[TODO: External cover]

Colophon This thesis was composed with LuaLaTeX (LuaHBTeX, Version 1.22.0 – TeX Live 2025) using a custom class `tlbook` developped for the occasion (credit is due to the `kabook` class and Ken Arroyo Ohori's PhD thesis for the inspiration, themselves based on Edward Tufte's *Beautiful Evidence*). Figures were created with PGF/TikZ, and the bibliography with the multiscript branch of BibLaTeX/Biber.

The main typeface is David Jonathan Ross's Fern; `JetBrains Mono` and NewComputerModern are respectively used for monospaced text and mathematics; other scripts use Gentium Plus (Cyrillic and Greek) and LXGW WenKai/霞鹜文楷 (Chinese characters).

Draft: *June 5, 2025 at 14:45.*

PhD Thesis

Soficity of multidimensional subshifts

Antonin Callard

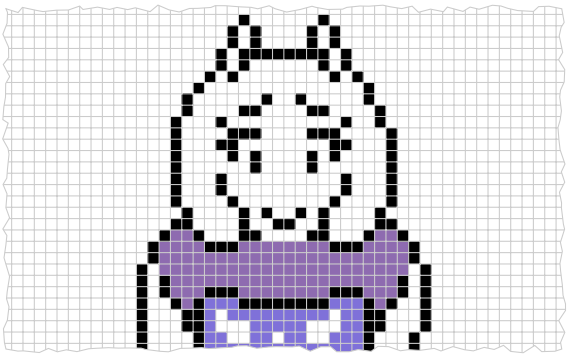
Acknowledgements

Table of contents

1 Introduction

Let us imagine for a second that a professional tiler has infinitely many copies from a finite collection of square tiles at their disposition; along with an infinite amount of time to devote to their tiling profession, which they actually are in dire need of as they specialize in decorating infinite bathroom walls. To appeal to different artistic sensibilities, our tiler allows their clients – who, for some reason, all happen to be mathematicians[1] – to provide them with sets of constraints on the patterns that can appear on their walls.

[1] Who else than an mathematician would delight in having an infinite decorated bathroom wall or a "cut and project" tiled wooden floor?



Some immediate questions arise from this fairly common renovation work situation. For example, following the set of constraints provided by a given client, will the tiler succeed in decorating the infinite wall (or has the mathematician client been too greedy in their esthetic requirements by providing contradictory constraints that prevents an infinite tiling from being valid?). This problem, officially known as the *Domino problem*, surprisingly turned out to be impossible to solve [Ber64]: more precisely, there exists no recipe for our tiler to follow that answers the question correctly for all sets of possible constraints.

[Ber64] Berger, "The undecidability of the Domino problem".

Since tiling infinite walls while following infinitely many constraints involves a considerable amount of effort, our tiler began a few eons ago to look for building tricks that would make their task easier. For example, they tried to convince their clients to only provide "local" constraints: in order to place a new tile on the wall, our tiler would no longer have to check infinitely many constraints around this position, but only look at the immediately adjacent tiles that have already been placed; unfortunately, they never found the clientele for such restrictive tilings.

Undeterred, our tiler may have recently found a potential workaround by studying the ancient scriptures of their craft [Wei73]: from now on, they will only accept the so-called "sofic" aesthetic constraints. Indeed, such constraints would allow the tiler to draw some construction lines on their tiles and build the wall locally – and make their task ever so easier – and then erase the construction lines as if they never were. This, for example, would satisfy their cook client who wanted a single "egg yellow" tile on a white background; or their number theoretist client, who wanted a spiral of squares who side lengths follow, in order, the sequence of prime numbers[2].

[Wei73] Weiss, "Subshifts of finite type and sofic systems".

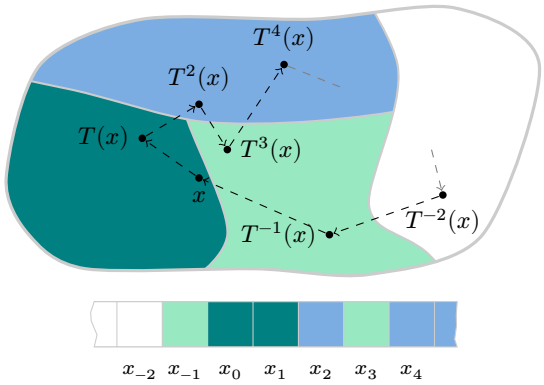
In this thesis, we are interested in answering the main question that will, likely, occupy both our professional tiler and their mathematician clients in the near future: when can a client's artistic choices be defined by sofic constraints? In other words, when listing the constraints given by a client, does there exists an equivalent *sofic* set of constraints that would define the same tilings?

[2] Even though the construction lines for this one are a bit tricky [Wes17].

[Wes17] Westrick, "Seas of squares with sizes from a Π^0_1 set".

(Multidimensional) symbolic dynamics

Unbeknownst to our tiler's knowledge, these questions can be formalized in the field of *symbolic dynamics*. Symbolic dynamics was originally introduced as a way to discretize continuous dynamical systems by cutting the phase space into finitely many subsets, assigning a symbol to each, and coding the trajectories of points in the original space into bi-infinite sequences of these symbols [Mor21; MH38].



While the field retains its dynamical origin (study of orbits, conjugacy invariants...), it was independently studied in a multidimensional setting under the formalism of Wang tiles [Wan61]: this introduced information-theoretic tools like combinatorics and computability in symbolic dynamics, with a famous milestone in the undecidability proof of the Domino problem [Ber64], to complete its traditional measure-theoretic and dynamical ones.

Nowadays, symbolic dynamics is the study of *subshifts*: given a finite alphabet of symbols $\mathcal{A}$, a *subshift* is a closed and shift-invariant subset of the Cantor space $\mathcal{A}^{\mathbb{Z}^d}$. It turns out that subshifts can also be defined combinatorially in terms of forbidden finite patterns, which allows to consider several complexity classes of subshifts: local subshifts, defined by adjacency constraints; subshifts of finite type (or SFTs), defined by finitely many forbidden patterns; effective subshifts, defined by computably enumerable families of forbidden patterns; and sofic subshifts which – as informally defined by our hypothetical infinite-time bathroom tiler – are projections of local subshifts and subshifts of finite type.

In this thesis, we are particularly interested in *multidimensional subshifts*, *i.e.* subshifts whose configurations color the discrete space $\mathbb{Z}^d$ for arbitrary dimensions $d \in \mathbb{N}$. If the first undecidability results on multidimensional subshifts from [Ber64; Rob71, ...], which have since then extended to a large variety of other decision problems [*e.g.* Har86], were largely considered as an *obstacle* to the study of multidimensional subshifts[3], many recent findings strongly suggest a revision of this rather pessimistic statement.

Indeed, computability theory has provided a rich forest of theorems, lemmas, ideas and constructions about the computational expressiveness of multidimensional symbolic dynamics: numerous dynamical invariants and properties of subshifts have actually been characterized computationally – such as entropies and entropy-like information-measuring quantities [HM10; Mey11, ...], sets of periods [JV15], directions of non-expansivity [Zin15], algebraic invariants [PV23]... –, while [Hoc09; AS13; DRS12; Wes17; Des21] embed arbitrary Turing machine computations into subshifts in order to directly control the geometrical structure of their configurations.

Thus, the convergence of computability theory and multidimensional symbolic dynamics has brought forward many fruitful interactions, and this thesis follows directly in these steps as it looks at the *soficity* of multidimensional subshifts.

[Mor21] Morse, "Recurrent geodesics on a surface of negative curvature".

[MH38] Morse and Hedlund, "Symbolic dynamics".

[Wan61] Wang, "Proving theorems by pattern recognition – II".

[Ber64] Berger, "The undecidability of the Domino problem".

[Rob71] Robinson, "Undecidability and nonperiodicity for tilings of the plane".

[Har86] Harel, "Effective transformations on infinite trees, with applications to high undecidability, dominoes, and fairness".

[3] This is the so-called "swamp of undecidability": if everything is undecidable, and that undecidability threatens to bog the carefree and careless mathematician down, what is there left to do?

[HM10] Hochman and Meyerovitch, "A characterization of the entropies of multidimensional shifts of finite type".

[Mey11] Meyerovitch, "Growth-type invariants for $\mathbb{Z}^d$ subshifts of finite type and arithmetical classes of real numbers".

[JV15] Jeandel and Vanier, "Characterizations of periods of multi-dimensional shifts".

[Zin15] Zinoviadis, "Hierarchy and expansiveness in 2D subshifts of finite type".

[PV23] Paviet Salomon and Vanier, "Realizing finitely presented groups as projective fundamental groups of SFTs".

[Hoc09] Hochman, "On the dynamics and recursive properties of multidimensional symbolic systems".

[AS13] Aubrun and Sablik, "Simulation of effective subshifts by two-dimensional subshifts of finite type".

[DRS12] Durand, Romashchenko, and Shen, "Fixed-point tile sets and their applications".

[Wes17] Westrick, "Seas of squares with sizes from a Π^0_1 set".

[Des21] Destombes, "Algorithmic complexity and soficness of shifts in dimension two".

Multidimensional soficity and subshifts

In this thesis, we present our work on the separation between the aforementioned classes of complexity on subshifts, in particular relating to the class of *sofic subshifts*. Since sofic subshifts are actually effective, our main question could be expressed as follows: **when does an effective subshift turn out to be sofic?**

As is often the case with subshifts, very different pictures emerge between the one-dimensional and the multidimensional cases. Since *sofic* subshifts are symbol-by-symbol projections of local subshifts, they can be considered as an infinite and multidimensional variation of the classical "regular" languages of finite words[4]:

- On $\mathbb{Z}$, the separation between sofic and effective subshifts is entirely solved: indeed, sofic subshifts being analogous to regular languages implies that they are easily characterized by their "extender sets" (see Chapter 6);
- However, the expressive power of sofic subshifts on $\mathbb{Z}^d$ for $d \geq 2$ is harder to grasp: for example, many "complex" multidimensional effective subshifts have surprisingly turned out to be sofic, including [Moz89; Cas10; Wes17, ...].

As a consequence, the exact separation between the classes of sofic and effective multidimensional subshifts has never been characterized, though many proofs of soficity and many sufficient conditions for non-soficity have been described over the years (see Chapter 7).

While answering this problem will probably require many more efforts, this thesis contains our progress on this separation question. In particular, we consider the expressive power of sofic multidimensional subshifts in two distinct contexts:

- Many proofs of non-soficity rely on quantifying how many patterns of the same size can be freely exchanged within a subshift. Using the formalization of "extender sets" (a point of view that, in fact, characterizes soficity on $\mathbb{Z}$), we look at the (asymptotic growth of) the number of extender sets of sofic and effective subshifts from a computational point of view, and aim at comparing the expressive power of sofic and effective multidimensional subshifts *via* their extender sets.
- While various conditions of non-soficity appear within the litterature, proving the soficity of a multidimensional subshift is often done on a case-by-case basis. Looking for more generic tools, we prove a sufficient condition for soficity in the multidimensional setting on the amount of "useful information" contained in the patterns. In addition to being a rather general condition of soficity, this condition enriches the known expressive power of multidimensional sofic subshifts by proving the newfound soficity of some effective sofic subshifts.

[4] Among the *numerous* equivalent definitions of regular languages, they are the letter-by-letter projections of local languages of finite words...

[Moz89] Mozes, "Tilings, substitution systems and dynamical systems generated by them".

[Cas10] Cassaigne, *Odd shift*.

Structure of the thesis and contributions

The first part of this thesis, "Gentle definitions", serves as a presentation of the many concepts that we use throughout this thesis, from symbolic dynamics (subshifts and classes of subshifts) and computability (computable functions and the RAM computational model). Since most results from these sections are folklore, the advanced reader can probably skip them safely.

The second part, "Context: soficity of subshifts", contains the real introduction of this document. Once proper definitions have been established, we establish this thesis' goal to study the soficity of multidimensional subshifts and present an overview of what is already known about it.

This thesis' contributions appear in the last two parts of this document, respectively titled "Multidimensional extender sets" and "Soficity and small representations". More precise motivations and a list of contributions are available at the beginning of each part, which are briefly summarized below.

▶ Part: "Multidimensional extender sets"

This part studies the generalization of the classical notion of *extender sets* from formal language theory [RS59] to the multidimensional setting $\mathbb{Z}^d$ [KM13; OP16]. Intuitively, extender sets define an equivalence between two patterns if any occurence of one can be replaced by the other in all the configurations of a given subshift.

[RS59] Rabin and Scott, "Finite automata and their decision problems".

[KM13] Kass and Madden, "A sufficient condition for non-soficness of higher-dimensional subshifts".

[OP16] Ormes and Pavlov, "Extender sets and multidimensional subshifts".

[FP19] French and Pavlov, "Follower, predecessor, and extender entropies".

In particular, we focus on a conjugacy invariant called *extender entropy*, which measures the exponential growth rate of the number of extender sets, and was originally introduced in [FP19] on $\mathbb{Z}$ subshifts. Instead of quantifying the direct number of patterns as with topological entropies, extender entropies quantify the number of equivalence classes of patterns for the exchange relation of the previous paragraph.

Inspired by other studies of entropy-like measures on subshifts [HM10; Mey11; CV21; GS23, ...], we characterize the whole set of possible extender entropies achieved by well-known classes of subshifts, especially effective and sofic subshifts, using the arithmetical hierarchy of real numbers. Our main contributions include:

[HM10] Hochman and Meyerovitch, "A characterization of the entropies of multidimensional shifts of finite type".

[Mey11] Meyerovitch, "Growth-type invariants for $\mathbb{Z}^d$ subshifts of finite type and arithmetical classes of real numbers".

[CV21] Callard and Vanier, "Computational characterization of surface entropies for $\mathbb{Z}^2$ subshifts of finite type".

[GS23] Gayral and Sablik, "Arithmetical hierarchy of the Besicovitch-stability of noisy tilings".

Theorem 9.2. *For $d \geq 1$, the set of extender entropies of $\mathbb{Z}^d$ effective subshifts is exactly $[0, +\infty) \cap \Pi_3$.*

Theorem 9.13. *For $d \geq 2$, the set of extender entropies of $\mathbb{Z}^d$ sofic subshifts is exactly $[0, +\infty) \cap \Pi_3$.*

In particular, these results show that sofic subshifts are as expressive as their effective counterparts when it comes down to the sets of possible extender entropies they can achieve, even under mixing dynamical restrictions; and that counting the number of extender sets cannot separate sofic from effective subshifts. These results were published in [CPV25].

[CPV25] Callard, Paviet Salomon, and Vanier, "Computability of extender sets in multidimensional subshifts".

▶ Part: "Soficity and small representations"

This part continues work initiated in [Des21] by considering sufficient conditions of multidimensional soficity based on the so-called "fixpoint construction" of tilings [DRS12], which has already been used to prove the soficity of some computationally complex subshifts [DRS10; Wes17; Des21].

[Des21] Destombes, "Algorithmic complexity and soficness of shifts in dimension two".

[DRS12] Durand, Romashchenko, and Shen, "Fixed-point tile sets and their applications".

[DRS10] Durand, Romashchenko, and Shen, "Effective closed subshifts in 1D can be implemented in 2D".

[Wes17] Westrick, "Seas of squares with sizes from a Π^0_1 set".

Our main contribution is a set of abstract sufficient conditions for multidimensional soficity based on a quantification of the "useful information" contained in the patterns of a given subshift. More precisely, we introduce

the notion of *inductive representations*, which provide a succint way to encode said "useful information" from the patterns of a given subshift. Intuitively, these representations should be understood as a way to summarize the information contained in square patterns, computed inductively from their pixels to the full domain, in order to decide whether the concatenation of 2×2 patterns (on $\mathbb{Z}^2$, and 2^d patterns on $\mathbb{Z}^d$) forms a locally valid pattern.

It is often intuited that, in a sofic subshift, communications between d-dimensional patterns of domain $[\![n]\!]^d$ and their configurations must go through the border $\partial([\![n]\!]^d)$, and are thus limited to $O(n^{d-1})$ bits of information. Our main contribution can be understood as a partial converse statement:

Theorem 10.11. *Fix $d \in \mathbb{N}$ and a finite alphabet $\mathcal{A}$. Let $\mathcal{R} : \mathcal{A}^{*d} \rightrightarrows \{0,1\}^*$ be a representation function and $\mathcal{I} \subseteq (\{0,1\}^*)^{2^d} \times \{0,1\}^*$ be an induction for $\mathcal{R}$ such that:*

(i) There exists $\alpha \in \mathbb{R}_+$ with $\alpha < d - 1$ such that, for any $w \in \mathcal{A}^{[\![n]\!]^d}$ and every representation $r \in \mathcal{R}(w)$, the size of r verifies $|r| = O(n^\alpha)$;

(ii) There exists $\beta \in \mathbb{R}_+$ with $\alpha \cdot \beta < d - 1$ such that $\mathcal{I}$ is computable in time $t(s) \leq O(s^\beta)$ in the log-RAM *model;*

Then

$$X_{\mathcal{R},\mathcal{I}} = \Big\{ x \in \mathcal{A}^{\mathbb{Z}^d} : \forall w \sqsubseteq x, \exists n \in \mathbb{N}, \exists w' \in \mathcal{A}^{[\![2^n]\!]^d}, \\ w \sqsubseteq w' \sqsubseteq x \text{ and } w' \text{ is inductively valid} \Big\}$$

is a sofic subshift.

Intuitively, this theorem proves that subshifts in which the validity of patterns of domain $[\![n]\!]^d$ can be verified with only $O(n^\alpha)$ bits of information about the pattern (for $\alpha < d - 1$)[5] are actually sofic (assuming some time restrictions on the computations, namely that computations happen in time $t(s) = O(s^\beta)$ for β such that[6] $\alpha \cdot \beta < d - 1$).

The proof of Theorem 10.11 mostly consists in building upon the "fixpoint contruction" from [DRS12]. However, due to the very tight time and space constraints of our problems, the computations are implemented within macro-tiles as space-time diagrams of a highly parallel computational model: *mesh-connected multicomputers*.

[5] *i.e.* with a sublinear amount of information relatively to the size of the border.

[6] Intuitively, the computations should fit within the domain $[\![n]\!]^d$.

Gentle definitions

2 Notations and conventions

Before the thesis begins, we settle on a few conventions regarding traditional notations of mathematics (sets of integers, cardinality, vectors, finite words…) and prove the generalization of the well-known "subadditive lemma" to a multivariate context.

2.1 General mathematics

Integers As the author is a computer scientist, we start counters at 0 and denote by $\mathbb{N} = \{0, 1, \dots\}$ the set of natural integers. For any two integers $a, b \in \mathbb{N}$, we denote $[a \,..\, b] = \{a, a+1, \dots, b\} \subseteq \mathbb{N}$ the interval of integers ranging between a and b (bounds included).

Additionally, we denote $[\![n]\!] \subseteq \mathbb{N}$ the interval of integers $\{0, \dots, n-1\}$.[7] Extending this notation from intervals of $\mathbb{N}$ to hyperrectangles of $\mathbb{N}^d$, we denote $[\![n_1, \dots, n_d]\!] = \{(i_1, \dots, i_d) \in \mathbb{N}^d : 0 \leq i_j < n_j \text{ for every } j\}$.

[7] The upper bound is not included, so that $[\![n]\!]$ has cardinality n.

Orderings and terminology All notions related to orderings are assumed to be inclusive by default[8]. In other words, if $(O, \leq)$ is an ordered set, then $o_1 \in O$ is *smaller* (resp. *larger*) than $o_2 \in O$ if $o_1 \leq o_2$ (resp. $o_1 \geq o_2$). If the inequalities are strict, then o_1 will be *strictly* smaller (resp. larger) than o_2. These considerations also apply to positive and negative real numbers.

Similarly, if two sets O, O' are ordered and $f : O \to O'$ is a function, f is said to be *increasing* (resp. decreasing) if $o_1 \leq o_2$ implies that $f(o_1) \leq f(o_2)$ (resp. $f(o_1) \geq f(o_2)$); and *strictly* increasing (resp. decreasing) if the inequalities are strict.

[8] Which follows the French conventions. While the author is very intimidated by foreign readers whose "non-decreasing" functions are not the functions that do not decrease, he is considerably more afraid of his former teachers – who are far closer and far more terrifying.

Sets For a finite set S, we denote by $|S|$ the cardinal of S, *i.e.* its number of elements. Similarly, if $S \subseteq X$ is a subset, we denote by $S^c = X \setminus S$ its complement. The union of two sets S_1, S_2 is denoted $S_1 \cup S_2$; and $S_1 \sqcup S_2$ if this union is disjoint.

Multidimensional geometry Working with d-dimensional objects, we will often consider tuples $\mathbf{i} = (i_1, \dots, i_d) \in \mathbb{Z}^d$ as either vectors, positions… typeset in bold letters.

For $k \in \mathbb{N}$ and a set of positions $F \subseteq \mathbb{Z}^d$, we denote by $\partial_k(F)$ the *k-border* of F, *i.e.* the set of all elements of F at distance less than k from $\mathbb{Z}^d \setminus F$ (for the L_1 or "Manhattan" distance). Similarly, we denote $\mathcal{I}_k(F) = F \setminus \partial_k(F)$ the *k-interior* of F, *i.e.* the set of all elements of F at distance strictly more than k from $\mathbb{Z}^d \setminus F$.

Words and patterns For a finite set $\mathcal{A}$, called an *alphabet*, we denote by $\mathcal{A}^* = \bigcup_{n \in \mathbb{N}} \mathcal{A}^n$ the set of all finite words over the alphabet $\mathcal{A}$. For two words $u, v \in \mathcal{A}^*$, we denote by uv (or $u \cdot v$) the concatenation of u and v; $\bar{u}$ will denote the mirror of the word u.

In Definition 3.1, we denote patterns as colorings of arbitrary domains $D \subseteq \mathbb{Z}^d$ by symbols of the alphabet $\mathcal{A}$. A pattern is said to be finite when its

domain is finite. Generalizing the previous notation, we denote by $\mathcal{A}^{*d}$ the set of all d-dimensional finite patterns, and $\mathcal{A}^{\circledast d}$ the set of all (potentially infinite) d-dimensional patterns[9].

Given $w, w' \in \mathcal{A}^{\circledast d}$ two patterns and $F \subseteq \mathrm{dom}(w)$, we denote by $w \ltimes_F w'$ the patterns whose symbols are $w_{\mathbf{i}}$ for $\mathbf{i} \in F$, and $w'_{\mathbf{i}}$ otherwise.

[9] The two symbols are distinct but voluntarily close, since no confusion should be possible in a given context.

Asymptotics We use classical notations for asymptotic comparison of functions: given two functions $f, g \colon \mathbb{R}_+ \to \mathbb{R}_+^*$, we say that:

(i) $f(x) = o(g(x))$ if $\frac{f(x)}{g(x)} \xrightarrow[x\to+\infty]{} 0$;
(ii) $f(x) = O(g(x))$ if there exists $K \in \mathbb{R}_+$ such that $|f(x)| \leq K \cdot |g(x)|$;
(iii) $f(x) = \Theta(g(x))$ if $f(x) = O(g(x))$ and $g(x) = O(f(x))$;
(iv) $f(x) = \tilde{O}(g(x))$ if $f(x) = O(g(x) \cdot \log(g(x)))$.

2.2 Multivariate analysis

We take a quite detour through the world of multivariate analysis to settle on a definition for multivariate limits, which we will later use to define multidimensional entropies of subshifts.

Considering the usual ordering of $\mathbb{N}$, let us consider the (partial) product ordering $\leq$ on $\mathbb{N}^d$ defined as[10]: $(n_1, \dots, n_d) \leq (m_1, \dots, m_d)$ if $n_i \leq m_i$ for every $1 \leq i \leq d$. The classical notion of limit from real analysis can be generalized to a multivariate setting as follows:

[10] We denote by $\leq$ both the classical ordering on $\mathbb{N}$ and its extension to $\mathbb{N}^d$.

Definition 2.1. *A function $f \colon \mathbb{N}^d \to \mathbb{R}$ admits a limit $l \in \mathbb{R}$ if:*

$$\forall \varepsilon > 0,\ \exists \mathbf{N} \in \mathbb{N}^d,\ \forall \mathbf{n} \in \mathbb{N}^d,\ \mathbf{n} \geq \mathbf{N} \implies |f(\mathbf{n}) - l| \leq \varepsilon;$$

in other words, if l is the limit of $f(n_1, \dots, n_d)$ when the variables $n_1, \dots, n_d$ all grow to $+\infty$. We denote this by $\lim_{n_1,\dots,n_d \in \mathbb{N}^d} f(n_1, \dots, n_d) = l$.

Other classical notions from real analysis (tending to infinity, limit inferior and superior...) generalize similarly with the ordering $\leq$ on $\mathbb{N}^d$.

In this manuscript, we will in particular need a multivariate version of the subadditive lemma[11]. On multivariate functions:

[11] The subadditive lemma states that for every subadditive function $f : \mathbb{N} \to \mathbb{R}$ such that $f(m+n) \leq f(m) + f(n)$, the limit $\lim_{n\to+\infty} \frac{f(n)}{n}$ exists and equals $\inf_{n\in\mathbb{N}} \frac{f(n)}{n}$.

Definition 2.2. *A multivariate function $f : \mathbb{N}^d \to \mathbb{R}$ is* subadditive *if it is subadditive in every variable. In other words, if for every $n_1, \dots, n_d \in \mathbb{N}$, every $1 \leq i \leq d$ and every $m_i \in \mathbb{N}$, we have:*

$$f(n_1, \dots, n_i + m_i, \dots, n_d) \leq f(n_1, \dots, n_i, \dots, n_d) + f(n_1, \dots, m_i, \dots, n_d).$$

A multivariate version of the subadditive lemma was proved in [Cap08][12]:

[Cap08] Capobianco, "Multidimensional cellular automata and generalization of Fekete's lemma".

[12] I include the statement and its proof here for the sake of exhaustiveness, and because I had somehow managed to never hear about them.

Lemma 2.3 ([Cap08, Theorem 1]). *Let $f : \mathbb{N}^d \to \mathbb{R}$ be a subadditive multivariate function. Then the limit of $\frac{f(n_1,\dots,n_d)}{n_1 \cdots n_d}$ exists and:*

$$\lim_{n_1,\dots,n_d \in \mathbb{N}^d} \frac{f(n_1, \dots, n_d)}{n_1 \cdots n_d} = \inf_{n_1,\dots,n_d \in \mathbb{N}^d} \frac{f(n_1, \dots, n_d)}{n_1 \cdots n_d}.$$

The proof is very similar to the one-dimensional case:

Proof. We follow the proof of [Cap08] and prove that:

$$\limsup_{n_1,\dots,n_d \in \mathbb{N}^d} \frac{f(n_1, \dots, n_d)}{n_1 \cdots n_d} \leq \inf_{n_1,\dots,n_d \in \mathbb{N}^d} \frac{f(n_1, \dots, n_d)}{n_1 \cdots n_d},$$

where $\limsup_{\mathbf{n}\in\mathbb{N}^d} g(\mathbf{n}) = \inf_{\mathbf{n}\in\mathbb{N}^d} \sup_{\mathbf{m}\geq\mathbf{n}} g(\mathbf{m})$.

Fix $k_1, \dots, k_d \in \mathbb{N}$. For $n_1, \dots, n_d \in \mathbb{N}^d$, by the division theorem there exists $q_1, \dots, q_d \in \mathbb{N}^d$ and $r_1, \dots, r_d \in \mathbb{N}^d$ (with $r_i < k_i$) such that $n_i = k_i \cdot q_i + r_i$. Using the subadditivity of f on n_1, we obtain:

$$\begin{aligned}
\frac{f(n_1, \dots, n_d)}{n_1 \cdots n_d} &= \frac{f(k_1 \cdot q_1 + r_1, n_2, \dots, n_d)}{n_1 \cdots n_d} \\
&\leq \frac{q_1 \cdot f(k_1, n_2, \dots, n_d)}{n_1 \cdots n_d} + \frac{f(r_1, n_2, \dots, n_d)}{n_1 \cdots n_d} && \text{(By subadditivity on } n_1\text{)} \\
&\leq \frac{q_1 \cdot f(k_1, n_2, \dots, n_d)}{n_1 \cdots n_d} + \frac{n_2 \cdots n_d \cdot f(r_1, 1, \dots, 1)}{n_1 \cdots n_d} && \text{(By subadditivity on } n_2, \dots, n_d\text{)}
\end{aligned}$$

Thus, by iterating subadditivity on each variable of f, we obtain:

$$\begin{aligned}
\frac{f(n_1, \dots, n_d)}{n_1 \cdots n_d} &\leq \frac{q_1 \cdots q_d \cdot f(k_1, \dots, k_d)}{n_1 \cdots n_d} \\
&\quad + \sum_{i=1}^{d} \frac{q_1 \cdots q_{i-1} \cdot n_{i+1} \cdots n_d \cdot f(r_1, \dots, r_i, 1, \dots, 1)}{n_1 \cdots n_d} \\
&\leq \frac{q_1 \cdots q_d \cdot f(k_1, \dots, k_d)}{n_1 \cdots n_d} + \sum_{i=1}^{d} \frac{f(r_1, \dots, r_i, 1, \dots, 1)}{n_i}
\end{aligned}$$

Since $n_i = k_i \cdot q_i + r_i$, we have $\frac{q_i}{n_i} \leq \frac{1}{k_i}$. Moreover, since all the terms $f(r_1, \dots, r_i, 1, \dots, 1)$ are bounded, we obtain that for any $\varepsilon > 0$, the previous inequalities become

$$\frac{f(n_1, \dots, n_d)}{n_1 \cdots n_d} \leq \frac{f(k_1, \dots, k_d)}{k_1 \cdots k_d} + \varepsilon.$$

when all the n_i are large enough. In particular, for any $k_1, \dots, k_d \in \mathbb{N}^d$, we have:

$$\limsup_{n_1, \dots, n_d \in \mathbb{N}^d} \frac{f(n_1, \dots, n_d)}{n_1 \cdots n_d} \leq \frac{f(k_1, \dots, k_d)}{k_1 \cdots k_d}.$$

By taking the infimum over $k_1, \dots, k_d \in \mathbb{N}^d$, we conclude that:

$$\limsup_{n_1, \dots, n_d \in \mathbb{N}^d} \frac{f(n_1, \dots, n_d)}{n_1 \cdots n_d} \leq \inf_{n_1, \dots, n_d \in \mathbb{N}^d} \frac{f(n_1, \dots, n_d)}{n_1 \cdots n_d}.$$

On the other hand, we have:

$$\inf_{n_1, \dots, n_d \in \mathbb{N}^d} \frac{f(n_1, \dots, n_d)}{n_1 \cdots n_d} \leq \liminf_{n_1, \dots, n_d \in \mathbb{N}^d} \frac{f(n_1, \dots, n_d)}{n_1 \cdots n_d}.$$

We conclude by equality of limit inferior and limit superior. □

3 Symbolic dynamics

This chapter contains a brief glance into the wonders of multidimensional symbolic dynamics. It introduces subshifts as combinatorial objects, but also as topological dynamical systems. Morphisms/factor maps (*i.e* cellular automata) define factors and conjugacy between subshifts. Classical dynamical properties of subshifts are introduced (minimality, mixingness...). Finally, we consider the usual computational classes of subshifts: local subshifts and subshifts of finite type, sofic subshifts and effective subshifts...

3.1 Subshifts

3.1.1 Patterns and configurations

An *alphabet* is a finite set of *symbols* $\mathcal{A}$, whose elements are often called *symbols* or *colors*. For $d \in \mathbb{N}$ a *dimension*, we define d-dimensional patterns as:

Definition 3.1 (Pattern). A (*d-dimensional*) pattern *is a mapping* $w \colon D \to \mathcal{A}$, *i.e. a coloring of* D *with the symbols of* $\mathcal{A}$, *where* D *is a subset* $D \subseteq \mathbb{Z}^d$ *called the* domain *of* w, *denoted* $\mathrm{dom}(w)$. *Given* $\mathbf{i} \in \mathrm{dom}(w)$, *we denote by* $w_\mathbf{i} \in \mathcal{A}$ *the symbol taken by* w *at position* $\mathbf{i}$.

A *pattern* w *is* finite *if* $\mathrm{dom}(w)$ *is a finite subset of* $\mathbb{Z}^d$. *We denote by* $\mathcal{A}^{*d}$ *the set of* d*-dimensional finite patterns over the alphabet* $\mathcal{A}$, *and* $\mathcal{A}^{\circledast d}$ *the set of* d*-dimensional (possibly infinite) patterns of the alphabet* $\mathcal{A}$.

Figure 3.1: A 2-dimensional pattern on the alphabet $\mathcal{A} = \{\blacksquare, \square\}$.

As is usual in symbolic dynamics, we prefer the subscript notation $w_\mathbf{i}$ instead of the traditional functional notation $w(\mathbf{i})$ to denote the color of the pattern w at position $\mathbf{i}$.

It is often convenient to consider patterns "only up to translation". Indeed, if there exists two patterns $u, v \in \mathcal{A}^{\circledast d}$ and some translation $\mathbf{i} \in \mathbb{Z}^d \setminus \{0\}$ such that $u = v|_{\mathbf{i}+\mathrm{dom}(v)}$, then u and v are distinct patterns formally but are often considered to represent the same drawing. In this thesis, we will avoid working with equivalence classes of patterns explicitely, but will try to explicitely mention it when the distinction matters.

We say that a pattern u *appears in* a pattern v, denoted $u \sqsubseteq v$ (or, conversely, that v *contains* the pattern u), if there exists a position $\mathbf{i} \in \mathbb{Z}^d$ such that $\mathrm{dom}(u) \subseteq (\mathbf{i} + \mathrm{dom}(v))$ and:

$$\forall \mathbf{i} \in \mathrm{dom}(u),\ u_\mathbf{j} = v_{\mathbf{i}+\mathbf{i}}.$$ [13]

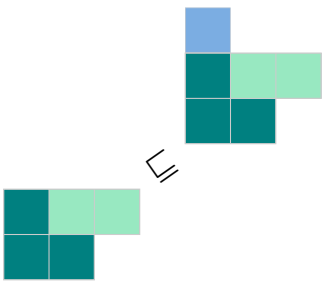


Figure 3.2: Pattern inclusion.

Conversely, a pattern u *avoids* a pattern v if, for every position $\mathbf{i} \in \mathbb{Z}^d$ such that $\mathrm{dom}(u) \subseteq (\mathbf{i} + \mathrm{dom}(v))$, there exists $\mathbf{j} \in \mathrm{dom}(u)$ such that $u_\mathbf{j} \neq v_{\mathbf{i}+\mathbf{j}}$.

Definition 3.2 (Configuration). A (*d-dimensional*) configuration *is a mapping* $x \colon \mathbb{Z}^d \to \mathcal{A}$, *i.e. a pattern whose domain is the entire space* $\mathrm{dom}(x) = \mathbb{Z}^d$. *Infinite patterns may sometimes be called* partial configuration. *The set of all* d*-dimensional configurations over* $\mathcal{A}$ *is denoted* $\mathcal{A}^{\mathbb{Z}^d}$.

[13] By definition, pattern inclusion does not require the exact inclusion of domains, *i.e.* is a notion of inclusion "up to translation".

3.1.2 Subshifts

Subshifts In this thesis, we are interested in *shift spaces*, or *subshifts*, which are sets of configurations defined in terms of forbidden patterns:

Definition 3.3 (Subshift). *In dimension $d \in \mathbb{N}$ and given an alphabet $\mathcal{A}$, a (possibly infinite) family of finite patterns defines the set of configurations that avoid the patterns of $\mathcal{F}$:*

$$X_{\mathcal{F}} = \left\{ x \in \mathcal{A}^{\mathbb{Z}^d} : \forall w \in \mathcal{F},\ w \not\sqsubset x \right\}.$$

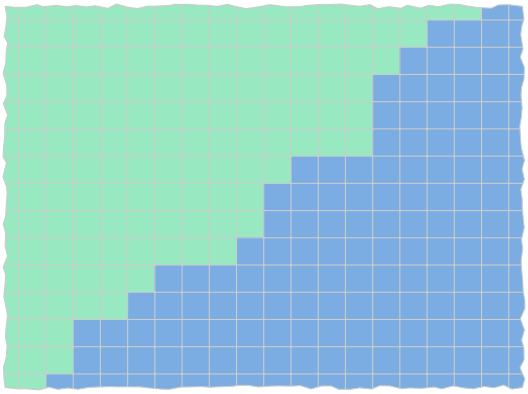
A subshift is a set of configurations $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ for which there exists a family of finite "forbidden" patterns $\mathcal{F}$ such that $X = X_{\mathcal{F}}$.

The reader should notice that a subshift can be defined by several (in fact, infinitely many) families of forbidden patterns.

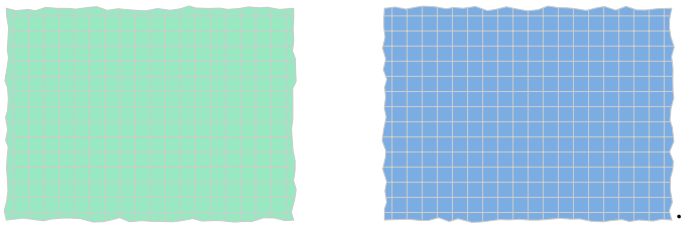
Example 3.4. *In dimension $d = 2$, consider the two-color alphabet $\mathcal{A} = \{$□, ■$\}$, and the family of forbidden patterns*

$$\mathcal{F} = \{$$  $$\}.$$

Then $X_{\mathcal{F}} \subseteq \mathcal{A}^{\mathbb{Z}^2}$ is the subshift whose configurations are composed of ascending ■-colored stairs over a □ background, such as:



along, of course, the full ■ and the full □ configurations:



Notice that translations act upon patterns $\mathcal{A}^{\circledast d}$ and configurations $\mathcal{A}^{\mathbb{Z}^d}$. More precisely, we define the *shift maps* as:

Definition 3.5 (Shift maps). *For a vector $\mathbf{t} \in \mathbb{Z}^d$, the $\mathbf{t}$-shift map is the translation function $\sigma^{\mathbf{t}} : \mathcal{A}^{\circledast d} \to \mathcal{A}^{\circledast d}$ defined as:*

$$\forall w \in \mathcal{A}^{\circledast d},\ \sigma^{\mathbf{t}}(w) = (\mathbf{i} \in (\mathrm{dom}(w) - \mathbf{t}) \mapsto w_{\mathbf{i}+\mathbf{t}}).$$

On $\mathbb{Z}$, we often denote the left-shift $\sigma = \sigma^1$.

The terminology *shift spaces/subshifts* comes from the invariance of these objects by translations/shift maps: indeed, a configuration $x \in \mathcal{A}^{\mathbb{Z}^d}$ avoids a pattern $w \in \mathcal{A}^{\star d}$ if and only if all its translations $\sigma^{\mathbf{t}}(x)$ for $\mathbf{t} \in \mathbb{Z}^d$ avoid w too.

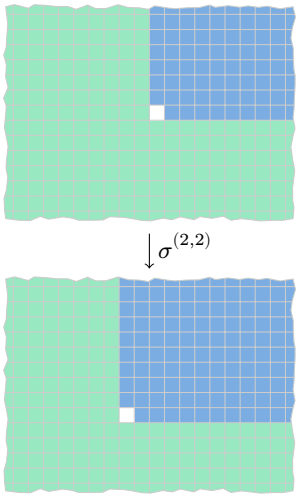


Figure 3.3: A configuration and its $(2,2)$-shift.

Languages The set of patterns that appear in a subshift, also called the set of *valid* patterns of a subshift, defines a *language*:

Definition 3.6 (Language). *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift. For a domain $D \subseteq \mathbb{Z}^d$, the* language *of all valid patterns of X over the domain D is:*

$$\mathcal{L}_D(X) = \{x|_D : x \in X\}.$$

The language *of X is the set of all its* valid *patterns, i.e.*

$$\mathcal{L}(X) = \bigcup_{D \subseteq \mathbb{Z}^d} \mathcal{L}_D(X).$$

It is sometimes useful to consider the language of *finite* valid patterns of X, which we will often denote $\mathcal{L}(X) \cap \mathcal{A}^{*d}$.

By counting the number of valid finite patterns appearing in a subshift, one defines the notion of *pattern complexity*. More formally:

Definition 3.7 (Pattern complexity). *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift. The* pattern complexity *of X is the function:*

$$N_X : n_1, \dots, n_d \in \mathbb{N}^d \mapsto |\mathcal{L}_{[\![n_1,\dots,n_d]\!]}(X)|.$$

Since the complexity function $N_X : \mathbb{N}^d \to \mathbb{N}$ is submultiplicative, it yields a well-defined limit called the *topological entropy* by the classical subadditive lemma[14]:

[14] See Lemma 2.3 for the generalization of the subadditive lemma to multivariate functions, and Definition 2.1 for the definition of multivariate limit used below.

Definition 3.8 (Topological entropy). *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift. The* (topological) entropy *of X, denoted $h(X)$, is the value:*

$$h(X) = \lim_{n_1,\dots,n_d \in \mathbb{N}^d} \frac{\log |\mathcal{L}_{[\![n_1,\dots,n_d]\!]}(X)|}{n_1 \cdots n_d}.$$

Following the information-theoretic point of view, we informally view the entropy of a subshift as a measure of the average amount of information that is learned about a whole configuration by only seeing one of its cells. Topological entropy is actually a classical notion in topological dynamics, and is a conjugacy invariant[15].

[15] See Definition 3.27 for more details.

3.1.3 Operations on subshifts

Cartesian products, layers and projections

Cartesian products The (Cartesian) product of two subshifts is defined as follows:

Definition 3.9 (Cartesian product). *Given two alphabets $\mathcal{A}$ and $\mathcal{B}$, and two d-dimensional subshifts $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ and $Y \subseteq \mathcal{B}^{\mathbb{Z}^d}$, the* Cartesian product $X \times Y$ *is the subshift:*

$$X \times Y = \{x \in (\mathcal{A} \times \mathcal{B})^{\mathbb{Z}^d} : \pi_{\mathcal{A}}(x) \in X \text{ and } \pi_{\mathcal{B}}(x) \in Y\},$$

where $\pi_{\mathcal{A}} : \mathcal{A} \times \mathcal{B} \to \mathcal{A}$ and $\pi_{\mathcal{B}} : \mathcal{A} \times \mathcal{B} \to \mathcal{A}$ denote the natural projections[16].

[16] Although these projections are defined on alphabets, we generalize to configurations (and sets of configurations) by denoting $\pi(x)$ the configuration formally defined as $\pi(x)_{\mathbf{i}} = \pi(x_{\mathbf{i}})$.

Remark. *When dealing with Cartesian products, we very often abuse notations and identify $(\mathcal{A} \times \mathcal{B})^{\mathbb{Z}^d}$ with $\mathcal{A}^{\mathbb{Z}^d} \times \mathcal{B}^{\mathbb{Z}^d}$. This should not be a source of confusion.*

Superimposition Very often in subshifts, especially for proofs that involve the construction of a subshift, we will consider the Cartesian product of several subshifts as a *superimposition* of layers. More precisely,

Definition 3.10 (Layers). *Given subshifts $L_1 \subseteq \mathcal{A}_1^{\mathbb{Z}^d}, \dots, L_k \subseteq \mathcal{A}_k^{\mathbb{Z}^d}$, a subshift X is a* superimposition *of $L_1, \dots, L_k$ if*

$$X \subseteq L_1 \times \cdots \times L_k$$

is a subshift of $L_1 \times \cdots \times L_k$ with (optionally) some additional forbidden patterns.

The subshifts $L_1, \dots, L_k$ are called the layers *of X.*

Given a subshift $X \subseteq L_1 \times \cdots \times L_k$ on k layers, and for any subset of these layers $I \subseteq [1 \,..\, k]$, we denote by $\pi_{\prod_{i\in I} L_i} : \prod_{i=1}^{k} L_i \to \prod_{i\in I} L_i$ the natural projection onto the layers of I.

Lifts

Given a d-dimensional subshift, we consider two constructions to obtain related d'-dimensional subshifts for $d' \geq d$: the periodic lift, and the free lift.

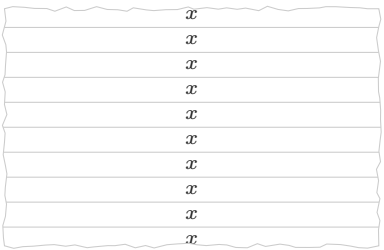


Figure 3.4: Periodic lift of a $\mathbb{Z}$ configuration to $\mathbb{Z}^2$.

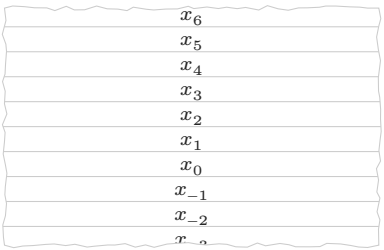


Figure 3.5: Free lift configuration from $\mathbb{Z}$ to $\mathbb{Z}^2$.

Definition 3.11 (Periodic lifts). *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift. For $d' \geq d$, the d'-*periodic lift *of X is the subshift $X^{\Uparrow d'}$ defined as:*

$$X^{\Uparrow d'} = \{x' \in \mathcal{A}^{\mathbb{Z}^{d'}} : \exists x \in X,\ \forall \mathbf{i} \in \mathbb{Z}^{d'-d},\ x'|_{\{\mathbf{i}\}\times\mathbb{Z}^d} = x\}.$$

For $d' = d+1$, we often denote $X^{\Uparrow d+1}$ as $X^{\Uparrow}$.

In other words, configurations of $X^{\Uparrow}$ are obtained by repeating configurations of X vertically. On the other hand, configurations in the free lift are obtained by stacking independent configurations of X:

Definition 3.12 (Free lifts). *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift. For $d' \geq d$, the d'-*free lift *of X is the subshift $X^{\rightleftharpoons d'}$ defined as:*

$$X^{\rightleftharpoons d'} = \{x \in \mathcal{A}^{\mathbb{Z}^{d'}} : \forall \mathbf{i} \in \mathbb{Z}^{d'-d},\ x|_{\{\mathbf{i}\}\times\mathbb{Z}^d} \in X\}.$$

For $d' = d+1$, we often denote $X^{\rightleftharpoons d+1}$ as $X^{\rightleftharpoons}$.

Periodic lifts gained popularity when initial work began on Proposition 3.44, while free lifts have mostly been considered in the context of sofic subshifts, especially relating to Question 15.10.

3.2 Topology

Subshifts can also be introduced as topological dynamical systems. Many properties of subshifts follow from the fact that they form compact spaces (see Corollary 3.19).

Definition 3.13 (Full shift). *In dimension $d \in \mathbb{N}$ and given an alphabet $\mathcal{A}$, the* full-shift *is the set of all possible configurations $\mathcal{A}^{\mathbb{Z}^d}$.*

The full-shift is a subshift, since it can be defined by the the empty family of forbidden patterns $\mathcal{F} = \emptyset$. More interestingly, it can be equipped with the *product topology* to define a topological space.

3.2.1 Product topology

Cylinders Cylinders define a basis of the product topology:

Definition 3.14 (Cylinder). *In dimension $d \in \mathbb{N}$ and given an alphabet $\mathcal{A}$, let $w \in \mathcal{A}^{*d}$ be a finite pattern. It defines a* cylinder *$[\![w]\!]$ as:*

$$[\![w]\!] = \{x \in \mathcal{A}^{\mathbb{Z}^d} : x|_{\mathrm{dom}(w)} = w\}.$$

In other words, the cylinder $[\![w]\!]$ denotes the set of all configurations that agree with w on $\mathrm{dom}(w)$.

Product topology

Proposition 3.15 (Product topology). *The topology $\mathcal{T}$ obtained by taking all cylinders $\{[\![w]\!] : w \in \mathcal{A}^{*d}\}$ as a basis of open sets is exactly the product of the discrete topology on each space $\mathcal{A}$.*

The topology $\mathcal{T}$ is thus called the *product topology*. By definition, a set $U \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is *open* in the product topology if it can be written as an arbitrary union of cylinders; it is closed if its complement U^c is open.

Claim 3.16. *Every cylinder $[\![w]\!]$ is a clopen*[17] *set in the product topology.*

[17] Open and closed set.

Proof. Let $D \subseteq_f \mathbb{Z}^d$ be a finite subset of $\mathbb{Z}^d$, and $w \in \mathcal{A}^D$ be a pattern of domain D. Then $[\![w]\!]$ is open as an element of the basis; and $[\![w]\!]^c$ is open as a union of open sets:

$$[\![w]\!]^c = \bigcup_{w' \in \mathcal{A}^D \setminus \{w\}} [\![w']\!]. \qquad \square$$

A complete presentation of all the classical definitions from general topology is outside the scope of this manuscrit. In fact, this thesis can be read with only intuitive notions of *limits* and *compactness*, considered here as "extraction arguments".

Compactness One of the most important aspect of this topology is that all full shifts are compact spaces, and thus extraction arguments can be applied to sequences of configurations:

Proposition 3.17. *For any dimension $d \in \mathbb{N}$ and alphabet $\mathcal{A}$, the full shift $\mathcal{A}^{\mathbb{Z}^d}$ is compact for the product topology.*

Proof. Since $\mathcal{A}$ is finite, thus compact for the discrete topology, by Tychonoff's theorem[18] we can conclude that any $\mathcal{A}^{\mathbb{Z}^d}$ is compact. However, we prefer a more combinatorial approach, based on a diagonal argument and the socket-drawer principle. Let $(x^{(n)})_{n\in\mathbb{N}}$ be a sequence of configurations in $\mathcal{A}^{\mathbb{Z}^d}$:

- Considering the position $0 \in \mathbb{Z}^d$ and using the socket-drawer principle, there exists $a \in \mathcal{A}$ and infinitely many configurations from $(x^{(n)})_{n\in\mathbb{N}}$ such that $x^{(n)}_0 = a$. We denote these configurations $(x^{0:(n)})_{n\in\mathbb{N}}$;
- More generaly, assume that there exists a subsequence $(x^{k:(n)})_{n\in\mathbb{N}}$ of configurations such that, for any $n, n' \in \mathbb{N}$, $x^{0:(n)}|_{[-k..k]^d} = x^{0:(n')}|_{[-k..k]^d}$. By the socket-drawer principleon these configurations, there exists a pattern $w \in \mathcal{A}^{[-k-1..k+1]^d}$ and infinitely many configurations from $(x^{k:(n)})_{n\in\mathbb{N}}$ such that $x^{k:(n)}|_{[-k-1..k+1]^d} = w$. We denote these configurations $(x^{k+1:(n)})_{n\in\mathbb{N}}$.

Then the sequence $(x^{n:(0)})_{n\in\mathbb{N}}$, which is by construction a subsequence of $(x^{(n)})_{n\in\mathbb{N}}$, is converging in the product topology towards the limit configuration $x \in \mathcal{A}^{\mathbb{Z}^d}$ defined by $x|_{[-n..n]^d} = x^{n:(0)}|_{[-n..n]^d}$. $\square$

[18] A product of compact spaces is compact.

3.2.2 Subshifts

In this thesis, we chose to introduce subshifts as combinatorial objects based on forbidden patterns. Since forbidding patterns define closed sets for the product topology[19], it should not come as a surprise that subshifts also enjoy a very topological definition:

[19] For a pattern w, $\{x : w \sqsubseteq x\}$ is open as an infinite union of cylinders.

Proposition 3.18. *In dimension $d \in \mathbb{N}$ and for an alphabet $\mathcal{A}$, the subshifts of $\mathcal{A}^{\mathbb{Z}^d}$ are exactly the closed and shift-invariant subsets of $\mathcal{A}^{\mathbb{Z}^d}$.*

Proof.

$\Longrightarrow$ Let $X_{\mathcal{F}} \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift for $\mathcal{F}$ a family of forbidden finite patterns. Then X is shift-invariant, since a configuration avoids a given pattern $w \in \mathcal{F}$ if and only if all its translations do. To prove that X is closed, notice that X^c is open as the following union of open sets:

$$\begin{aligned} X^c &= \{x \in \mathcal{A}^{\mathbb{Z}^d} : \exists w \in \mathcal{F},\, w \sqsubseteq x\} \\ &= \bigcup_{w \in \mathcal{F}} \bigcup_{t \in \mathbb{Z}^d} [\![\sigma^t(w)]\!]. \end{aligned}$$

$\Longleftarrow$ Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a closed and shift-invariant subset of $\mathcal{A}^{\mathbb{Z}^d}$. Denoting $\mathcal{F} = \bigcup_{n \in \mathbb{N}} \mathcal{A}^{[-n..n]^d} \setminus \mathcal{L}(X)$, we prove that $X = X_{\mathcal{F}}$.
If $x \in X$, then all patterns $w \sqsubseteq x$ and all their shifts appear in $\mathcal{L}(X)$, and consequently do not belong in $\mathcal{F}$: thus, $x \in X_{\mathcal{F}}$. Reciprocally, consider $x \in X_{\mathcal{F}}$: by definition of the language, for all $n \in \mathbb{N}$, we have $x|_{[-n..n]^d} \in \mathcal{L}_{[-n..n]^d}(X)$, so that there exists $x^{(n)} \in X$ such that $x^{(n)}|_{[-n..n]^d} = x|_{[-n..n]^d}$. Since $x^{(n)}$ converges towards x in $\mathcal{A}^{\mathbb{Z}^d}$, and that X is closed, we conclude that $x \in X$.

□

Corollary 3.19. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift. Then X is compact.*

Proof. $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is a closed subset of a compact space, so it is compact. □

Compactness is used in many proofs of this thesis, in the form of the following "limit argument": given a sequence of *locally admissible* patterns $(u_n)_{n \in \mathbb{N}}$ of domain $[-n \,..\, n]^d$ such that $u_n \sqsubseteq u_{n+1}$, then the limit configuration $x \in \mathcal{A}^{\mathbb{Z}^d}$ defined by $x|_{[-n..n]^d} = u_n$ is a valid configuration of the subshift.

Another consequence that we sometimes use is that it is possible to obtain a subshift by taking the topological closure of a set of shift-invariant configurations:

Claim 3.20. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a shift-invariant set of configuration. Then its topological closure $\bar{X}$ is a subshift $\bar{X} \subseteq \mathcal{A}^{\mathbb{Z}^d}$.*

3.3 Morphisms

3.3.1 Morphisms and factors

Since subshifts are topological objects, it makes sense to consider their *morphisms*, *i.e.* the transformations that preserve their structure.

Definition 3.21. *In dimension $d \in \mathbb{N}$ and given two subshifts $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ and $Y \subseteq \mathcal{B}^{\mathbb{Z}^d}$, a* morphism *between X and Y is a continuous function $f : X \to Y$ such that $f \circ \sigma^{\mathbf{t}} = \sigma^{\mathbf{t}} \circ f$ for every $\mathbf{t} \in \mathbb{Z}^d$.*

In the formula $f \circ \sigma^{\mathbf{t}} = \sigma^{\mathbf{t}} \circ f$, one should note that $\sigma^{\mathbf{t}}$ acts on two distinct spaces. On the left-hand side of the equality, $\sigma^{\mathbf{t}}$ acts on $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$; on the right-hand side, $\sigma^{\mathbf{t}}$ acts on $Y \subseteq \mathcal{B}^{\mathbb{Z}^d}$. In terms of vocabulary, a function $f : X \to Y$ that verifies the σ-commuting condition is said to be *σ-equivariant*.

As implied by the name, subshifts are preserved under morphisms: given a morphism $f : X \to \mathcal{B}^{\mathbb{Z}^d}$, the set $f(X) \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is actually a subshift: its shift-invariance comes from the shift-equivariance of f; and since $\mathcal{B}^{\mathbb{Z}^d}$ is Hausdorff[20], while $f(X)$ is compact as the image of a compact set by a continuous function, we deduce that $f(X)$ is closed.

[20] In fact, it is even metrizable.

Since the product topology is defined in terms of cylinders, the continuity of morphisms implies that the value of an image configuration can only depend on finitely many positions of the initial configuration. This provides another equivalent definition for morphisms, as the well-known cellular automata:

Proposition 3.22 (Curtis-Hedlund-Lyndon theorem[21] [Hed69]). *In dimension $d \in \mathbb{N}$ and given two subshifts $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ and $Y \subseteq \mathcal{B}^{\mathbb{Z}^d}$, a function $f\colon X \to Y$ is a* morphism *if and only if it is a* cellular automaton, *i.e. there exists some finite $N \Subset \mathbb{Z}^d$ and a function $F\colon \mathcal{A}^N \to \mathcal{B}$ such that:*

$$\forall x \in \mathcal{A}^{\mathbb{Z}^d}, \forall \mathbf{i} \in \mathbb{Z}^d,\ f(x)_{\mathbf{i}} = F(x|_{\mathbf{i}+N}).$$

[Hed69] Hedlund, "Endomorphisms and automorphisms of the shift dynamical system".

[21] Gustav A. Hedlund co-credited Morton L. Curtis and Roger Lyndon as co-discoverers of this result on $\mathbb{Z}$. The generalization to higher dimensions followed in [Ric72].

[Ric72] Richardson, "Tessellations with local transformations".

Proof.

$\Longleftarrow$ If f is given by a cellular automaton of finite neighborhood $N \Subset \mathbb{Z}^d$ and rule $F\colon \mathcal{A}^N \to \mathcal{B}$, then f is shift-equivariant by definition; and f is continuous for the product topology, since the preimage of a cylinder of domain $D \subseteq \mathbb{Z}^d$ is a union of cylinders of domain $\bigcup_{\mathbf{i}\in D}(\mathbf{i}+N)$, and is in particular open.

$\Longrightarrow$ Let $f\colon X \to Y$ be a morphism. For any $b \in \mathcal{B}$ (considered as a pattern of domain $[\![1]\!]^d$), the cylinder $[\![b]\!] \cap Y$ is open in Y: and since f is continuous, $f^{-1}([\![b]\!])$ is open in X. Thus, $\bigcup_{b\in\mathcal{B}^{[\![1]\!]^d}} f^{-1}([\![b]\!])$ is open and can be written as a (potentially infinite) union of cylinders.

Since $\bigcup_{b\in\mathcal{B}^{[\![1]\!]^d}} f^{-1}([\![b]\!]) \supseteq X$ and that X is compact, there exists finitely many cylinders $[\![w^{(1)}]\!], \dots, [\![w^{(k)}]\!]$ that cover $\bigcup_{b\in\mathcal{B}^{[\![1]\!]^d}} f^{-1}([\![b]\!])$. Denoting by N the finite set $N = \bigcup_{i=1}^{k} \operatorname{dom}(w^{(i)})$, we obtain that for any $x \in X$, the symbol $f(x)_0$ is entirely determined by $x|_D$; and there exists a function $F\colon D \to \mathcal{B}$ such that $f(x)_0 = F(x|_D)$ for every $x \in X$. Since f is a morphism, it is σ-equivariant, and we conclude that f is the map defined by the cellular automaton of rule F and neighborhood N. □

Since subshifts are preseved under morphisms, we define the notion of *factors* of a subshift, which are the images of said subshift by morphisms:

Definition 3.23 (Factor). *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ and $Y \subseteq \mathcal{B}^{\mathbb{Z}^d}$ be two subshifts. Y is a* factor *of f if there exists a morphism $f\colon X \to Y$ such that $Y = f(X)$.*

In this case, $f\colon X \to Y$ is also the *factor map*. We will often use the terminology "factor map" in this thesis, as sofic subshifts will be defined in Section 3.4 as factors of subshifts of finite type.

3.3.2 Isomorphisms and conjugacy

Isomorphisms are, in mathematics, structure-preserving mappings between two objects of the same type. In the context of subshifts, they would be defined as bijective morphisms whose inverse are also morphisms. However, the latter condition is actually not needed:

Proposition 3.24. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ and $Y \subseteq \mathcal{B}^{\mathbb{Z}^d}$ be two subshifts, and $f\colon X \to Y$ be a bijective morphism. Then $f^{-1}\colon Y \to X$ is a morphism.*

Proof. Since $f\colon X \to Y$ is a continuous bijection between two compact spaces, f^{-1} is also continuous. Since f is shift-equivariant, f^{-1} is also shift-equivariant, and we conclude that f^{-1} is a morphism. □

Thus, we define isomorphisms as:

Definition 3.25 (Isomorphism)**.** *For $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ and $Y \subseteq \mathcal{B}^{\mathbb{Z}^d}$ two subshifts, an* isomorphism *$f\colon X \to Y$ is a bijective morphism.*

Isomorphisms define a way to identify subshifts that share similar properties. Such subshifts are said to be *conjugate*:

Definition 3.26 (Conjugacy)**.** *Two subshifts $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ and $Y \subseteq \mathcal{B}^{\mathbb{Z}^d}$ are* conjugate *if there exists a bijective morphism $f\colon X \to Y$.*

3.3.3 Conjugacy invariants

Since conjugacy might be difficult to determine (when given two subshifts, are they conjugate?), it is useful to consider the notion of *conjugacy invariants*:

Definition 3.27 (Conjugacy invariant)**.** *Let f be a function that associates to a subshift X a mathematical object. Then f is a* conjugacy invariant *if, for any two conjugate subshifts $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ and $Y \subseteq \mathcal{B}^{\mathbb{Z}^d}$, we have $f(X) = f(Y)$.*

Informally, a conjugacy invariant is a mathematical object that represents a subshift property that is shared by all conjugate subshifts. For example:

[22] From Definition 3.8: the entropy of a subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is:

$$h(X) = \lim_{n_1,\dots,n_d \in \mathbb{N}^d} \frac{\log |\mathcal{L}_{[\![n_1,\dots,n_d]\!]}(X)|}{n_1 \cdots n_d}.$$

Example 3.28. *Topological entropy*[22] *is invariant under conjugacy.*

Proof. If Y is a factor of X by a morphism $f\colon X \to Y$, let us consider an associated cellular automaton of neighborhood $[-k \,..\, k] \subseteq \mathbb{Z}^d$ and local rule $F\colon \mathcal{A}^{[-k..k]} \to \mathcal{B}$. Then for $n \in \mathbb{N}$, said cellular automaton defines a surjective mapping between languages $F_n\colon \mathcal{L}_{[\![n_1,\dots,n_d]\!]}(X) \to \mathcal{L}_{[\![n_1-2k,\dots,n_d-2k]\!]}(Y)$, so that $|\mathcal{L}_{[\![n_1-2k,\dots,n_d-2k]\!]}(Y)| \leq |\mathcal{L}_{[\![n_1,\dots,n_d]\!]}(X)|$.

In other words, entropy decreases under factor maps: if X and Y are conjugate, this implies by symmetry that $h(X) = h(Y)$. □

[23] See Definition 3.34: a subshift is of finite type (SFT) if it can be defined by a finite family of forbidden patterns.

Example 3.29. *Being of finite type*[23] *is a conjugacy invariant.*

Proof. Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ and $Y \subseteq \mathcal{B}^{\mathbb{Z}^d}$ be two subshifts and $f\colon X \to Y$ be an isomorphism. By definition, there exists a finite neighborhood $N \subseteq_f \mathbb{Z}^d$ and a rule $F\colon \mathcal{A}^N \to \mathcal{B}$ that defines f. Let us also assume that Y is an SFT: there exists $n \in \mathbb{N}$ such that $\mathcal{F}_Y = \mathcal{B}^{[\![n]\!]^d} \setminus \mathcal{L}(Y)$ is a forbidden pattern family defining Y. Since $N \subseteq \mathbb{Z}^d$ is finite, let us consider $\mathcal{F}_X = \mathcal{B}^{[\![n]\!]^d+N} \setminus \mathcal{L}(X)$.[24] We prove that $\mathcal{F}_X$ is a forbidden pattern family defining X:

[24] Where $A + B = \{\mathbf{i} + \mathbf{j}\colon \mathbf{i} \in A, \mathbf{j} \in B\}$.

- Since $\mathcal{F}_X \subseteq \mathcal{A}^{*d} \setminus \mathcal{L}(X)$, we have $X \subseteq X_{\mathcal{F}_X}$.
- On the other hand, let $x \in X_{\mathcal{F}_X}$ be a configuration containing no pattern of $\mathcal{F}_X$. Since every $x|_{\mathbf{i}+N}$ for $\mathbf{i} \in \mathbb{Z}^d$ is locally valid in X, and that f has neighborhood N, the configuration $f(x)$ is well-defined; and since every pattern of domain $[\![n]\!]^d + N$ in x is valid in X, every pattern of domain $[\![n]\!]^d$ is valid in Y. Thus, $f(x) \in Y$, and since f is bijective we conclude that $x \in X$. □

By definition, conjugacy invariants provide a sufficient condition for non-conjugacy: for example, two subshifts X and Y of distinct entropies cannot be conjugate. However, this condition is not necessary: two subshifts of the same entropy may or may not be conjugate.

To illustate the variety of mathematical objects that can occur as conjugacy invariants, we also mention an algebraic conjugacy invariant: namely, automorphism groups.

3.3.4 Automorphism groups

A special class of isomorphisms are the isomorphisms that map a subshift onto itself:

Definition 3.30 (Automorphism). *Given a subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$, an* automorphism *is an isomorphism $f : X \to X$ from X to itself.*

As the composition of two automorphisms is an automorphism, we define:

Definition 3.31 (Automorphism group). *Given a subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$, the* automorphism group $\mathrm{Aut}(X)$ *is the set of all its automorphisms equipped with map composition.*

Informally, the automorphism group of a subshift defines its set of "symmetries", *i.e.* all the ways to map it to itself while preserving its structure.

Proposition 3.32. *The automorphism group is a conjugacy invariant.*

Proof. Let X and Y be two conjugate subshifts, and $f : X \to Y$ be an isomorphism between X and Y. Then the two groups $\mathrm{Aut}(X)$ and $\mathrm{Aut}(Y)$ are isomorphic by the group isomorphism $\phi : g \in \mathrm{Aut}(X) \mapsto f \circ g \circ f^{-1} \in \mathrm{Aut}(Y)$. □

3.4 Classes of subshifts

As for languages of finite words, which are classified into local, regular, context-free languages... depending on their computational properties, several classes of subshifts have been introduced to stratify subshifts depending on the complexity of their presentations. In this section, we define and relate the classes of local subshifts, subshifts of finite type, sofic subshifts and effective subshifts.

3.4.1 Definitions

Local subshifts and subshifts of finite type

We begin with the definition of *local subshifts*, which are a straightforward generalization of local languages of finite words:

Definition 3.33 (Local subshift). *A subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is* local *if there exists a set of local validity rules $V_1, \dots, V_d \subseteq \mathcal{A}^2$ such that*[25]*:*

$$X = \{x \in \mathcal{A}^{\mathbb{Z}^d} : \forall \mathbf{i} \in \mathbb{Z}^d, \forall k \in [1 \,..\, d],\ (x_{\mathbf{i}}, x_{\mathbf{i}+\mathbf{e}_k}) \in V_k\}.$$

In other words, if X is a local subshift, the validity of a configuration x is checked by considering the symbols coloring every pair of adjacent positions. While they might seem very restricted, the Domino problem[26] is already undecidable on local subshifts[27].

Extending forbbiden patterns to larger domains leads to the class of subshifts of finite type:

Definition 3.34 (Subshift of finite type). *A subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is* of finite type (SFT) *if there exists a* finite *family of forbidden finite patterns $\mathcal{F}$ such that $X = X_{\mathcal{F}}$.*

Example 3.35. *The subshift drawn in Example 3.4 is a subshift of finite type (in fact, it is even a local subshift).*

[25] Recall that, on $\mathbb{Z}^d$, the standard unit vector $\mathbf{e}_k \in \mathbb{Z}^d$ along the k^{th} coordinate is defined as $(\mathbf{e}_k)_j = 0$ if $j \neq k$, and $(\mathbf{e}_k)_j = 1$ otherwise.

[26] Given a family of forbidden patterns $\mathcal{F}$, is the subshift $X_{\mathcal{F}}$ empty?

[27] It is undecidable on Wang tiles [Ber64], and Wang tiles are a special case of local subshift (see Section 12.1).

[Ber64] Berger, "The undecidability of the Domino problem".

[28] As do effective subshifts; but a problem turning out undecidable on effective subshifts is probably less surprising than its undecidability on subshifts defined by finitely many patterns.

Subshifts of finite type (SFTs) form the most classical class of multidimensional subshifts. Most decision problems about subshifts are usually defined on SFTs, because SFTs naturally provide a finite presentation to be given as input to an algorithm[28].

As we already mentioned after Definition 3.3, a given subshift can be defined by many distinct families of forbidden finite patterns; and it is of finite type if *at least one* of them is finite.

Effective subshifts

Definition 3.36 (Effective subshift)**.** *A subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is* effective *if there exists a* computably enumerable *family of forbidden finite patterns $\mathcal{F}$ such that $X = X_{\mathcal{F}}$.*

Effective subshifts are probably the most natural class of subshifts, at least from an intuitive point of view, since they can be defined according to arbitrary "algorithmic" conditions.

They are also the most natural class of subshifts on which it makes sense to consider decision problems. In this thesis, we will sometimes say that a decision problem takes as input an effective subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ – by which we actually mean a Turing machine enumerating an effective family of forbidden patterns $\mathcal{F}$ such that $X = X_{\mathcal{F}}$.

One could ask about the subshifts defined by *computable* families of patterns. By definition, such subshifts are effective; and conversely:

Proposition 3.37. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be an effective subshift. There exists a computable family of forbidden patterns $\mathcal{F}$ such that $X = X_{\mathcal{F}}$.*

Proof. Let $\mathcal{F}'$ be a computably enumerable family of forbidden patterns for X: it is enumerated by a Turing machine $\mathcal{M}$. We use a standard fuel argument from computability[29] and define:

$$\mathcal{F} = \bigcup_{n \in \mathbb{N}} \{w \in \mathcal{A}^{[\![n]\!]^d} : \exists w' \in \mathcal{A}^{*d}, w' \sqsubseteq w \text{ and } w' \text{ has been enumerated by } \mathcal{M} \text{ in less than } n \text{ steps}\}.$$

Then $\mathcal{F}$ is computable and also defines the subshift X. □

[29] While it undecidable to know whether a given patterns $w \in \mathcal{A}^{*d}$ is enumerated by $\mathcal{M}$, it is decidable to know, for a given n, if it is enumerated in less than n steps of computation.

Sofic subshifts

Sofic subshifts were originally introduced on $\mathbb{Z}$ in [Wei73]:

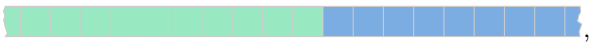
[Wei73] Weiss, "Subshifts of finite type and sofic systems".

Definition 3.38 (Sofic subshift)**.** *A subshift $Y \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is* sofic *if it is the factor of a subshift of finite type $X \subseteq \mathcal{B}^{\mathbb{Z}^d}$.*

In the previous definition, the subshift X is called an SFT *cover* of Y. **Sofic subshifts are the main objects of interests in this thesis.**

Example 3.39. *The most classical example of a sofic subshift is the* sunny-side-up *subshift. On the alphabet $\mathcal{A} = \{□, ■\}$, it is the set of configurations containing at most a single ■ cell, i.e. $Y = \{y \in \mathcal{A}^{\mathbb{Z}} : \forall i, j \in \mathbb{Z},\ y_i = y_j = ■ \implies i = j\}$.*

Indeed, consider the alphabet $\mathcal{B} = \{□, ■\}$ and define $X \subseteq \mathcal{B}^{\mathbb{Z}}$ by the family of forbidden patterns $\mathcal{F} = \{■■\}$. Then X is the set of configurations of the form



along, of course, the full □ and the full ■ configurations. Then X is an SFT cover of Y by the cellular automaton of neighborhood $N = \{-1, 0\} \subseteq \mathbb{Z}$ and local rule $F : \mathcal{B}^N \to \mathcal{A}$ defined by $F(■■) = ■$, and $F(w) =$ □ otherwise.

In Corollary 3.43, we prove an alternative characterization for sofic subshifts (as letter-by-letter projections of local subshifts).

3.4.2 Relating classes of subshifts

The aforementioned classes of subshifts are naturally included into the next. If local subshifts and SFTs are conjugated by Proposition 3.42, all the other inclusions are strict[30]:

$$\{\text{Local subshifts}\} \subseteq \{\text{SFTs}\} \subsetneq \{\text{Sofic subshifts}\} \subsetneq \{\text{Effective subshifts}\}.$$

[30] The sunny-side-up subshift is sofic but not of finite type; and the mirror subshift is effective but not sofic. See Chapter 6 and Chapter 7.

The only non-trivial inclusion is between sofic and effective subshifts:

Proposition 3.40. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a sofic subshift. Then X is an effective subshift.*

which follows from the more general statement:

Proposition 3.41. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be the factor of an effective subshift. Then X is an effective subshift.*

Proof. Let $X' \subseteq \mathcal{B}^{\mathbb{Z}^d}$ be an effective subshift defined by a computably enumerable family of forbidden patterns $\mathcal{F}'$; and $f\colon X' \to X$ be a morphism of finite neighborhood $N \subseteq_f \mathbb{Z}^d$ and a local rule $F\colon \mathcal{B}^N \to \mathcal{A}$.

For a pattern $w \in \mathcal{A}^{*d}$ of domain $D = \mathrm{dom}(w)$, denote by $f^{-1}(w)$ the set of all preimages of w by f, *i.e.* $f^{-1}(w) = \{u \in \mathcal{B}^{D+N} : \forall \mathbf{i} \in D, F(u|_{\mathbf{i}+N}) = w_{\mathbf{i}}\}$. And let us define:

$$\mathcal{F} = \bigcup_{n\in\mathbb{N}} \{w \in \mathcal{A}^{[\![n]\!]^d} : \forall w' \in f^{-1}(w), \exists u \in \mathcal{F}' \text{ such that } u \sqsubseteq w'\}.$$

Then $\mathcal{F}$ is a computably enumerable family of forbidden patterns that yields the subshift X. □

The rest of this section is dedicated to partial converse inclusions between these classes.

SFTs and local subshifts

Proposition 3.42. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift of finite type. There exists a local subshift $X' \subseteq \mathcal{B}^{\mathbb{Z}^d}$ that is conjugated to X.*

The proof just consists in taking a *higher block code* of the SFT X:

Proof. For $n \in \mathbb{N}$, let us denote the alphabet $\mathcal{B} = \mathcal{A}^{[-n..n]^d}$ and the subshift $X^{[n]} \subseteq \mathcal{B}^{\mathbb{Z}^d}$ defined as:

$$X^{[n]} = \{x \in \mathcal{B}^{\mathbb{Z}^d} : \exists x' \in X,\ x_{\mathbf{i}} = x'|_{\mathbf{i}+[-n..n]^d}\}.$$

$X^{[n]}$ is called a *higher block code* of the subshift X. Since X is of finite type, there exists some $n \in \mathbb{N}$ such that $X^{[n]}$ is a local subshift. Since $X^{[n]}$ and X are conjugate for every $n \in \mathbb{N}$, this completes the proof. □

Corollary 3.43. *Let $Y \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a sofic subshift. There exists an alphabet $\mathcal{B}$, a local subshift $X \subseteq \mathcal{B}^{\mathbb{Z}^d}$ and a projection $\pi\colon \mathcal{B} \to \mathcal{A}$ such that $Y = \pi(X)$.*

In this case, the subshift X is called a *local cover* of Y.

Proof. By definition, there exists an SFT X' and a factor map $f\colon X' \to Y$. By Proposition 3.22, f is the map associated to a cellular automaton of neighborhood $N \subseteq_f \mathbb{Z}^d$. Thus, there exists $n \in \mathbb{N}$ such that the higher block code (see Proposition 3.42) $X^{[n]}$ is a local subshift and such that $N \subseteq [-n\,..\,n]^d$: this concludes the proof. □

Sofic and effective subshifts

By Proposition 3.40, sofic subshifts are effective. The following proposition shows that effective subshifts turn out to be sofic, *up to an increase in dimension*: this initially appeared in [Hoc09], and was independently improved in [AS13] and [DRS10]:

[Hoc09] Hochman, "On the dynamics and recursive properties of multidimensional symbolic systems".

[AS13] Aubrun and Sablik, "Simulation of effective subshifts by two-dimensional subshifts of finite type".

[DRS10] Durand, Romashchenko, and Shen, "Effective closed subshifts in 1D can be implemented in 2D".

Proposition 3.44 ([AS13; DRS10])**.** *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be an effective subshift. Then $X^{\Uparrow} \subseteq \mathcal{A}^{\mathbb{Z}^{d+1}}$ is a sofic subshift.*

Proving this statement is completely outside the scope of "gentle definitions"; however, we actually provide a novel proof of this result in Section 14.3.1.

3.4.3 Computational considerations

To conclude this section, let us briefly mention some results about the computability of languages of subshifts:

Proposition 3.45. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be an effective subshift. Then the set of finite patterns $\mathcal{L}(X) \cap \mathcal{A}^{*d}$ is a Π_1^0 language.*

Proof. Let $\mathcal{F}$ be a computably enumerable family of forbidden patterns defining X, and let $w \in \mathcal{A}^{*d}$ be a pattern of finite domain $D \subseteq_f \mathbb{Z}^d$. Applying compactness, w is not valid in X if and only if:

$$\exists n \in \mathbb{N}, \forall w' \in \{w' \in \mathcal{A}^{[-n..n]^d} : w'|_D = w\}, \exists u \in \mathcal{F} \text{ such that } u \sqsubseteq w'.$$

Since this condition is computably enumerable (*i.e.* Σ_1^0), this concludes the proof. □

[Ber64] Berger, "The undecidability of the Domino problem".

How hard can the languages of subshifts be? Since the domino problem is undecidable on local subshifts [Ber64], we conclude that all these classes of subshifts can have Π_1^0-hard, thus Π_1^0-complete, languages.

Since many properties of subshifts can actually be formulated on their languages, this implies that many properties of subshifts (even local subshifts) are undecidable. This motivates the introduction of a new class/property of subshifts:

Definition 3.46 (Computable subshift)**.** *A subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is* computable *if its set of finite patterns $\mathcal{L}(X) \cap \mathcal{A}^{*d}$ forms a computable language.*

By definition, computable subshifts are effective. Yet, as mentioned above, the class of computable subshifts is somewhat orthogonal to local, SFTs and sofic subshifts: indeed, there exists effective and non-sofic subshifts with computable languages, and local subshifts whose languages are undecidable.

3.5 Dynamics

Considering the action of $\mathbb{Z}^d$ by translations on a subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$, we can study the behavior of the configurations of X under the action $\mathbb{Z}^d \curvearrowright X$, which results in a topological dynamical system. This allows to study the orbits of the configurations of X under various dynamical properties, including periodicity, recurrence properties...

3.5.1 Minimality

Definition 3.47 (Minimality). *A subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is* minimal *if it contains no non-empty proper subshift (i.e. a subshift $Y \subseteq X$ verifies $Y = \emptyset$ or $Y = X$).*

In the case of subshifts, this can be rephrased into the following statement:

Proposition 3.48. *A subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is minimal if and only if for every finite pattern $w \in \mathcal{L}(X) \cap \mathcal{A}^{*d}$, there exists a size $N \in \mathbb{N}$ such that, for every $w' \in \mathcal{L}_{[\![N]\!]^d}(X)$, we have $w \sqsubseteq w'$.*

Proof. If X is minimal, then every finite pattern $w \in \mathcal{L}(X) \cap \mathcal{A}^{*d}$ appears in every configuration $x \in X$: the rest of the proposition follows from X being compact. □

Minimality is a well-known restriction on the possible behaviors of subshifts. For example:

Proposition 3.49. *Let $Y \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a sofic minimal subshift. Then $h(Y) = 0$.*

Proof. By [Des06, Corollary 4.5], a sofic shift Y of positive entropy is not minimal. □

[Des06] Desai, "Subsystem entropy for $\mathbb{Z}^d$ sofic shifts".

Proposition 3.50. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be an effective minimal subshift. Then its language of finite patterns $\mathcal{L}(X) \cap \mathcal{A}^{*d}$ is computable.*

Proof. By Proposition 3.45, the language $\mathcal{L}(X) \cap \mathcal{A}^{*d}$ is Π_1^0. We prove that it is Σ_1^0, *i.e.* computably enumerable. This holds by definition if X is empty; and if X is non-empty, for $w \in \mathcal{A}^{*d}$, let us consider the subshift $X_{\cup\{w\}} \subseteq X$ of X in which w is also forbidden: since X is minimal, w belongs in $\mathcal{L}(X)$ if and only if $X_{\cup\{w\}}$ is empty, which is computably enumerable. □

3.5.2 Transitivity and mixingness

Definition 3.51 (Transitivity). *A subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is* transitive *if, for all finite patterns $u, v \in \mathcal{L}(X) \cap \mathcal{A}^{*d}$, there exists a configuration x such that $u \sqsubseteq x$ and $v \sqsubseteq x$.*

Intuitively, in a transitive subshift, all pairs of pattern must appear together in a configuration, without any way to control the distance between their respective occurrences.

Mixingness is a stronger notion of recurrence which is well-defined on $\mathbb{Z}$:

Definition 3.52 (Mixingness). *A subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is* mixing *if, for every $u, v \in \mathcal{L}(X) \cap \mathcal{A}^*$, there exists $n \in \mathbb{N}$ and such that, for every $|n'| \geq n$, there exists $x \in X$ such that $x \in [\![u]\!] \cap [\![\sigma^{n'}(v)]\!]$.*

Intuitively, in a mixing subshift, all pairs of patterns can appear together in a configuration at all possible relative positions, according that said positions are sufficiently far from one another.

On $\mathbb{Z}^d$, defining mixingness is complicated by the fact that patterns can have arbitrary shapes (*e.g.* rings, which could be nested). To avoid these pathological cases, mixingness is often defined on rectangular patterns:

Definition 3.53 (Block-gluingness). *A subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is $f(n)$-*block gluing *if, for all patterns $u, v \in \mathcal{L}_{[\![n]\!]^d}(X)$, and for all positions $\mathbf{i} \in \mathbb{Z}^d$ such that $\|\mathbf{i}\| \geq n + f(n)$, there exists $x \in X$ such that $x \in [\![u]\!] \cap [\![\sigma^{\mathbf{i}}(v)]\!]$.*

The function f in the previous definition is called the *gluing distance*. We will often say that X is c-block gluing for $c \in \mathbb{N}$ if it is $f(n)$-block gluing for the constant function $f : n \mapsto c$.

4 Computability

This chapter introduces some elementary definitions from computability theory: namely, the computable functions. Through encodings, they allow to define computable sets and decisions problems. We also present the arithmetical hierarchy, which stratisfies decision problems depending on their degree of (un)computability.

Finally, we specify the log-RAM model that is used in this thesis to state precise polynomial-time complexity results. We also reprove several well-known computable transformations of programs (S_n^m theorem, recursion theorems…) in the case of the log-RAM model, and prove their preservation of time complexity.

4.1 Basic definitions

4.1.1 Computation model

To avoid relying too much on the specifics of a computational model, the whole definitions will be stated for an informal but fixed Turing-complete model[31]. Intuitively, a Turing-complete computational model should be able to read a memory, write said memory, count and perform comparisons… Since all Turing-complete computational models define the same set of computable functions, this choice has no actual influence on the computability notions it induces.

[31] Which is something most modern programming languages are, unless specifically designed not to be.

As with modern programming languages, *programs* (*i.e.* sequences of instructions performed by said fixed computational model) will be represented as binary words $e \in \{0,1\}^*$, which are called *codes*. To fix one coding scheme, $e \in \{0,1\}^*$ will be the string (encoded in binary) representing the program in our fixed programming language. To keep the difference between codes and functions explicit, we denote:

Definition 4.1. *Let $\varphi_e : \subseteq \{0,1\}^* \to \{0,1\}^*$ be the (partial) function computed by the program of code $e \in \{0,1\}^*$.*

As is often the case in programming languages, the function φ_e is a *partial* function from $\{0,1\}^*$ to itself. This may be induced by computations that do not terminate properly, *e.g.* if the program e runs forever on an input, or if the program e crashes due to some programming error.

We denote $\varphi_e(u)\downarrow$ to signify that the program of code e halted properly on the input word $u \in \{0,1\}^*$, and $\varphi_e(u)\downarrow = v$ to denote that $v \in \{0,1\}^*$ was the ouput of the program of code e on the input word u.

[32] These will actually be non-deterministic functions, *i.e.* multifunctions. But let us not worry about that here.

Note. *In this thesis, programs will be written (unless otherwise specified) for the Random-Access Machine model: in this context, φ_e will denote the function[32] implemented by the RAM program of code e. See Section 4.3 for more details.*

4.1.2 Computable functions, computable sets

Since we have a definition of programs, computable functions are now the partial functions that can be implemented as codes in our fixed programming language:

Definition 4.2 (Computable function)**.** *A function $f\colon \subseteq \{0,1\}^* \to \{0,1\}^*$ is* computable *if there exists a code $e \in \{0,1\}^*$ such that $\varphi_e(u)\!\downarrow = f(u)$ for $u \in \mathrm{Dom}(f)$, and $\varphi_e(u)\!\uparrow$ otherwise.*

To define computations on arbitrary countable sets, we represent the elements of these sets as binary strings. More precisely, for a countable set S, an *encoding* is an injective map $\eta\colon S \to \{0,1\}^*$. In this context:

Definition 4.3 (Computable function)**.** *Let S, S' be two arbitrary countable sets equipped with respective encoding $\eta\colon S \to \{0,1\}^*$ and $\eta'\colon S' \to \{0,1\}^*$. A function $f\colon \subseteq\colon S \to S'$ is* computable *if its encoding is computable, i.e. if the function $\eta(s) \in \{0,1\}^* \mapsto \eta'(f(s)) \in \{0,1\}^*$ is computable.*

By definition, computability on a set S depends on the chosen encoding $\eta\colon S \to \{0,1\}^*$. