94}(p_u) = F_{94}(p_u(x_1 \ldots x_n))$. This can be decided with a finite number of communications.
- Or $F_{94}(p_u(x_1 \ldots x_n))$ is not additive. Then it must contain $10^{2t+1}1$ or $01^{2t+3}0$ as a subword, for some $t \geqslant 0$, because, as in the proof of lemma , if the image of a configuration contains a 010, then it must also contain 101. Consider the leftmost and the rightmost occurrences of this kind of words. Since walls appear above the middle of these two occurrences after a time equal to half their lengths (by lemma 3.5.19, the invasion does not depend on the

contents of the part of the configuration between the two occurrences. It takes $O(\log n)$ bits of communication for Alice to know the position of these two occurrences, and the exact words present at their positions (of type $10^{2t+1}1$ and $01^{2t+3}0$). Moreover, as soon as Alice knows this, she also knows that on the left of the leftmost occurrence, and on the right of the rightmost occurrence, the configuration is additive. If there is no difference with p_u in these additive parts, then there is no invasion. Else, then Alice has got enough information to decide the problem alone. Deciding in which case we are can be done within constant communication. □

Proposition 3.5.22 There is a protocol in $O(1)$ for SINV for rule 104.

Proof. Let us first notice that rule 104 is symmetric, and that for any value of a and b, $F_{104}(a00b) = 00$. Thus, we need to find the configurations on which this wall does not appear. Looking at all the configurations of size 4, we find the following:

1. 0110: the only possible way to avoid walls is to have a 1 on each side of it, that is, 101101, but then $F_{104}^2(101101) = 00$.
2. 0101 and 1010
3. 1011 and 1101
4. 0111 and 1110

Thus, the only possible repetitions of 1s can be three 1s or one 1. Let us look at the word of length four that can occur after 0111, without creating a wall. We first notice that this word cannot begin with a 1 without creating a wall. Two possibilities are left:

1. 0111
2. 0101: this case creates a wall after three steps.

Since rule 104 is symmetric, this shows that the only configurations on which there is no 00 wall are $(01)^*$ and $(0111)^*$. Thus, only these configurations may be invaded. These two configurations are each stable under rule 104. To see that they actually are when the configuration is finitely changed, it is sufficient to simulate what happens by creating a wall at $i, i+1, i+2, i+3$ for some position i, because they are both 4-periodic, and the content of the configuration after the wall does not matter. □

Proposition 3.5.23 There is a protocol in $O(\log n)$ for PRED of rule 132.

Proof. For any $a, b \in \{0,1\}$, $f(a0b) = 0$. Thus, Alice only needs to send the length of the longest string of 1s she has from the center. □

Proposition 3.5.24 There is a protocol in $O(1)$ for SINV for rule 152.

Proof. Let us first remark that no finite group of at least two 1s can be in the orbit of rule 152. This comes from the three following subrules of rule 152:

1. for $a \in \{0,1\}$, $f(a011) = 01$
2. $f(111) = 1$
3. $f(110) = 0$

Henceforth, if all the 1-blocks are finite (there is at least one 0 in the periodic pattern), then no invasion can occur, since after a finite number of steps, the rule behaves like rule 32; that is, a right shift.

Else, the periodic pattern has only 1s, thus any 0 in the input word invades the configuration: towards its left because of subrule 3, and upright because of subrule 1. The fact that Alice and Bob have only 1s in their respective inputs can be checked with only one bit of communication. □

Proposition 3.5.25 Rule 156 has a protocol in $O(1)$ for SINV.

Proof. Let us first notice that 01 is a wall in rule 156: for any $a, b \in \{0,1\}$, $F_{156}(a01b) = 01$. Thus, the only case where invasion could occur would be when the background pattern has only 0s or only 1s (else, a wall appears on both sides). If there are only 0s, the first 1 creates a wall, and since $f_{156}(100) = 1$, the right of the configuration get invaded by the last 1. Since $f(110) = 0$, the same happens when the background pattern has only 1s. □

Proposition 3.5.26 Rule 184 has a protocol in $O(\log n)$ for PRED, and this protocol is optimal.

Proof. Let us see what happens to the blocks of two cells in rule 184. Let $A = 00$, $B = 01$, $C = 10$ and $D = 11$. Then, for all $n \geqslant 0$:

$$F^n_{184}(A\{B,C\}^n) = A$$

$$F^n_{184}(\{B,C\}^n D) = D$$

$$F^n_{184}(AD) = B$$

$$F^n_{184}(DA) = B$$

Thus, let $\sharp A_{\text{Alice}}$ be the number of A Alice has, $\sharp D_{\text{Alice}}$ her number of Ds, $\sharp A_{\text{Bob}}$ the number of A Bob has, and $\sharp D_{\text{Bob}}$ his number of Ds.

Moreover, we say that position i is *free* if:

$$\sharp D(w_0 ... w_i) \geqslant \sharp A(w_0 ... w_i)$$

$$\sharp D(w_{i+1} ... w_n) \geqslant \sharp A(w_{i+1} ... w_n)$$

Then, the following is a valid protocol for F_{184}:

- Alice sends $N_A = \max(0, \sharp A_{\text{Alice}} - \sharp D_{\text{Alice}})$ to Bob.
- If $N_B > N_A$, then Bob knows the answer (if he has a C particle in a free zone, the result is C, else it is B).
- Else, if $N_B < N_A$, then Alice knows the answer: if she has a C particle in a free zone, then the result is C, else it is B.

The following fooling set shows that this protocol is optimal:

$$S = \left\{ A^i B^{n-i}_{,} B^{n-i} D^i \mid i \in \{1 \ldots n\} \right\} \qquad \square$$

With a slight modification, the protocol we had for rule 184 can also predict rule 56:

Proposition 3.5.27 There is a protocol in $O(1)$ for $\textsc{Pred}_{F_{56}}$.

Proof. Using the same bulking parameters, there are only two differences:

$$F(DD) = A \quad \text{and} \quad F(BD) = C$$

But fortunately, none of these two problems have any antecedent, thus they disappear after one step, which requires only two bits of communication to be simulated. □

3.5.2 The last candidates to universality

In last section, we have shown simple protocols for a large number of elementary cellular automata, and essentially problems $\textsc{Pred}$ and $\textsc{SInv}$. The following elementary automata remain without proof of simplicity nor of universality:

We do not know simple protocols for the following 33 automata: 3, 6, 9, 11, 14, 18, 22, 25, 26, 30, 33, 35, 37, 41, 43, 45, 54, 57, 58, 62, 73, 74, 106, 110, 122, 126, 134, 142, 146, 152, 154, 164, 204.

The main open problem of this section on experiments remains:

Open problem 3.5.1 Is there an intrinsically universal cellular automaton among the elementary cellular automata ?

The smallest known intrinsically universal cellular automaton is the one of Ollinger and Richard (see [19]), with radius one and four states. Our conjecture is that no elementary cellular automaton is intrinsically universal. In chapter 4, we will see extensions of the experimental approach presented here, that seem promising to the study of this conjecture.

3.6 Two-dimensional extensions

In the beginning of this chapter, we did most of our definitions, problems and theorems in the context of one-dimensional cellular automata. This degree of generality is far from being satisfying, as many definitions, well understood in the one-dimensional case, lose their sense, or become undecidable in two dimensions. We will see a few examples below. Therefore, as we will see, the generalization of our problems to two dimensions is quite non-trivial.

The first thing that we will need to generalize is the way of cutting the inputs. In one dimension, this was not quite a problem, since once we had defined that the cut should cut at most one bulking block, there was not a large choice of cuts; the notion of *border* was obvious. In two dimensions, this is one more thing we need to redefine:

Definition 3.6.1 Let $S = \{0 \dots n-1\}^2$ be a square of $\mathbb{Z}^2$. A *simple path* is a finite sequence $(u_n)_n$ of points of S such that: $\|u_{n+1} - u_n\|_1 = 1$ and $\forall i,\ j,\ u_i \neq u_j$.

We call a *border* on S a minimal simple path dividing S into two connected components.

With this definition, it is not hard to see why all our problems are still valid here. We define them below, but we will begin with a general idea to adapt the results.

First, the exact same proofs apply for relation $\sqsubseteq$: it is still true that if $F \sqsubseteq G$, then $cc(\mathcal{P}_F) \preccurlyeq cc(\mathcal{P}_G)$. However, the *bulking* relation is a little more complicated to define. What we need to do is to show how to use a protocol for some automaton F using a protocol for $F^{\langle m,t,z \rangle}$, and vice-versa. When we have the protocol for F, the adaptation is trivial. In the other case, in one dimension, we needed to communicate a constant number of bits, in order for one player to know the bulked state around the border. We can do the same here, but for one player to know all the states around the border, they need to communicate $\Omega(m)$ bits, where m is the length of the border. This cost is significantly higher than in the one-dimensional case, as it may be in $\Omega(n)$. However, for an automaton to be intrinsically universal, the same condition clearly holds: its communication complexity must not be in $o(n^2)$.

In the adaptations we define below, we can always construct a cellular automaton in $\Omega(n^2)$: assume there is a one-dimensional cellular automaton F, of communication complexity $\Omega(n)$ for problem $\mathcal{P}$. If the problem of simulating n instances of F together has communication complexity in $\Omega(n^2)$, then we can build a two-dimensional cellular automaton with maximal complexity. Indeed, we can simply simulate F on each row separately to get the result.

Now, for all the hard problems we have constructed in section 3.2, we were largely reducing problems Eq or Disj, for which computing two independent instances is the same as computing instances twice larger. Hence, our hypothesis that computing n independent instances at once is no easier than computing them n times separately always holds for our problems.

We can now define the generalizations of our problems:

- The prediction problem does not really need to be generalized, as definition 3.2.2 is defined in terms of F^*, which is valid in any dimension.

- The temporal invasion problem does not need to be generalized either, since definition 3.2.4 is formulated in terms of periodic configurations (which still exist in two dimensions), and differences between configurations.
- The spatial invasion problem can be generalized by replacing δ_l and δ_r by the following definitions:

$$\delta^2(t) = \max\left\{ \|i-j\| \mid F^t(p_u(x))[i] \neq F^t(p_u)[i] \quad \text{and} \quad F^t(p_u(x))[j] \neq F^t(p_u)[j] \right\}$$

  And $\textsc{SInv}_F^u$ is the problem of deciding if $\delta^2(t) \to \infty$.
- The cycle length problem could well be defined in any dimension with the same definition as definition 3.2.6

The two-dimensional case is thus an easy generalization of our one-dimensional problems. However, they make up an interesting source of open problems, in particular to study special cases of long-standing open problems such as the *direct sum* problem, that we have already defined (open problem 3.2.17).

Moreover, several results in two dimensions let us think that this extension of our approach may be interesting. In particular, Applebaum, Ishai and Kushilevitz in [52], show how to use a special kind of two-dimensional cellular automata, namely cellular automata in which the rule is allowed to change at each step. Our approach here with communication complexity may enable us to prove (or disprove) the stronger result that their cryptographic primitives are also implementable in plain cellular automata with a uniform rule.

