Obtaining tilings by self-assembly and self-stabilization

Damien Regnault

École doctorale Stic *(Sciences et Technologies de l'Information et de la Communication)*s

HABILITATION A DIRIGER DES RECHERCHES

Spécialité: Informatique

Présentée et soutenue publiquement le 8 novembre 2023

par Damien REGNAULT

Formation de pavages par auto-assemblage et auto-stabilisation

---

Tuteur: Éric Rémila, Professeur émérite, université de Saint-Étienne

Membres du jury:

Rapporteurs:	*Emmanuel Jeandel*	*Université de Lorraine*
	Jarkko Kari	*University of Turku*
	Matthew J. Patitz	*University of Arkansas*
Examinateurs:	*Valérie Berthé*	*Université Paris-Cité*
	Johanne Cohen	*Université Paris-Saclay*
	Pierre Fraigniaud	*Université Paris-Cité*

Laboratoire Informatique, BioInformatique, Systèmes Complexes EA 4528
Institut de Biologie Génétique et Bioinformatique (IBGBI), 23 boulevard de France, 91037 Évry-Courcouronnes.

À Amélie et Julien

Remerciements

Lors de ma carrière universitaire, j'ai eu la chance de pouvoir rencontrer de nombreux chercheurs, ce qui a abouti à des collaborations enrichissantes. Je souhaiterai donc tout d'abord remercier mes nombreux co-auteurs sans qui ce document serait beaucoup moins fourni. Malheureusement, je ne suis pas doué pour garder contact mais je n'ai oublié aucun d'entre eux. Parmi ceux-ci, j'ai une pensée particulière envers Nicolas Schabanel qui a continué à me conseiller même après ma thèse et qui m'a transmis sa curiosité. Je suis heureux que nous soyons finalement amenés à retravailler ensemble. Il y aussi Sylvain Sené, mon meilleur ami rencontré sur les bancs de l'ENS Lyon. Quant à Pierre-Étienne Meunier et Damien Woods, leur persévérance nous a permis de résoudre la conjecture de la température 1. Il nous a fallu six ans pour résoudre cette conjecture. Je m'excuse auprès de toutes les personnes que j'ai lassées à force de répéter que cela sera fini le mois prochain (en particulier ma femme). Durant ces années, j'ai aussi eu la chance d'encadrer Sébastien Morais, avec l'aide Éric Angel et de Franck Ledoux, qui a heureusement fait preuve d'une maturité et d'une autonomie exceptionnelle. Enfin, je remercie Éric Rémila que j'ai eu comme professeur, collègue, co-auteur et qui a accepté d'être mon tuteur pour ce manuscript.

Je tiens aussi à remercier les membres du jury d'avoir accepté de consacrer de leur temps à la lecture de ce document et Nicole Bidoit, la conseillère HDR de l'école doctorale Stic de l'université Paris-Saclay qui m'a suivi durant la rédaction de ce document.

Je remercie l'ancienne équipe Escape du Lif à Marseille en particulier Thomas Fernique et Victor Poupet ainsi que Vincent Bernardi et Jérémie Chalopin qui m'ont accueilli chaleureusement à Marseille lors de mon année d'ATER durant une période personnellement difficile et qui m'ont grandement aidé à rebondir. Je remercie les membres du laboratoire Ibisc d'avoir suffisamment cru en moi pour me donner ma chance et d'avoir réussi à créer un lieu de travail accueillant. Je termine par Violette Ruggeri qui m'a épaulée pour gérer la L2.

Bien entendu, je tiens à remercier ma femme Amélie et mon fils Julien, j'ai une chance incroyable de vous avoir dans ma vie.

Contents

Chapter 1

Introduction

This document presents my works since I obtained a PhD in 2008 (my curriculum vitae is available in appendix A). Only the bibliographies, key ideas, main results and theirs implications and perspectives are presented here. Detailed proofs are available in the different published articles.

This document mainly focusses on two series of articles about models of computation based on sets of dominoes. These models are of interest into two different domains of nano-technology: self-assembly and crystallography. The introduction summarizes some fundamental notions, introduces the different models, theirs similarities and differences. Then, Chapter 2 is dedicated to self-assembly while Chapter 3 deals with crystallography. Finally, Chapter 4 is an activity report about my other works (algorithmics, cellular automata and boolean automata networks) and perspectives.

1.1 Models of computation

In language theory, a specific finite set Σ is called an *alphabet* whose elements are called *letter*. A *word* is a finite sequence of letters and Σ^* is the set of all finite words. A *language* L is a subset of Σ^* (possibly finite or infinite). The language L can also be interpreted as a function $f : \Sigma^* \to \{0, 1\}$ such that $L = \{w \in \Sigma^* : f(w) = 1\}$ and a *model of computation* aims to describe f as a finite composition of several simple functions. These simple functions aim to be easily implementable by real life mechanisms: for example any boolean function $\{0, 1\}^n \to \{0, 1\}$ with $n \geq 0$ can be decomposed into a finite amount of And and Not gates and such gates can be implemented in real life using

diodes and transistors. A model of computation defines the simple functions and the rules of composition, *i.e.* how to do a computation and how to link them. A *program* describes how to use these functions and rules in order to *recognize* one specific language L. A *run* of the program on a given word x is the sequence of computations done to determine if x belongs to L [1]. Finally, implementing a program by a real-life mechanism provides a machine which recognizes the language L, *i.e.* which is able to test if a word x belongs to L automatically.

Nevertheless, this approach has some limits. By definition, a language is a countable set but for some countable sets, there is no program to recognize them. Indeed, the Church–Turing thesis defines a limit of what can be achieved by a "real-life" machine, called the class of *decidable* languages. Several models of computation are able to recognize the languages of this class: Turing machines, RAM machines, cellular automata, boolean automata networks, Wang tiles, abstract Tile Assembly Model, ... [2] It is possible to build real-life mechanisms which are able to simulate a computation done by these models but two limits appear during the run of a program: the time constraint (the number of successive computations needed to obtain the result) and the space constraint (the number of mechanisms needed to simulate the computation). After running for millions of years, some program may stop since the memory space of the computer is not large enough to finish the computation. Thus modern computers are weaker than their mathematical abstraction. Even if computers become faster and if the size of their memory storages increases, their theoretical limit stays the same: they are only able to compute decidable languages. Note that building a stronger model of computation is possible theoretically but its implementation in real life would fall under science fiction.

A language which is not decidable is *undecidable* and Rice's theorem stipulates that any non trivial question about programs of a model of computation which recognizes the decidable languages is undecidable. There is no program to determine if another program will stop, if another program is optimal, if two other programs compute the same language, ...

[1] For a Turing Machine, one of the most well-known model of computation, a program is called a transition function and a run is the sequence of configurations (internal state and tapes).

[2] Among these models of computation, Turing machines are often used in theoretical works to study decidable languages whereas RAM machines is an abstraction of modern computers.

The field of complexity focus on some sub-classes of the decidable language such as P, NP, PSPACE, ... Here, we are interested in a class called the *regular* languages. These languages are recognized by finite automata, a model of computation which is weaker than the ones previously cited. Interestingly, non-trivial questions about finite automata are decidable: there exist programs which are able to test if two finite automata recognize the same language, which are able to optimize a finite automaton (to minimize its number of states, to determinize it, ...). Moreover, a regular language admits other representations besides a finite automaton: regular expression, regular grammar. These representations are useful depending on the context of use: finite automata provide an easy test to determine if a word belongs to a language, regular expressions provide a more readable and easy way to define a regular language, grammars provide a parse tree which is useful for further analysis of the word. Programs are also able to transform one of these representations into another one. Then, depending on the context, programs are able to compute the best and optimal representation of a regular language. Thus, this weaker model of computation is still commonly used since it is possible to optimize automatically its programs.

More formally, the finite automata are defined as follows (see Figure 1.1): consider a quintuplet $A = (Q, \Sigma, \delta, q_0, q_f)$ where Q is a finite set of states, Σ is the alphabet, $\delta : (Q \times \Sigma) \to Q$ is the transition function and q_0 and q_f are two states of Q. For a word $w \in \Sigma^*$ and a letter $a \in \Sigma$, we define $\delta(q, wa)$ as $\delta(\delta(q, w), a)$ and w is accepted by the finite automata if and only if $\delta(q_0, w) = q_f$. As stated before, finite automata are not able to recognize all decidable languages. For example, the decidable language $\{0^n 1^n : n \geq 0\}$ defined over the alphabet $\{0, 1\}$ cannot be recognized by a finite automaton. By contradiction, if a finite automaton $(Q, \Sigma, \delta, q_0, q_f)$ recognizes $\{0^n 1^n : n \geq 0\}$, then it must accept $0^n 1^n$ for some $n > |Q|$ and the finite automaton will *loop* along its run: there exist $0 \leq i < j \leq n$ such that $\delta(q_0, 0^i) = \delta(q_0, 0^j)$ and for all $k \geq -1$, the words $0^{n+k(j-i)}1^n$ should be accepted by the finite automaton which is a contradiction. This reasoning was generalized into a result known as the *pumping lemma*: any word long enough accepted by a finite automaton will generate an infinity of "*pumped*" words which are also accepted by the finite automaton (see Figure 1.1). The pumping lemma is a strong tool used to prove that some decidable languages are not regular and its proof relies only on a simple combinatorial argument.

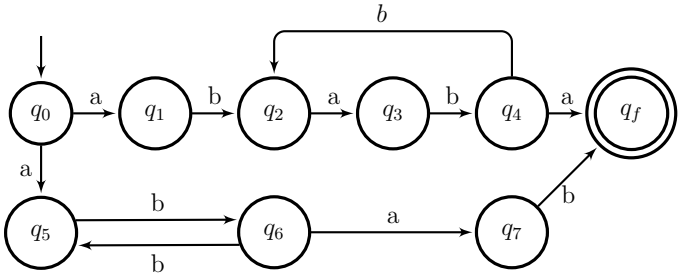


Figure 1.1: A finite automaton made of nine states and eleven transitions over the alphabet $\Sigma = \{a, b\}$. Note that $\delta(ab) = \delta(ababb) = q_2$, there is a loop and the two words ab and $ababb$ end in the same states. This finite automaton recognizes the regular language $\{abab(bab)^*a + ab(bb)^*ab\}$

1.2 Dominoes

A *domino* is a square characterized by *glues*, on each of its four sides. Any regular language L can be transformed into an equivalent set of dominoes (see Figure 1.2). Indeed, consider a finite automaton $A = (Q, \Sigma, \delta, q_0, q_f)$ which recognizes L. Each transition $\delta(q_i, a) = q_j$ of A is transformed into a domino labeled by the letter a where the left (resp. right) glue is q_i (resp. q_j). Then two specials dominoes are added: one, called the *seed*, with only one glue q_0 on its right side and one with only one glue q_f on its left side. Initially, only the seed is put on the $2D$ grid and then copies of dominoes are placed on the $2D$ grid to the right of the previous one such that the glues on their abuttal sides are identical. The dominoes on the grid form an *assembly* which grows each time another one is added, we say that the new domino *binds* with the assembly. At all times, only the right glue of the rightmost domino of the assembly is *free*, *i.e.* this glue does not bind with a glue of another domino. Consider a word $w \in L$, the sequence of transitions of A which leads from the state q_0 to q_f can be simulated by the dominoes: for each transition of the sequence, the corresponding domino binds with the assembly. Along the simulation, the free glue of the assembly corresponds to the state of the automata. At the end, the glue q_f is free and the last domino with only q_f on its left side binds with the assembly. Then, there is no more free glue and the assembly becomes *terminal*. The dominoes of the terminal assembly are labelled by the word w. Thus, there is an equivalence between the words of L and the labels of the finite terminal assemblies obtained with

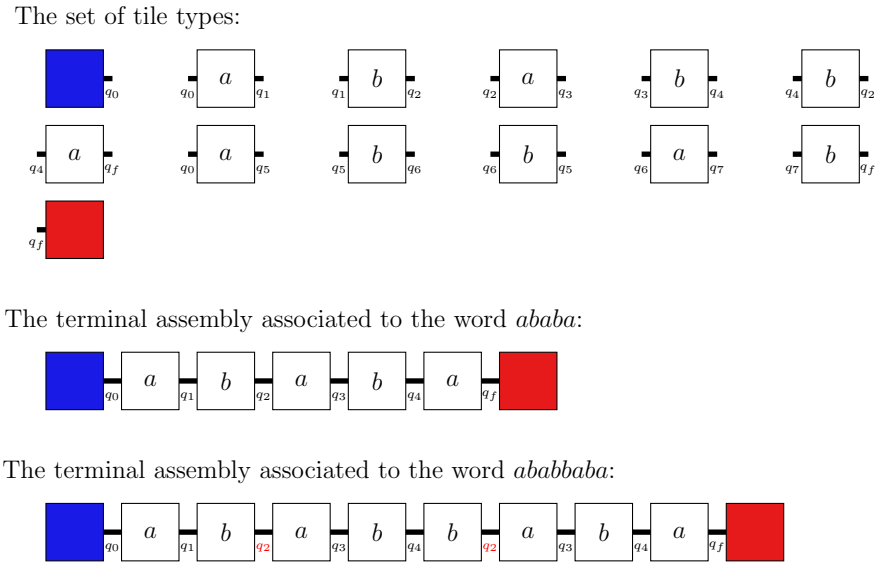


Figure 1.2: A set of dominos generated from the finite automaton of Figure 1.1. We represent the terminal assemblies corresponding to the words *ababa* and *ababbaba*. Note that for the second terminal assembly, the glue q_2 (in red) appears two times and the tiles between these two glues can be pumped.

this set of dominoes.

Dominoes are able to simulate finite automata but they can do more. Indeed, in this transformation the dominoes form a line and by adding glues on the north and south sides we obtain a new $2D$ model of computation. Introduced by H. Wang in [47], dominoes were initially defined with a glue on each of their four sides. In this case, the assembly grows until it covers the whole $2D$ grid. Nevertheless, while growing the assembly can create *mismatch*: a new domino can bind with a domino of the assembly while, at the same time, one of its glue is different from the glue of a neighboring tile of the assembly. Initially, H. Wang was interested in solving a logical problem and he needed to avoid any mismatch. In this case, a terminal assemblies covers the $2D$ grid without any mismatch and is called a *tiling*. This model became known as *Wang tiles*.

A natural question known as the *domino problem* was to determine if a finite set of dominoes is able to assemble at least one tiling. Initially, H. Wang conjectured that this problem was simple, *i.e.* decidable. Indeed, a first intuition was to do a reasoning similar to the pumping lemma. By

starting from a single domino, new dominoes can bind on a line until the same glue appears two times. Then, the pattern between the two glues can be extended into a periodic horizontal line. By doing the same reasoning on columns, either we obtain a mismatch or we found one bi-periodic tiling which can be described by a finite rectangle. Nevertheless in [2], R. Berger has shown that this conjecture was false and that the domino problem is undecidable.

One of the key step to prove this result was to encode a Turing machine into a set of Wang tiles. Then, Wang tiles is a model of computation which can compute the class of decidable language. A set of dominoes can be seen as a program and binding the dominoes together is a run of this program. Nevertheless, there is a major problem with this interpretation. Indeed, the previous reasoning and the result of R. Berger imply that some tilings cannot be assembled carelessly otherwise a mismatch may appear. In other words, there exist tilings such that the local constraints imposed by the glues are not sufficient to encode how to assemble them. This was not a problem for H. Wang which introduced this model to solve a theoretical problem. However, this remark became problematic when practical applications of Wang tiles appeared in crystallography and self-assembly.

1.3 Self-assembly and aTam

E. Winfree introduced in [48] a computation model called *abstract tile assembly model* (aTAM)[3]. This model adds several modifications to Wang tiles. Firstly, the notion of assembly and binding are formally defined in $aTAM$ in order to model the computations done during a run of a program. Secondly, mismatches are authorized. Thirdly, some dominoes may not have glues on some of their sides which implies that a terminal assembly may be finite. Fourthly, some tiles are initially puts on the $2D$ grid before computations (*i.e.* bindings) start, these tiles form an assembly called the *seed*. By considering these modifications, we obtain the model of *non-cooperative* aTAM or aTAM at temperature 1. The temperature is a parameter introduced by E. Winfree. When it is greater than 2, the model is called *cooperative* aTAM and is able to simulate Turing machine similarly as Wang tiles. This document focuses on non-cooperative aTAM and investigates which kind of

[3]Formal definition of aTAM is available in chapter 2.1

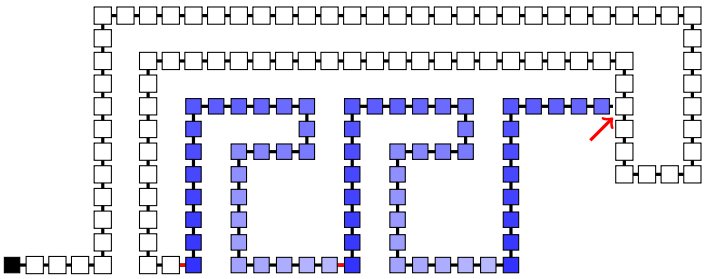


Figure 1.3: An assembly where the seed is in black. The start of the assembly is in white. We suppose that the two glues in red are identical. Thus it is possible to try to pump the blue tiles between them. In this example, this pumping fails since a collision occurs with the start of the assembly. Here, this collision creates a mismatch: two neighboring tiles do not have the same glue on their abuttal side.

computation this model can achieve.

At first glance, non-cooperative aTAM does not seems far more expressive than its $1D$ counterpart. One key question is to develop a formalism to describe efficiently the set of $2D$ shapes assembled by a set of dominoes. Then, the same kind of results could be expected as for finite automata: developing programs which are able to find a set of dominoes whose terminal assemblies are the $2D$ shapes described by this formalism (or conversely), developing a program which is able to optimize the number of dominoes, ... To show that such questions are decidable, a reasonable approach is to proceed like the reasoning done for finite automata: to prove a kind of $2D$ pumping lemma and to use it to show that some $2D$ shapes cannot be assembled by aTAM. Surprisingly, all the attempts to show a $2D$ pumping lemma relying on a simple combinatorial arguments have failed. A new phenomenon, called *collision*, appears due to the $2D$ grid: some previous dominoes may block the growth of an infinite periodic path as shown in Figure 1.3.

Until recently, few results were known about noncooperative aTAM. In chapter 2, we present three recent developments about the limits and possibilities of this model of computation. Firstly, about the decidabillity of this model, we present a $2D$ pumping lemma which strongly hints that this model cannot simulate Turing machine. Secondly, this result allows to conclude that *directed* non cooperative aTAM (a special case where there is only one terminal assembly) is decidable. Thirdly, about the complexity of this

model, we exhibit a set of dominoes of size n whose terminal assemblies are all finite, bounded and always contain the same path of length $\Omega(n \log n)$. Such a path is called *efficient* and its existence implies that noncooperative aTAM is able to achieve computations relying on collision which cannot be done by finite automata. This result also hints that the non-directed case of non-cooperative aTAM may be more powerful than the directed one.

E. Winfree introduced aTAM for practical application in self-assembly. Indeed, it is possible to build DNA tiles which are expected to behave as the dominoes of cooperative aTAM. These experiments (see Chapter 2.1) were successful and were followed by theoretical and/or experimental studies. For non-cooperative aTAM, collisions are challenging to reproduce experimentally.

1.4 Crystallography and flip

In order to solve the domino problem, R. Berger had to encode computations done by a Turing machine using Wang tiles. The classical transformation of a run of a Turing machine into in a set of Wang tiles relies on the following idea: the column t of the tiling is used to represent the configuration of the Turing machine after t time steps. Nevertheless, the Turing machine uses only a finite amount of its available space (which is infinite). Then, one domino should be able to fill the space which is unused by the simulation, *i.e.* this domino is able to tile an infinite part of the $2D$ grid. Then, such a domino is also capable to tile the whole grid to create a biperiodic tiling. In such a case, the simulation does not start [4] and this transformation is not sufficient to conclude: this set of dominoes allows to assemble a non-periodic tiling but also a biperiodic one.

R. Berger had to encode a mechanism into his set of Wangs tile to force the simulation to start whatever the seed is. In other words, no finite assembly is able to tile an arbitrarily large area of the $2D$ grid periodically. Such a tiling is called *aperiodic*. Following the work of R. Berger, aperiodicity became a subject of research of its own with the aim to simplify the construction of R. Berger. This result was achieved in 2015 with a set of eleven Wang tiles and four glues developed by E. Jeandel and M. Rao [23]. Aperiodicity was also studied when dominoes are not restricted to square, the most famous

[4] In aTAM, the introduction of the seed allows to avoid this problem.

one is the Penrose tilings.

Several years after the work of R. Berger, aperiodicity became relevant in crystallography. In physics, a crystal is a periodic structure of organized atoms. Physicists believed that any organized structure had to be periodic. In 1982, D. Shechtman et al [45] exhibited a structure where atoms are organized without any symmetries: a *quasi-crystal*. Tilings became an abstract model of crystallography: dominoes represent the atoms and the glues represent the interactions between the atoms. Aperiodic tilings explain the existence of quasi-crystals. Therefore, understanding how to assemble an aperiodic tiling using only local constraints gained a new interest.

Manufacturing a quasi-crystal is challenging and is generally done by cooling: at high temperature the interactions between the atoms are negligible but they become progressively more important as the temperature decreases. In chapter 3, we present a stochastic process using flips (permutation of neighboring dominoes) to model the cooling of a crystal and analyze it. We show that our model manage to explain the formation of a crystal but it fails to model the formation of quasi-crystal if the parameters are too simple. Finding a model which can explain the formation of a quasi-crystal is still open.

Chapter 2

Self-assembly

This chapter is a summary of three articles [30, 31, 32]. The bibliography, definitions and theorems come from these articles and no new result is presented here. This document aims to harmonize notation, to regroup the results, to present the open questions and to sketch the main proofs and their key ideas.

2.1 Introduction

Self-assembly is the process by which independent, unsynchronised components coalesce into complex forms and patterns, using geometry and local constraints to exchange information, and perform different sorts of computations. In particular, self-assembly is the process by which molecules, and in particular biomolecules, acquire their shape (and therefore their function).

A computational theory of self-assembly has a wealth of applications in a large range of fields and scales. At the molecular level, programming molecules would enable us to interact with living organisms, potentially defeating the geometric strategies used by nasty viruses to penetrate cells. Smart materials with new properties such as self-reproduction and self-repairing are another example. At a much larger scale, industrial processes could also benefit from a better understanding of self-assembly, as it could streamline processes and make industrial robots simpler.

This theory has already yielded experimental realisations such as DNA Origami [41], allowing anyone to make their own molecules of any prescribed shape of a diameter between 10nm and 500nm. DNA Self-Assembly has

also been used to build fractal shapes [43], information retrieval circuits [36], cyclic machines using DNA as machine material *and* as fuel [49]. Another recent application has been the amplification of minuscule concentrations of a molecular compound in solution, by using it as a "seed" for self-assembling large structures [34]. DNA storage [6] has also been proposed and implemented as a technique to store a tremendous amount of information in a tiny space, with millions of years of potential durability.

These developments have happened in parallel to, and with interactions with work on the computer science theory of tile assembly. The most studied model in that direction is the abstract Tile Assembly Model (aTAM), created by Winfree [48, 42] with inspiration from Wang tilings [47]. This model studies assemblies made of *square tiles* with colours on their borders. Using a finite set of tile *types*, and an assumed infinite supply of each type, the assembly process starts with an initial "seed" assembly, and proceeds nondeterministically and asynchronously, one tile at a time. Unlike Wang tilings, which is mostly concerned with (potentially undecidable) full covers of the plane, the abstract Tile Assembly Model studies the *assembly sequence* of an assembly, which is the sequence of binding events necessary to build a shape.

In the fully general abstract Tile Assembly Model, tile borders have a *glue strength* on their border, and the model has a global assembly threshold called the "*temperature*": in order to remain stably attached, the sum of glue strengths on the attached borders of a tile must be at least equal to the temperature. One of the key complexity measures of this model is *program-size complexity*, meaning the number of tile types in the tileset. The fact that this model can simulate Turing machines has been used to encode complex shapes with a number of tile types logarithmic in the Kolmogorov complexity of the shapes [46]. Moreover, the aTAM model is also *intrinsically universal*, meaning that there is a single finite "universal" tileset capable of simulating any other tileset up to a constant scaling factor [11]. Over the years, a number of consequences and extensions of that result have also been studied [12, 9, 10], and intrinsic universality has also been used to classify models according to their simulation power [28].

2.1.1 Noncooperative self-assembly

Noncooperative self-assembly is a restriction of the aTAM to a temperature of 1, meaning that tiles always attach to an existing assembly as soon as

at least one side has its colour matching the colour of the current assembly. In other words, the assembly cannot "wait" for two different "branches" to meet at a point in the plane before growing further (see section 2.2 for detailed definitions). The restriction of this model to one-dimension is exactly equivalent to finite automata, where tiles map to the edges of the automaton, and border colours to states (see Figure 1.2).

The only form of synchronisation in this model is by geometric "blocking", where two branches compete for a position in the plane, and the first one to get there can continue to grow. The fundamental question of noncooperative self-assembly is whether this rather weak form of communication is sufficient to achieve synchronisation. This has been an open problem since the early days of the field, and research in variants of the model has shown surprising results, in that every variation of the noncooperative model, however minor, seems to endow it with arbitrary computational capabilities. In the three-dimensional extension, for example, one can arrange little "bridges" and "tunnels" to block one branch of a test while allowing the other one to continue, which allows one to read and write bits [7]. In two dimensions, using random assembly sequences rather than asynchronous ones yields the same result [7], and so do negative glues [35], polyomino tiles (2×1 is enough) [17], polygonal tiles, provided they have at least seven sides [21]. Separating the assembly process into stages with different sets of tiles available at each stage also makes the model Turing-universal [1], which is also the case for a model with detachable tiles [24].

On the negative side, no tile set is intrinsically universal at temperature 1 [33], meaning that no tile set can simulate all temperature 1 tile assembly systems, even when rescaled. This result strongly hints that 2D noncooperative tile assembly is not capable of performing Turing computation, since it is in particular not capable of simulating Turing machines inside a rectangle [33], which is the only known form of Turing computation in tile assembly. Moreover, we have recently shown that long enough *paths* built by a temperature-1 tile assembly system are *pumpable* or *fragile*, meaning that their growth can only be controlled within a finite radius, after which they degenerate into simple periodic paths or their growth can be stoped [32], we present this result and the roadmap of its proof in section 2.3. Combining this result with [13] allowed us to conclude that directed temperature-1 tile assembly system are decidable in [31] (see section 2.4) and to give a classification of the different kinds of terminal assemblies (see section 2.5.1).

One particularly puzzling fact about 2D noncooperative self-assembly is

that even though it seems computationally weak, a handful of nontrivial algorithms have been designed, including assemblies of diameter $\Omega(n \log n)$, produced by a tileset of size n [27, 30], we present one this construction in section 2.5.2. In three-dimensions, recent results have also shown how to build thin rectangles [20, 19] with almost matching upper and lower bounds.

2.2 Definitions

As usual, let $\mathbb{R}$ be the set of real numbers, let $\mathbb{Z}$ be the set of all integers, let $\mathbb{N}$ be the set of all natural numbers including 0, and let $\mathbb{N}^*$ be the set of all natural numbers excluding 0. The domain of a function f is denoted $dom(f)$, and its range (or image) is denoted $f(dom(f))$.

2.2.1 Abstract tile assembly model

A *tile type* is a unit square with four sides, each consisting of a glue *type* and a nonnegative integer *strength*. Let T be a a finite set of tile types. The sides of a tile type are respectively called north, east, south, and west.

An *assembly* is a partial function $\alpha : \mathbb{Z}^2 \dashrightarrow T$ where T is a set of tile types and the domain of α (denoted $dom(\alpha)$) is connected.[1] Two tile types in an assembly are said to *bind* (or *interact*, or are *stably attached*), if the glue types on their abutting sides are equal, and have strength ≥ 1. An assembly α induces an undirected weighted *binding graph* $G_\alpha = (V, E)$, where $V = dom(\alpha)$, and there is an edge $\{a, b\} \in E$ if and only if the tiles at positions a and b interact, and this edge is weighted by the glue strength of that interaction. The assembly is said to be τ-stable if every cut of G_α has weight at least τ.

A *tile assembly system* is a triple $\mathcal{T} = (T, \sigma, \tau)$, where T is a finite set of tile types, σ is a τ-stable assembly called the *seed*, and $\tau \in \mathbb{N}$ is the *temperature*. Throughout this chapter, $\tau = 1$.

Given two τ-stable assemblies α and β, we say that α is a *subassembly* of β, and write $\alpha \sqsubseteq \beta$, if $dom(\alpha) \subseteq dom(\beta)$ and for all $p \in dom(\alpha)$, $\alpha(p) = \beta(p)$. We also write $\alpha \rightarrow_1^{\mathcal{T}} \beta$ if we can obtain β from α by the binding of a single tile type, that is: $\alpha \sqsubseteq \beta$, $|dom(\beta) \setminus dom(\alpha)| = 1$ and the tile type at the

[1] Intuitively, an assembly is a positioning of unit-sized tiles, each from some set of tile types T, so that their centers are placed on (some of) the elements of the discrete plane $\mathbb{Z}^2$ and such that those elements of $\mathbb{Z}^2$ form a connected set of points.

position $dom(\beta) \setminus dom(\alpha)$ stably binds to α at that position. We say that γ is *producible* from α if there is a (possibly empty) sequence $\alpha_1, \alpha_2, \ldots, \alpha_n$ where $n \in \mathbb{N} \cup \{\infty\}$, $\alpha = \alpha_1$ and $\alpha_n = \gamma$, such that $\alpha_1 \to_1^{\mathcal{T}} \alpha_2 \to_1^{\mathcal{T}} \ldots \to_1^{\mathcal{T}} \alpha_n$.

The set of *productions*, or *producible assemblies*, of a tile assembly system $\mathcal{T} = (T, \sigma, \tau)$ is the set of all assemblies producible from the seed assembly σ and is written $\mathcal{A}[\mathcal{T}]$. An assembly α is called *terminal* if there is no β such that $\alpha \to_1^{\mathcal{T}} \beta$. The set of all terminal assemblies of $\mathcal{T}$ is denoted $\mathcal{A}_\square[\mathcal{T}]$. If there is a unique terminal assembly, *i.e.* $|\mathcal{A}_\square[\mathcal{T}]| = 1$, then $\mathcal{T}$ is *directed* and in this case, this unique terminal assembly is denoted γ.

Given two τ-stable assemblies α and β, the union of α and β, write $\alpha \cup \beta$, is an assembly defined if and only if and for all $p \in dom(\alpha) \cap dom(\beta)$, $\alpha(p) = \beta(p)$ and either at least one tile of α binds with a tile of β or $dom(\alpha) \cap dom(\beta) \neq \emptyset$. Then, for all $p \in dom(\alpha)$, we have $(\alpha \cup \beta)(p) = \alpha(p)$ and for all $p \in dom(\beta)$, we have $(\alpha \cup \beta)(p) = \beta(p)$.

2.2.2 Paths

Let T be a set of tile types. A *tile* is a pair $((x, y), t)$ where $(x, y) \in \mathbb{Z}^2$ is a position and $t \in T$ is a tile type. Intuitively, a path is a finite, one-way-infinite or two-ways-infinite simple (non-self-intersecting) sequence of tiles placed on points of $\mathbb{Z}^2$ so that each tile in the sequence interacts with the previous one, or more precisely:

Definition 1 (Path)**.** *A* path *is a (finite or infinite) sequence $P = P_0 P_1 P_2 \ldots$ of tiles $P_i = ((x_i, y_i), t_i) \in \mathbb{Z}^2 \times T$, such that:*

- *for all P_j and P_{j+1} defined on P it is the case that t_j and t_{j+1} interact, and*
- *for all P_j, P_k such that $j \neq k$ it is the case that $(x_j, y_j) \neq (x_k, y_k)$.*

By definition, paths are simple (or self-avoiding). For a tile P_i on some path P, its x-coordinate is denoted x_{P_i} and its y-coordinate is denoted y_{P_i}. Then, the position of P_i is $\mathrm{pos}(P_i) = (x_{P_i}, y_{P_i})$, and we denote $\mathrm{type}(P_i)$ its tile type. Hence if $P_i = ((x_i, y_i), t_i)$ then $\mathrm{pos}(P_i) = (\mathrm{x}_{P_i}, \mathrm{y}_{P_i}) = (x_i, y_i)$ and $\mathrm{type}(P_i) = t_i$. A "*position of P*" is an element of $\mathbb{Z}^2$ that appears in P (and therefore appears exactly once).

Whenever P is finite, i.e. $P = P_0 P_1 P_2 \ldots P_{n-1}$ for some $n \in \mathbb{N}$, n is termed the *length* of P and denoted by $|P|$. An *index* i of a finite path P is a

natural number $i \in \{0, 1, \ldots, |P| - 1\}$. The vertical height of a finite path P is defined as $\max\{|y_{P_i} - y_{P_j}| : 0 \leq i \leq j \leq |P| - 1\}$ and its horizontal width is $\max\{|x_{P_i} - x_{P_j}| : 0 \leq i \leq j \leq |P| - 1\}$.

The *concatenation* of a finite path P with a path Q is the concatenation PQ of these two paths as sequences, and is a path if and only if (1) the last tile of P interacts with the first tile of Q and (2) P and Q do not intersect each other. For a path $P = P_0 \ldots P_i P_{i+1} \ldots P_j \ldots$, we define the notation $P_{i,i+1,\ldots,j} = P_i P_{i+1} \ldots P_j$, i.e. "the subpath of P between indices i and j, inclusive". In the special case of a subpath where $i = 0$, we say that $P_{0,1,\ldots,j}$ is a *prefix* of P. When P is finite and $j = |P| - 1$, we say that $P_{i,\ldots,|P|-1}$ is a *suffix* of P.

Although a path is not an assembly, we know that each adjacent pair of tiles in the path sequence interact implying that every path uniquely represents an assembly containing exactly the tiles of the path, more formally: for a path $P = P_0 P_1 P_2 \ldots$ we define the set of tiles $\text{asm}(P) = \{P_0, P_1, P_2, \ldots\}$ which we observe is an assembly[2] and we call $\text{asm}(P)$ a *path assembly*. Given a tile assembly system $\mathcal{T} = (T, \sigma, 1)$ the path P is a *producible path of* $\mathcal{T}$ if $\text{asm}(P)$ does not intersect[3] the seed σ and the assembly $(\text{asm}(P) \cup \sigma)$ is producible by $\mathcal{T}$, i.e. $(\text{asm}(P) \cup \sigma) \in \mathcal{A}[\mathcal{T}]$, and P_0 interacts with a tile of σ. Consider an assembly α (resp. a path Q), as a convenient abuse of notation we sometimes write $\sigma \cup P$ (resp. $P \cup Q$) as a shorthand for $\sigma \cup \text{asm}(P)$ (resp. $\text{asm}(P) \cup \text{asm}(Q)$). Intuitively, although producible paths are not assemblies, any producible path P has the nice property that it encodes an unambiguous description of how to grow $\text{asm}(P)$ from the seed σ to produce the assembly $\text{asm}(P) \cup \sigma$. Given a tile assembly system $\mathcal{T} = (T, \sigma, 1)$, we define the set of producible paths of $\mathcal{T}$ to be $\mathbf{P}[\mathcal{T}]$. Given a directed tile assembly system $\mathcal{T} = (T, \sigma, 1)$ and its unique terminal assembly γ, the path P is *a path of* γ if $\text{asm}(P)$ is a subassembly of γ. We define the set of paths of γ to be:

$$\mathbf{P}[\gamma] = \{P \mid P \text{ is a path and } \text{asm}(P) \text{ is a subassembly of } \gamma\}$$

[2]I.e. $\text{asm}(P)$ is a partial function from $\mathbb{Z}^2$ to tile types, and is defined on a connected set.

[3]Formally, non-intersection of a path $P = P_0 P_1, \ldots$ and a seed assembly σ is defined as: $\forall t$ such that $t \in \sigma$, $\nexists i$ such that $\text{pos}(P_i) = \text{pos}(t)$.

2.2.3 Intersections

If two paths, or two assemblies, or a path and an assembly, share a common position we say that they *intersect* at that position. Furthermore, we say that two paths, or two assemblies, or a path and an assembly, *agree* on a position if they both place the same tile type at that position and *conflict* if they place a different tile type at that position. We say that a path P is *fragile* to mean that there is a producible assembly α that conflicts with P. Intuitively, if we grow α first, then there is at least one tile that P cannot place.

Definition 2 (Fragile)**.** *Let $\mathcal{T} = (T, \sigma, 1)$ be a tile assembly system and $P \in \mathbf{P}[\mathcal{T}]$. We say that P is fragile if there exists a producible assembly $\alpha \in \mathcal{A}[\mathcal{T}]$ and a position $(x, y) \in (dom(\alpha) \cap dom(\text{asm}(P)))$ such that $\alpha((x, y)) \neq \text{asm}(P)((x, y))$.*[4]

For the special case of directed tile assembly systems, since the terminal assembly is unique there are no fragile paths in $\mathbf{P}[\gamma]$.

Let P and Q be two paths. We say that Q *grows from* P at index i, if the only intersection between Q and P occurs at $\text{pos}(Q_0) = \text{pos}(P_i)$ and is an agreement.

2.2.4 Periodicity and pumpability

The translation of an assembly α by a vector $\overrightarrow{v} \in \mathbb{Z}^2$, written $\alpha + \overrightarrow{v}$, is the assembly β defined for all $(x, y) \in (dom(\alpha) + \overrightarrow{v})$ as $\beta(x, y) = \alpha((x, y) - \overrightarrow{v})$. Translations of tiles and paths are defined similarly. For $A, B \in \mathbb{Z}^2$, we define $\overrightarrow{AB} = B - A$ to be the vector from A to B, and for two *tiles* $P_i = ((x_i, y_i), t_i)$ and $P_j = ((x_j, y_j), t_j)$, we define $\overrightarrow{P_i P_j} = \text{pos}(P_j) - \text{pos}(P_i)$ to mean the vector from $\text{pos}(P_i) = (x_i, y_i)$ to $\text{pos}(P_j) = (x_j, y_j)$.

An infinite (resp. bi-infinite) path P is *periodic* (resp. bi-periodic) if and only if there exists $j \geq 0$ such that for all $i \geq 0$ (resp. $i \in \mathbb{N}$), we have $P_{i+j} = P_i + \overrightarrow{P_0 P_j}$. In this case, we call $Q = P_{0,\ldots,j}$ a period of the path P and we denote this path by $(Q)^*$ (resp. ${}^*(Q)^*$).

[4]Here, it might be the case that α and P conflict at only one position by placing two different tile types t and t', but that t and t' may place the same glues along P. In this case P is not producible when starting from the assembly α because one of the tiles along the positions of P is of the wrong type.

An infinite path P is *ultimately periodic* if and only if it can be written as $R(Q)^*$ where R is a finite path called the *transient* part of P and $(Q)^*$ is the periodic path of period Q called the *periodic* part of P.

A producible path P is *pumpable* if and only if there exists a producible ultimately periodic path $R(Q)^*$ such that RQ is a prefix of P. In this case, Q is a subpath of P and we say that Q is pumpable[5] in P.

2.2.5 Glues and columns

For $0 \leq i \leq |P| - 2$, the glue used to bind tiles P_i and P_{i+1} is designed by glue(P_iP_{i+1}). This glue is *horizontal* if $y_{P_i} = y_{P_{i+1}}$ and *vertical* otherwise. An horizontal glue *points to the east* if $x_{P_i} < x_{P_{i+1}}$ (pointing to the west, north or south is defined similarly). An horizontal glue is *located* at *column* $c = \min\{x_{P_i}, x_{P_{i+1}}\}$ and we say that P *crosses trough* column c. Without loss of generality, the path P crosses trough the columns 0 to $w - 1$ where w is the horizontal width of P.

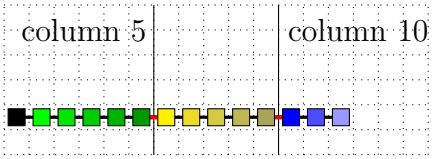
2.3 A $2D$ Pumping Lemma

The aim of this section is to sketch the proof the following result.

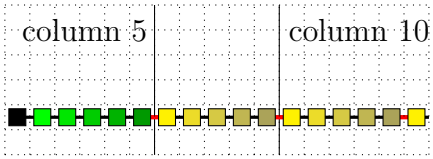
Theorem 1 ($2D$ Pumping lemma)**.** *Let $\mathcal{T} = (T, \sigma, 1)$ be any tile assembly system in the non-cooperative abstract Tile Assembly Model (aTAM), and let P be a path producible by $\mathcal{T}$. If P has vertical height or horizontal width at least $(8|T|)^{4|T|+1}(5|\sigma| + 6)$, then P is pumpable or fragile.*

The sketch proof is divided in four subsections, each dedicated to different key tools. Roughly, *spans* introduced in subsection 2.3.1 are needed to emulate the combinatorial aspect of the proof of the pumping lemma for finite automata. Subsection 2.3.2 is about *visible glues* which are used as the starting point of any geometrical reasoning of this section. In subsection 2.3.3, we explain how a sequence of three visible glues forming a pattern called *shield* can be found in any large enough path. This result combines some geometrical arguments about visible glues and combinatorial arguments about spans. Finally, we conclude in subsection 2.3.4 with the shield lemma: any

[5]Note that, if a producible path P can be written as $P = RQS$ then it is logical to define P is pumpable as for all $n \in \mathbb{N}$, $P = RQ^nS$ is producible. Nevertheless, the suffix S of P is dealt later separately in section 2.4 as a $2D$ artefact called a comb.



a) A path with two glues of the same type on columns 5 and 10.



b) The subpath between the two columns can be pumped.

Figure 2.1: A case where each column crossed by the path is crossed exactly one time. Here, the glues on columns 5 and 10 are identical and the subpath between the two columns can be pumped into an infinite periodic path.

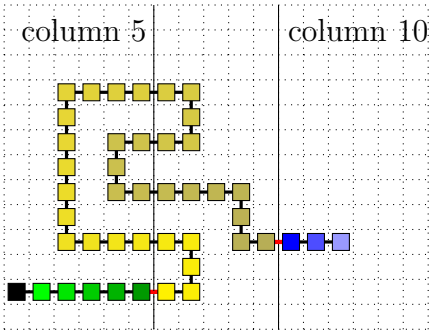
producible path with a shield is pumpable or fragile. This last part of the proof is the most difficult one and relies on complex geometrical arguments. Along this section, we consider a fixed tile assembly system $\mathcal{T} = (T, \sigma, 1)$.

2.3.1 Span and visible glues

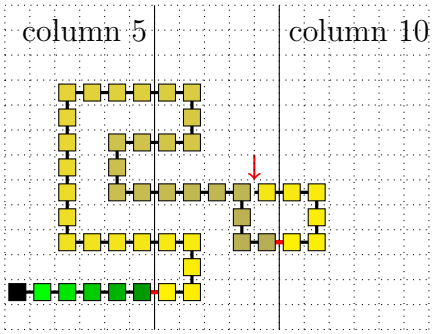
Consider a producible path P whose last tile is the unique easternmost one. Foremost, let's try to emulate the reasoning of the pumping lemma for finite automata in the simple example shown in Figure 2.1 where the path P grows horizontally. In this case, for each index i of P, glue(P_iP_{i+1}) points to the east and is the only glue located at column $i+1$. The path is pumpable as soon as two of its glues have the same type. Now, consider a more complex example as the one shown in Figure 2.2. Here, P goes back and forth several times trough the same column. Having two glues of the same type is not sufficient anymore to be pumpable.

A first result towards a kind of pumping lemma was achieved by Meunier et al [28], this result known as the *windows movie lemma* relies on the key notion of *movie*. In our context [6], the movie of P through column i is the ordered sequence of glues of P which are located on column i. The glues of a movie are ordered by their order of appearance in P and each glue is characterized by its type, its position and the direction pointed by the glue. The windows movie lemma claims that if two movies are identical up to some translation then it is possible to copy and paste the tiles between the two movies to obtain an ultimately periodic path, see Figure 2.3. It is a first step toward a pumping lemma but it is not sufficient by itself. Indeed, movies

[6] Note that, the result of [28] is more general than the simplified version used here.



a) A path crossing several times columns 5. The two glues in red have the same type.



b) The subpath between the two glues cannot be pumped.

Figure 2.2: A case where a column is crossed several times by a path. Here, finding two identical glues is not sufficient: the subpath between the two glues cannot be used to generate a bi-periodic path.

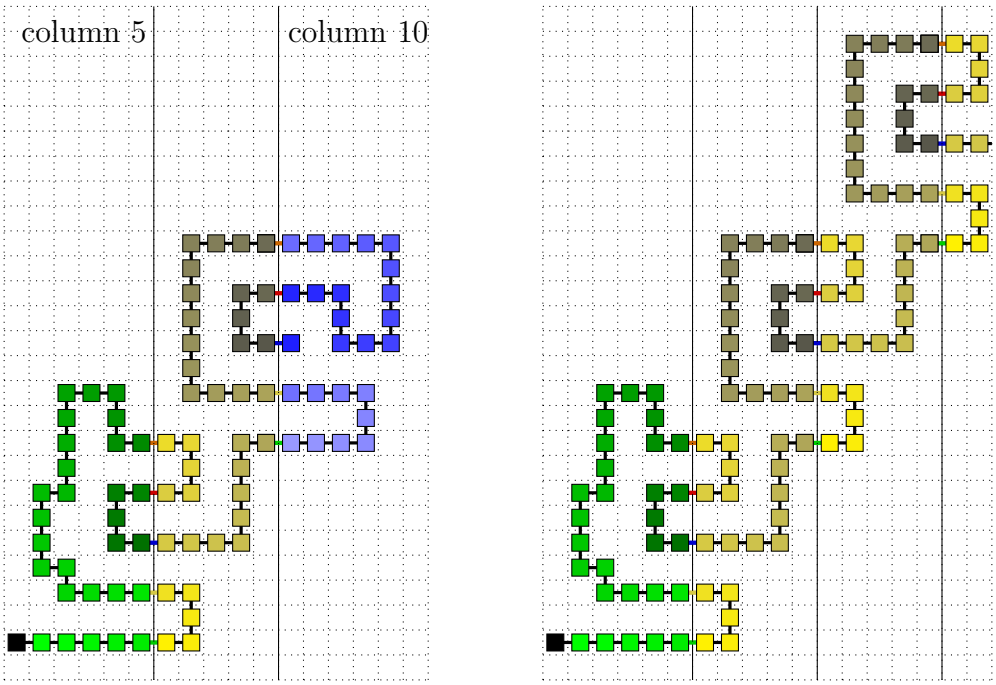
can become larger and larger as P grows to the east and thus two identical movies up to some translation cannot be found by a simple combinatorial argument.

In fact, a movie memorizes too much information for each column. Indeed, we will conclude by keeping only two glues per column. We simplified the notion of windows by introducing the notion of of *span* (see Figure 2.4).

Definition 3. *Consider a path P and $c \in \mathbb{Z}$, the span of P trough column c is defined as a couple of indices (s, n) such that* glue(P_sP_{s+1}) *(resp.* glue(P_nP_{n+1})*) is the southernmost (resp. northernmost) glue among the glues of P located at column c.*

The glue(P_sP_{s+1}) and the glue(P_nP_{n+1}) are called *visible* glues. Moreover, glue(P_sP_{s+1}) (resp. glue(P_nP_{n+1})) is said to be *visible from the south* (resp. from the north). The *glue ray* l^s (resp. l^n) associated to glue(P_sP_{s+1}) (resp. glue(P_nP_{n+1})) is the ray going south (resp. north) and starting at the midway point of P_s and P_{s+1} (resp. P_n and P_{n+1}), see Figure 2.4. By definition of a visible glue, the path P crosses a glue ray only one time at its start-point and can go back and forth trough column i only between the start-points of l^s and l^n. The size of this gap is called the *width* of the span defined formally by $x_{P_n} - x_{P_s}$.

Now, if $s < n$ then glue(P_sP_{s+1}) is before glue(P_nP_{n+1}) according to their



a) The windows of columns 5 and 10 are identical up to some translation.

b) The part between the two columns can be pumped.

Figure 2.3: A case where two columns have the same windows up to some translation. The part of the path between the two columns can be used to generate a bi-periodic path.

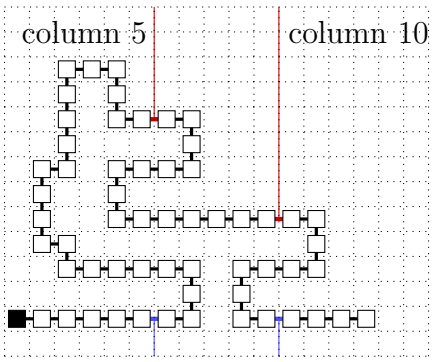


Figure 2.4: The orientation of the span though column 5 is up (the blue glue comes before the red one), it is directed to the east and its width is 8. The orientation of the span though column 10 is down (the red glue comes before the blue one), it is directed to the east and its width is 4. The red glues are visible from the north while the blue ones are visible from the south.

order of appearance in P. In this case, the *orientation* of the span is *up*, otherwise the orientation of the span is *down*. If the orientation of the span is up (resp. down), the type of the span is the type of glue(P_sP_{s+1}) (resp. glue(P_nP_{n+1})) and the span points in the same direction as glue(P_sP_{s+1}) (resp. glue(P_nP_{n+1})).

There exist $4|T|$ kinds of span when taking into account the orientation, the direction pointed by the span and the type of the span.

2.3.2 Distribution of visible glues

Consider a producible path P whose last tile is the unique easternmost one, then the distribution of its visible glues is in fact heavily constrained by three rules which were initially proven in [33], see Figure 2.5. Firstly, the order of the visible glues is coherent with the order the columns.

Lemma 1. *Consider a producible path P whose last tile is the unique easternmost one and $0 \leq i < j < |P| - 1$ and $c, c' \in \mathbb{Z}$ such that* glue(P_iP_{i+1}) *is located at column c,* glue(P_jP_{j+1}) *is located at column c', both glues point to the east (resp. west) and are both visible from the same direction then $c < c'$ (resp. $c > c'$).*

Secondly, either all glues visible from the north point to the east or all glues visible from the south point to the east. Up to some horizontal sym-

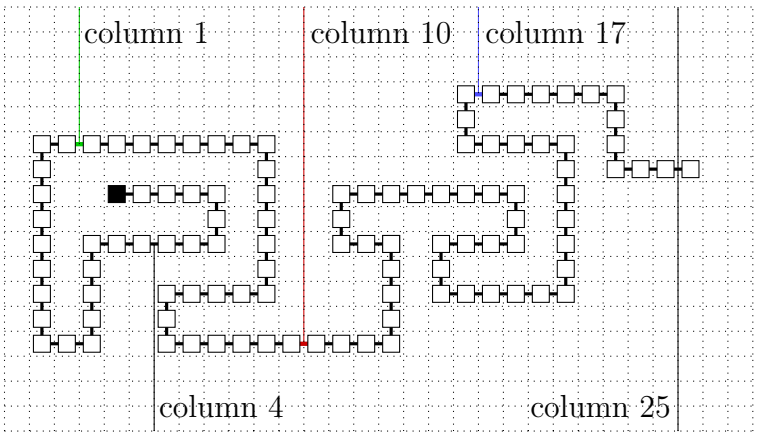


Figure 2.5: The last tile of the path is its easternmost one (the only one east of column 25). All glues visible from the south from columns 0 to 4 (resp. 5 to 25) point to the west (resp. to the east). All glues visible from the north points to the east. Among these glues, the ones of column 1 (green), column 10 (red) and column 17 (blue) appear in the following order along the path: green, red and blue.

metry, we can consider that all glues visible from the north points to the east.

Lemma 2. *Consider a producible path P whose last tile is the unique easternmost one then, up to some symmetry, all glues of P visible from the north point to the east.*

Thirdly, for the glues visible from the south, the ones pointing west are all gathered on the left side of P while the ones pointing east are all gathered on the right side of P. More formally,

Lemma 3. *Consider a producible path P whose last tile is the unique easternmost one then, there exists a column c such that all glues visible from the south on column $c' < c$ (resp. $c' \geq c$) points to the west (resp. east).*

The proofs of these lemmas all rely on cutting the $2D$ plane by using the glue rays of two different visible glues. Indeed, a bi-infiinite simple curve can be obtained by combining two glue rays and the subpath of P between the two visible glues. By the Jordan curve theorem, this curve cuts the plane in two parts, its right side and its left side (we consider that the orientation of

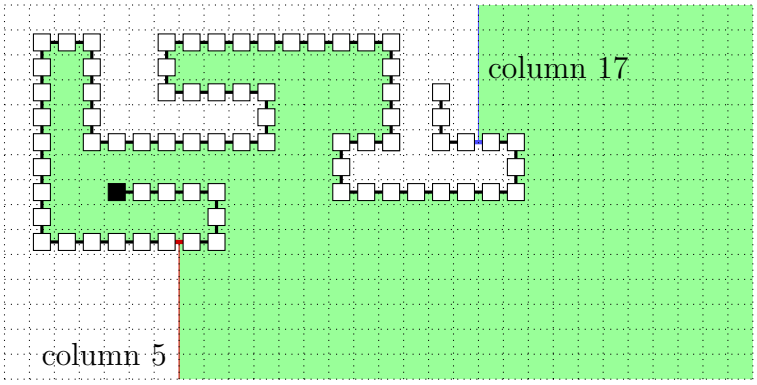


Figure 2.6: How to use visible glues: the glue in red (resp. blue) is visible from the south (resp. north) and points west. The two visible glues delimit a green area. The last tile of the path is the easternmost one and must be in the right side of the cut (in green). Since the glue in blue points west, the end of the path is in the left side of the cut which is a contradiction.

the curve is the same as P). For an example of such a reasoning, we sketch the proof of lemma 2, see Figure 2.6. By contradiction, consider glue(P_sP_{s+1}) visible from the south and pointing west and glue(P_nP_{n+1}) visible from the north and also pointing west. Without loss of generality suppose that $s < n$ and then the right side of the cut defined by the two glue rays contains the last tile of P since it is the unique easternmost one. Moreover, the left side of the cut contains the tile P_{n+1} since glue(P_nP_{n+1}) points to the west. Then $P_{n+1,\dots,|P|-1}$ must reach $P_{|P|-1}$ by crossing either one of two glue rays or by intersecting $P_{s+1,\dots,n}$. This is contradicting either the visibility of glue(P_sP_{s+1}), the visibility of glue(P_nP_{n+1}) or the fact that P is simple.

2.3.3 Shield

In this subsection, we explain how to find a pattern called *shield* in any path P large enough. In the next subsection, we conclude by showing that a path with a shield is fragile or pumpable (see Lemma 5). Figure 2.7 illustrates this definition.

Definition 4 (A shield (i, j, k) for P)**.** *Let P be a path producible by some tile assembly system $\mathcal{T} = (T, \sigma, 1)$. We say that the triple (i, j, k) of indices is a* shield for P *if $0 \leq i < j \leq k < |P| - 1$, and the following three conditions*

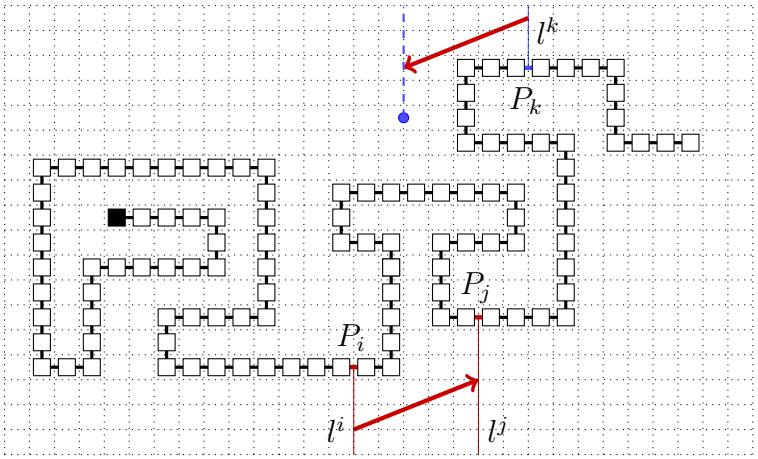


Figure 2.7: A shield (i, j, k): the two glues in red have the same type and are visible from the south. The glue in blue is visible from the north. In this example, the translation of l^k does not cross the path.

hold:

1. glue(P_iP_{i+1}) *and* glue(P_jP_{j+1}) *are both of the same type, visible from the south and pointing in the same direction; and*

2. glue(P_kP_{k+1}) *is visible from the north, for notation let* l^k *be its glue ray; and*

3. *If* $l^k + \overrightarrow{P_jP_i}$ *intersects with* $P_{i,i+1,\ldots,k}$ *(which may not be the case), there is exactly one intersection, which is at the start-point of the ray* $l^k + \overrightarrow{P_jP_i}$.

Throughout the paper, l^i, l^j and l^k denote the glue rays of glue(P_iP_{i+1}), glue(P_jP_{j+1}) and glue(P_kP_{k+1}), respectively.

Lemma 4. *If* P *has vertical height or horizontal width at least*

$$(8|T|)^{4|T|+1}(5|\sigma| + 6),$$

then there is a shield in P *or in a prefix of* P.

Proof. Consider a path P of horizontal width L, for the first part of the proof see Figure 2.8. Without loss of generality, we consider that the glues of P are located between columns 0 and $L - 1$ and that the last tile of P

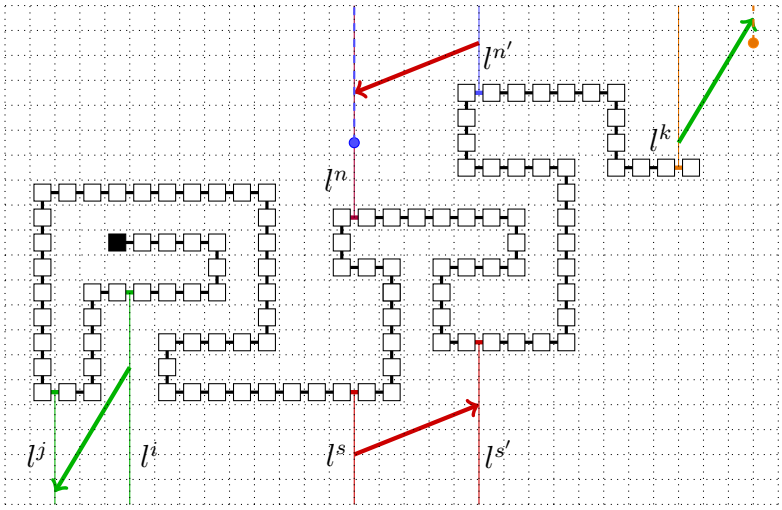


Figure 2.8: This path contains two shields. The first one is (i, j, k) where i and j are pointing to the west and k is the easternmost glue of the path. The second one (s, s', n') is obtained with two spans (s, n) and (s, n') of decreasing width.

is the easternmost one. To avoid interaction with the seed, we do not take into account the columns which are crossed by the seed. Then, there are $L - |\sigma|$ spans to consider. By lemma 2, we suppose that all the glues visible from the north are pointing east. Now, suppose that there is two indices $0 \leq i < j \leq |P| - 1$, such that glue$(P_iP_{i+1})$ and glue(P_jP_{j+1}) are visible from the south, of the same type and are pointing west. In this case, we claim that $(i, j, |P| - 1)$ is a shield, see Figure 2.8. Indeed, condition 1 of definition 4 is satisfied by hypothesis and condition 2 of definition 4 is satisfied since the last tile of P is the unique easternmost one. For condition 3 of definition 4, by lemma 1, we have $x_{\overrightarrow{P_jP_i}} > 0$ and thus $l^k + \overrightarrow{P_jP_i}$ is east of column L and cannot intersect with P and $|\sigma|$.

Now consider that there are less than $|T|$ glues of P which are visible from the south and pointing west. By lemma 3, any span of column $c \geq |T| + |\sigma|$ is pointing east whatever its orientation is (since the two visible glues point to the east). Consider the first column $c \leq |T| + |\sigma|$ where such a span appears. This span is called (s, n), its width is w and w.l.o.g. we suppose that its orientation is up. We claim that if more than w spans share the same orientation and type with (s, n) then there is a shield in P. Indeed consider

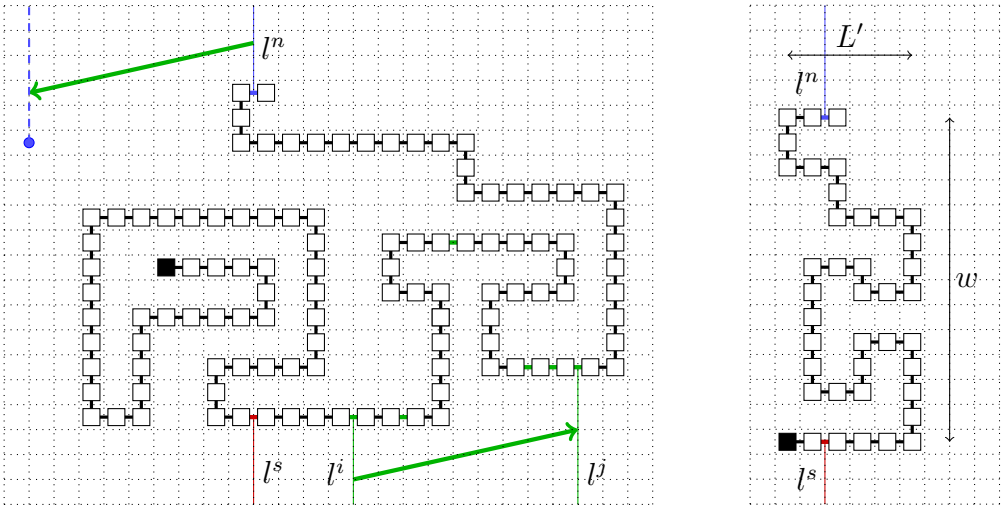


Figure 2.9: Consider the path $P_{0,\dots,n+1}$ and (s,n), its span trough column c. In the left case, the horizontal width L' of $P_{0,\dots,n+1}$ is far larger than c. In this case, there are enough visible glue of the same type (in green) to find a shield (i,j,n). In the right case, the width of the span w is far larger than L'. By switching the x and y axis, we obtain a path of horizontal width w and the widths of its spans are bounded by L'.

another span (s',n') of width w' with the same type and orientation than (s,n), see Figure 2.8. If $w' \geq w$, we claim that (s,s',n') is a shield. Indeed, (s',n') is east of (s,n) and then by lemma 1, $s < s'$. Since the orientation of (s',n') is up, then $s' \leq n'$. By definition of a span, glue$(P_{n'}P_{n'+1})$ is visible from the north while glue(P_sP_{s+1}) and glue$(P_{s'}P_{s'+1})$ are visible from the south. Since glue$(P_{s'}P_{s'+1})$ and glue$(P_{n'}P_{n'+1})$ are on the same column then $l^{n'} + \overrightarrow{P_{s'}P_s}$ and l^n are on the same column. Since $w' \geq w$ then the start-point of $l^{n'} + \overrightarrow{P_{s'}P_s}$ is north of the start-point of l^n then $l^{n'} + \overrightarrow{P_{s'}P_s}$ can intersect with $P_{s+1,\dots,n'}$ only at its start-point. Thus (s,s',n') is a shield. To conclude this second part, either there is a shield in P or the widths of spans sharing the same orientation and type are strictly decreasing.

Now, suppose that there are less than w spans which share the same orientation and type than (s,n). In order to bound w, consider the prefix $Q = P_{0,\dots,n+1}$. If the horizontal width of Q is greater than $|T|(c+1)+|\sigma|+|T|$ then there are at least $c+1$ glues visible from the south pointing east of the same type in Q, see Figure 2.9. Let glue(P_iP_{i+1}) (resp. glue(P_jP_{j+1})) be

the first (resp. last) of these visible glues. Since $\overrightarrow{Q_jQ_i} < -c$, by a reasoning similar to the one of the first paragraph $(i, j, n+1)$ is a shield of Q. Now, if the horizontal width of Q is bounded by $|T|(c+1)+|\sigma|+|T|$, we rotate Q such that its horizontal span is w and the widths of its spans are bounded by $|T|(c+1)+|\sigma|+|T|$ (see Figure 2.9). If w is large enough, two identical spans of increasing width can be found and it is possible to conclude as in the second paragraph of the proof.

To conclude, we iterate this reasoning for the $2|T|$ kinds of span to obtain the bound stated in the lemma. The detailed computations are available in [32]. □

2.3.4 Proving the shield lemma: right priority path and dominant tiles

In this subsection, we present a roadmap of the shield lemma 5. Note that, the whole proof is long, complicated and available in [32]. Here, many special and technical cases are omitted.

Lemma 5 (Shield Lemma)**.** *Let P be a path producible by some tile assembly system $\mathcal{T} = (T, \sigma, 1)$. If there exists a shield (i, j, k) in P then P is pumpable or fragile.*

Consider a path P with a shield (i, j, k). First of all, we are looking for an area of the $2D$ grid devoided of any obstacles. It will be used to assemble paths which are either periodic or conflicting with P. This area is the right side of the cut delimited by the two glue rays l^i and l^k and it is called the *workspace* $\mathcal{C}$ (see Figure 2.10). Indeed, since glue(P_iP_{i+1}) points to the east, σ and $P_{0,\dots,i}$ are in the left side of the cut. Then, workspace $\mathcal{C}$ may contain only the tiles $P_{i+1,\dots,|P|-1}$. Therefore, for any path Q whose first tile binds with P_i and which is inside $\mathcal{C}$, the path $P_{0,\dots,i}Q$ is a producible path.

By property 1 of a shield, glue(P_iP_{i+1}) and glue(P_jP_{j+1}) have the same type. We denote by R the path $P_{j+1,\dots,|P|-1} + \overrightarrow{P_jP_i}$, we start by assembling $P_{0,\dots,i}$ first and then we try to assemble R, see Figure 2.11. Of course, R may not fully grow but in this case, it must leave $\mathcal{C}$. To do so, R cannot intersect with l^i otherwise $P_{j+1,\dots,|P|-1}$ would intersect l^j (contradicting property 1 of a shield). If R leaves $\mathcal{C}$ by crossing $P_{i+1,\dots,k}$ then either R conflicts with $P_{i+1,\dots,k}$ (and P is fragile and the proof ends) or we can modify R by connecting it with $P_{i+1,\dots,k}$ as shown in Figure 2.11. Now, let's start again by assembling

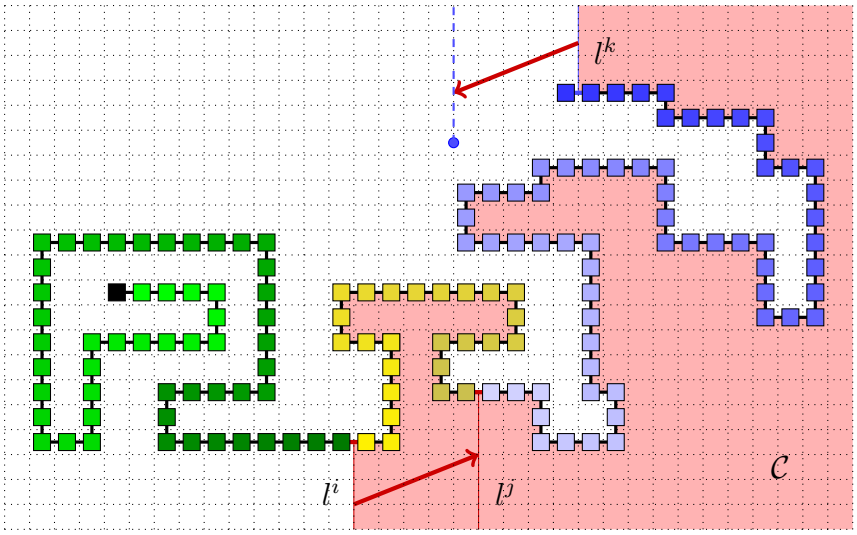


Figure 2.10: A path P with a shield (i, j, k). The beginning of the path $P_{0,\dots,i}$ is in green, its middle $P_{i+1,\dots,j}$ is in yellow and its end $P_{j+1,\dots,|P|-1}$ is in blue. Its workspace is in red and the translation of l^k is not in the workspace.

$P_{0,\dots,j}$ first and then let's try to assemble $R + \overrightarrow{P_iP_j}$, see Figure 2.12. By a similar reasoning, we can connect $R + \overrightarrow{P_iP_j}$ and $P_{j+1,\dots,k}$. By iterating this reasoning, either P is fragile or the path R is modified until it obtains the following properties (see Figure 2.13):

1. R and $R + \overrightarrow{P_iP_j}$ are in $\mathcal{C}$ and $R_0 = P_{j+1} + \overrightarrow{P_jP_i}$;
2. $R_{|R|-1}$ is "near" l^k;
3. R is a patchwork made of subpaths of $P_{i,\dots,k}$ and $P_{j,\dots,k} + \overrightarrow{P_jP_i}$.

We omitted several details in the construction of path R but note that the hypothesis 3 of the shield lemma is required to avoid that $P_{j,\dots,k} + \overrightarrow{P_jP_i}$ ends inside $\mathcal{C}$ without intersecting with l^k or $P_{i+1,\dots,k}$. In such a case, R would not end near l^k (property 2 of R would be false) and R would not be long enough for the next steps. Also, our example represents a simple case where the intersection between R and $P_{i,\dots,k}$ is reduced to a single position. In a more general context, there may be several intersections between the two paths and choosing the good one must be done carefully. This problem is solved in [32] by using an order introduced in [33]:

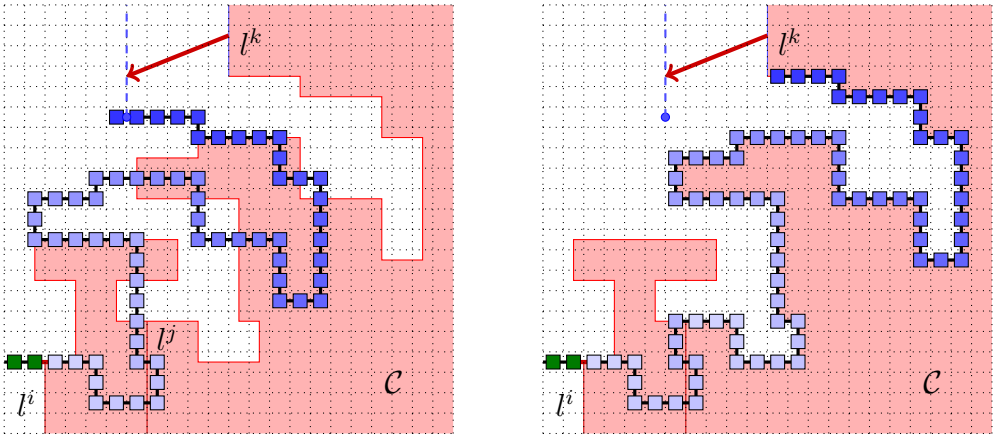


Figure 2.11: On the left side, segment $P_{i+1,\dots,j}$ of Figure 2.10 is removed. The translation of the end of the path, called R, is glued directly at the end of $P_{0,\dots,i}$. Note that, R must leave the workspace to reach the translation of l^k. On the the right side, R is modified and connected with the original path before leaving the workspace.

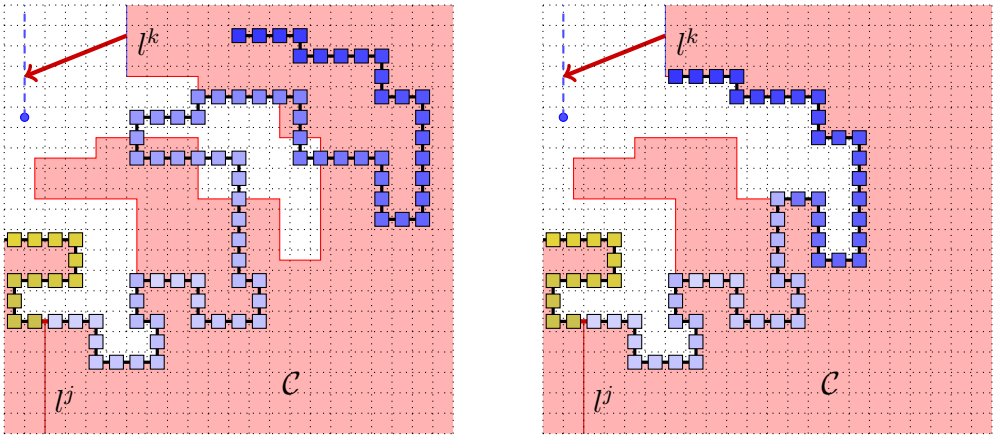


Figure 2.12: On the left side, the path R obtained at the end of Figure 2.11 is translated and glued at the end of the yellow segment of P, see Figure 2.10. If the translation of R leaves the workspace then it is modified as previously by connecting it with the end of P as shown on the right side.

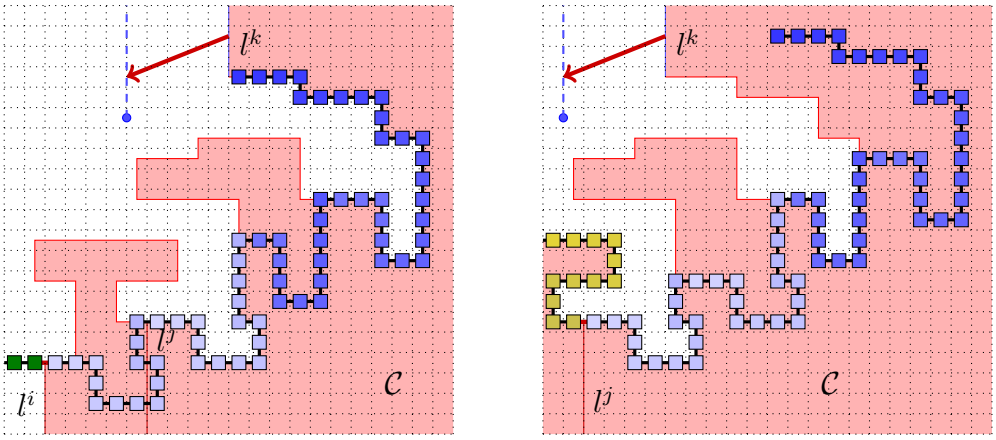


Figure 2.13: By iterating the reasoning of Figure 2.11 and 2.12, we obtain this path R. When R is glued at the end of $P_{0,\dots,i}$ (on the left side) then it stays inside the workspace and ends near ends l^k. When R is glued at the end of $P_{0,\dots,j}$ (on the right side) then it stays inside the workspace.

Definition 5 (The right priority path of a set of paths)**.** *Let P and Q be two paths, where $P \neq Q$ and moreover neither is a prefix of the other, and with* $\text{pos}(P_0) = \text{pos}(Q_0)$ *and* $\text{pos}(P_1) = \text{pos}(Q_1)$*. Let i be the smallest index such that $i \geq 0$ and $P_i \neq Q_i$. We say that P is the* right priority path *of P and Q if either (a) $P_{0,1,\dots,i}$ is a right turn from Q or (b)* $\text{pos}(P_i) = \text{pos}(Q_i)$ *and the type of P_i is smaller than the type of Q_i in the canonical ordering of tile types.*

For any finite set S of paths, we extend this definition as follows: let $p_0 \in \mathbb{Z}^2, p_1 \in \mathbb{Z}^2$ be two adjacent positions. If for all $P \in S$, we have $\text{pos}(P_0) = p_0$ *and* $\text{pos}(P_1) = p_1$*, we call the* right-priority path of S *the path that is the right-priority path of all other paths in S.*

The path R is the *right priority* path among all the paths which start by P_i and P_{i+1}, use tiles of $P_{i,\dots,k}$ and $P_{j,\dots,k} + \overrightarrow{P_jP_i}$ and end near l^k. The notion of right priority is key in most recent results about temperature 1.

Now, we aim to show that P is pumpable. Remind that we want to work inside the workspace $\mathcal{C}$, then we start by looking for a tile P_d such that $P_d + a\overrightarrow{P_iP_j}$ belongs to $\mathcal{C}$ for all $a \in \mathbb{N}$. Finding such a tile can be done by drawing the tangent to $P_{i+1,\dots,k}$ of direction $\overrightarrow{P_iP_j}$, see Figure 2.14. This tile P_d is called a *dominant* tile and it has the following properties, see Figure 2.14:

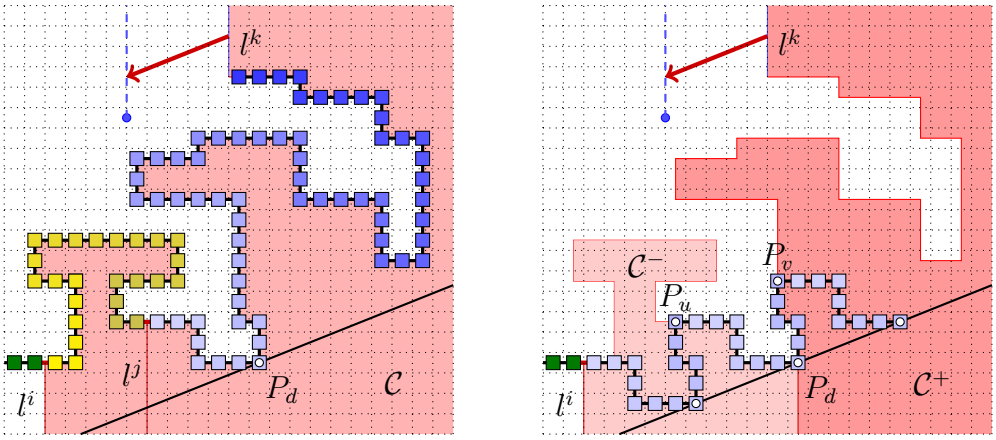


Figure 2.14: On the left side, the dominant tile P_d is found by drawing the tangent to $P_{i+1,\dots,k}$. There exist two indices $u \leq d \leq v$ such that $P_u = P_v + \overrightarrow{P_jP_i}$ and $P_{d,\dots,v} + \overrightarrow{P_jP_i}$ is a subpath of R. Then, R and its translation can be used to assemble the path on the right side which is the beginning of a periodic path.

1. The ray l^d starting at $\text{pos}(P_d)$ and going south intersects R only at $\text{pos}(P_d)$;
2. This ray l^d cuts $\mathcal{C}$ in two parts: $\mathcal{C}^-$ (resp. $\mathcal{C}^+$) which contains l^i and $P_{i+1,\dots,d}$ (resp. $P_{d,\dots,k}$ and l^k);
3. If $d > j$, $\text{pos}(P_d) + \overrightarrow{P_jP_i}$ is in $\mathcal{C}^-$ and for all $a \in \mathbb{N}$, $P_d + a\overrightarrow{P_iP_j}$ is in $\mathcal{C}^+$.

For property 1 of a dominant tile, remark that the path R starts in $\text{pos}(P_{i+1})$ which is in $\mathcal{C}^-$ and ends near l^k (by property 2 of path R) which is in $\mathcal{C}^+$ thus it must cross l^d. By definition of d and property 3 of path R, the only possible intersection between R and l^d is at $\text{pos}(P_d)$. For property 3 of a dominant tile, the special case $d \leq j$ is omitted here.

Now, the definition of a right priority path and property 3 of the dominant tile imply that $P_d + \overrightarrow{P_jP_i}$ is a tile of R which belongs to $\mathcal{C}^-$, see Figure 2.14. Thus, R must intersect with $P_{i+1,\dots,d}$ after passing by $P_d + \overrightarrow{P_jP_i}$. This intersection implies that there exist $u \leq d \leq v$ such that $P_u = P_v + \overrightarrow{P_jP_i}$. If $P_{u,\dots,v}$ is pumpable, then the proof ends.

Otherwise, we claim that there exists another dominant tile $P_{d'}$ with $d' > d$. To find this tile, consider the shortest suffix of $P_{u,\dots,d}$ such that

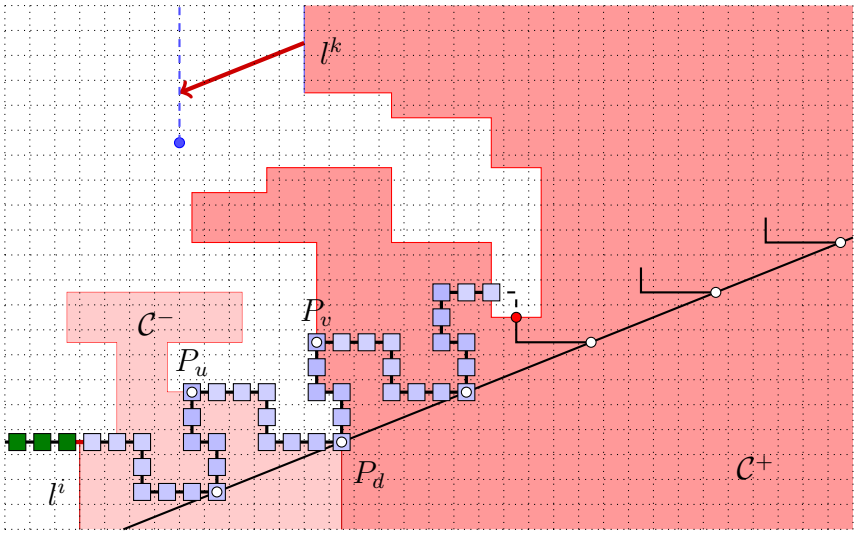


Figure 2.15: If the subpath $P_{u,\dots,v}$ of Figure 2.14 is used to generate a periodic path then it leaves the workspace. In this case, we will find a new dominant tile $P_{d'}$ where the red dot is (see Figure 2.16).

$P_{u,\dots,d} + a\overrightarrow{P_iP_j}$ intersects with $P_{d+1,\dots,k}$ for some $a > 1$ (see Figure 2.15), this intersection occurs at $P_{d'}$. We claim that $P_{d'}$ is a dominant tile. Indeed, by concatenating $P_{u,\dots,d} + a\overrightarrow{P_iP_j}$ and $l^d + a\overrightarrow{P_iP_j}$ we obtain a curve which cuts $\mathcal{C}$ into two parts and which satisfies the three properties of a dominant tile, see Figure 2.16. By iterating this reasoning, the last dominant tile of P belongs to a pumpable subpath.

2.3.5 Discussion

The bound of the pumping lemma 1 seems obviously unconventional and we conjecture that a polynomial bound is achievable. One way to proceed would be to upgrade the shield lemma. Indeed, in a previous unpublished version [29], we relied on a weaker version of the shield lemma called the U-turn lemma. This lemma required far more complex combinatorial arguments to produce a bound which is tower exponential. Combining the shield lemma and the window movie lemma may produce interesting results. Indeed, glue ray are straight line while we could use more complex curve (which is done in the recurrence of the proof of the shield lemma) or by including horizontal glue in the reasoning. This upgraded shield lemma may allow more efficient

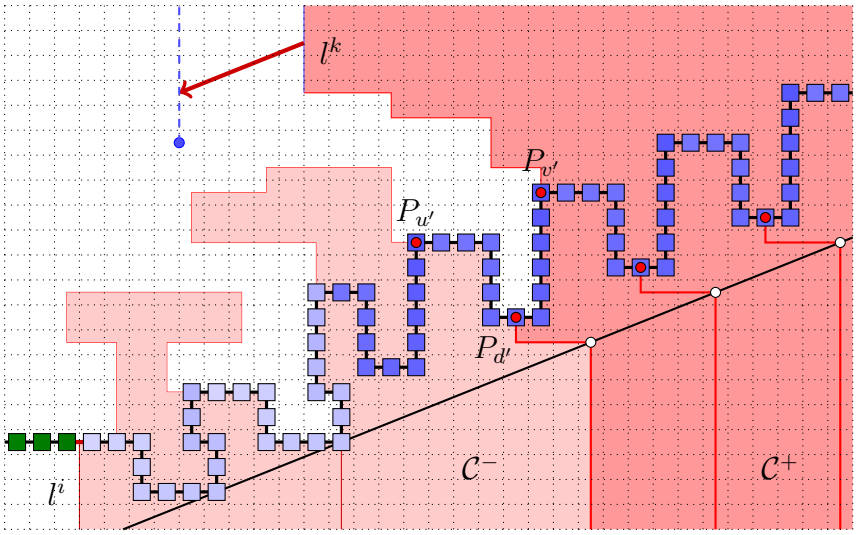


Figure 2.16: Following the failed attempt of Figure 2.15, we find a new dominant tile $P_{d'}$ and a new subpath $P_{u',\dots,v'}$ by iterating the same reasoning. This time, the subpath is pumpable.

combinatorial arguments.

Also, in subsection 2.5.2, we show a construction which implies a lower bound of $\Omega(|T|\log|T|)$ for the pumping lemma. We discuss more about these questions later since they concern the complexity and not the decidability of non-cooperative aTAM. Concluding about the decidability of non-cooperative aTAM was the main motivation to find a pumping lemma.

2.4 Decidability

Unfortunately the pumping lemma 1 is not enough to conclude about the decidability of non-cooperative aTAM. Indeed, there are still two problems to address. For the first problem, see Figure 2.17 where we consider a producible path P. If P is large enough then the pumping lemma may conclude that P is fragile by finding a producible path Q which conflicts with P. Nevertheless, Q may be the prefix of a large path and the pumping lemma is required again. We may end up in a case where a producible path R is assembled first in order to block the path Q and then P is allowed to growth. Dealing with a

path was hard but dealing with all the paths at the same time is even harder[7]. The second problem occurs when the pumping lemma produces an ultimately periodic path P. Then, another ultimately periodic path Q may grow on the periodic part of P, see Figure 2.18. We say that Q is a *comb* of P (a notion introduced by Doty et al [13]). By continuing this reasoning, it may be possible to assemble an infinite sequence of ultimately periodic paths which are able to achieve some complex computation together. Moreover, it should noted that Cook et al [7] have shown that non-cooperative aTAM is able to simulate almost surely a Turing machine if probabilities are introduced in the model. Then we aim to show that any tile assembly system has a simple (easy to describe) terminal assembly.

Nevertheless, the pumping lemma allows to conclude in the directed subcase. Indeed, the first problem does not exist anymore since fragility is not possible. The second problem was solved by Doty et al in [13]. A comb Q may grow on an ultimately periodic path P but no other comb can grow on Q. Indeed, the intersections of the different combs will form a cycle and copies of this cycle will invade the whole $2D$ grid, see Figure 2.19. The terminal assembly becomes bi-periodic and is easily described by two vectors and a finite pattern.

Note that, the bibliography is confusing on this point. Indeed, the result of Doty et al [13] was published before ours [32] and relied on a different version of the $2D$ pumping lemma. This version is stronger than the version presented in Theorem 1 and published in [32]. Nevertheless, as stated in [32], the proof of [13] still holds with our version of the $2D$ pumping lemma. Only one remark is needed to deal with a comb Q growing on a periodic path P. Indeed, the comb Q is not producible since its first tile does not bind with σ. The trick is to consider a "new" seed made of σ and P as shown in Figure 2.20. To end any confusion, this patch was published in [31] with several improvements concerning the complexity of the terminal assembly.

[7]Note that, a similar problem was solved in [33] in an other context.

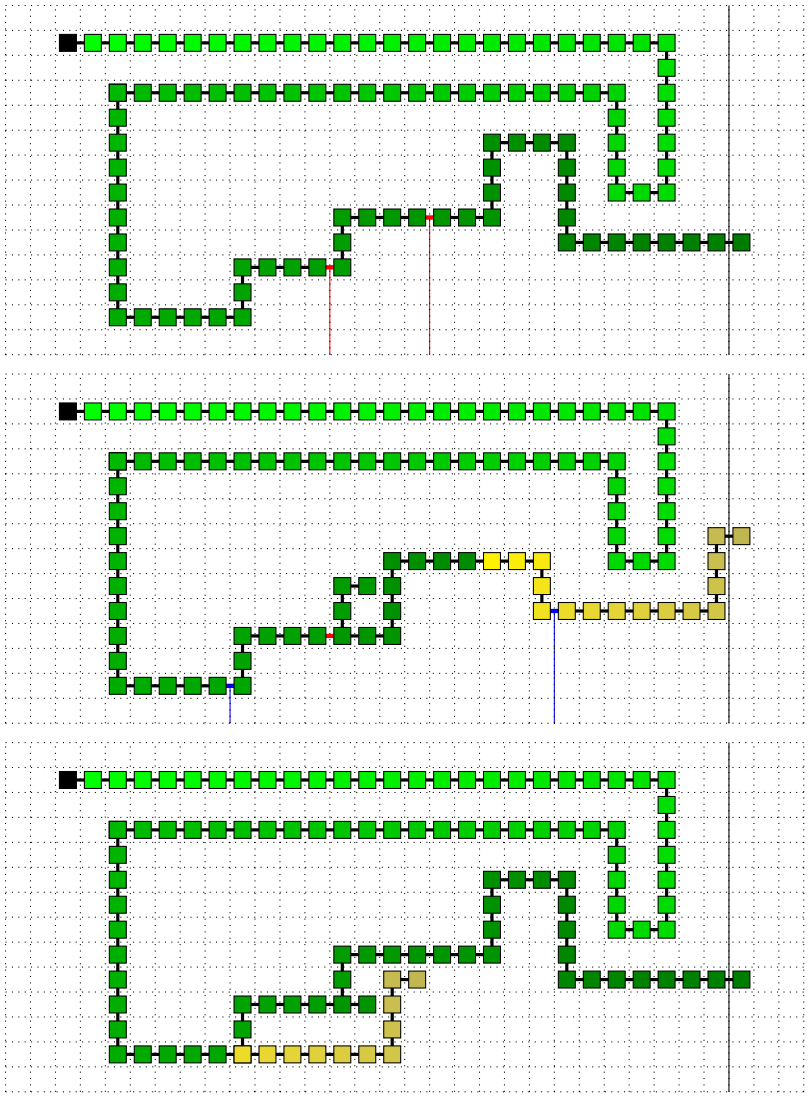


Figure 2.17: The first problem: (Top) A producible path P is able to perform some "computation". (Middle) Using the pumping lemma, a path R blocks P but R is also able to do some "computation". (Bottom) Similarly, a path Q blocks R, allowing P to grow again.

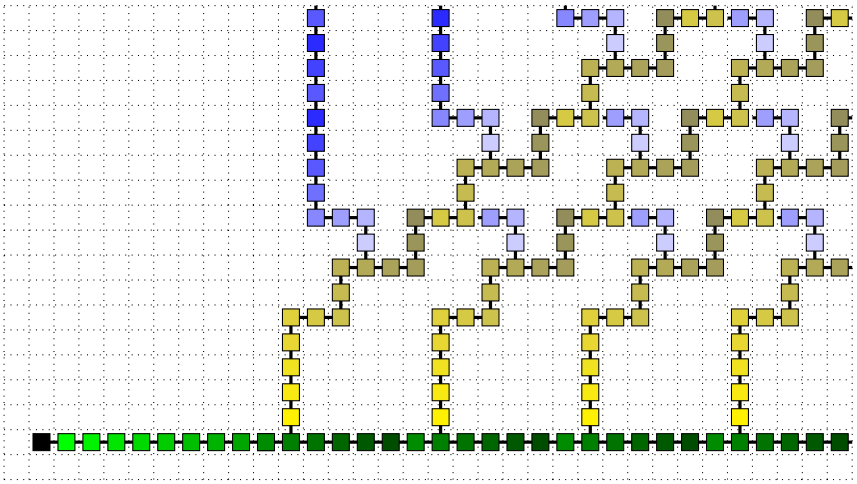


Figure 2.18: The second problem: an ultimately periodic path (in green) grows from the seed (in black). Then, another ultimately periodic path (in yellow) grows on the periodic part of the green path. This path is called a *comb*. This sequence of combs may be infinite, hard to describe and may encode complex computation.

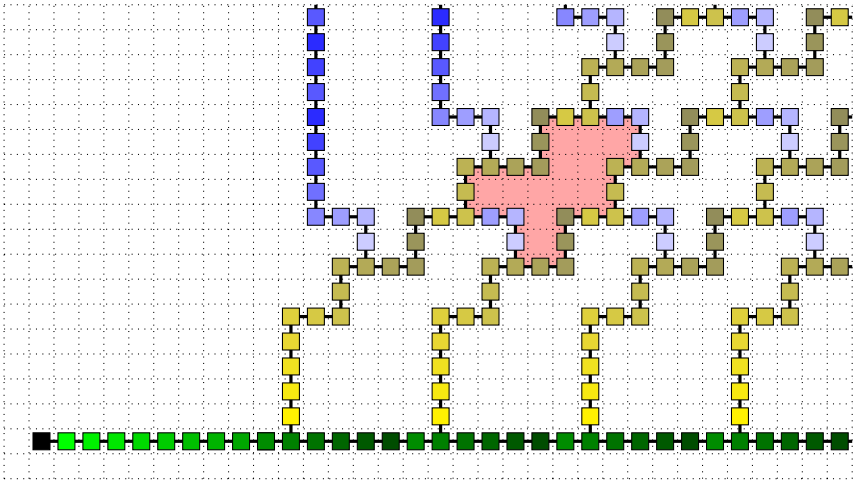


Figure 2.19: The solution of the second problem in the directed case: the intersections of the different combs create a cycle which delimits a red area. In fact, this area is enough to describe the terminal assembly since the cycle is able to invade the whole $2D$ grid and the configuration is bi-periodic.

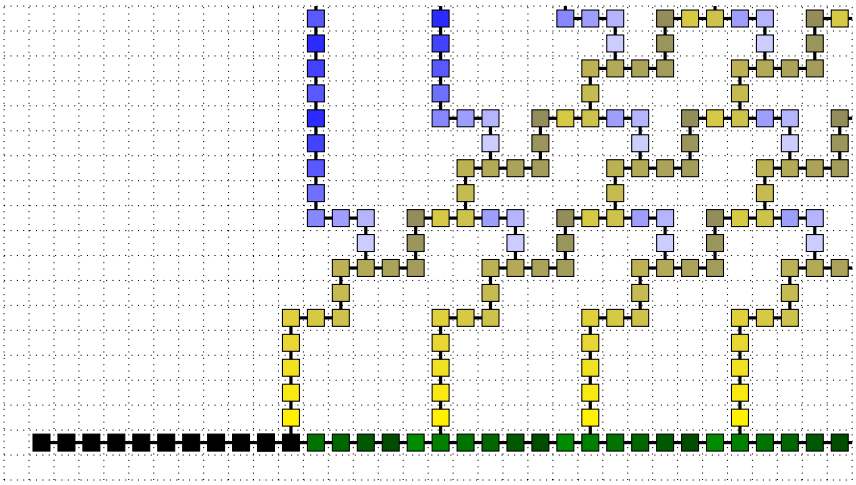


Figure 2.20: The pumping lemma should be applied on a producible path. In order to pump a comb (here in yellow), we "cheat" as follow: the path (in green in Figure 2.19) growing from the seed (in black) to the comb is considered as part of the seed.

2.5 Complexity

2.5.1 The unique terminal assembly in the directed case

In [31], we investigate the complexity of the unique terminal assembly in directed non-cooperative aTAM. The previous study of Doty et [13] implies that there exist four kinds of terminal assembly, see Figure 2.21:

- type (i): finite terminal assembly whose size is bounded by the $2D$ pumping lemma;
- type (ii): non-periodic infinite terminal assembly characterized by a finite number of finite paths, ultimately periodic paths and combs;
- type (iii): periodic terminal assembly characterized by a finite assembly, one vector and a finite number of finite paths and combs;
- type (iv): bi-periodic terminal assembly characterized by a finite assembly and two vectors.

Since type(iii) is an hybrid case between type(ii) and type(iv), these terminal assemblies are made of three different kinds of structures: non-periodic finite assembly, ultimately periodic paths/comb and periodic/bi-periodic assembly. In [31], we give an optimal characterization of the periodic/bi-periodic assemblies: they can be described with one/two vectors and a finite path where each tile type appears at most one time. In other words, the periodic/bi-periodic structures have to be hardcoded and there is no clever way to reuse any tile type. The other improvements of [31] are more technical: we bound where a comb appears for the first time on an ultimately periodic path, we use the pumping lemma only two times to conclude by using a combinatorial argument to deal with the combs (instead of three times in [13]). Note that finding the maximum length of the finite assembly/ultimately periodic path/comb is still open. Among these questions, we are currently investigating the maximum size of a finite path and we hope to obtain a linear bound soon. This result may lead to new tools in order to improve several other bounds. Moreover, it would also prove that the directed case is not equivalent to the non-directed one. Indeed, this chapter ends by showing a positive result about non-cooperative aTAM.

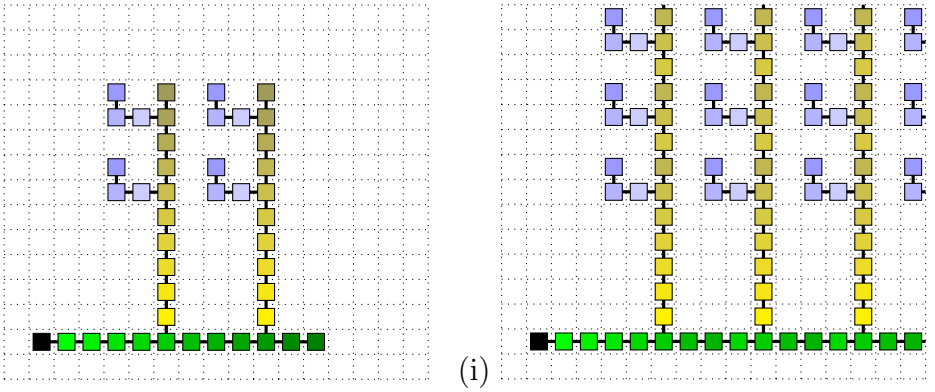
2.5.2 Efficient path for non-cooperative aTAM

In this section, we present the positive result of [30] which exhibits a non-trivial program relying on collisions.

Definition 6. *Consider a tile assembly system $\mathcal{T} = (T, \sigma, 1)$, a path P is efficient for $\mathcal{T}$ if and only if all assemblies of $\mathcal{A}_{\square}[\mathcal{T}]$ contain the path P.*

Theorem 2. *For all $t \geq 0$, there is a tile assembly system $\mathcal{T} = (T, \sigma, 1)$ with an efficient path of horizontal width $w = \Theta(|T| \log |T|)$, where $|\sigma| = 1$, $|T| \geq t$, and all assemblies of $\mathcal{A}_{\square}[\mathcal{T}]$ are of horizontal width w and vertical height less than $|T|$.*

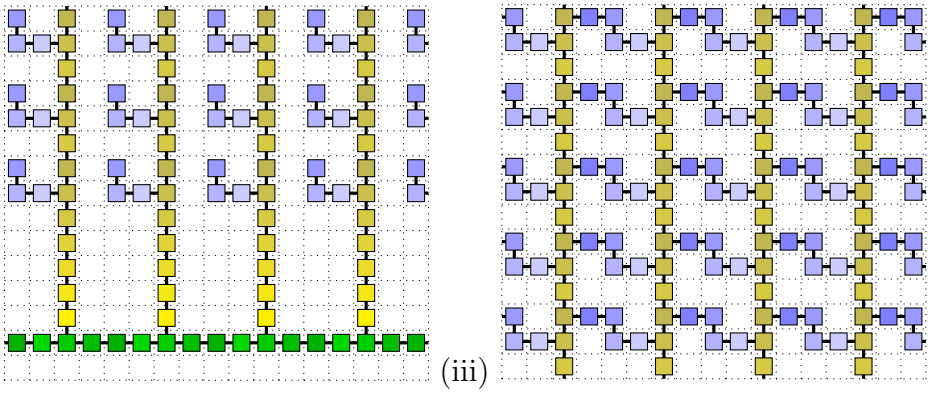
The seed is always made of a unique tile type. The tile assembly systems are defined using two parameters n and k and two functions $h : \mathbb{N} \to \mathbb{N}$ and $s : \mathbb{N} \to \mathbb{N}$ whose values are available in [30]. Figure 2.22 shows a producible path P using each tile type of such a tile assembly system exactly one time. The path P can be decomposed as $P = GYBR$ and each of these four subpaths of P will be used as follows:



(i)

a finite terminal assembly

(ii) a non-periodic infinite one



(iii)

a periodic terminal assembly

(iv) a bi-periodic one

Figure 2.21: The four different kinds of terminal assembly of a directed tile assembly system.

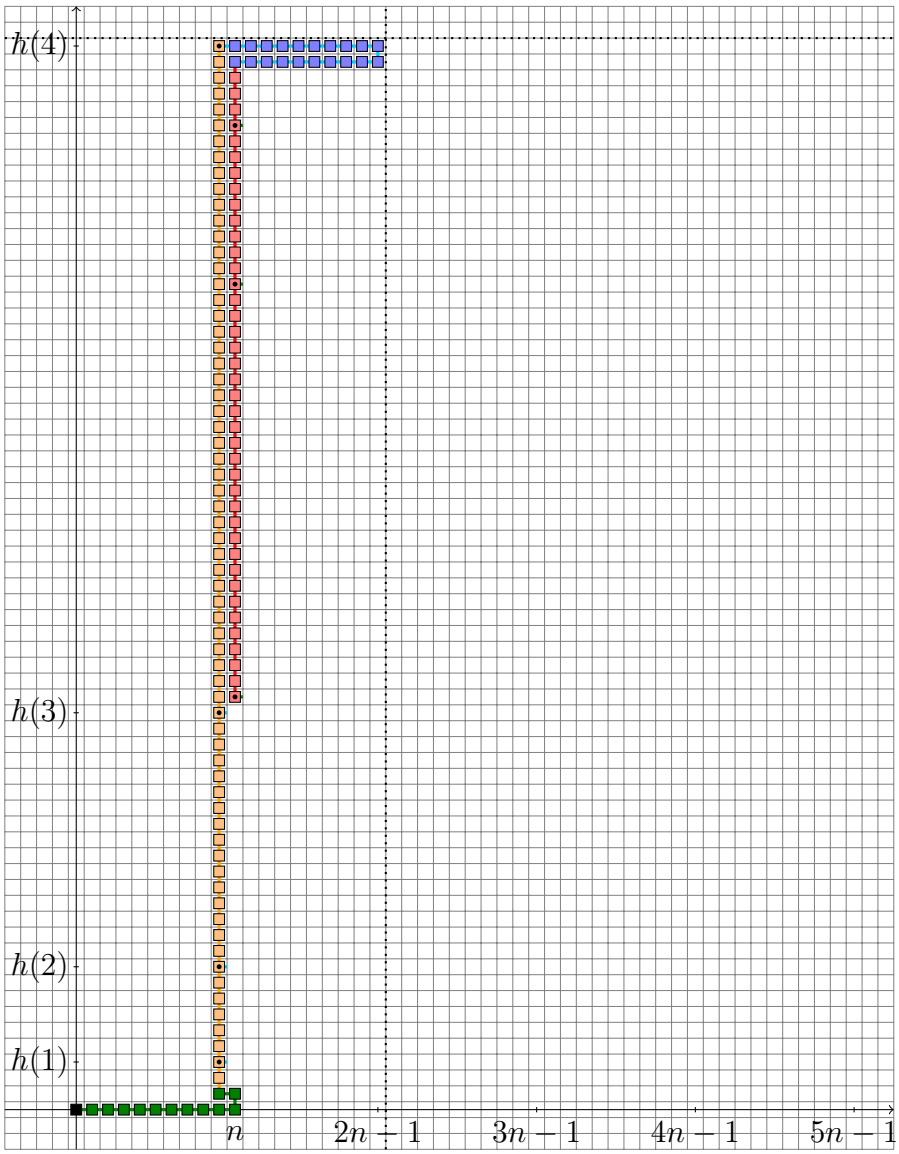


Figure 2.22: In our examples, we consider $k = 4$ and $n = 10$. The seed (in black) is at position $(0, 0)$ and we represent the producible path $P = GYBR$. This figure contains exactly one occurrence of each tile type. The height of P is $h(4)$ and its width is $2n - 1$. The tiles of Y and R with a glue on their east side are marked by a black dot.

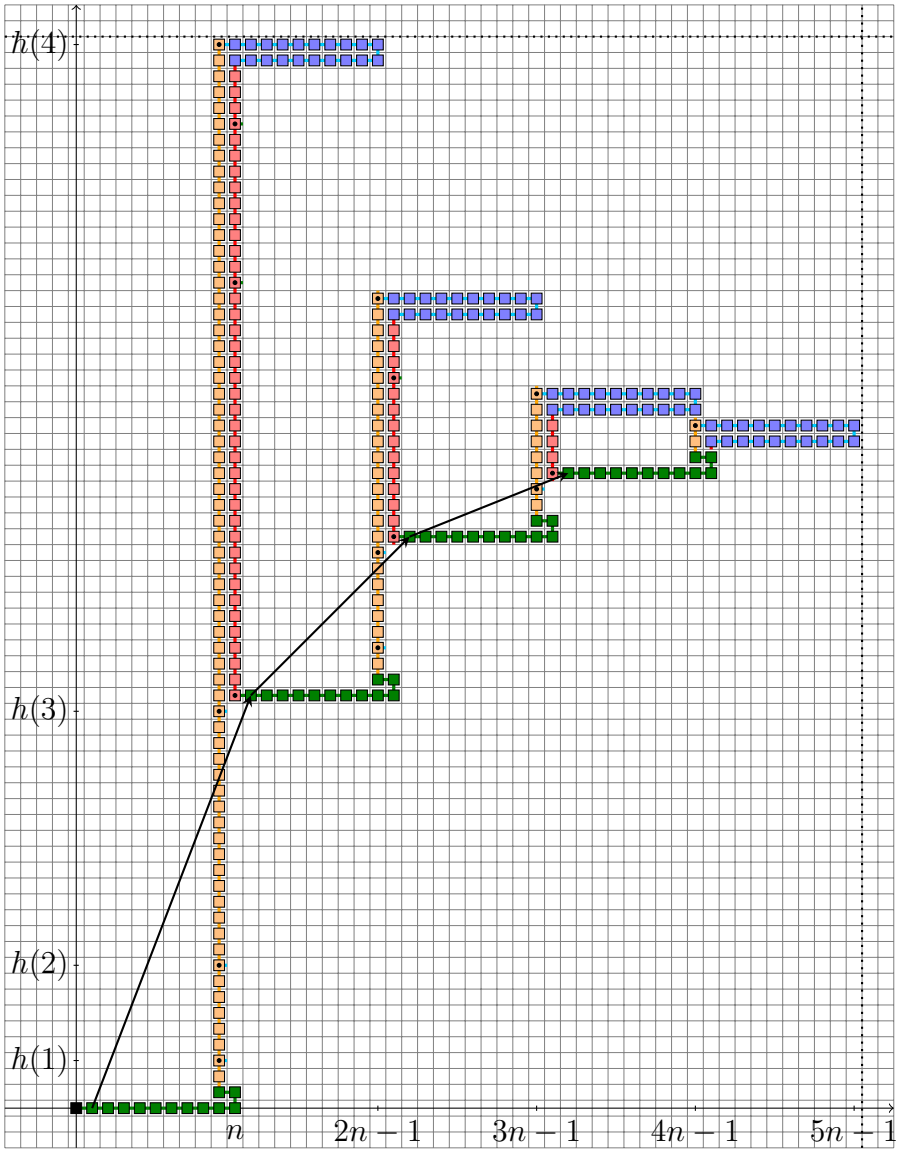


Figure 2.23: The efficient path E of horizontal width $(n+1)k-1$ which appears in any terminal assembly. Several smaller replicas of P are glued together.

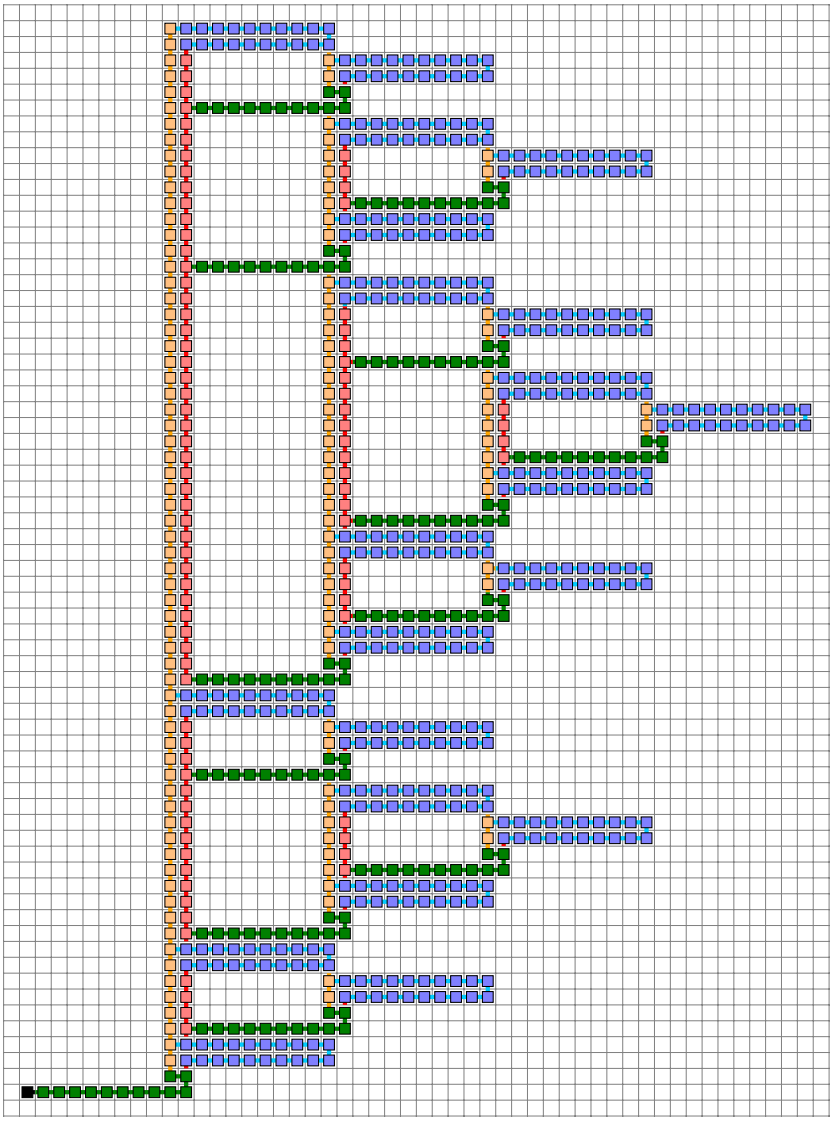


Figure 2.24: One the possible terminal assembly.

- the path G (in green in Figure 2.22) grows from the seed by n steps in direction of the east. A small "hook" grows at the end of G and the only free glue of G is at its end. The hook and the tile set are designed such that any assembly growing on the free glue at the end of G will stay to the north-east of G.

- the path Y (in yellow in Figure 2.22) grows from the free glue of G in direction of the north. Its length is $h(k)$ and there are k free glues on the east sides of the $h(1)th$, $h(2)th$, ... and $h(k)th$ tiles of G. The tile set is designed such that any assembly growing on one of these free glues will stay to the east of Y.

- the path B (in blue in Figure 2.22) grows from the kth free glue at the end of path Y. This path is a large "hook" and the only free glue of B is at its end. This hook and the tile set are designed such that any assembly growing on the free glue at the end of B will stay to the south-east of B.

- the path R (in red in Figure 2.22) grows from the free glue of B in direction of the south. Its length is $s(k-1)$ and there are $k-1$ free glues on the east sides of the $s(1)th$, $s(2)th$, ... and $s(k-1)th$ tiles of R. The tile set is designed such that any assembly growing on one of these free glues will stay to the east of R.

Now, a smaller replica of P (where the length of Y is only $h(k-1)$ and the length of R is only $s(k-2)$) can grow from the free glue at the end of P (the $(k-1)th$ free glue of R). Then an even smaller replica of P can grow at the end of the first replica and this operation can be repeated $k-1$ times in total creating a path E of horizontal width $(n+1)k-1$ (see Figure 2.23). Moreover E is efficient. Indeed, when an assembly grows on the ith free glue of Y or of one of its replica, then the assembly has to start by growing a replica of B protecting the efficient path (functions h and s are designed such that the efficient path is over this free glue). A similar reasoning can be done for the free glues of R. Figure 2.24 shows one of the terminal assemblies which contains E. When choosing the parameters n and k wisely, the efficient path has horizontal width $\Theta(|T|\log|T|)$ (see [30], for the details of the calculus).

2.5.3 Conclusion and ongoing work

Collisions allow a kind of computation which cannot be done by finite automata. Currently, we did not manage to obtain a similar result for the directed case. In fact, we are almost sure that there exists no finite assembly of horizontal width $\Theta(|T| \log |T|)$ in the directed case. We hope to prove this result soon. The tool developed for this upcoming result may lead to an improvement of the upper bound for the pumping lemma in the non-directed case. Ultimately, we aim to prove that there exists an "easy to describe" terminal assembly for any non-directed tile assembly system at temperature 1.

Chapter 3

Self-Stabilization

This chapter is a summary of four articles [4, 5, 18, 38]. This document aims to regroup their results and to present the open questions. Details of the proof are omitted since the papers are self-content.

3.1 Introduction

Crystallography focusses on periodic and organized structures made of atoms. An important breakthrough occurred in this domain when non-periodic organized structures where observed, see [45]. Aperiodic tilings became a theoretical model of these objects called quasi-crystals. A classical example of such an aperiodic tiling is the *Penrose* tiling whose tiles are rhombus. Unfortunately, it is not possible to grow efficiently a Penrose tiling by self-assembly. Indeed, when the tiles are assembled carelessly a mismatch may occur.

We proposed in [4] another approach based on self-stabilization. Inspired by C. Janot [22], the aim is to model a cooling process where at "high temperature", the constraints imposed by the glues are negligible. In this case, only the shapes of the tiles are relevant. When the temperature decreases, the glues start to influence the interactions and the tiles move in order to minimize the number of mismatches in their neighborhood. These movements are called *flips*. The aim of this section is to present a stochastic process which at each time step, selects randomly a flip in order to erase the mismatches after a reasonable time. More details about this approach are available in [4].

This cooling process relies on the Minority Rule, previously studied on

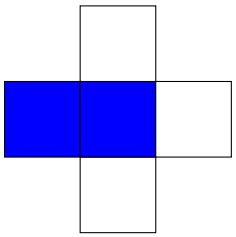
stochastic cellular automata (some of these results are quickly summarized in section 3.2). Instead of directly studying the Penrose tiling, we started by studying simpler tilings which belong to the family of tilings by cut and projection. In [8], N. G. de Bruijn interpreted the Penrose tiling as a projection of a space of dimension 5 on a plane of dimension 2. This reasoning can be extended for any $n > d$ to obtain a tiling called $n \to d$ which is the projection of a space of dimension n onto a surface of dimension n. Depending on n and d, these tilings can be either periodic or aperiodic, more details are available in [4]. We present the analysis of the Minority rule on the tiling $3 \to 2$, called the dimer tiling, in section 3.3. In this first example, the cooling process is able to obtain a periodic tiling after erasing the mismatches in polynomial time according to the number of tiles. In section 3.4, we present the analysis of the tiling $2 \to 1$. This tiling is simple but we study it in a more general setting where the different kinds of tiles have different densities. In section 3.5, we conclude by presenting the perspectives of these studies.

3.2 Minority Rule

The $2D$ Minority rule on cellular automata with von Neumann neighborhood and asynchronous updates was studied in [39]. Consider $n, m \in \mathbb{N}$ and a grid of automata of size $n \times m$ with periodic boundary conditions: an automaton is a couple (i, j) with $0 \leq i \leq n-1$ and $0 \leq j \leq m-1$ and its five neighbors are the automaton itself and $(i+1 \mod n, j)$, $(i, j+1 \mod m)$, $(i-1 \mod n, j)$ and $(i, j-1 \mod m)$. Each automaton is characterized by a state 0 (white) or 1 (blue) and a *configuration* associated a state to each automata, see Figure 3.1. Time is discrete and at each time step, an automaton is selected randomly and uniformly. This automaton is updated and its state switches to the minority state in its neighborhood. An automaton is *active* if and only if its state changes when it is updated otherwise the automaton is *inactive*. A configuration is *stable* if all automata are inactive.

Starting from some initial random configuration, the simulations show two successive behaviors, see Figure 3.2. Firstly, some different regions made of checkerboard patterns quickly appear and spread, erasing the initial configuration. All the automata inside these regions are inactive. The only active automata are on the borders between these regions. Secondly, these borders will move until the configuration becomes stable. This event occurs when either the borders manage to stabilize or a unique region absorbs all

(a) initial configuration

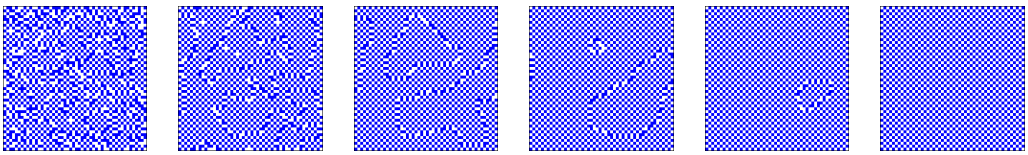


(b) Von Neumann neighborhood

Number of neighbors in state 1	0	1	2	3	4	5
State	1	1	1	0	0	0

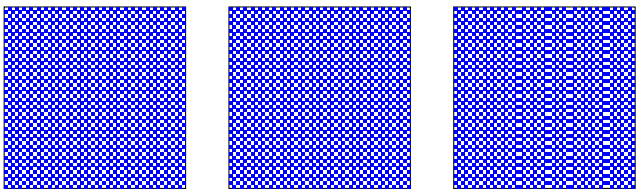
(c) the transition function of the Minority rule.

Figure 3.1: The definition of the Minority rule on a two dimensional grid with the Von Neumann neighborhood. Each automaton of the initial configuration (a) has a probability $\frac{1}{2}$ to be in the state 0 (white) and $\frac{1}{2}$ to be in the state 1 (blue). The Von Neumann neighborhood (b) of an automaton includes the automaton itself and its four nearest neighbors. When updated, an automaton follows the transition table (c): its new state depends on the number of automata in its neighborhood in the state 1. For the example, the neighborhood (b) has two automata in the state 1 and then its central automaton is inactive: when updated its state will not change.



$t = 3$ $t = 10$ $t = 30$ $t = 100$ $t = 218$ $t = 247$

Figure 3.2: The evolution of the initial configuration of Figure 3.1(a) for the Minority rule when only one random automaton is updated as each time step. The configurations are obtained after t iterations (one iteration is 2500 time steps). After 247 iterations, the configuration is stable. These figures were obtained with the software FiatLux [16].



(a) (b) (c)

Figure 3.3: Three stable configurations. Configurations (a) and (b) have an energy of 0 and are made of a checkerboard pattern. Configuration (c) is made of vertical stripes of checkerboard patterns.

the other ones.

Among the stable configurations, two of them of them are special. They are the configurations made of a checkerboard pattern: the one where the state of automaton (i, j) is 0 if and only if $i + j = 0 \mod 2$ and the one where the state of automaton (i, j) is 1 if and only if $i + j = 0 \mod 2$, see Figure 3.3. For these two configurations, the state of each automaton is different from the state of the four others automata in its neighborhood. The other stable configurations are made of vertical or horizontal stripes of checkerboard patterns.

The dynamics of the random process can be explained by introducing an energy function defined as follow: each automaton receives a potential equals to the number of automata in its neighborhood (excluding itself) with the same state as itself. The energy of a configuration is the sum of the potential of all its automata. Since the potential of an automata is between 0 and 4, the energy of a configuration is between 0 and $4nm$. The two configurations of energy 0 are the two stable configurations made of an unique checkerboard pattern. The two configurations of energy $4nm$ are the one where all automata are in the state 0 and the one where all automata are in the state 1. Moreover, the energy is non-increasing along time. Indeed, the transition function of the Minority rule, see Figure 3.1(c), can be rewritten using the potential as follows:

Potential of the automaton	0	1	2	3	4
Automaton is active?	No		Yes		
Variation of energy when updated	0	0	0	-4	-8

Then, automata with a potential of 0 or 1 are inactive and updating such automata does not change the energy. When an automaton with a potential 3

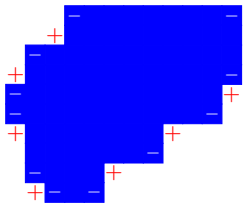


Figure 3.4: A zoom on the surrounded region of Figure 3.2 at $t = 218$ after switching the states of half of the automata. Now, this region appears as a blue convex polyomino. The $+$ (resp. $-$) denote the active automata which increase (resp. decrease) the size of the blue region when updated.

or 4 is updated then the energy drops and when an automaton of potential 2 is updated then the energy does not change (the automaton is still active after such an update and the transition is reversible). The emergence of the checkerboard patterns is explained by local arguments, see [37]: the energy quickly drops when automata of potential 3 and 4 of the initial configuration are updated. Later, generally all automata have a potential of 2 or less and then the energy cannot decrease in one time step. Here, the borders between the different regions must move in order to create an automaton of potential 3 or 4, allowing the energy to drop.

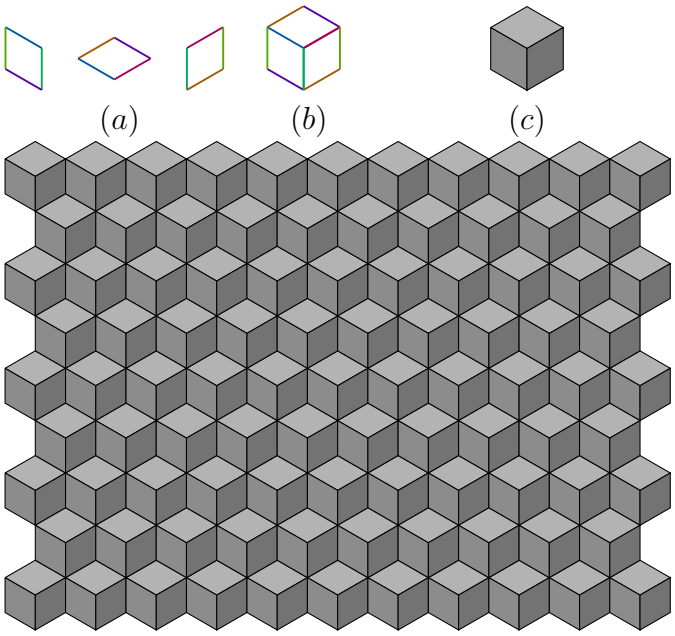
When a region is surrounded by another one, as in Figure 3.2 at $t = 218$, it quickly shrinks and disappears. To understand this special case, consider Figure 3.4 which is obtained by switching the state of all automata (i, j) where $i + j = 1 \mod 2$ in Figure 3.2 at $t = 218$. The surrounded region becomes a finite blue polyomino of size A which is convex in this example. In such a case, the automata of potential 2 are in the angles of this polyomino. Updating an automaton in a reflex (resp. salient) angle increases (resp. decreases) the size of the polyomino. Since along a polyomino, there are four more salient angles than reflex angles, then its size tends to decrease. This reasoning can be generalized for any number of surrounded regions, see [39]. Thus, the size of the surrounded region can be coupled with a biased random walk and it will reach 0 after updating $\Theta(A)$ active automata on expectation.

We studied the Minority rule with different topologies and generally the dynamics tends to converge quickly to a stable configuration made of an uniform pattern [40]. Nevertheless, in some cases, the convergence time can

be exponentiel on expectation [44].

3.3 A periodic tiling via self-stabilization

A dimer tiling is the projection of a space of dimension 3 on a $2D$ plane. It is made of three tile types which represent the three visible sides of a cube in side view, see Figure 3.5. These three tile types can assemble the tiling of Figure 3.5(d) which can be interpreted as the $2D$ plane covered by a layer of cubes aligned one next to each other. Note that this tiling is periodic and in [18], we used it as a first step to develop our process before considering more complex cases.

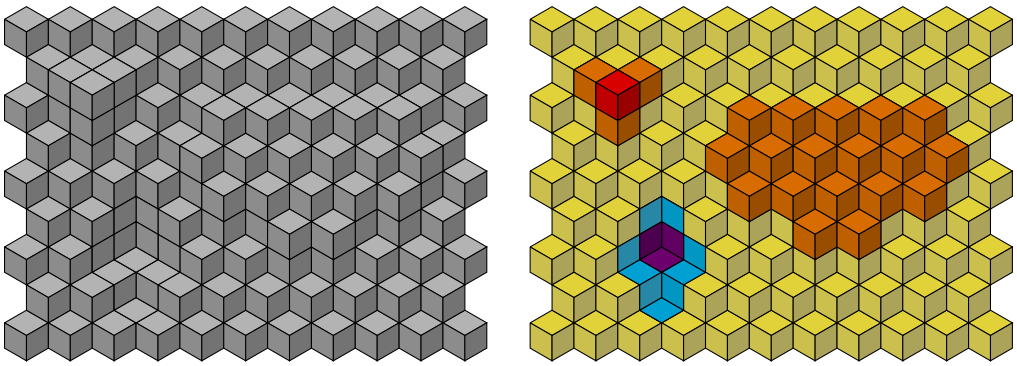


(d) A dimer tiling with no mismatch

Figure 3.5: The three tile types of a dimer tiling (a) can assemble the three visible sides of a cube (b). At high temperature the glues become negligible (c).

At high temperature, only the shapes of the tiles matter. Then, other tilings may be assemble such as the one in Figure 3.6 (a). This tiling is, in fact, made of several layers of cubes (see Figure 3.6(b), where a color is as-

signed to each layer). Intuitively, some cubes are missing, creating some holes while some other cubes are stacked creating some islands. In our study, we consider an area made of A tiles where the holes and islands are surrounded by tiles of the same layer. We consider that this layer has a height of 0. Then, the maximum (resp. minimum) height of an island (resp. hole) is bounded by $\sqrt{A}$ (resp. $-\sqrt{A}$).



(a) (b)

Figure 3.6: A dimer tiling with mismatches (a). For each tile type, this tiling has the same number of tiles as the one of Figure 3.5(d). In (b), the colors have been modified to show the different layers: the layer of height 0 is in yellow, height 1 is orange, height 2 is in red, height -1 is in blue and height -2 is in purple.

For a dimer tiling, a flip permutes three tiles, as shown in Figure 3.7, and it is active if and only if it does not increase the number of mismatches (the Minority rule). Intuitively, when a flip is done, a cube is either removed or put over the others. Then, these flips can be used to remove the islands and fill the holes. The dynamics of the Minority rule selects randomly and uniformly at each time step a flip in the area, the flip is done if and only if it is active. Experimentally, this dynamics seems to correct all mismatches in reasonable time.

The analysis is similar to the one on the $2D$ grid (see section 3.2) except that there are now several layers. Note that the active flips of Figure 3.7 cannot increase (resp. decrease) the maximum (resp. minimum) height. Then, we just need to focus on the layers of maximum and minimum height. The islands and holes of these layers can be formally defined by introducing a border between the tiles with a mismatch (see Figure 3.8). Here, instead of

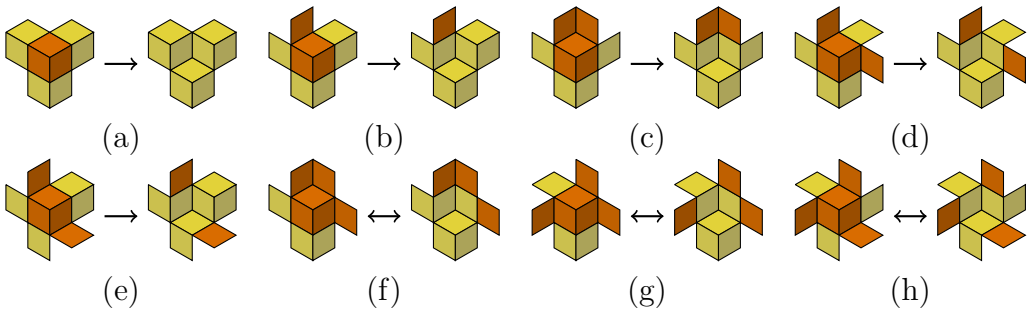


Figure 3.7: The different active flips up to some symmetries. There is a mismatch between two tiles if and only if one is orange and the other one is yellow. Note that flip (a) corresponds to the case where the last cube of an island of size 1 is removed. For all the other active flips, the cube is in contact with two layers of height i and $i+1$. Thus, it is not possible to increase (resp. descrease) the maximum (resp. minimum) height with an active flip.

a polyomino made of squares, the top islands and bottom holes are made of hexagons. As previously, the only active flips of the top islands and bottom holes are near their borders, their interiors are stable. Consider the example of Figure 3.8 where there is one convex island. In this simple case, there are more flips decreasing the size of the island than flips increasing it. This reasoning can be generalized for any kind and number of top islands or bottom holes, see [18]. Then, the top islands (resp. the bottom holes) tend to shrink and they will disappear after updating $\Theta(A)$ active flips on expectation in the top islands (resp. bottom holes)[1]. When such an event occurs either the maximum height decreases or the minimum height increases. Then after updating $\Theta(A^{\frac{5}{2}})$ active flips on expectation, there is only one layer of height 0 left. This layer is the dimer tiling with no mismatch of figure 3.5(d).

3.4 Two-letters words

A tiling $2 \to 1$ is the projection of a two dimensional space on a line. In other words, it is the discretization of a continuous line, see Figure 3.9(a). Consider a two dimensional grid, a line ℓ of slope α and an alphabet $\{a, b\}$

[1]Note that in the worst case, there is a probability $\frac{1}{A}$ to select a flip in the top islands or in the bottom holes

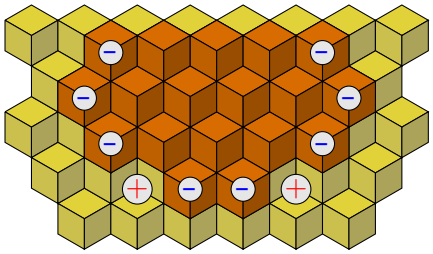


Figure 3.8: A zoom on the large island of layer 1 of Figure 3.6. On the one hand, two active flips (in red) will increase the size of the island by one if selected. On the other hand, eight active flips (in blue) will decrease the size of the island by one if selected. Then the size of the island tends to decrease.

such that the letter a (resp. b) encodes an intersection between ℓ and an horizontal line (resp. vertical line) of the grid. Then, ℓ can be transformed into a bi-infinite two-letters word w representing the sequence of intersections between ℓ and the grid. Moreover, if the letter a (resp. b) is transformed into a vertical (resp. horizontal) segment then the word w becomes a path of the grid which is a good approximation of ℓ, see Figure 3.9(b). If the slope α is rational then the word w is periodic and called a *Christoffel* word otherwise it is a *Sturmian* word [3].

In our context, letters a and b are also seen as tiles with glues on their left and right sides which assemble a tiling. This can be achieved only when the slope of the line is 1: consider the two tile types of Figure 3.9(c). In [5], we have analyze the cooling process on such a tiling. Nevertheless, in [38] we have generalized this process for any Christoffel word and we obtained a negative result for Sturmian words.

Consider a $n \times n$ grid, a configuration is a word w of length $2n$ with $|w|_a = |w|_b = n$. This word is equivalent to a path P of the grid from $(0, 0)$ to (n, n) of length $2n$, see Figure 3.10(b). A flip switches two consecutive letters: $ab \to ba$ and it is active if and only if it decreases the number of mismatches. Here, there are two kinds of active flip up to some symmetries: $aabb \to abab$ which removes two mismatches and $aaba \to abaa$ which does not change the number of mismatches, see Figure 3.10(a). Simulations show that this dynamics efficiently removes the mismatches. To analyze it, remark that the path P is inside an area delimited by two lines ℓ and ℓ' of slope 1, see Figure 3.10(b). The *thickness* of P (and w) is defined as the distance between ℓ and ℓ'. The thickness is non-increasing over time and the configuration

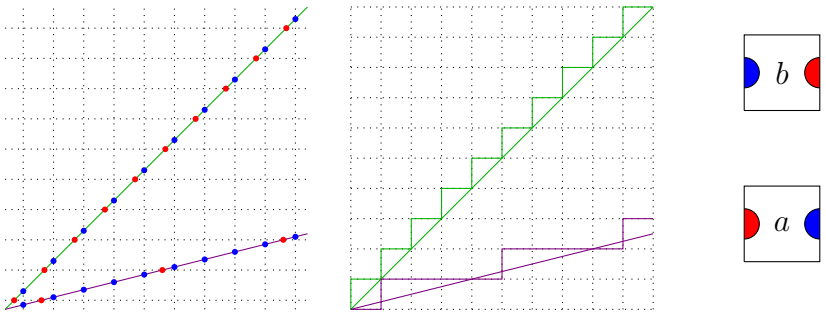


Figure 3.9: In (a), we consider a line ℓ (resp. ℓ') of slope 1 in green (resp. $\frac{1}{4}$ in purple) and a $2D$ grid. The intersections between these two lines and an horizontal (resp. vertical) line of the grid are indicated by a red (resp. blue) dot. For ℓ (resp. ℓ'), the sequence of dots along ℓ (resp. ℓ') is a word $w = \dots ababababababababab\dots$ (resp. $w' = \dots babbbbabbbbab\dots$) on a two-letters alphabet $\{a, b\}$ with a for a red dot and b for a blue dot. In (b), the words w and w' are transformed into a path of the grid where a (resp. b) is a vertical (resp. horizontal) segment. These paths are a good discrete approximation of the continuous line. The word w can be interpreted as a tiling made of the two tiles type of (c).

with minimum thickness is the stable configuration with no mismatch, *i.e.* the Christoffel word associated to a line of slope 1. In the simple case of Figure 3.10(b), the number of intersections between P and ℓ can be coupled with a random walk [2]. Then after $O(n^3)$ updates on expectation, there is no more intersection between the configuration and ℓ, thus the thickness has decreased. Since the thickness of a configuration can decrease at most $O(n)$ times, the process converges after $O(n^4)$ updates on expectation.

This cooling process can be extended to any Christoffel word. Indeed, consider t_a, t_b two relatively prime numbers, $n \in \mathbb{N}$ and $m = n(t_a + t_b)$, a configuration is either a word w of length m such that $|w|_a = nt_a$ and $|w|_b = nt_b$ or a path P of the grid from $(0, 0)$ to (nt_b, nt_a) of length m. In this case, we cannot consider anymore that a configuration is a tiling using the tiles type of Figure 3.9(c). The thickness of a configuration can be defined similarly, see Figure 3.11 and an active flip should not be able to increase it.

[2]Note that the reasoning presented here is inspired by [38], the one done in [5] is different and provides a better bound but it is also more complicated.

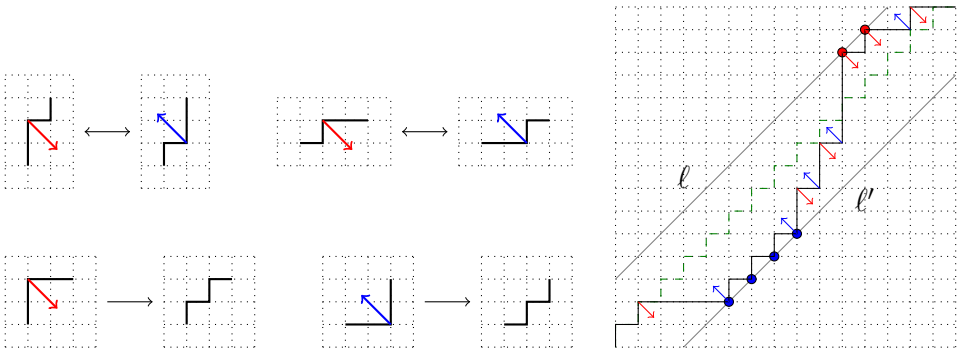


Figure 3.10: The active flips are listed in (a). The configuration of (b) is included between the two lines ℓ and ℓ'. Active flips are marked by an arrow and the intersections between the configuration and ℓ (resp. ℓ') are marked by red (resp. blue) dots. The stable configuration is in dark green.

To achieve this goal, we need more information. Consider an integer $s \in \mathbb{N}$ and two words w', w'' of length s, we define a test $\delta(w', w'')$, explained in Figure 3.12(a). To determine if a flip ab is active, we compute $\delta(w_l, w_r)$ and $\delta(w_r, w_l)$ where w_l (resp. w_r) is the left (resp. right) neighborhood of size s of the flip, see Figure 3.12(b). If one of these two tests is positive then the flip is active. If the size s of the neighborhood is greater than $t_a + t_b$ then an active flip cannot increase the thickness of the configuration, see Figure 3.13. For the case where $s = 2$ and $t_a = t_b = 1$, we obtain the active flips and the process of Figure 3.10. A similar analysis shows that the cooling process converges towards the Christoffel configuration in $O(m^4)$ updates on expectation, see [38] for the details.

To conclude, we have also shown in [38] that s should be at least $t_a + t_b - 1$ to develop such a cooling process. Otherwise either the Christoffel word is not stable or there are other stable configurations. Then, this approach cannot be extended to Sturmian words. Also, we were initially trying to develop a cooling process to explain the formation of a crystal but the process described here is linked to two other problems in distributed computing. The first one is to maintain a chain of relays between an explorer and a base camp [25] where several agents aim to align on a line. Defined in the continuous case, the Go-To-The-Middle strategy [14] allows the agents to reach their objective by communicating with only their two nearest neighbors. Our result can be

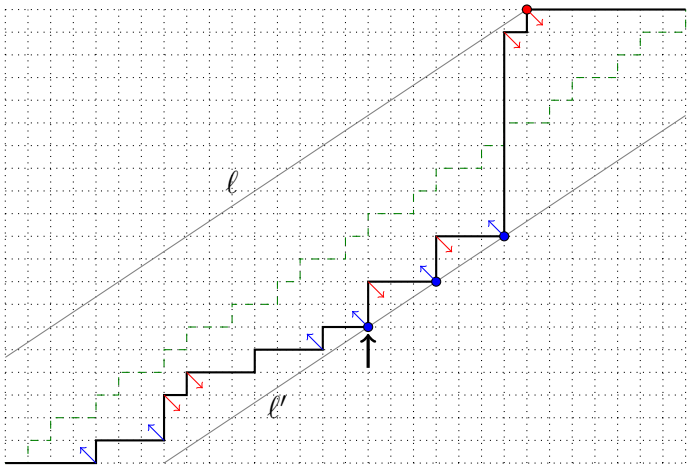
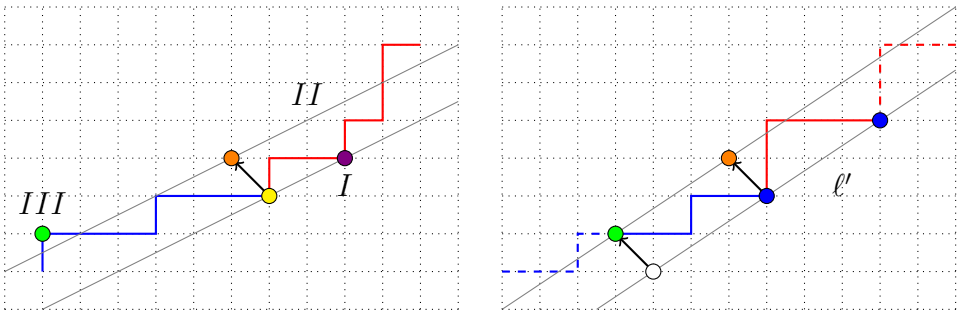


Figure 3.11: A case with $t_a = 2$, $t_b = 3$ and $n = 10$, the configuration is included between the two lines ℓ and ℓ' of slope $\frac{2}{3}$. The stable configuration is in dark green. Active flips are computed for $s = 5$ and are marked by an arrow. The intersections between the configuration and ℓ (resp. ℓ') are marked by red (resp. blue) dots. The Figure 3.12(b) is a detailed analysis of the flip pointed by an arrow.

seen as a discretization of this strategy. Secondly, this result is linked to the density classification problem [15]. This problem consider a line of agents which can only memorized one bit of information. They must determine if there is a majority of agents with the bit 0 or with the bit 1 in the initial configuration. Our result can be seen as a generalization of this problem for an arbitrary density. Indeed, for a given s, the cooling process will converge for any relatively prime numbers t_a and t_b such that $t_a + t_b \leq s$. Since a Christoffel word is periodic of period $t_a + t_b$ then after the convergence, the periodic pattern can be read locally by the agents to determine t_a and t_b. For the example of Figure 3.11 with $t_a = 2$ and $t_b = 3$, the configuration is the Christoffel word $(babab)^n$ when the cooling process has converged. The agents are then able to determine that the periodic pattern is *babab* and that $t_a = 2$ and $t_b = 3$.

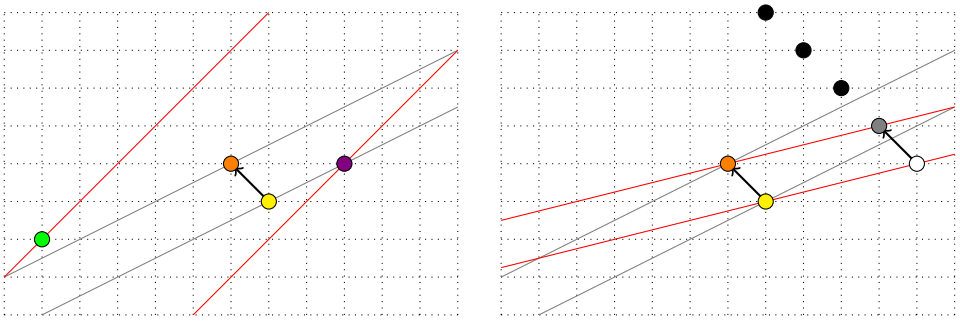


(a) (b)

Figure 3.12: In Figure (a), the flip moves the site in yellow to the position in orange. In this example, $s = 8$ and the right (resp. left) neighborhood is in red (resp. blue). Firstly, the line (in gray) passing by the yellow site and tangent to the right neighborhood is drawn. Here, this line passes by the purple site and its slope is $\frac{1}{2}$. Secondly, a line (in gray) passing by the orange site and parallel to the previous one is drawn. Thirdly, the flip is active if there exists at least one site in the left neighborhood which is on or over the second line (as the green one in this example). The test is also applied similarly by switching the right and left neighborhoods. In Figure (b), this test is applied to the flip pointed by an arrow in Figure 3.11. Here, $s = 5$ and the tangent passes by two sites (in blue) which are on ℓ' then its slope is $\frac{2}{3} = \frac{t_a}{t_b}$. On the left neighborhood, the site at distance $5 = t_a + t_b$ (in green) is not on ℓ' (the white position is empty) then this site will allow the flip to be active.

3.5 Conclusion

These results show that it is possible to obtain periodic tilings via self-stabilization. For the aperiodic case, it is not possible to obtain such a result via a generalization of the tiling $2 \to 1$. For the Penrose tiling, the simulations are not conclusive. Currently, we are not able to determine if the Minority rule on a Penrose tiling does not converge, converges in exponentiel time or in polynomial time (with a bound larger than in the previous cases).



(a) (b)

Figure 3.13: An active flip cannot increase the thickness of a configuration. In the example of Figure 3.12(a), the slope of the tangent (in gray) is $\frac{1}{2}$. If $\frac{t_a}{t_b} \geq \frac{1}{2}$, see the case shown in Figure (a): the orange dot is between the two lines of slope $\frac{t_a}{t_b}$ (in red) passing by the purple and green dots. If $\frac{t_a}{t_b} < \frac{1}{2}$, see the case shown in Figure (b): on the one hand, the orange dot is on the line of slope $\frac{t_a}{t_b}$ (in red) passing by the gray site. The white site at distance $t_a + t_b$ is under the tangent thus the configuration must pass by one of the sites in gray or black. This means that ℓ is parallel and over the line in red passing by the gray dot. On the other hand, ℓ' is under the line in red passing by the yellow dot. Thus, the thickness cannot increase in both cases.

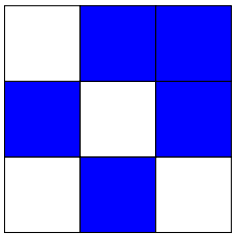
Chapter 4

Activity Report and Perspectives

My research mainly focuses on unconventionnel computational models. My list of publications is available in the annex A. I have published six articles in international peer-reviewed journals, one in a national peer-reviewed journal, twenty-two in international peer-reviewed conferences and two in national peer-reviewed conferences.

In 2006, I started my PhD by studying cellular automata under the supervision of Nicolas Schabanel and Éric Thierry at ENS Lyon. This model of computation is made of automata disposed on a d-dimensional grid such that all automata share the same neighborhood. Each automaton is characterized by a state. Time is discrete and at each time step, automata update their state according to the same transition function which takes as input the states in the neighborhood of the automaton. Usually, the grid is infinite and all automata are updated at each time step, this update mode is called *parallel*. In our case, we consider a finite grid with periodic boundary condition, the state is either 0 (white) or 1 (blue) and the update mode is α-*asynchronism* where each automaton has a probablity $0 < \alpha < 1$ to be updated and $1-\alpha$ to keep its current state. When $\alpha = 0$, this update mode is called *totally asynchronous*, only one automaton chosen at random is updated. We developed tools in order to study the Elementary Cellular Automata (ECA), a class of 256 one-dimensional cellular automata, and we were able to classify different kinds of behavior. The most interesting ones are phase transitions where two different behaviors are exhibited depending on the value of α. In some cases, we were able to prove the existence of these two phases but determining the

(a) initial configuration



(b) Moore neighborhood

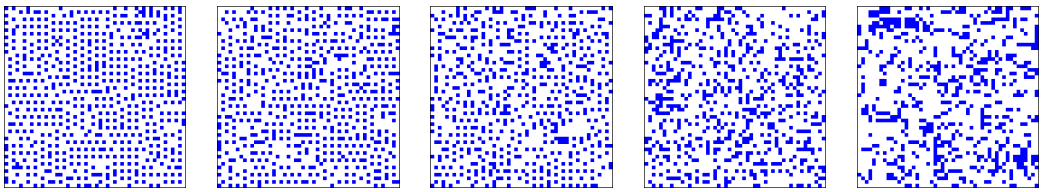
Number of neighbors in state 1	0	1	2	3	4	5	6	7	8	9
State	0	1	0	0	0	1	0	0	0	0

(c) the transition function of TOTM**34**.

Figure 4.1: The definition of a two dimensional cellular automaton, called *totalistic* **34** with *Moore* neighborhood (TOTM**34**). Each automaton of the initial configuration (a) has a probability $\frac{1}{2}$ to be in the state 0 (white) and $\frac{1}{2}$ to be in the state 1 (blue). The Moore neighborhood (b) of an automaton includes the automaton itself and its eight nearest neighbors. When updated, an automaton follows the transition table (c): its new state depends on the number of automata in its neighborhood in the state 1. For the example, the neighborhood (b) has five automata in the state 1 and then its central automaton is active: it will switch from state 0 to state 1 when updated.

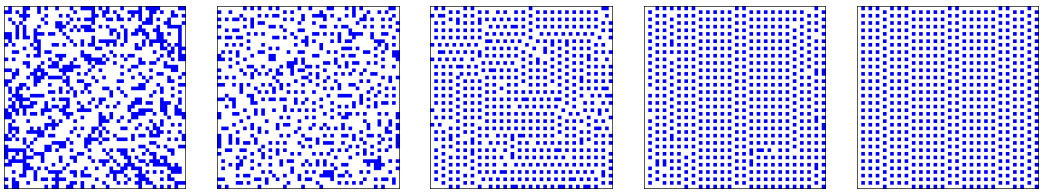
exact value of the transition is an open question linked to directed percolation. Following these studies, we looked for $2D$ cellular automata to study. The Minority rule caught our attention, see section 3. I was active on this topic from 2006 to 2013. I tried to relaunch these studies with the analysis of a $2D$ cellular automaton, called TOTM**34**, which presents a behavior similar to the Minority rule (see Figures 4.1, 4.2 and 4.3). Three students worked under my supervision on this cellular automaton during their internships for bachelor's degree or Master's degree. They made some progress but not enough for a publication. The analysis of this cellular automaton seems to require more sophisticated combinatorial and probabilistic tools. I am the co-author of three articles in international peer-reviewed journals and seven articles in international peer-reviewed conferences on this topic.

At the end of my PhD thesis, I encountered Thomas Fernique who was aiming to obtain tilings via self-stabilisation. The Minority rule seemed efficient to achieve this goal and for one year I joined his team at Université de



$\alpha = 5$ $\alpha = 10$ $\alpha = 20$ $\alpha = 50$ $\alpha = 100$

Figure 4.2: The evolution of the initial configuration of Figure 4.1(a) for cellular automaton TOTM**34** after 10000 time steps under α-asynchronous dynamics for different values of α. There is a phase transition between two different behaviors depending on the values of α. These figures were obtained with the software FiatLux [16].



$t = 1$ $t = 3$ $t = 100$ $t = 500$ $t = 1400$

Figure 4.3: The evolution of the initial configuration of Figure 4.1(a) for cellular automaton TOTM**34** under the totally asynchronous dynamics where only one random automaton is updated as each time step. The configurations are obtained after t iterations (one iteration is 2500 time steps). After 1400 iterations, the configuration is stable. These figures were obtained with the software FiatLux [16].

Provence as an ATER to work on this model. Later, I was recruited as an assistant professor at Université d'Évry-Val-d'Essonne joining the AROBAS team of the IBISC laboratory. Meanwhile T. Fernique obtained an ANR jeune chercheur funding (called Quasicool) to develop his thematic. My work among this community is described in section 3. We were able to model the formation of some periodic tilings by self-stabilisation. On dimension 1, we have negative results on two-letter words. The next step would be to investigate the Minority rule on an aperiodic tilings (such as Penrose Tiling). Concerning this topic, I was active from 2009 to 2018 but I am currently inactive due to time constraint. I am the co-author of one article in an international peer-reviewed journal and four articles in international peer-reviewed conferences or workshops on this topic.

While working on self-stabilisation, I met Pierre-Étienne Meunier and Damien Woods who were studying non-cooperative ATAM. From 2013 to 2021, I worked with them until we solved the main temperature 1 conjecture by showing a $2D$ pumping lemma, details of this collaboration are available in section 2. Combining this lemma with the previous result of Doty et al [13] implies that the directed non-cooperative ATAM is not able to do computations as complex as the ones done by a Turing Machine. Moreover, we have also built a non-directed non-cooperative tile assembly system which is able to build finite terminal assembly, called efficient path, of width $\Theta(n \log(n))$ where n is the size of the tile assembly system. Unfortunately, after publishing three articles in international peer-reviewed conferences, this collaboration ended due to practical constraints. Pierre-Étienne Meunier left the scientific community and Damien Woods moved from INRIA Paris to Maynooth University. To keep working on this topic, I have started a new local collaboration with Sergiu Ivanov from IBISC at UEVE. On the one hand, we are currently writing an article showing that the maximal width of a terminal assembly of a directed non-cooperative tile assembly system is bounded by $\Theta(n)$. This result would imply that the technique used for the efficient paths cannot be generalized to the directed case. Moreover, we hope that the new techniques developed for this result will lead to new results for the non-directed case. On the other hand, we are trying to develop a site of experimental manipulations in Évry. To achieve this goal, we were trained in May 2022 by Nicolas Schabanel and his PhD students who developed such a site at ENS Lyon. We used this experience to create a M2 lecture about self-assembly and the software ENSnano [26] in the Master Geniomhe for students in bio-informatics. In Évry, we contacted Marco Mendoza to have

access to his wet lab at Télécom Sud Paris and Guillaume Lamour from the laboratory LAMBE of UEVE trained us on their AFM. We hope to start a first round of manipulations soon in order to show the viability of this project.

When I was recruited at UEVE in 2010, I joined the AROBAS team specialized in algorithmic and I started working with Éric Angel. We study approximation algorithms and together we published one article in an international peer-reviewed journal, three articles in international peer-reviewed conferences or workshops and two articles in national peer-reviewed conferences. We supervised with Franck Ledoux the PhD thesis of Sébastien Morais on mesh partioning with memory constraints. In September 2022, we start supervising the PhD thesis of Xiechen Zhang with Feng Chu on multi-criteria and stochastic optimization for partitioning. I did not developed this axis in this document since it is not linked with my work on computation models.

Finally, I have worked with Sylvain Sené and Tarek Melliti from the COSMO team of IBISC on the dynamics of boolean automata networks (BANs). This model of computation is a generalization of cellular automata where the automata are disposed on a arbitrary graph instead of a grid. Moreover, each automata has its own neighborhood and transition function. BANs are used to model gene regulation networks. Similarly as cellular automata, this model is usually considered under the parallel dynamics. Recently other dynamics where studied such as the sequential one where automata are updated one after the other according to a predetermined order. I am the co-author of one article in an international peer-reviewed journal, five articles in international peer-reviewed conferences or workshops and one article in a national peer-reviewed journal. I did not developed this topic in this document since this model does not rely on dominoes. I was active on this topic from 2011 to 2016. Currently, these studies are on hold since Sylvain Sené leaved UEVE to become a full professor at Aix-Marseille Université where he founded the CANA team in the laboratory LIF. We hope to start this collaboration again soon. In 2021, we supervised Lucas Venturini during its internship for his Master's degree. Unfortunately due to Covid, it was not followed by a PhD thesis.

Bibliography

[1] Bahar Behsaz, Ján Maňuch, and Ladislav Stacho. Turing universality of step-wise and stage assembly at Temperature 1. In *DNA: Proceedings of 18th International Conference on DNA Computing and Molecular Programming*, volume 7433 of *LNCS*, pages 1–11. Springer, 2012.

[2] Robert Berger. *The Undecidability of the Domino Problem*. American Mathematical Society, 1966.

[3] Jean Berstel, Aaron Lauve, Christophe Reutenauer, and Franco Saliola. *Combinatorics on Words: Christoffel Words and Repetition in Words*, volume 27 of *CRM monograph series*. American Mathematical Society, 2008.

[4] Olivier Bodini, Thomas Fernique, and Damien Regnault. Cristallization by stochastic flips. *Proceedings of Aperiodics 2009, Journal of Physics: conference series*, 226(012022):131–136, 2010.

[5] Olivier Bodini, Thomas Fernique, and Damien Regnault. Stochastic flip of two-letters words. In *ANALCO: Proceedings of the 7th Workshop on Analytic Algorithmics and Combinatorics*, pages 48–55. SIAM, 2010.

[6] Luis Ceze, Jeff Nivala, and Karin Strauss. Molecular digital data storage using DNA. *Nature Reviews Genetics*, 20(8):456–466, 08 2019.

[7] Matthew Cook, Yunhui Fu, and Robert T. Schweller. Temperature 1 self-assembly: deterministic assembly in 3D and probabilistic assembly in 2D. In *SODA: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 570–589. SIAM, 2011. Arxiv preprint: `arXiv:0912.0027`.

[8] Nicolaas Govert de Bruijn. Algebraic theory of penrose's non-periodic tilings of the plane. i,ii. *Indagationes Mathematicae (Proceedings)*, 84(1):39–52, 53–66, 1981.

[9] Erik D. Demaine, Martin L. Demaine, Sándor P. Fekete, Matthew J. Patitz, Robert T. Schweller, Andrew Winslow, and Damien Woods. One tile to rule them all: Simulating any tile assembly system with a single universal tile. In *ICALP: Proceedings of the 41st International Colloquium on Automata, Languages, and Programming*, volume 8572 of *LNCS*, pages 368–379. Springer, 2014. Arxiv preprint: `arXiv:1212.4756`.

[10] Erik D. Demaine, Matthew J. Patitz, Trent A. Rogers, Robert T. Schweller, Scott M. Summers, and Damien Woods. The two-handed tile assembly model is not intrinsically universal. In *ICALP: Proceedings of the 40th International Colloquium on Automata, Languages, and Programming*, volume 7965 of *LNCS*, pages 400–412. Springer, July 2013. Arxiv preprint: `arXiv:1306.6710`.

[11] David Doty, Jack H. Lutz, Matthew J. Patitz, Robert T. Schweller, Scott M. Summers, and Damien Woods. The tile assembly model is intrinsically universal. In *FOCS: Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science*, pages 439–446. IEEE Computer Society, 2012. Arxiv preprint: `arXiv:1111.3097`.

[12] David Doty, Jack H. Lutz, Matthew J. Patitz, Scott M. Summers, and Damien Woods. Intrinsic universality in self-assembly. In *STACS: Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science*, volume 5 of *LIPIcs*, pages 275–286, 2010. Arxiv preprint: `arXiv:1001.0208`.

[13] David Doty, Matthew J. Patitz, and Scott M. Summers. Limitations of self-assembly at temperature 1. *Theoretical Computer Science*, 412(1–2):145–158, 2011. Arxiv preprint: `arXiv:0903.1857v1`.

[14] Miroslaw Dynia, Jarosław Kutyłowski, Paweł Lorek, and Friedhelm Meyer auf der Heide. Maintaining communication between an explorer and a base station. In *BICC: Proceedings of Biologically Inspired Cooperative Computing, IFIP 19th World Computer Congress*, volume 216 of *IFIP*, pages 137–146. Springer, 2006.

[15] Nazim Fatès. Stochastic cellular automata solve the density classification problem with an arbitrary precision. In *STACS: Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science*, volume 9 of *LIPIcs*, pages 284–295, 2011.

[16] Nazim Fatès, Nicolas Gauvi, Émilien Brun, Océane Chazé, Olivier Bouré, Nikolaos Vlassopoulos, Antoine Spicher, and Alexandre Bryskowski. Fiatlux. https://project.inria.fr/fiatlux/.

[17] Sándor P. Fekete, Jacob Hendricks, Matthew J. Patitz, Trent A. Rogers, and Robert T. Schweller. Universal computation with arbitrary polyomino tiles in non-cooperative self-assembly. In *SODA: Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 148–167. SIAM, 2015.

[18] Thomas Fernique and Damien Regnault. Stochastic flip on dimer tilings. In *AofA: Proceedings of the 21st International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms*, volume AM, pages 207–220. DMTCS proceedings, 2010.

[19] David Furcy and Scott M. Summers. Optimal self-assembly of finite shapes at temperature 1 in 3D. *Algorithmica*, 80(6):1909–1963, 2018.

[20] David Furcy, Scott M. Summers, and Christian Wendlandt. New bounds on the tile complexity of thin rectangles at temperature-1. In *DNA: Proceedings of the 25th International Conference on DNA Computing and Molecular Programming*, volume 11648 of *LNCS*, pages 100–119. Springer, 2019.

[21] Oscar Gilbert, Jacob Hendricks, Matthew J Patitz, and Trent A Rogers. Computing in continuous space with self-assembling polygonal tiles. In *SODA: Proceedings of the 27h Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 937–956. SIAM, 2016. Arxiv preprint: `arXiv:1503.00327`.

[22] Christian Janot. *Quasicrystals, a Primer*. Oxford: Oxford University Press, 1992.

[23] Emmanuel Jeandel and Michaël Rao. An aperiodic set of 11 wang tiles. 2015. Arxiv preprint: `arXiv:1506.06492v4` [cs.CC].

[24] Nataša Jonoska and Daria Karpenko. Active tile self-assembly, part 1: Universality at temperature 1. *International Journal of Foundations of Computer Science*, 25(2):141–164, 2014.

[25] Jarosław Kutyłowski and Friedhelm Meyer auf der Heide. Optimal strategies for maintaining a chain of relays between an explorer and a base camp. *Theoretical Computer Science*, 410:3391–3405, 2009.

[26] Nicolas Levy and Nicolas Schabanel. Ensnano. http://www.ens-lyon.fr/ensnano/.

[27] Pierre-Étienne Meunier. Non-cooperative algorithms in self-assembly. In *UCNC: Proceedings of 14th Unconventional Computation and Natural Computation*, volume 9252 of *LNCS*, pages 263–276. Springer, 2015.

[28] Pierre-Étienne Meunier, Matthew J. Patitz, Scott M. Summers, Guillaume Theyssier, Andrew Winslow, and Damien Woods. Intrinsic universality in tile self-assembly requires cooperation. In *SODA: Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 752–771. SIAM, 2014. Arxiv preprint: `arXiv:1304.1679`.

[29] Pierre-Étienne Meunier and Damien Regnault. A pumping lemma for noncooperative self-assembly. 2015. Arxiv preprint: `arXiv:1312.6668v4`.

[30] Pierre-Étienne Meunier and Damien Regnault. Non-cooperatively assembling large structures. In *DNA: Proceedings of the 25th International Conference on DNA Computing and Molecular Programming*, volume 11648 of *LNCS*, pages 120–139. Springer, 2019.

[31] Pierre-Étienne Meunier and Damien Regnault. Directed Non-Cooperative Tile Assembly Is Decidable. In *DNA: Proceedings of the 27th International Conference on DNA Computing and Molecular Programming*, volume 205 of *LIPIcs*, pages 6:1–6:21, 2021.

[32] Pierre-Étienne Meunier, Damien Regnault, and Damien Woods. The program-size complexity of self-assembled paths. In *STOC: Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 727–737. ACM, 2020. Arxiv preprint: `arXiv:2002.04012` [cs.CC].

[33] Pierre-Étienne Meunier and Damien Woods. The non-cooperative tile assembly model is not intrinsically universal or capable of bounded Turing machine simulation. In *STOC: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 328–341. ACM, 2017. Arxiv preprint with full proofs: `arXiv:1702.00353v2`.

[34] Dionis Minev, Christopher M. Wintersinger, Anastasia Ershova, and William M. Shih. Robust nucleation control via crisscross polymerization of highly coordinated DNA slats. *Nature Communications*, 12(1741), 2021.

[35] Matthew J. Patitz, Robert T. Schweller, and Scott M. Summers. Exact shapes and Turing universality at Temperature 1 with a single negative glue. In *DNA: Proceedings of the 17th International Conference on DNA Computing and Molecular Programming*, volume 6937 of *LNCS*, pages 175–189. Springer, 2011. Arxiv preprint: `arXiv:1105.1215`.

[36] Lulu Qian, Erik Winfree, and Jehoshua Bruck. Neural network computation with DNA strand displacement cascades. *Nature*, 475(7356):368–372, 2011.

[37] Damien Regnault. Quick energy drop in stochastic 2D minority. In *ACRI: Proceedings of the 8th International Conference on Cellular Automata for Reseach and Industry*, volume 5191 of *LNCS*, pages 307–314. Springer, 2008.

[38] Damien Regnault and Éric Rémila. Lost in self-stabilization: A local process that aligns connected cells. *Theoretical Computer Science*, 736:41–61, 2018.

[39] Damien Regnault, Nicolas Schabanel, and Éric Thierry. Progresses in the analysis of stochastic 2D cellular automata: A study of asynchronous 2D minority. *Theoretical Computer Science*, 410:4844–4855, 2009.

[40] Damien Regnault, Nicolas Schabanel, and Éric Thierry. On the analysis of "simple" 2D stochastic cellular automata. *Discrete Mathematics & Theoretical Computer Science*, 12(2):263–294, 2010.

[41] Paul W. K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440(7082):297–302, March 2006.

[42] Paul W. K. Rothemund and Erik Winfree. The program-size complexity of self-assembled squares (extended abstract). In *STOC: Proceedings of the 32nd annual ACM Symposium on Theory of Computing*, pages 459–468. ACM, 2000.

[43] Paul W.K. Rothemund, Nick Papadakis, and Erik Winfree. Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biology*, 2(12):2041–2053, 2004.

[44] Jean-Baptiste Rouquier, Damien Regnault, and Éric Thierry. Stochastic minority on graphs. *Theoretical Computer Science*, 412(30):3947–3963, 2011.

[45] D. Shechtman, I. Blech, D. Gratias, and J. W. Cahn. Metallic phase with long-range orientational order and no translational symmetry. *Physical Review Letters*, 53:1951–1953, Nov 1984.

[46] David Soloveichik and Erik Winfree. Complexity of self-assembled shapes. *SIAM Journal on Computing*, 36(6):1544–1569, 2007.

[47] Hao Wang. Proving theorems by pattern recognition – II. *The Bell System Technical Journal*, XL(1):1–41, 1961.

[48] Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, June 1998.

[49] Bernard Yurke, Andrew J. Turberfield, Allen P. Mills, Friedrich C. Simmel, and Jennifer L. Neumann. A DNA-fuelled molecular machine made of DNA. *Nature*, 406(6796):605–608, 2000.

Appendix A

CV - Damien Regnault

Maître de conférences (Assistant Professor)
Address: IBISC, Université d'Évry-Val-d'Essonne (UEVE)
Institut de Bioligie Génétique et Bio-Informatique (IBGBI),
23 boulevard de France, 91037 Évry cedex, France
Tel: +33 1 64 85 34 77
Email: damien.regnault@univ-evry.fr
www: https://www.ibisc.univ-evry.fr/∼dregnault/
Spoken languages: French, English.

Academic history:

2005-2008: Preparation of a doctoral thesis at LIP, ENS Lyon.

Defended november 24th 2008 at École Normale Supérieure de Lyon.
Sur les automates cellulaires probabilistes:
comportements asynchrones.
PhD advisors: Nicolas Schabanel, Éric Thierry.
Referees: Alexander Shen, Bernard Ycart, Cristopher Moore
Jury : Jean Mairesse, Alexander Shen, Bernard Ycart, Pascal Koiran

2005: Master's degree in Computer Science, ENS Lyon.

2002-2006: *Élève Normalien*, École Normale Supérieure (ENS) de Lyon.

Employment record:

2010- *Maître de conférences* (Assistant Professor), IBISC, Université d'Évry Val-d'Essonne (UEVE).

2009-2010 *ATER*, LIF, Université de Provence.

2006-2009 *PhD Student*, LIP, École Normale Supérieure de Lyon.

Teaching

I started teaching as a moniteur at ENS Lyon from 2006 to 2009 doing 64 hours of tutorials per year. Then, I worked one year as ATER at université de Provence Aix-Marseille 1. From 2010, I am an assistant professeur (maître de conférence) at the computer science department of UEVE with an obligation of 192 hours of tutorials per year (one hour of lecture is counted as 1.5 hour of tutorial). On average, I have done 250 hours of tutorials per year (120% of my obligation).

Most of my teaching (112 hours per year) focuses on algorithmic: introduction to computer science (L1), introduction to algorithmics (L2), calculability and complexity (L3) and approximation algorithms (M2). These lectures are theoretical and focus on developing algorithms, proving their correctness and analyzing their complexity. Generally, there are two hundred students in L1 and around fifteen students in M2. Most of them come from the computer science department but some are from the department of mathematics, biology or law. Grabbing the attention of all these students requires to choose adequate examples. Otherwise, I am frequently doing tutorials and sometimes lectures on applied topics (networks, programming, database) when my help is required. Along the years, I have been involved in most teaching done during the three years of a licence's (bachelor's) degree.

This year, I have created with Sergiu Ivanov a new M2 lecture about self-assembly. It is addressed to students proficient in both computer science and biology. It relies on the software ENSnano developed by Nicolas Levy and Nicolas Schabanel.

Scientific popularization ("fête des sciences")

2014 – 2016	Creation of the stand "Tilings" (UEVE).
2013 – 2014	Exhibitor at the stand "Turing Machine" (UEVE).

Collective responsability

Sept 2022 –	In charge of M1 MIAGE apprentissage (UEVE).
Mars 2019 – Mars 2021	Substitute member of IBISC's council laboratory.
Sept 2014 – Sept 2018	Co-organizer of IBISC Seminar.
Sept 2012 – Sept 2019	In charge of L2 (UEVE).
Sept 2010 – Sept. 2011	In charge of L3 ASR (UEVE).
Nov. 2008 – Sept 2009	Co-organizer of MC2 team meeting.

Communications

I regularly review articles for international conferences (MFCS 2009, STACS 2009, ACRI 2012 and 2014, WAOA 2012, STACS 2019, SODA 2021 and 2023) and international journals (journal of cellular automata, NACO, TCS, TSI, Physica D: Nonlinear Phenomena, ACA, Neural Networks, Biosystems). I have presented my scientific results in approximately forty team or laboratory seminars and talks in conferences and workshops. In particular, I have been invited to the workshop PCA EURANDOM at Eindhoven (Netherlands) the 11th January 2013, to the workshop "Discrete Models of Complex Systems" at Orléans organized by *le Stadium* in March 2018 and the conference in honor of the 60 years of Éric Rémila at ENS-Lyon in May 2019.

Supervision (internship)

2020-2021	L. Venturini (Internship during 4th year of ENS-Lyon) *Poursuite des travaux démarrés durant le stage de M2.* Founding by FRR action 2
2020	M2 - L. Venturini (ENS-Lyon). *Analyse du comportement asynchrone de l'automate cellulaire TOTM34.* Founding by IBISC.
2014	L3 - M. Hassani (UEVE). *Recherche des configurations stables de l'automate cellulaire OTM 38.*
2013	L3 - J. Sobieraj (UEVE). *Isomorphismes et intersections de cycles dans les réseaux d'automates booléens asynchrones.* Published in UCNC 2015. Done with Tarek Melliti and Sylvain Sené.
2012	L3 - V. Verhille (ENSIIE). *Importance de la dynamique synchrone dans l'étude des réseaux d'automates booléens non monotones.* Done with Sylvain Sené.
2011	L3 - F. Rabin (ENSIIE). *Étude des circuits k-xor circulants.* Done with Sylvain Sené.

Supervision (thesis)

2013-2016: Preparation of a doctoral thesis by S. Morais at IBISC, UEVE.

Defended november 23th 2016 at Université Paris-Saclay.
Étude et obtention d'heuristiques et d'algorithmes exacts et approchés pour un problème de partitionnement de maillage sous contrainte mémoire.
PhD advisors: É. Angel (50%), F. Ledoux (25%), D. Regnault (25%).
Referees: C. Durr, F. Pellegrini.
Jury : C. Bazgan, I. Todinca, J.-C. Janodet.
Published Articles: International conference EuroPAR [4],
National conferences COMPAS [1] et ROADEF [2].
Sébastien Morais is currently working at CEA as an ingénieur-chercheur.

He is also working as a PAST in the computer science department of UEVE.

2022- : Preparation of a doctoral thesis by X. Zhang at IBISC, UEVE.

Optimisation multicritère et stochastique pour l'ordonnancement
PhD advisors: É. Angel (50%), F. Chu (25%), D. Regnault (25%).

Articles published in international peer-reviewed journal.

[1] Damien Regnault and Éric Rémila. Lost in self-stabilization: A local process that aligns connected cells. *Theoretical Computer Science (TCS)*, volume 736: pages 41-61, 2018.

[2] Éric Angel, Kim Thang Nguyen and Damien Regnault. Improved local search for universal facility location. *Journal of combinatorial Optimization*, volume 29: pages 237-246, 2015.

[3] Mathilde Noual, Damien Regnault, and Sylvain Sené. About non-monotony in Boolean automata networks. *Theoretical Computer Science (TCS)*, volume 504: pages 12-25, 2013.

[4] Damien Regnault, Jean-Baptiste Rouquier, and Éric Thierry. Stochastic minority on graphs. In *Theoretical Computer Science (TCS)*, special issue on computation theory of cellular automata and discrete dynamical systems, volume 412, issue 30: pages 3947-3963, 2011.

[5] Damien Regnault, Nicolas Schabanel, and Éric Thierry. On the analysis of "simple" 2D cellular automata. In *Discrete Mathematics and Theoretical Computer Science (DMTCS)*, special issue, volume 12, number 2, online publication, 2010.

[6] Damien Regnault, Nicolas Schabanel, and Éric Thierry. Progresses in the analysis of stochastic 2D cellular automata: A study of asynchronous 2D minority. *Theoretical Computer Science (TCS)*, volume 410(47-49): pages 4844–4855, 2009.

Articles published in national peer-reviewed journal.

[1] Tarek Melliti, Mathilde Noual, Damien Regnault, Adrien Richard, and Sylvain Sené. Cycles, double-cycles d'interactions et modes de mise à jour. Technique et Science Informatiques (TSI), volume 34(4), pages 401-430, 2015.

Articles published in international peer-reviewed conferences or workshops.

[1] Jacques Demongeot, Tarek Melliti, Mathilde Noual, Damien Regnault and Sylvain Sené. On Boolean Automata Isolated Cycles and Tangential Double-Cycles Dynamics. In *Automata and Complexity*, Springer, ECC 42, page 145–178, 2022.

[2] Éric Angel, Sébastien Morais and Damien Regnault. A Bi-Criteria FPTAS for Scheduling with Memory Constraint on Graphs with Bounded Tree-width. In *Euro-Par: Proceedings of the 28th International Conference on Parallel and Distributed Computing*, LNCS 13440, pages 136-151, 2022.

[3] Pierre-Étienne Meunier and Damien Regnault. Directed non-cooperative tile assembly is decidable. In *DNA: Proceedings of the 27th International Conference on DNA Computing and Molecular Programming*, LIPIcs 205, pages 6:1–6:21, 2021.

[4] Pierre-Étienne Meunier, Damien Regnault and Damien Woods. The program-size complexity of self-assembled paths. In *STOC: Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 727-737, 2020.

[5] Pierre-Étienne Meunier and Damien Regnault. Non-cooperatively Assembling Large Structures. In *DNA: Proceedings of the 25th International Conference on DNA Computing and Molecular Programming*, LNCS 11648, pages 120-139, 2019.

[6] Tarek Melliti, Damien Regnault, Adrien Richard and Sylvain Sené. Asynchronous Simulation of Boolean Networks by Monotone Boolean Networks. In *ACRI: Proceedings of the 12th International Conference on Cellular Automata for Research and Industry*, LNCS 9863, page 182-191, 2016.

[7] Éric Angel, Cédric Chevalier, Franck Ledoux, Sébastien Morais and Damien Regnault. FPT Approximation Algorithm for Scheduling with Memory Constraints. In *Euro-Par: Proceedings of the 22nd International Conference on Parallel and Distributed Computing*, LNCS 9833, pages 196-208, 2016.

[8] Damien Regnault and Éric Rémila. Lost in Self-Stabilization. In *MFCS: Proceedings of the 40th International Symposium on Mathematical Foundations of Computer Science*, LNCS 9234, page 432-443, 2015.

[9] Tarek Melliti, Mathilde Noual, Damien Regnault, Sylvain Sené and Jérémy Sobieraj. Asynchronous Dynamics of Boolean Automata Double-Cycles. In *UCNC: Proceedings of the 14th International Conference on Unconventional Computing and Natural Computing*, LNCS 9252, page 250-262, 2015.

[10] Tarek Melliti, Damien Regnault, Adrien Richard and Sylvain Sené. On the convergence of boolean automata networks without negative cycles. In *AUTOMATA: Proceedings of the 19th International Workshop on Cellular Automata and Discrete Complex Systems*, LNCS 8155, page 124-138, 2013.

[11] Éric Angel, Kim Thang Nguyen and Damien Regnault. Improved local search for universal facility location. In *COCOON: Proceedings of the 19th International Conference on Computing and Combinatorics*, LNCS 7936, page 316-324, 2013.

[12] Damien Regnault. Proving a phase transition in cellular automata under asynchronous dynamics. In *DLT: Proceedings of 17th International Conference on Developments in Language Theory*, LNCS 7907, page 433-444, 2013.

[13] Mathilde Noual, Damien Regnault and Sylvain Sené. Boolean networks synchronism sensitivity and XOR circulant networks convergence time.

In *AUTOMATA & JAC: Proceedings of 18th international workshop on Cellular Automata and Discrete Complex Systems and 3rd international symposium Journées Automates Cellulaires*, EPTCS 90, page 37-52, Open Publishing Association, 2012.

[14] Thomas Fernique and Damien Regnault. Stochastic flips on dimer Tilings. In *AofA: Proceedings of the 21st International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms*, DMTCS proceedings, volume (AM), page 207-220, online publication, 2010.

[15] Olivier Bodini, Thomas Fernique and Damien Regnault. Stochastic flips on two-letter words. In *ANALCO: Proceedings of the 7th Workshop on Analytic Algorithmics and Combinatorics*, page 48-55, SIAM, 2010.

[16] Olivier Bodini, Thomas Fernique and Damien Regnault. Cristallization by stochastic flips. In Proceedings of Aperiodics 2009, *Journal of Physics: Conference Series*, volume 226(012022):131-136, 2010.

[17] Damien Regnault. Quick energy drop in stochastic 2D Minority. In *ACRI: Proceedings of the 8th International Conference on Cellular Automata for Reseach and Industry*, volume 5191 of *LNCS*, pages 307–314. Springer, 2008.

[18] Damien Regnault. Directed percolation arising in stochastic cellular automata. In *MFCS: Proceedings of the 33rd International Symposium on Mathematical Foundations of Computer Science*, volume 5162 of *LNCS*, pages 563–574. Springer, 2008.

[19] Damien Regnault, Nicolas Schabanel, and Éric Thierry. On the analysis of "simple" 2D cellular automata. In *LATA: Proceedings of the 2nd International Conference on Language and Automata Theory and Applications*, volume 5196 of *LNCS*, pages 463–474, Springer 2008.

[20] Damien Regnault, Nicolas Schabanel, and Éric Thierry. Progresses in the analysis of stochastic 2D cellular automata: a study of asynchronous 2D Minority. In *MFCS: Proceedings of the 32rd International Symposium on Mathematical Foundations of Computer Science*, volume 4708 of *LNCS*, pages 320–332. Springer, 2007.

[21] Damien Regnault. Abrupt behaviour changes in cellular automata under asynchronous dynamics. In *ECCS: Proceedings of European Conference on Complex Systems, 6 pages*, 2006.

[22] Nazim Fatès, Damien Regnault, Nicolas Schabanel, and Éric Thierry. Asynchronous behaviour of double-quiescent elementary cellular automata. In *LATIN: Proceedings of the 7th Latin American Symposium*, volume 3887 of *LNCS*, pages 455–466. Springer, 2006.

Articles published in national peer-reviewed conferences or workshops.

[1] Éric Angel, Cédric Chevalier, Franck Ledoux, Sébastien Morais, and Damien Regnault. Partitionnement de maillages sous contrainte mémoire à l'aide de la programmation linéaire en nombres entiers. In *Conférence d'informatique en Parallélisme, Architecture et Système (COMPAS) 2016*.

[2] Éric Angel, Cédric Chevalier, Franck Ledoux, Sébastien Morais, Kim Thang Nguyen, and Damien Regnault. Algorithme approché pour un problème de partionnement de maillage sous contrainte mémoire. In *Conférence d'informatique en Recherche Opérationnelle et Aide à Décision En France (ROADEF) 2014*.

Résumé: Cette *habilitation à diriger des recherches* est consacrée aux pavages, un modèle de calcul utilisant des tuiles pour former des assemblages capables de faire des calculs d'une complexité équivalente à ceux d'une machine de Turing. En tant que modèle de calcul, les pavages présentent néanmoins un défaut majeur: les tuiles doivent être placées minutieusement pour obtenir l'assemblage souhaité. Les pavages étant utilisés pour étudier des structures nanoscopiques en cristallographie ou pour modéliser l'interaction de brins d'ADN, il est nécessaire d'envisager que l'assemblage du pavage ne se déroule pas parfaitement. Nous nous intéressons ici à deux approches pour gérer ce problème: l'auto-assemblage où les différentes possibilités de placement des tuiles lors de l'assemblage sont prises en compte et l'auto-stabilisation où les tuiles peuvent se réorganiser localement pour corriger d'éventuelles erreurs survenues lors de leur placement.

Mots-clés: modèles de calcul, pavages, auto-assemblage, auto-stabilisation.

Abstract: This *habilitation à diriger des recherches* deals with tilings, a computational model where tiles are bound together in order to do computation as complex as the ones done by Turing Machines. Nevertheless, tilings have a major flaw as a computational model: the tiles should be carefully assembled in order to produce the desired result. Since tilings are used for studying nanoscopic structures such as crystals or for modeling interactions of DNA strands, errors appearing during the assembly should be taken in consideration. We are focusing here on two ways to deal with this problem: self-assembly where all possibilities are taken into account during the assembly and self-stabilization where tiles can locally move in order to correct faults done during the assembly.

Keywords: Computational model, tilings, self-assembly, self-stabilization.