4 Extensions to other simulations

4.1 Communication ideals

In definition 1.2.1, we required our simulation relation to be preorders. In fact, this hypothesis allows for a nice formalism of simulations in cellular automata, already defined in [10], using order theory. It is quite easy to see that the set of automata simulated by some automaton $\mathcal{A}$ with relation $<$ is an ideal, in the sense of order theory, that is:

Definition 4.1.1 Let S be a set, $<$ be a quasiorder relation on $S \times S$. An ideal for $<$ is a subset $\mathcal{I}$ of S such that:

1. If $F_2 \in \mathcal{I}$ and $F_1 < F_2$, then $F_1 \in \mathcal{I}$.
2. For any $F_1, F_2 \in \mathcal{I}$, there is some $F_3 \in \mathcal{I}$ such that $F_1 < F_3$ and $F_2 < F_3$.

A *principal ideal* $\mathcal{I}$ is an ideal with a maximal element, that is, there is an $i \in \mathcal{I}$ such that $\forall x \in \mathcal{I}$, $x < i$.

Several results have been proved by Delorme et al. in [10] about this formalism. The following proposition summarizes the results useful for us:

Proposition 4.1.1 *(from [10])* $\mathcal{I}$ is an ideal for $\preccurlyeq_{\sqsubseteq}$ (respectively $\preccurlyeq_{\trianglelefteq}$) if:

1. $\forall\ m, t \in \mathbb{N}, z \in \mathbb{Z},\ F \in \mathcal{I} \Leftrightarrow F^{\langle m,t,z \rangle} \in \mathcal{I}$
2. $F_2 \in \mathcal{I} \ \wedge\ F_1 \sqsubseteq F_2$ (resp. $F_1 \trianglelefteq F_2$) $\Rightarrow F_1 \in \mathcal{I}$
3. $F_1 \in \mathcal{I} \ \wedge\ F_2 \in \mathcal{I} \Rightarrow F_1 \times F_2 \in \mathcal{I}$

We can reconsider, with this definition, all the results of section 3.2. Indeed, since both simulation relations $\sqsubseteq$ and $\trianglelefteq$ are preorders, we can summarize these results as:

Proposition 4.1.2 Let $(X_F)_F$ a family of communication problems, defined for each cellular automaton F, of complexity increasing with respect to simulation $\preccurlyeq$ (that is, if $\varphi \preccurlyeq \psi$, then $cc(X_\varphi) \prec cc(X_\psi)$). Let f be a non-decreasing function from $\mathbb{N} \to \mathbb{N}$, then the following set is an ideal for $\preccurlyeq$:

$$\mathcal{I} = \{\varphi \mid cc(X_\varphi) \prec f\}$$

Proof. This is a direct consequence of the definition of $\prec$. □

In the literature about simulation ideals in cellular automata and symbolic dynamics, the two most studied relations are the ones that we have defined in chapter 1: $\preccurlyeq_{\sqsubseteq}$ and $\preccurlyeq_{\trianglelefteq}$. However, until now, we have mostly restricted our study to the first one, namely, subsystem simulation. In this chapter, we generalize our approach to the other simulation. We begin with a presentation of the tools we are going to use: non-determinism and randomization. Then, the main result of this chapter, in section 4.3, shows an example of an automaton universal for $\preccurlyeq_{\sqsubseteq}$, while in a quite small ideal for $\preccurlyeq_{\trianglelefteq}$.

4.2 Determinism and sub-system simulation

As we said in chapter 1, it is not known whether there is a universal cellular automaton for the coloring simulation relation. In this part, we present an interesting link between this problem, and two different generalizations of communication complexity, namely *non-deterministic* complexity, and *randomized* complexity.

4.2.1 The limit set

The problem we are going to define is the problem of deciding if a given pattern can appear arbitrarily late in the evolution of the automaton. We first need to give the following classical definition:

Definition 4.2.1 Let F be a cellular automaton. The *limit set* of F is defined by:

$$\Omega_F = \left\{ x \in Q_F^{\mathfrak{C}_F} \mid \forall t,\ \exists\, y \in Q_F^{\mathfrak{C}_F},\ F^t(y) = x \right\}$$

This set is obviously a subshift, and it is not difficult to see why its language is co-recursively enumerable. Several examples of automata with non-recursive limit sets are shown in [53] or [30].

Based on this definition, we can now ask Alice and Bob to compute if a given word can appear in a configuration of the limit set, that is, if the input is in the *language* of the limit set. This is precisely the $\textsc{Limit}_F$ problem:

Definition 4.2.2 Let F be a cellular automaton on alphabet Q, and x a word of Q^*. The problem $\textsc{Limit}_F$ is defined by:

$$\textsc{Limit}_F(x) = \chi_{\mathcal{L}(\Omega_F)}(x)$$

Where for any set S, χ_S is its characteristic function.

As we proved in [54], it is quite easy to see why this problem is compatible with $\trianglelefteq$. We first need to show that the property of "being in the language of the limit set" is conserved by $\trianglelefteq$:

Proposition 4.2.1 If Φ is a factor map of (X,F) onto (Y,G), then $\Omega_G = \Phi(\Omega_F)$.

Proof. The proof is two-fold:

- $\Phi(\Omega_F) \subseteq \Omega_G$. Indeed:

$$\Phi(\Omega_F) = \Phi\Big(\bigcap_{t\in\mathbb{N}} F^t(X)\Big) \subseteq \bigcap_{t\in\mathbb{N}} \Phi(F^t(X)) = \bigcap_{t\in\mathbb{N}} G^t(X)$$

- Now, let $y \in \bigcap_{t\in\mathbb{N}} G^t(Y)$, and, for $t \in \mathbb{N}$, $X_t = \Phi^{-1}(y) \cap F^t(X)$. Note that X_t is closed, since Φ and F are continuous, and X is compact. Moreover, X_t is nonempty, for Φ is onto. By Cantor's intersection theorem, $\bigcap_{t\in\mathbb{N}} X_t = \Phi^{-1}(y) \cap \Omega_F$ is nonempty, and thus $y \in \Phi(\Omega_F)$. □

Next, we only need to find a way to use a protocol for LIMIT$_F$ to solve LIMIT$_G$ where G is a factor of F. Unfortunately, the simple method we used in section 3.2.2.1 does not work anymore, since proposition 4.2.1 only proves the existence of one antecedent of the input, by the induced map Φ, that is in the limit language of F. See for instance the following construction:

Proposition 4.2.2 For any cellular automaton F, there is a cellular automaton F' such that $F \trianglelefteq F'$ and for each $x \in \Omega_F$, there is a $y \in Q_{F'}^{\mathbb{Z}} \setminus \Omega_{F'}$ such that $\Phi(y) = x$, where Φ is the map induced by $\trianglelefteq$.

Proof. The idea of the proof is to duplicate the alphabet to artificially construct configurations not in the limit set of F'. Concretely, for each $q \in Q_F$, F' has two corresponding states q and q', and $F'(a,b,c) = F(\alpha(a),\alpha(b),\alpha(c))$, where $\alpha : Q_{F'} \to Q_F$ "removes the prime", in the sense that $\forall q \in Q_F$, $\alpha(q) = q$, and $\forall q' \in Q_F \setminus Q_{F'}$, $\alpha(q') = q$.

No configuration with a state in $Q_F \setminus Q_{F'}$ can ever be in the limit set, since these configurations have no antecedent by F'. Yet, the factors of F are actually factors of F'. □

However, if x is a configuration of Ω_F, then its image by the coloring map induced by the relation $G \trianglelefteq F$, $y = \Phi(x)$, is in Ω_G, since $F^{-t}(\Phi^{-1}(y))$ is nonempty.

4.2.2 Non-deterministic protocols

One possible generalization of the deterministic framework introduced in section 3.1 is adding non-determinism. In the original definition, given by Lipton and Sedgewick [44], this corresponds to the scenario where, "an all powerful prover, who sees x and y, is trying to convince Alice and Bob that $f(x,y) = z$" (this description comes from [43]). If it is actually the case that $f(x,y) = z$, then Alice and Bob must agree, else they must be able to detect it, whatever the prover says. In this case, the complexity of the protocol is the total speaking time of the prover, Alice, and Bob.

The example given in [43] is a protocol for the non-equality function, hereafter written NEQ, which is the negation of function EQ that we defined in section 3.1.1. In this problem, if $x \neq y$, the prover can simply give Alice and Bob an index where their inputs differ. The hint given by the prover is seen by both players; it is *public*. In the case of NEQ, it takes at most $O(\log(n))$ bits for the prover to transmit a position, then Alice and Bob can verify it by sending each other their bit at this position in $O(1)$. The cost of this protocol is thus $O(\log n)$.

In our definition, however, we allow the prover to give a "hint" *for free* to each player. But contrarily to the classical definition, this information is private; the players do not know each other's hint. It is clear that any function, within this framework, is going to be no harder than in the previous definition: if there is a protocol of cost n for some function f, for the definition of [44], then there is a protocol with at most n bits with our definition: the prover just gives the same hint, for free but privately, to Alice, who then transmits it to Bob. Conversely, if there is a private protocol for f, then Alice and Bob can simulate it using a public protocol. At each round of the private protocol where a private hint is useful, the prover gives the outcome of the round. Since this information is public, Alice and Bob can skip the round: they both know what to do next. Formally, the definition of a non-deterministic protocol goes like this:

Definition 4.2.3 Let $f : X{\times}Y \to \{0,1\}$. A *non-deterministic* communication protocol $\mathcal{P}$ for f is a deterministic protocol for a *completion* of f, that is, for a function $f' : (X{\times}X'){\times}(Y{\times}Y') \to \{0,1\}$, such that:

$$\forall x \in X,\ y \in Y,\ f(x,y) = 1 \Rightarrow \exists x' \in X',\ y' \in Y,' f'((x,x'),(y,y')) = 1$$

$$\forall x \in X,\ y \in Y,\ f(x,y) = 0 \Rightarrow \forall x' \in X',\ y' \in Y,' f'((x,x'),(y,y')) = 0$$

A non-deterministic protocol $\mathcal{P}$ is said to *compute* f if:

- for all (x,y) such that $f(x,y) = 1$, it answers 1 on input $((x,x'),(y,y'))$ for at least one value of (x',y').
- for all (x,y) such that $f(x,y) = 0$, there is no value of (x',y') for which it answers 1 on input $((x,x'),(y,y'))$.

We also need a definition of non-deterministic communication complexity:

Definition 4.2.4 For $z \in \{0,1\}$, we say that f has z-non-deterministic communication complexity n, and we write $N^z(f) = n$, if there is a non-deterministic protocol $\mathcal{P}$ for f, such that the paths leading to a z are of length at most n.

Now, how can we lower-bound the non-deterministic complexity of a function? If we represent our protocols also by trees, where each node is labeled by a function depending on the input, and also on the private hint, then each leaf is again a monochromatic rectangle. The difference is that the same input may lead to different leaves, because there may be several hints for it. The following proposition proves by a combinatorial argument the equivalence between the two definitions of non-deterministic communication complexity:

Proposition 4.2.3 Let f be a function of $X{\times}Y \to \{0,1\}$, and $N^1(f)$ its private or public communication complexity, and $C^1(f)$ the size of a minimal cover of the 1-inputs of f's matrix by rectangles. Then:

$$N^1(f) = \lceil \log_2(C^1(f)) \rceil + 1$$

Proof.

- For public protocols, any cover of f's matrix induces a proof system, in which the prover gives the name of a 1-rectangle in exactly $\lceil \log_2(C^1(f)) \rceil$ bits, where $C^1(f)$ is the number of 1-rectangles in f's matrix.
- For private protocols, the prover can give Alice the same hint as in the public protocol (the name of a rectangle), and she transmits it to Bob. This also costs exactly $\lceil \log_2(C^1(f)) \rceil$.
- Conversely, for public protocols, we can reuse the proof of Theorem 3.1.1, since at each step of the protocol, whoever speaks splits the inputs into two parts. The same induction works to prove that each leaf corresponds to a monochromatic rectangle in the matrix. Now, since several proofs might be used on the same inputs, the rectangles may overlap.
- For private protocols, once Alice and Bob know their respective hints, their behavior is essentially the same as in a deterministic protocol; the proof of Theorem 3.1.1 works without any tweak: a leaf always corresponds to a monochromatic rectangle. And, again, different hints on the same input might lead to different (overlapping) monochromatic rectangles. □

Almost the same fooling set technique allows to prove the same lower bounds for EQ and DISJ:

Proposition 4.2.4 Let $z \in \{0,1\}$. If there is a 1-*fooling set* of size n for f, then the non-deterministic communication complexity of f is at least $\log n$.

Proof. A 1-fooling set S is a fooling set for the 1-inputs of f: $\forall (s,s') {\in} S$, $f(s,s') = 1$, and for any $(s_1, s_1') \in S$, $(s_2, s_2') \in S$, either $f(s_1, s_2') = 0$ or $f(s_2, s_1') = 0$. The same proof is still valid, as a fooling set only proves that no two elements of it can be in the same rectangle. □

An example 1-fooling set for EQ is the same as in the deterministic case ($S_1 = \{(x,x) \mid x \in \{0,1\}^n\}$). To lower-bound the 0-non-deterministic complexity of EQ, however, we need other arguments. For instance, Exercise 2.6 of [43] uses the fact that a cover of size n of the z-inputs of the matrix gives a deterministic protocol: Alice can tell Bob all the 0-rectangles her input is in; at round i, she says 1 if and only if her input is in rectangle R_i. At the end, Bob has enough information to answer. This proves that $n \leqslant D(f) \leqslant C^0(f) + 1$, and thus $\log n \leqslant N^0(f)$.

Remark here that in our definitions of non-determinism, we only define protocols for one result of the function: either 0 or 1. In reality, this does not really matter; we could as well have defined N^0 in a completely symmetric way, and have chosen the maximum of N^0 and N^1 (let us call it N) as *the* definition of non-deterministic complexity. The proofs of hardness would have had the same difficulty, since proving that $ncc^1 \in \Omega(n)$ implies $N \in \Omega(n)$. But the proofs of low complexity would have been harder, and this definition would make fewer functions simple. Since we are mainly interested in finding simple protocols for cellular automata, it would be a less powerful tool for our purpose.

Before applying it to cellular automata, we need to cope again with the problem of input splitting. Remember the i-concatenation function $\mathcal{C}_i$ defined in section 3.1.3. We can define the non-deterministic communication complexity as before:

Definition 4.2.5 Let φ_n be an application of $Q^n \to \{0,1\}$. The 1-non-deterministic communication complexity of φ, hereafter denoted as $ncc(\varphi)$, is defined by:

$$ncc^1(\varphi_n) = \max_{i \leqslant n} N^1(\varphi_n \circ \mathcal{C}_i)$$

Finally, the definition of private protocols is justified by their use in factor simulations: indeed, assume that $F \trianglelefteq G$, and that we have a non-deterministic private protocol for some problem $\mathcal{P}_G$ on G. When solving the corresponding problem on F, the non-determinism gives Alice and Bob a way to choose a right configuration for G:

Lemma 4.2.5 Let f be a function of $X \to \{0,1\}$, and $g : Y \to \{0,1\}$ be a *coloring* of this function, that is, a function such that there is an onto function $\varphi : X \to Y$ such that for all $y \in Y$, $g(y) = 1 \Leftrightarrow (\exists x \in \varphi^{-1}(y), f(x) = 1)$. Then $ncc(g) \leqslant ncc(f)$.

Proof. We use the private formulation of non-deterministic communication protocols. Since φ is an onto application of $X \to Y$, then when on input (y_0, y_1), if there is an element $(x_0, x_1) \in \varphi^{-1}(\{y_0\}) \times \varphi^{-1}(\{y_1\})$ such that $\mathcal{P}_F(x_0, x_1) = 1$, then the prover gives x_0 to Alice, x_1 to Bob, and they can use the protocol for $\mathcal{P}_F$, which proves the result. Remark that we implicitly use the fact that the input for the protocol is already split, that is, the input for $\mathcal{P}_F$ is from $X \times X$, if f is from $X \to X$. □

A direct application is this is problem Limit: by proposition 4.2.1, the hypotheses are clearly verified:

Proposition 4.2.6 Let F and G two cellular automata such that $F \leqslant_{\trianglelefteq} G$. Then:

$$ncc^1(\text{Limit}_F) \prec ncc^1(\text{Limit}_G)$$

Proof. This is only a matter of verifying that all the basic ingredients of $\trianglelefteq$ are compatible with the protocols:

- It is clear that $F^{\langle m,t,z\rangle}$ and F have the same protocol for Limit, since temporal and spatial rescaling are both bijective operations and do not change the complexity of the limit language.
- Then, Lemma 4.2.5 finishes the proof by showing that:

$$\mathrm{ncc}^1(\textsc{Limit}_{F^{\langle m_F,t_F,z_F\rangle}}) \prec \mathrm{ncc}^1(\textsc{Limit}_{G^{\langle m_G,t_G,z_G\rangle}}) \qquad \square$$

Finally, we must show that there is a cellular automaton F such that $\textsc{Limit}_F$ has maximal non-deterministic communication complexity. We proved this in [48], with the following automaton:

Proposition 4.2.7 There is a cellular automaton F such that:

$$\mathrm{ncc}^1(\textsc{Limit}_F) \in \Omega(n)$$

Proof. Automaton F is a product of three layers:

1. A shift to the left on alphabet $\{0,1\}$.
2. A shift to the right on alphabet $\{0,1\}$.
3. A test layer with alphabet $\{\emptyset,\top,\bot\}$. This is a cellular automaton of radius 0, with the rule that:
   - The blank state $\emptyset$ remains blank
   - State $\bot$ remains $\bot$.
   - Whenever a $\top$ sees a 1s on both layers, it becomes a $\bot$. Otherwise, it remains $\top$.

The problem of deciding if an input of the following form:

$$(x_1,y_1,\emptyset)...(x_n,y_n,\emptyset)(x_{n+1},y_{n+1},\top)(x_{n+2},y_{n+2},\emptyset)...(x_{2n},y_{2n},\emptyset)$$

is in F's limit set amounts to deciding if $x_1...x_n = y_{2n}...y_{n+2}$, and $y_1...y_n = x_{2n}...x_{n+2}$, since it is the only way to preserve a $\top$ state on the third layer. This is another variant of the Disj problem. According to the lower bound we showed above, we can still use the fooling set technique. Therefore, $\mathrm{ncc}^1(\textsc{Disj}) \in \Omega(n)$, and thus $\mathrm{ncc}^1(\textsc{Limit}_F) \in \Omega(n)$. □

4.2.3 Randomized protocols

In the model of deterministic communication complexity, the communications at each stage of the protocols are completely determined by deterministic functions of the input, and of the communications that have happened until the current stage. On the contrary, randomized communication complexity allows the players to use random bits in their communications. There are several variants defined in the literature; in the one we will use, Alice and Bob have access to a common random string of arbitrary length, and they are allowed to make errors with constant probability (over the random string).

To formalize this framework, we set up the same context as with deterministic protocols; a protocol tree, with the nodes labeled by functions, this time depending on the inputs, but also on the random string, and the leaves labeled by results:

Definition 4.2.6 Let f be a function of $X{\times}Y \rightarrow Z$. A *randomized public coin protocol* $\mathcal{P}$ computing f is a tree, where the internal nodes are labeled either by $a_i : X{\times}\{0,1\}^* \rightarrow \{0,1\}$ (for the rounds where Alice speaks), or by $b_i : Y{\times}\{0,1\}^* \rightarrow \{0,1\}$, and the leaves are labeled by values of Z.

For an input (x,y) and a random string $r \in \{0,1\}^*$, we write $\mathcal{P}(x,y,r)$ the deterministic protocol over (x,y) where each node i is labeled by $a_{i,r} : (x,y) \mapsto a_i(x,y,r)$.

Since we want to allow errors in the computations, we need a notion of computation with errors. Let f be a function of $X{\times}Y \rightarrow Z$, and $\epsilon > 0$. We say that $\mathcal{P}$ *computes* f with error probability ϵ if for any input (x,y), the probability (over the random string $r \in \{0,1\}^*$) that $\mathcal{P}(x,y,r) \neq f(x,y)$ is bounded by ϵ.

Definition 4.2.7 For $\epsilon \geqslant 0$, the randomized communication complexity of f, denoted as $R_\epsilon(f)$, is the complexity of the best randomized protocol achieving error probability ϵ.

Before defining this model in our context with cellular automata, we shall note that the randomized communication complexity only defines a probability distribution μ over deterministic protocols, over which the probability of error is defined. For each input, at least a fraction $1-\epsilon$ (weighted by μ) of the protocols must give a correct answer.

Again, the input splitting issue is resolved by considering the cut maximizing the randomized communication complexity:

Definition 4.2.8 Let φ_n be an application of $Q^n \rightarrow \{0,1\}$, and $0 \leqslant \epsilon \leqslant 1$. The randomized communication complexity of φ, hereafter denoted as $rcc(\varphi)$, is defined by:

$$rcc^1_\epsilon(\varphi_n) = \max_{i \leqslant n} R_\epsilon(\varphi_n \circ \mathcal{C}_i)$$

To use it on cellular automata, we need to prove that it is increasing with colorings, and that there are hard problems for it. The first result comes from a theorem by Yao [55], relating *distributional* complexity to *randomized* complexity. In the model that we have just described, a randomized protocol is a probability distribution over the protocols; that is, the probability we consider is a probability on the coins tossed by the players. On the contrary, in distributional complexity, we consider a distribution probability μ *over the inputs*, and a fixed deterministic protocol, whose probability of error, when the inputs are chosen according to distribution μ, is no greater than ϵ. Formally:

Definition 4.2.9 The μ-distributional complexity of f, denoted as $D_\epsilon^\mu(f)$, is the least complexity a deterministic protocol such that $Pr_\mu(\mathcal{P}(x,y) \neq f(x,y)) \leqslant \epsilon$.

Now, we need to prove that this notion of communication complexity is useful to our approach, that is, it is compatible with simulation by coloring. We first need the following result by Yao, whose proof is to be found in [43].

Theorem 4.2.8 *(from [55])*$R_\epsilon(f) = \max_\mu D_\epsilon^\mu(f)$

The following lemma is the equivalent of Lemma 4.2.5 for public coin randomized complexity.

Lemma 4.2.9 Let f be a function of $X \rightarrow \{0,1\}$, and $g : Y \rightarrow \{0,1\}$ be a *coloring* of this function, that is, a function such that there is an onto function $\varphi : X \rightarrow Y$ such that for all $y \in Y$, $g(y) = 1 \Leftrightarrow \exists x \in \varphi^{-1}(y),\ f(x) = 1$. Then $rcc_\epsilon^1(g) \leqslant rcc_\epsilon^1(f)$.

Proof. Let μ be a probability distribution on Y. Then we can transform it into a distribution μ' on X, in the following way: for each $x \in X$, let $S_x = \varphi^{-1}(\varphi(\{x\})) \cap f^{-1}(\{1\})$. Then:

- If $S_x \neq \emptyset$, we set $\mu'(x) = \mu(\varphi(x))$ if $x = \min S_x$, and 0 else.
- Else, we set $\mu'(x) = \mu(\varphi(x))$ if $x = \min(\varphi^{-1}(\varphi(\{x\})))$, and 0 else.

We use min here to choose a unique element of each S_x. This choice is not relevant, and, on finite sets like X or Y, it is a correct choice function.

Now, by Theorem 4.2.8, we know that there is a μ'-distributional protocol $\mathcal{P}'$ for f, operating in at most $R_\epsilon(f)$ bits, with error ϵ. Transforming it into a μ-distributional protocol $\mathcal{P}$ for g is quite easy: we keep the same protocol tree, and only turn the labels of $\mathcal{P}'$, at each node i, from a_i' or b_i' into a_i and b_i, as follows: At each node where Alice speaks, and for all $y \in Y$, set $a_i(y) = 1$ if and only if, for $x \in \varphi^{-1}(\{y\})$ such that $\mu'(x) > 0$ (there is only one such x, by construction of μ'), $a_i'(x) = 1$. We convert Bob's nodes (those labeled by a b_i') similarly.

For the inputs such that $f(x,y) = 0$, the protocol does not change: that is, the same bits are exchanged for x in $\mathcal{P}'$ and $\varphi(x)$ in $\mathcal{P}$, considering an adequate splitting of x and $\varphi(x)$. This is because we have not changed the nodes' labels for these inputs. Else, for v such that $f(v) = 0$, there is only one element of $u \in \varphi^{-1}(v)$ such that $\mu'(v) > 0$, thus the protocol is still correct with probability ϵ with distribution μ'. If there is set of weight more than ϵ with distribution μ, on which the protocol $\mathcal{P}$ we have just built makes errors, then we can transform it into a set of weight more than ϵ for μ' on which $\mathcal{P}'$ is incorrect, which contradicts the definition of $\mathcal{P}'$. □

Like in the case of non-deterministic communication complexity, we can also apply this definition to the case of the LIMIT problem, exactly with the same proof as for proposition 4.2.6:

Proposition 4.2.10 Let F and G two cellular automata such that $F \preccurlyeq_{\trianglelefteq} G$, and $\epsilon > 0$. Then:

$$\mathrm{rcc}_\epsilon(\textsc{Limit}_F) \prec \mathrm{rcc}_\epsilon(\textsc{Limit}_G)$$

4.2.4 Applications to natural examples

The two new definitions of communication complexity we have given in this section offer a new playground for experiments on natural examples, such as the elementary cellular automata. Let us recall our goal of section 3.5, that was: *showing that no elementary cellular automaton is intrinsically universal.* Covering the matrices is a much easier problem than finding a partition for them; indeed, it is an instance of Set - Cover problem, that has a polynomial time $O(\log n)$-approximation (see [56]).

It is not hard to see that these two definitions are compatible with sub-automata:

Proposition 4.2.11 Let P_F be a problem on cellular automata, F be a cellular automaton on alphabet Q_F, and $\epsilon > 0$. Then, for any $G \preccurlyeq_{\sqsubseteq} F$:

$$\mathrm{ncc}^1(P_G) \prec \mathrm{ncc}^1(P_F)$$

$$\mathrm{rcc}_\epsilon(P_G) \prec \mathrm{rcc}_\epsilon(P_F)$$

However, as in the case of the logrank conjecture (open problem 3.1.2), we do not know of any natural example on which this approach really helps:

Open problem 4.2.1 What is the minimal number of states such that there is a radius one cellular automaton F for which $\mathrm{ncc}^1(\textsc{Pred}_F) \in o(\mathrm{cc}(\textsc{Pred}_F))$?

4.3 Separating colorings and sub-system

The result we are going to show now is a separation result, that we published in [54], between the simulations $\trianglelefteq$ and $\sqsubseteq$. We show the existence of a $\sqsubseteq$-universal cellular automaton with a limit language in NL, thus, according to Juraj Hromkovič's theorem (theorem 3.1.4), with communication complexity in $O(\log^2 n)$. This is actually far from our lower bound on the complexity of the limit set of a universal cellular automaton: indeed, the following result shows that for if that if the limit set of a cellular automaton F is an SFT (that is, by definition 1.1.6, if its language is finite), then F is *stable*, that is, it reaches its limit set after only a finite number of steps:

Definition 4.3.1 Let F be a cellular automaton over some alphabet Q. We say that F is *stable* if there is a step t_F such that $F^{t_F}(Q^{\mathbb{Z}}) = \Omega_F$.

Lemma 4.3.1 Let F be a cellular automaton. If Ω_F is an SFT, then F is stable.

Proof. An SFT subshift, as defined in Definition 1.1.6, is characterized by a finite set of forbidden patterns S. For each $w \in S$, by compactness, there is a first time step t_w such that for all $t \geqslant t_w$, w does not appear anymore in $F^{t_w}(Q^{\mathbb{Z}})$. Taking $t^* = \max_{w \in S} t_w$, we have:

$$\Omega_F = F^{t^*}(Q^{\mathbb{Z}}) \qquad \square$$

For those automata with an SFT limit set, the problem TInv is easy to solve:

Proposition 4.3.2 Let F be a cellular automaton with an SFT limit set. Then, for all $u \in Q^+$:

$$cc\left(\text{TInv}_F^u\right) \in O(1)$$

Proof. A protocol for TInv_F iterates F t^* times over $p_u(x)$, to reach the limit set. This can be done in $O(t^*)$ bits of communication, which is $O(1)$ here. Alice and Bob now need only check if there is a difference between $F^{t^*}(p_u(x))$ and $F^{t^*}(p_u)$. A classical result of symbolic dynamics (see [57]) shows that F is preinjective on an *irreducible subshift,* which means that it preserves finite differences.

This means that the differences will remain forever between the orbits of p_u and $p_u(x)$ if and only if $F^{t^*}(p_u(x)) \neq F^{t^*}(p_u)$. This completes the proof, and shows that there cannot be $\sqsubseteq$-intrinsically universal stable cellular automata. □

Returning to our goal of separating coloring and subsystem simulations, the construction is based on the existence of a firing-squad cellular automaton S, due to Kari in [58]. This cellular automaton has a so-called *firing state* γ, and a spreading state κ. We call its radius r_S, its alphabet Q, and Q' the set $Q \setminus \{\kappa, \gamma\}$. Furthermore, we define the set X_S of the configurations with an infinite history avoiding κ and γ, i.e. $X_S = \Omega_S \cap Q'^{\mathbb{Z}}$.

Lemma 4.3.3 S is such that:

1. For any $j \in \mathbb{N}$, there is a z such that $S^j(z) \in \{\gamma\}^{\mathbb{Z}}$, and $\forall t < j$, $S^t(z) \in Q'^{\mathbb{Z}}$.
2. $\forall i \in \mathbb{Z}$, $\Omega_S \cap [\gamma]_i \subseteq \{\kappa, \gamma\}^{\mathbb{Z}}$.
3. X_S is recognizable in NL.

Proof. Properties (1) and (2) are proved in [58]. We prove property (3) in section 4.3.1 below. □

We now prove the following theorem:

Theorem 4.3.4 Any cellular automaton is a sub-automaton of a cellular automaton whose limit set is recognizable in NL.

Proof. The idea of the proof is to add an extra component to a given cellular automaton F over alphabet A, which delays the apparition of any configuration of $A^{\mathbb{Z}}$ to a time arbitrarily late in the future. The limit set is thus completely flooded with configurations of $A^{\mathbb{Z}}$. The technical difficulty is to control the contribution of the additional component to the complexity of the final limit set.

Let F be a cellular automaton of radius r_F, local rule f, and alphabet A, with a spreading state $0 \in A$. We define cellular automaton $\Delta_{F,S}$, of local rule $\delta_{F,S}$, on alphabet $C = A \cup (A \times Q)$, with radius $r = \max(r_F, r_S)$, by:

1. $\delta_{F,S}(c) = f(c)$ if $c \in A^{2r+1}$
2. $\delta_{F,S}(c) = a_0$ if $c = (a_{-r}, \gamma) \ldots (a_r, \gamma)$
3. $\delta_{F,S}(c) = \left(a_0, s\left(b_{-r_S} \ldots b_{r_S}\right)\right)$ if $c = (a_{-r}, b_{-r}) \ldots (a_r, b_r) \in (A \times Q')^{2r+1}$
4. $\delta_{F,S}(c) = 0$ otherwise

Intuitively, this cellular automaton freezes its first component, while applying the firing squad on the second component, until the firing state appears somewhere. Only then, it starts the evolution of F on the first component. When the configuration is not consistent, or when κ appears, spreading state 0 is generated. It is not hard to see why F is a sub-automaton of $\Delta_{F,S}$. The structure of $\Delta_{F,S}$ is described by the following lemma:

Lemma 4.3.5 $A^{\mathbb{Z}} \subseteq \Omega_{\Delta_{F,S}}$

Proof. Let $x \in A^{\mathbb{Z}}$ and $j \in \mathbb{N}$. From point 1 of lemma 4.3.3, there is a configuration $z \in Q^{\mathbb{Z}}$ such that $F^{j-1}(z) \in \{\gamma\}^{\mathbb{Z}}$, and for any $t < j-1$ and any $i \in \mathbb{Z}$, $F^t(z)[i] \notin \{\gamma, \kappa\}$. Consider now the configuration $y = (x_i, z_i)_{i \in \mathbb{Z}}$. By an easy induction on $t < j$, we can see that for any cell $i \in \mathbb{Z}$, only case (3) of the local rule is used, and $\Delta^t_{F,S}(y)[i] = \left(x_i, S^t(z)[i]\right)$. At time j, since $S^{j-i}(y)[i] = \gamma$, part (2) of the rule is applied, and $\Delta^j_{F,S}(y)[i] = x[i]$. As a result, $x \in \bigcap_{j \in \mathbb{N}} \Delta^j_{F,S}(C^{\mathbb{Z}})$. □

Lemma 4.3.6 Let $x \in \Omega_{\Delta_{F,S}}$, $i < j$ two integers. If $x[i] = (a[i], \gamma)$ and $x[j] = (a[j], b[j])$, then $b[j] \in \{\gamma, \kappa\}$.

Proof. Assume, on the contrary, that $b[j] \notin \{\gamma, \kappa\}$. Let $(x^t)_{t \in \mathbb{Z}}$ be a biorbit of $x = x^0$, that is, a bisequence of configurations such that $\forall t \in \mathbb{Z}$, $\Delta_{F,S}(x^t) = x^{t+1}$. By an easy recurrence, and the fact that states of $A \times Q$ are obtained only by case (3) of the rule, we see that:

$$x^{-t}[i - rt \ldots i + rt] \in (A \times Q)^{1+2rt}$$

Similarly, $x^{-t}[j - rt \ldots j + rt] \in (A \times Q)^{1+2rt}$ and $s^t\left(b^{-t}[j - r_S t \ldots j + r_S t]\right) = b_j$. Then, for any $t > \frac{j-i-1}{2r}$, $x^{-t}[i - 2rt \ldots j + 2rt]$ is in $(A \times Q)^{j-i-1+4rt}$ and the image $s^t\left(x^{-t}[i - r_S t \ldots j + r_S t]\right)$ contains b_i and b_j. In other words, the cylinder $\left[b_i Q^{j-i-1} b_j\right]_i$ intersects $S^t(Q^{\mathbb{Z}})$ for any t, and by compactness, intersects Ω_S, which contradicts point 2 of lemma 4.3.3. □

If $\Sigma \subseteq A^{\mathbb{Z}}$ is a subshift and $0 \in A$, then we consider the set $0{\circ}\Sigma{\circ}0$ of configurations or pieces of configurations of Σ surrounded by 0:

$$0{\circ}\Sigma{\circ}0 = \bigcup_{-\infty \leqslant l \leqslant m \leqslant \infty} \left\{x \in A^{\mathbb{Z}} \mid x[l...m] \in \mathcal{L}(\Sigma) \text{ and } \quad \forall i \notin \{l...m\},\ x[i] = 0\right\}$$

Lemma 4.3.7 $\Omega_{\Delta_{F,S}} \setminus A^{\mathbb{Z}} \subseteq 0{\circ}(A\times Q)^{\mathbb{Z}}{\circ}0$.

Proof. By shift-invariance, it is sufficient to prove that:

$$\Omega_{\Delta_{F,S}} \cap [A\times Q]_0 \subseteq 0{\circ}(A\times Q)^{\mathbb{Z}}{\circ}0$$

Let us prove by induction on n that the patterns of $(A\times Q)(A^{2rn} \setminus \{0^{2rn}\})$ are forbidden in $\Omega_{\Delta_{F,S}}$. The base case is trivial, since there are no such patterns. Now, assume it holds for $n \in \mathbb{N}$, and there exists a configuration $x \in \left[(A\times Q)0^{2rn+k}(A\setminus\{0\})\right]_0 \cap \Omega_{\Delta_{F,S}}$, with $1 \leqslant k \leqslant 2r$. Consider a preimage $i \in \Omega_{\Delta_{F,S}}$ of x.

On the one hand, in cell 0 of y, we must have applied case (3) of the rule, so that $y[-r...+r] \in (A\times Q)^{2r+1}$, and this word does not involve γ.

On the other hand, if we have applied case (1) in cell $2nr+k+1$ of y, then:

$$y[(2n-1)r+k+1...(2n+1)r+k+1] \in (A\setminus\{0\})^{2r+1}$$

but the space between these two neighborhoods is $(2n-1)r+k+1-r-1 \leqslant 2nr-1$, which contradicts the induction hypothesis.

The other possibility was that we had applied case (2) in cell $2nr+k+1$, which involves a state γ among cells of $y[(2n-1)r+k+1...(2n+1)r+k+1]$, which contradicts lemma 4.3.6. In the limit, and with a symmetric argument on the left, we obtain that all the configurations of $\Omega_{\Delta_{F,S}} \setminus A^{\mathbb{Z}}$ are in $0{\circ}\Sigma{\circ}0$. □

Lemma 4.3.8 Let us write $A^{\mathbb{Z}}{\times}X_S = \left\{(a_i, s_i)_{i\in\mathbb{Z}} \in (A\times Q)^{\mathbb{Z}} \mid (s_i)_{i\in\mathbb{Z}} \in X_S\right\}$. Then:

$$\Omega_{\Delta_{F,S}} = A^{\mathbb{Z}} \cup 0{\circ}\left(A^{\mathbb{Z}}{\times}X_S\right){\circ}0$$

Proof. Let $x \in C^{\mathbb{Z}}$ and $-\infty \leqslant l \leqslant m \leqslant \infty$ such that $x[l...m] \in (A\times Q)^{\mathbb{Z}}$, and for any $i \notin [l...m]$, $x[i] = 0$. First assume that $x \in \Omega_{\Delta_{F,S}}$, i.e. for any $t \in \mathbb{Z}$, there exists $x^t \in \Delta^t_{F,S}(x)$. By recurrence, we can see that $x^{-t}[i] \in A\times Q'$ for all $i \in \{l-rt...m-rt\}$ and $t \geqslant 1$ since states from $A\times Q$ only come from case (3) of the rule. Let $w^{-t} \in Q'^{m-l-2rt+1}$ be the projection of $(x^{-t})[l-rt...m+rt]$ on its second component. Clearly, w^0 is in the language of X_S. We deduce that $x = x^0 \in 0{\circ}\left(A^{\mathbb{Z}}{\times}X_S\right){\circ}0$.

Conversely, assume that $x \in A^{\mathbb{Z}} \times X_S$, that is, there are $l \leqslant m$ two integers, and a sequence (y^t) with, for $t \geqslant 1$, $y^t \in Q'^{\mathbb{Z}}$ and $y^t = S(y^{t+1})$ and, for any $t \in \mathbb{N}$ and any $i \in \{l ... m\}$, $x = (a[i], S^t(y^t)[i])$ for some $a[i] \in A$. Now, take the configuration $\overline{y}^t \in C^{\mathbb{Z}}$ such that for any $i \notin \{l - rt ... m + rt\}$, $\overline{y}^t[i] = 0$, and for any $i \in \{l - rt ... m + rt\}$, $\overline{y}^t[i] = \left(b_i, y^t_i\right)$ with $b_i \in A$ and $b_i = a_i$ if $i \in \{l, m\}$. By a direct recurrence, for any $j < t$ and any $i \notin \{l - rt + rj ... m + rt - rj\}$, we have $\Delta^j_{F,S}(\overline{y}^j) = 0$ and for any $i \in \{l - rt ... m + rt - rj\}$, we have $\Delta^j_{F,S}(\overline{y}^j)[i] = \left(b_i, S^j(y^j)[i]\right)$ (since $y_j \in Q'$, case (3) of the definition of $\Delta_{F,S}$ applies at position i of $\overline{y}^j$). This proves that $\Delta_{F,S}(y) = x$. Thus:

$$\Omega_{\Delta_{F,S}} \cap 0 \circ (A \times Q)^{\mathbb{Z}} \circ 0$$

We can conclude thanks to lemmas 4.3.7 and 4.3.5. □

Corollary 4.3.9 $\mathcal{L}\left(\Omega_{\Delta_{F,S}}\right)$ is NL-recognizable.

Proof. From lemma 4.3.8 and point 2 of definition 4.3.3, the limit language is a finite boolean formula of finite concatenation of NL-recognizable languages. □

Let F be a cellular automaton on some alphabet A. We can artificially add some spreading state $0 \notin A$ to build a cellular automaton F' on alphabet $A \cup \{0\}$ which admits F as a sub-automaton. Now, we have seen that F' is a sub-automaton of $\Delta_{F',S}$. From corollary 4.3.9, its limit set has an NL-recognizable language. □

4.3.1 A firing squad cellular automaton

Let S be the firing squad cellular automaton defined in [58]. Its radius is 1. Let us call its alphabet Q. It has size 16, with a killer state κ. The complete rule is given in Kari's article [58], and figure 4.1 shows an example of each neighborhood on which the rule does not yield κ.

The analysis of S's limit set goes by finding the form of the configurations that can be in the limit set. For this purpose, we will use arguments about distances ran by *signals*. However, since cellular automata are by essence discrete objects, it will often be convenient to use a continuous version of signals, made possible by the following lemma:

We call a *euclidean* history diagram a set of labeled points, lines, segments and half-lines of $\mathbb{R}^2$. Two lines cannot cross; instead, they are transformed into a number of other lines (possibly null). The lines are labeled by symbols among $\left\{L_1, l_1, l_2, \sharp', r_2, r_1, R_1\right\}$, and the points by the same symbols, X, Y, Z or ♯. The rules are the same as in the local rule of the automaton. We only need to take care of possible "rounding problems":

Lemma 4.3.10 To each history diagram D, we can associate a valid euclidean history diagram E such that, at any integer coordinate of E containing a point or a single signal, the label gives the state of the corresponding position in D.

Proof. There is no rounding problem for signals of slope -1, 0 and 1, since the coordinates on both axes are always either both integers or both non-integers. For slopes $1/2$ and $-1/2$, whenever two points p_0 and p_1 with integer coordinates are connected by a vector with that slope, it is easy to see that the cells at positions $\{p_0+(i,2i) \mid 0 \leqslant i \leqslant n\}$ (or $\{p_0+(-i,2i) \mid 0 \leqslant i \leqslant n\}$ in the case of slope $-1/2$), that is, the cells with integer coordinates on this segment, are in state r_2 (respectively l_2). □

We prove now that the limit language of S is recognizable in logarithmic space by a non-deterministic Turing machine (that is, $\mathcal{L}(\Omega_S) \in \textsc{nl}$).

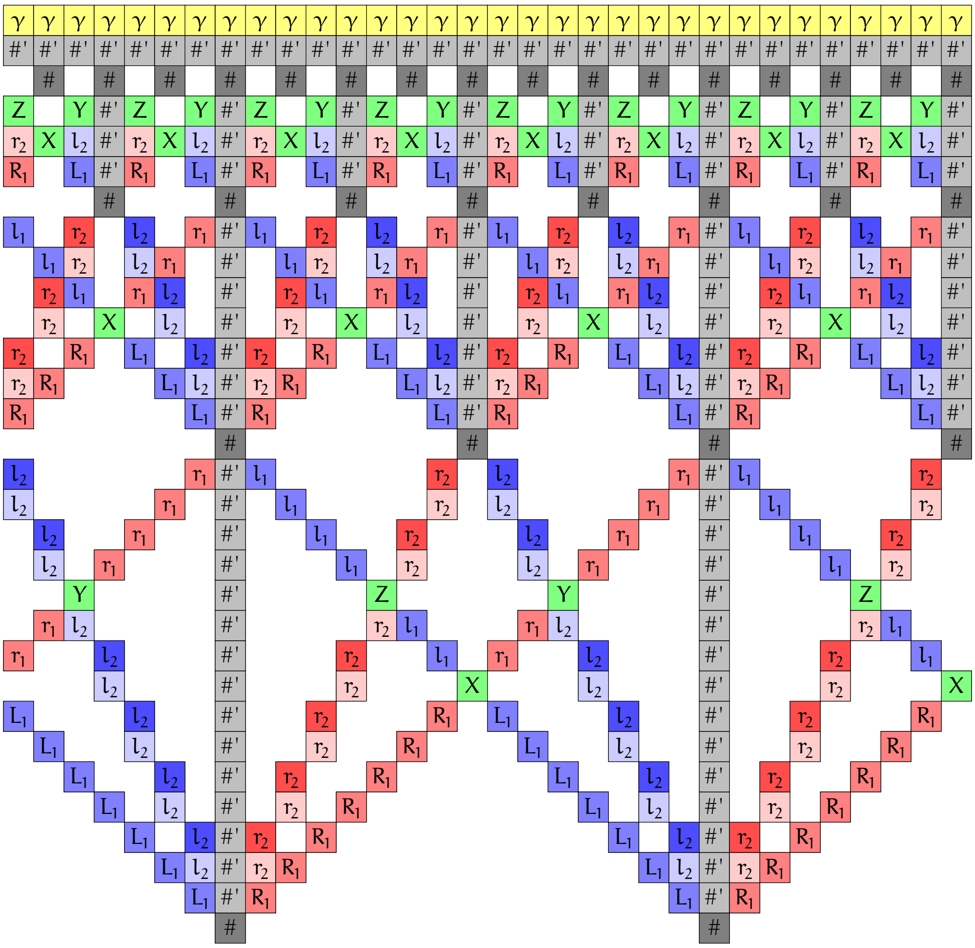


Figure 4.1 - The firing squad automaton of [58]

4.3.1.1 Synchronization of the $\sharp$s.

The first four lemmas show that if there is both a $\sharp$ and a $\sharp'$ in a word, without κ between them, then this word is not in the limit language.

Lemma 4.3.11 Consider a history diagram containing a word $w \in \sharp Q \sharp'$ at time t_0. Let z_1 be the position of w's first cell, and z_2 the position of w's last cell. If the $\sharp$ at z_1 was created by an l_1 signal, then there is a time step t_1, before t_0, in which cell z_1 was in state $\sharp$, and a time step t_2, also before t_0, in which z_2 was in state $\sharp$.

Proof. First let us prove the existence of t_1. Assume that there is no $\sharp$ in the past of cell z_1. The only possible post of this cell, in this case, is necessarily an infinite column of $\sharp$'s. But we assumed that the left $\sharp$ was created by an l_1 signal, which necessarily either crossed this column, or was generated by it. In both cases, it is a contradiction.

Now that we know this, we prove the existence of t_2. Assume that cell z_1 has only $\sharp'$ in its past. Since the $\sharp$ in the right column was necessarily created by a signal coming from the left, be it r_1 or r_2, this means that the signal should also have crossed the left column, which is impossible. Thus, there is at least one $\sharp$ in this column, that generated the l_1 signal. □

Now, this lemma allows us to prove that there is actually no configuration of this form in the limit set of S. Let us call $\Sigma = Q' \setminus \{\sharp, \sharp'\}$.

Lemma 4.3.12 There is no history diagram containing a word w of the form $\sharp\Sigma\sharp'$, where the left $\sharp$ was created by a r_1/l_1 pair of signals.

Proof.

Let us call $\mathcal{Z}$ the l_1 signal that created the left $\sharp$ of w at t_0. According to lemma 4.3.11, there is some time step t_1 in which a $\sharp$ appears in the past of the right $\sharp'$ (we call this column $\mathcal{C}$). Moreover, let t_2 be the most recent step in which a $\sharp$ appears in the past of the right $\sharp$ at t_0; we call this column $\mathcal{B}$. There are two cases:

- This $\sharp$ in column $\mathcal{C}$ was created by an r_1 signal $\mathcal{Z}_1$. In this case, this $\sharp$ is necessarily the same as the one that generated $\mathcal{Z}$. Else, applying the same argument as in lemma 4.3.11, there would be another $\sharp$ between $\mathcal{Z}$, and the $\sharp$ at t_1, and then $\mathcal{Z}_1$ would have crossed both $\mathcal{Z}$ and the L_1 signal emitted by this intermediate $\sharp$.

  We now have two subcases:

  - The $\sharp$ at time t_1 in column $\mathcal{B}$ was created by signals l_2 and r_2. Thus, the $\sharp$ on column $\mathcal{C}$, at time t_1, was created by signals l_1 and r_1, for otherwise, we would have an l_2/r_2 intersection between the two columns, which is would create a column of $\sharp$s between the two columns, that would still exist at time t_0. Thus:

1. Column $\mathcal{C}$ had $\sharp$s or $\sharp$'s before t_1, and we can thus consider the first $\sharp$ on it before t_1. This is the $\sharp$ which emitted the l_2 signal creating the $\sharp$ on column $\mathcal{B}$ at t_2, otherwise an l_1 or l_2 signal emitted by this $\sharp$ would intersect column $\mathcal{B}$ between t_0 and t_2, which is impossible by minimality of t_2.
2. The r_2 signal arriving on the $\sharp$ of column $\mathcal{B}$ at time t_2 and the r_1 arriving on the $\sharp$ of column $\mathcal{C}$ at time t_1 meet in the past on some $\sharp$. This allows to infer the existence of a third column $\mathcal{A}$ on the left of $\mathcal{B}$ and $\mathcal{C}$.

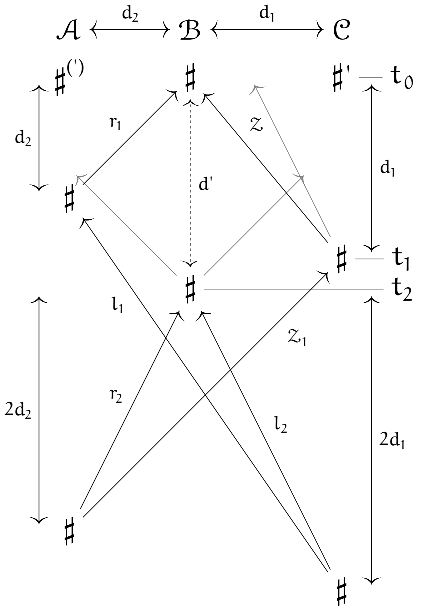


Figure 4.2

We can summarize this situation on figure 4.2.

We have the following equations on the distances:

$$d_1 + d_2 = 2d_2 + d' - d_1 \tag{1}$$

$$2d_1 = d_2 + d'$$

$$d_2 + d_1 = 2d_1 + d' - d_2 \tag{2}$$

$$2d_2 = d_1 + d'$$

Equation 1 comes from the r_1 signal from the leftmost bottom $\sharp$, and equation 2 comes from the rightmost bottom one. From this, we can conclude that $d_1 = d_2 = d'$, and that the R_1/r_1 signal emitted at time t_2 by the $\sharp$ in column $\mathcal{B}$ reaches the right column at time t_0, contradicting the fact that the state at time t_0 in this right column is $\sharp'$.

- It was created by an l_1/r_1 intersection, and so was the right $\sharp$ at t_1. In this case, one of them has to have another $\sharp$ in its past, by lemma 4.3.11. Thus, this new $\sharp$ sends an r_2 or l_2 signal (depending on which one we consider), that necessarily collides either with the other column of $\sharp$'s, or with the L_1 or R_1 signals represented in thick on the following figure 4.3:

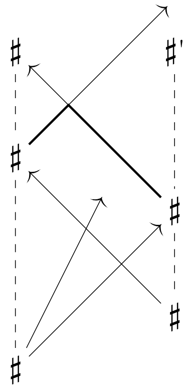


Figure 4.3

Since this L_1 signal creates the top left $\sharp$, this collision should occur necessarily before time t_0 and would create a κ, which is forbidden by hypothesis.

- Or it was created by an l_2/r_2 collision. In this case, the only possible option for the $\sharp$ at t_2 on the left column is a l_1/r_1, else there would be another $\sharp$/sharp' column between the $\sharp$ and the $\sharp'$ at time t_0, contrarily to our hypothesis.

  Therefore, we are in the exact symmetric of the case studied above, where we supposed that the $\sharp$ at t_2 was created by an l_2/r_2, and the left $\sharp$ at t_1 by l_1/r_1. For the same reason, this is a contradiction. □

The other case of creation of a $\sharp$ is the collision between signals l_2 and r_2. But then again, this kind of word is not in the limit language:

Lemma 4.3.13 There is no history diagram containing a word w of the form $\sharp\Sigma\sharp'$, where the left $\sharp$ was created by a signal pair l_2/r_2.

Proof. We use lemma 4.3.12 above to prove this one. To do this, we need to prove the existence of a configuration of the form $\sharp\Sigma\sharp'$ or $\sharp'\Sigma\sharp$, with in any case the $\sharp$ created by an l_1/r_1 pair of signals, in the past of our current configuration.

We begin by proving the existence of another column of $\sharp$, on the left of the left $\sharp$. Since the right column of $\sharp'$ collides with signal $\mathcal{Z}$ (the l_2 signal creating the left $\sharp$), there is necessarily a $\sharp$ in the past of this $\sharp'$, say at time t_1. But then, this $\sharp$ is the one which emitted $\mathcal{Z}$, for else it would have been formed by an l_2/r_2 pair of signals colliding with $\mathcal{Z}$, thus forming another $\sharp$ between the $\sharp$ and the $\sharp'$ of our hypothesis. This shows that $\mathcal{Z}$ was originated by a $\sharp$ in the same column as the right $\sharp'$.

But then, this $\sharp$ must also have emitted an L_1 signal, that cannot collide with the r_2 signal (let us call it $\mathcal{Z}'$), colliding with $\mathcal{Z}$ to form the left $\sharp$. Thus, this signal must be converted into an l_1 before meeting the r_2 of the hypothesis; more precisely, it must collide with an R_1 signal. But this one necessarily has a finite origin, since it could not have crossed $\mathcal{Z}'$. This shows the existence of another column of $\sharp^{(')}$ at t_2 (see figure 4.4):

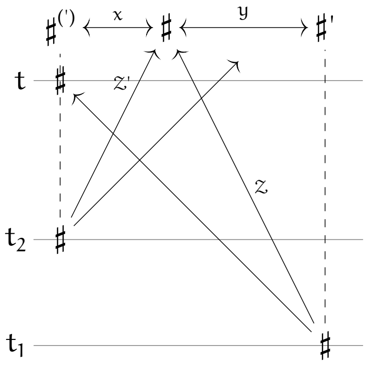


Figure 4.4

□

Lemma 4.3.14 Any configuration of $\mathcal{L}(\Omega_S)$ with at least two $\sharp$s is of the following form, for some value of n: ${}^{\omega}(\sharp B^n)^{\omega}$.

Proof. We first need to show that in a configuration $x \in \mathcal{L}(\Omega_S)$ containing $\sharp$s, there is no history diagram yielding x in which two consecutive $\sharp$s of x were both created by an l_2/r_2 pair, or both by an l_1/r_1 pair. This is the case, since:

- If two consecutive $\sharp$s were created by an l_2/r_2 pair, these signals would have collided before, yielding another $\sharp$ between them.
- If two consecutive $\sharp$s were created by an l_1/r_1 pair, the right r_1 would originate in the left column, and vice-versa (see figure 4.5).

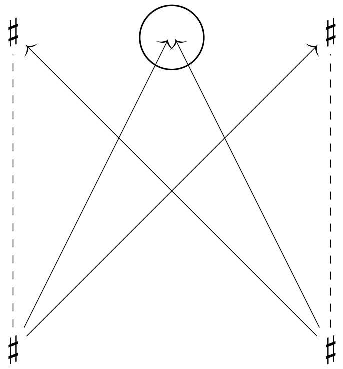


Figure 4.5

These signals would then have been created by $\sharp$s, which also would have emitted r_2s and l_2s, intersecting between the two $\sharp$s in question, thus creating another $\sharp$.

Now, if we have two consecutive $\sharp$s in a configuration, we can infer the position of the third one from the signals having created them, since they have different speeds, and thus must intersect. Then, we can reproduce this argument at the step where these signals were created.

The only problem with this is that the inferred $\sharp$ could be lay on the same side at each step of the argument, thus yielding a configuration with a rightmost $\sharp$ or a leftmost $\sharp$. Without loss of generality, let us assume there is a rightmost $\sharp$ or a leftmost $\sharp$. Then, the past of this cell would be an infinite column of $\sharp$s and $\sharp$'s, with infinitely many $\sharp$s.

Then, it is easy to see that there can be no signal between two consecutive $\sharp$s, for they would come from a desynchronized step in a history diagram of the configuration. □

4.3.1.2 Complexity of the limit set

Lemma 4.3.15 Let L be the language of configurations of Ω_S admitting a history diagram where two $\sharp$s occur at some time t in the past. Then $L \in \text{NL}$.

Proof. Applying lemma 4.3.14 at time t, we know that the configuration at time t is of the form $^{\omega}(\sharp B^{n})^{\omega}$. Moreover, using again the argumentation of lemma 4.3.14, we know that there is some time t' before t in the history, where the configuration is of the form $^{\omega}(\sharp B^{n'})^{\omega}$, with $n' > n$. Hence, L is exactly the language of forward orbits of periodic configurations of the form $^{\omega}(\sharp B^{n})^{\omega}$. It is straightforward to check from the definition of S (see figure 4.1) that the language of periods of such forward orbits is a finite union of languages of the form:

$$\sharp' B^{x_1} A_1 \dots B^{x_i} A_i$$

Where $i \leqslant 4$ (since there are no more than four different signals and points at any time between two $\sharp$s), $A_i \in \Sigma$ and where the x_i satisfy simple linear equations. The lemma follows. □

Lemma 4.3.16 The language of configurations from Ω_S which contain only one state from $\{\sharp, \sharp'\}$ is also in NL.

Proof. Let c be the column containing the state from $\{\sharp, \sharp'\}$. There are several possibilities for the number of $\sharp$s in its past. There may be at most two, for else, two of them would have been created by l_1 signals crossing the R_1/r_1 emitted by the third $\sharp$, and that intersection, circled in the following picture, can occur only once:

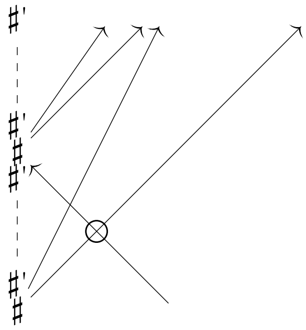


Figure 4.6

Thus, we are left with the following cases:

- Two $\sharp$s in the past: in this case, the r_2 emitted allows to guess the whole column. After the last r_2 signal, there may be signals in the opposite direction, that cannot collide in the past, that is, first l_2s, then L_1s and l_1s. Such configurations are described by the following regular expression, with $x = 2z + y$:

$$^{\omega}\{R_1, r_1, B\}\{r_2, B\}l_1B^xl_2B^yL_1B^zl_2B^z\{\sharp, \sharp'\}B^zr_2B^zR_1B^yr_2B^xr_1\{L_1, l_1, B\}^{\omega}$$

- With only one $\sharp$ in the past, there may still be some l_1 signal somewhere, such as in the following figure 4.7:

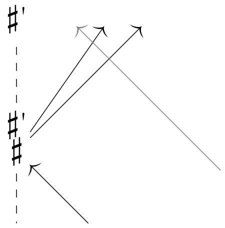


Figure 4.7

Thus, the configurations are of the following form:

$${}^{\omega}\{R_1,r_1,B\}L_1B^z l_2B^z\{\sharp,\sharp'\}B^z r_2B^z R_1\{L_1,l_1,B\}^{\omega}$$

With possibly a r_1 replacing some state on the left, and an l_1 on the right, like on the figure.

- With no $\sharp$ in the past, we have an infinite column of $\sharp'$, thus no signal can come from it. These configurations are of the form:

$${}^{\omega}\{R_1,r_1,B\}\{r_2,B\}\sharp'\{l_2,B\}\{L_1,l_1,B\}^{\omega}$$

The lemma easily follows from the characterization of the different possible forms of configurations discussed above. □

Lemma 4.3.17 Let L be the language of configurations from Ω_S which two or more $\sharp'$, and having a history diagram with no $\sharp$. Then L is regular.

Proof. In a history diagram satisfying the hypothesis, there can be no collision generating a $\sharp$, and even no signal crossing, because this would mean that two signal are going in opposite directions, in the same of a configuration. Thus, at least one would meet a $\sharp'$ in the past, which is forbidden by hypothesis. Therefore, we have:

$$L = \Big(B^+\{R_1,L_1\}\Big)\Big(B^+r_2\Big)(B^+\sharp')\Big(B^+l_2\Big)\Big(B^+\{L_1,l_1\}\Big)^* \qquad \square$$

Lemma 4.3.18 The language of configurations from Ω_S without any $\sharp$ nor $\sharp'$ is also regular.

Proof. Without any $\sharp$ nor $\sharp'$, we are left with blank states and signals. The configurations of the limit set are those configurations in which the possible signals do not have to collide in the past, thus they fall into one of these three possibilities:

$${}^{\omega}\{l_1,r_2,B\}^{\omega}$$

$${}^{\omega}\{r_1,l_2,B\}^{\omega}$$

$${}^{\omega}\{r_1,R_1,B\}\{r_2,B\}\{l_2,B\}\{l_1,L_1,B\}^{\omega} \qquad \square$$

Finally, we can state the following proposition, which does the disjunction of all the cases we have seen until now:

Proposition 4.3.19 The language of the limit set of S is recognizable in logarithmic space on a non-deterministic Turing machine, that is, $\mathcal{L}(\Omega_S) \in$ NL

Proof. There are several cases, and the disjunction on the form of the configurations allows to express the language of Ω_S as a union of "simple" NL languages:

- Configurations with $\sharp$'s or $\sharp$s. We can describe the set of these configurations by the following expression:
$$^{\omega}\{L_1, l_1, B\}\{l_2, B\}A\{r_2, B\}\{R_1, r_1, B\}^{\omega}$$
Where A is one of the following (possibly infinite) configurations:
  1. A has exactly one state from $\{\sharp, \sharp'\}$. We conclude, in this case, with lemma 4.3.16.
  2. A has one $\sharp$, and at least one other $\sharp$ or $\sharp'$. Lemmas 4.3.12, 4.3.13 and 4.3.14 show that the configuration satisfy the hypothesis of Lemma 4.3.15, which allows us to conclude.
  3. A has at least two $\sharp'$, but does not contain any signal. This case is treated by lemma 4.3.17.
  4. A has at least two $\sharp'$, along with some signal(s) between two of them. Denote by c the global configuration in this case. We can simply go back a few steps in the past to find a configuration c' of case 1.2. We can therefore apply lemma 4.3.15 to c.
- The configurations with no $\sharp$s nor $\sharp$'s are treated by lemma 4.3.18

The end of the proof goes by constructing a machine making a non-deterministic choice between all these cases, then doing the computation of the chosen one. By this case analysis, the global machine runs in logarithmic space. □

Simulation $\preccurlyeq_{\trianglelefteq}$ is far from being really understood, but this result gives a strong insight on the difference between the two simulations, by showing the long-term behaviors they imply on universal cellular automata.

Future work

This work asks many more questions than it gives answers. Some of the ideas and open problems presented here were already present in the literature, and we only gave a new light on them. This is the case of open problem 1.2.1, asking if there could exist an automaton intrinsically universal for relation $\trianglelefteq$? Others are relatively new, like the problematic developed in chapter 2, where we ask what algorithmic complexity implies on the dynamics of cellular automata. We conclude this manuscript by giving research directions that seem interesting continuations of the results presented so far, as well as possible generalizations and extensions of this work.

In circuits

The two main open problems that we studied in Chapter 2 are related to the implications of hypotheses on the algorithmic complexity of predicting a cellular automaton on its dynamics. The first question that remains open is whether cellular automata predictable by circuits of constant depth have all their dependencies at a bounded distance of lines. This question can be asked for any complexity class: do the modulo gates of AC circuits help to predict the evolution of cellular automata? The reverse question also holds: Neary and Woods proved in [33] that a particular automaton (elementary rule 110) has a P-complete prediction problem. The question is open for any other class than P-complete problems: what automata simulate circuits of a given depth and size? This seems a really promising research direction, as it gives a dynamical vision to computation, and replaces two obstacles in the way of understanding circuit families and Turing machines; namely, *reductions* and *uniformity*, by the elegant notion of *simulation*, and spatial and temporal *identity*, respectively.

Our second question was about the simulation power reachable by cellular automata with constraints similar to hypotheses of classical theorems in the theory of circuits. The example we studied first was monotonic cellular automata, which are only one level of Post's classification of boolean functions. Post's lattice is a classification of n-ary boolean functions, according to what basic bricks are needed to write them. More precisely, we call a *clone* a set of boolean functions, closed by composition. Post's result gives a classification of all boolean n-ary functions into a countable number of clones, which is a wide extension of Proposition 2.1.1. Applying this to our work, we could ask, for instance, what are the automata with a local rule writable as a composition of max and $\oplus$ gates, which is just one level above the monotone functions in Post's lattice? What about the lower levels of the hierarchy, such as compositions of threshold functions and simple logical operators of fan-in three? Moreover, results in clone theory show that the generalization of Post's classification to sets of more than two elements is much more complicated, and has an uncountable number of classes (see [59] for an introduction). But cellular automata are not arbitrary compositions of functions: the way they are composed is much more constrained. Can the circuits needed to predict them be decomposed into a countable number of clones?

In symbolic dynamics

The generalization of our deterministic approach, that we presented in chapter 4, asks an interesting question: are these all-powerful complicated communication models with non-determinism and randomization really needed? The answer is that we do not know, and it would be really interesting to prove that deterministic communication complexity cannot be applied to relation $\trianglelefteq$. This would give a useful insight on the differences between the two models of simulation we have presented in this thesis: colorings and sub-automata, and would allow to understand why we fail to find an intrinsically universal cellular automaton for relation $\trianglelefteq$.

In chapter 3, we quickly showed how our approach could be generalized to two dimensions. Classically, the favored non-deterministic two-dimensional objects are tilings. What do our results mean for tilings? Are they amenable to the same kind of analysis? The first basic extension that we can envision in this direction is to find a method to show that a given tileset cannot generate arbitrarily complicated tilings.

Another possible extension would be to bound the difficulty of computing a tiling *in parallel*, before assembling all the parts. For instance, if the communication complexity of a certain tileset is bounded by a constant d, then tiling an arbitrarily large rectangle could be done by computing the parts in parallel: each processor could compute its part for all the possible executions of the protocol, since there are at most 2^c of them.

In small machines and experiments

The field of *experimental theoretical computer science* is quite new, but seems really promising. Its goals can be to disprove, by experimental evidence, conjectures about large sets of data; this is similar to the role of experiments in physics. Or to suggest unexpected behaviors. The importance of this idea is also justified by results such as *natural proofs*, predicting that proofs of complexity must be complicated or applicable to a very small number of functions.

A natural extension to our work would be to find an algorithm to compute an optimal, or approximately optimal, partition of the matrices given by protocols for relations. This is a variant of communication complexity, in which Alice is given an element of $f^{-1}(\{0\})$, Bob is given an element of $f^{-1}(\{1\})$, for some $f : \{0,1\}^n \rightarrow \{0,1\}$. They are asked to compute an index where their configurations differ. If they have a circuit C computing f, they can use it to find a protocol of cost the depth of C. And the correspondence between monochromatic rectangles and communication complexity still holds. Hence, if we can compute such a partition experimentally, this would give a lower bound for the small input sizes. Natural examples that seem interesting, for this approach, include the two elementary rules 30 and 110.

In communication complexity of cellular automata

Several questions are asked by our work on the communication complexity of cellular automata. First, we have not managed to find any lower bound for a natural example. We highly suspect that automata such as elementary rule 30 do not have protocols of less than maximal complexity, but they resisted to all our proof attempts. The result of Neary and Woods [33] might make it possible to construct a fooling set of size $2^{P(n)}$, with $P(n) \geqslant n^{1/k}$ for some integer k, for rule 110, but a general method is yet to be found.

Then, our results using distributional communication complexity (Proposition) open the way to a study of statistical properties on the inputs of cellular automata, in relation with their simulation power. This would be complement elegantly the approach of [60] about stochastic cellular automata.

Also, studying the converse of Proposition 4.1.2 could be interesting. Our conjecture is that, to any simulation ideal without intrinsically universal cellular automaton, corresponds a communication problem easy for all the automata of the ideal, and hard for at least one cellular automaton.

In general

Finally, we believe that our approach is applicable to from other theories than the theory of computation. The most tempting extension would be algorithmic game theory, and economics, to show for instance that a certain *type of behavior* is impossible in a game, or on the contrary that given such and such a law, another type of behavior is frequently observed in a model of economics. The study of information transmission is by the way one of the key elements of today's economic science, as the works of Stiglitz [42] show.

Bibliography

[1] John von Neumann, The Theory of Self-reproducing Automata, 1966, University of Illinois Press, A. Burks (ed.)

[2] Gustav Arnold Hedlund, Endomorphisms and Automorphisms of the Shift Dynamical Systems, in: Mathematical Systems Theory, volume 3 (1969), 320-375

[3] Alan Cobham, The intrinsic computational difficulty of functions, 1964, 24-30

[4] Stephen A. Cook, The Complexity of Theorem-Proving Procedures, in: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA, 1971, ACM, Michael A. Harrison, Ranan B. Banerji, Jeffrey D. Ullman (eds.), 151-158

[5] Leonid Levin, Универсальные задачи перебора (Universal search problems), in: Проблемы передачи информации (Problems of Information Transmission), volume 9 (1973), 265–266

[6] Ryan Williams, Non-uniform ACC Circuit Lower Bounds, in: Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, June 8-10, 2011, 2011, IEEE Computer Society, 115-125

[7] Alexander A. Razborov, Steven Rudich, Natural proofs, in: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada, 1994, ACM, Frank Thomson Leighton, Michael T. Goodrich (eds.), 204-213

[8] Edward Forrest Moore, Machine models of self-reproduction, in: Proceedings of Symposia in Applied Mathematics, volume 14 (1962), American Mathematical Society, 17-33

[9] Guillaume Theyssier, Marianne Delorme, Jacques Mazoyer, Nicolas Ollinger, Bulking I: An abstract theory of bulking, in: Theoretical Computer Science, volume 412 (2011), 3866-3880

[10] Guillaume Theyssier, Marianne Delorme, Jacques Mazoyer, Nicolas Ollinger, Bulking II: Classifications of cellular automata, in: Theoretical Computer Science, volume 412 (2011), 3881-3905

[11] Edwin Roger Banks, Universality in cellular automata, in: Foundations of Computer Science, IEEE Annual Symposium on, volume 0 (1970), IEEE Computer Society, 194-215

[12] Andrew Chi-Chih Yao, Some Complexity Questions Related to Distributive Computing (Preliminary Report), in: Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA, 1979, ACM, Michael J. Fischer, Richard A. DeMillo, Nancy A. Lynch, Walter A. Burkhard, Alfred V. Aho (eds.), 209-213

[13] C. Shannon, A mathematical theory of communication, in: Bell system technical journal, volume 27 (1948)

[14] Ivan Rapaport, Guillaume Theyssier, Christoph Dürr, Cellular automata and communication complexity, in: TCS, volume 322 (2004), 355-368

[15] E. Post, The Two-valued Iterative Systems of Mathematical Logic. Princeton, Princeton University Press, 1941, 1965

[16] Ivan Rapaport, Jacques Mazoyer, Inducing an order on cellular automata by a grouping operation, in: Discrete Applied Mathematics, 1999, 177-196

[17] Nicolas Ollinger, Automates cellulaires : structures, 2002, PhD. thesis

[18] Guillaume Theyssier, Cellular automata : a model of complexities, 2005, PhD. thesis

[19] Nicolas Ollinger, Gaétan Richard, Four states are enough!, in: Theoretical Computer Science, volume 412 (2011), 22-32

[20] John R. Myhill, The converse of Moore's Garden-of-Eden theorem, in: Proceedings of the American Mathematical Society, volume 14 (1963), 204-205

[21] Serafino Amoroso, Yale N. Patt, Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tessellation Structures, in: Journal of Computer and System Sciences, volume 6 (1972), 448-464

[22] François Blanchard, Pierre Tisseur, Some Properties of Cellular Automata with Equicontinuity Points, 2000

[23] Jérôme Olivier Durand-Lose, Intrinsic Universality of a 1-Dimensional Reversible Cellular Automaton, in: STACS 97, 14th Annual Symposium on Theoretical Aspects of Computer Science, Lübeck, Germany, February 27 - March 1, 1997, Proceedings, volume 1200 (1997), Springer, Rüdiger Reischuk, Michel Morvan (eds.), 439-450

[24] Jarkko Kari, Anahí Gajardo, Andrés Moreira, On time-symmetry in cellular automata, in: Journal of Computer and System Sciences, volume 78 (2012), 1115-1126

[25] Petr Kůrka, Topological dynamics of cellular automata, 2009, Springer, 9246-9268

[26] Cristopher Moore, Predicting nonlinear cellular automata quickly by decomposing them into linear ones, in: Physica D: Nonlinear Phenomena, volume 111 (1998), 27 - 41

[27] Johannes Gütschow, Vincent Nesme, Reinhard F. Werner, Self-similarity of Cellular Automata on Abelian Groups, in: J. Cellular Automata, volume 7 (2012), 83-113

[28] Marcus Pivato, Reem Yassawi, The spatial structure of odometers in certain cellular automata, in: First Symposium on Cellular Automata Journ'ees Automates Cellulaires (JAC 2008), Uz`es, France, April 21-25, 2008. Proceedings, 2009, MCCME Publishing House, Moscow, Bruno Durand (ed.), 119-129

[29] A. Turing, On Computable Numbers, with an application to the Entscheidungsproblem, in: Proceedings of the London Mathematical Society, volume 42 (1936), 230-265

[30] Karel Culik II, Sheng Yu, Jan K. Pachl, On the Limit Sets of Cellular Automata, in: SIAM Journal of Computing, volume 18 (1989), 831-842

[31] Jarkko Kari, Tiling Problem and Undecidability in Cellular Automata, in: Encyclopedia of Complexity and Systems Science, 2009, Springer, Robert A. Meyers (ed.), 9158-9172

[32] Matthew Cook, Universality in elementary cellular automata, in: Complex Systems, volume 15 (2004), 1-40

[33] Turlough Neary, Damien Woods, P-completeness of Cellular Automaton Rule 110, in: Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part I, volume 4051 (2006), Springer, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, Ingo Wegener (eds.), 132-143

[34] Bruno Poizat, Les petits cailloux, Nur al-Mantiq wal-Ma'rifah, Aléas Editeur (ed.)

[35] Ingo Wegener, The complexity of Boolean functions, 1987, Wiley-Teubner

[36] Eric Goles Ch., Pierre-Etienne Meunier, Ivan Rapaport, Guillaume Theyssier, Communication complexity and intrinsic universality in cellular automata, in: Theoretical Computer Science, volume 412 (2011), 2-21

[37] Eric Goles Ch., Pierre-Etienne Meunier, Ivan Rapaport, Guillaume Theyssier, Erratum to: Communication Complexity and Intrinsic Universality in Cellular Automata [Theoretical Computer Science 412 (1-2) (2011) 2-21], in: Theoretical Computer Science, volume 412 (2011), 7169-7170

[38] R. Raz, A. Wigderson, Monotone circuits for matching require linear depth, in: J. ACM, volume 39 (7/1992), ACM, 736-744

[39] Sven Skyum, David A. Mix Barrington, Chi-jen Lu, Peter Bro Miltersen, On Monotone Planar Circuits, 1999

[40] Jarkko Kari, The Nilpotency Problem of One-Dimensional Cellular Automata, in: SIAM J. Comput., volume 21 (1992), 571-586

[41] Friedrich August Hayek, The Use of Knowledge in Society, in: The American Economic Review, volume 35(4) (1945), 519-530

[42] Joseph Eugene Stiglitz, Information and the change in paradigm in Economics, in: The American Economic Review, volume 92(3) (6/2002), 460-501

[43] Eyal Kushilevitz, Noam Nisan, Communication complexity, 1997, Cambridge university press

[44] Richard J. Lipton, Robert Sedgewick, Lower Bounds for VLSI, in: Proceedings of the 13th Annual ACM Symposium on Theory of Computing, May 11-13, 1981, Milwaukee, Wisconsin, USA, 1981, ACM, 300-307

[45] Juraj Hromkovič, Georg Schnitger, Communication Complexity and Sequential Computation, 1997, Springer-Verlag, 71-84

[46] Mathieu Sablik, Directional dynamics for cellular automata: A sensitivity to initial condition approach, in: TCS, volume 400 (2008), 1-18

[47] P. Kurka, Languages, equicontinuity and attractors in cellular automata, in: Ergodic theory and dynamical systems, volume 17 (1997), 417-433

[48] Pierre-Etienne Meunier, Raimundo Briceño, The structure of communication problems in cellular automata, in: 17th International Workshop on Cellular Automata and Discrete Complex Systems, Automata 2011. Proceedings, 2011, DMTCS, Nazim Fatès, Eric Goles, Alejandro Maass, Iván Rapaport (eds.)

[49] Eric Goles Ch., Ivan Rapaport, Pierre Guillon, Traced communication complexity of cellular automata, in: CoRR, volume abs/1102.3522 (2011)

[50] Marvin L. Minsky, Computation: Finite and Infinite Machines, 1967, Prentice-Hall

[51] Gregory Lafitte, Christophe Papazian, The fabric of small Turing machines, in: Computation and Logic in the Real World, Third Conference on Computability in Europe, 2007, Local Proceedings, 219-227

[52] Eyal Kushilevitz, Benny Applebaum, Yuval Ishai, Cryptography by Cellular Automata or How Fast Can Complexity Emerge in Nature?, in: Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings, 2010, Tsinghua University Press, Andrew Chi-Chih Yao (ed.), 1-19

[53] Lyman P. Hurd, Nonrecursive Cellular Automata Invariant Sets, in: Complex Systems, volume 4 (1990), 131-138

[54] Pierre-Etienne Meunier, Guillaume Theyssier, Pierre Guillon, Clandestine Simulations in Cellular Automata, in: Second Symposium on Cellular Automata Journées Automates Cellulaires, JAC 2010, Turku, Finland, December 15-17, 2010. Proceedings, 2010, Turku Center for Computer Science, Jarkko Kari (ed.), 133-144

[55] A. Yao, Lower bounds by probabilistic arguments, 1983, IEEE Computer Society, 420--428

[56] Vijay V. Vazirani, Approximation algorithms, 2001, Springer, I-IXI, 1-378

[57] Douglas Lind, Brian Marcus, An Introduction to Symbolic Dynamics and Coding, 1995, Cambridge

[58] Jarkko Kari, Rice's theorem for the limit sets of cellular automata, in: TCS, volume 127 (1994), Elsevier, 229-254

[59] D. Lau, Function Algebras on Finite Sets: Basic Course on Many-Valued Logic and Clone Theory (Springer Monographs in Mathematics), 2006, Springer-Verlag New York, Inc.

[60] Damien Regnault, Nicolas Schabanel, Eric Thierry, Progresses in the analysis of stochastic 2D cellular automata: A study of asynchronous 2D minority, in: Theor. Comput. Sci., volume 410 (2009), 4844-4855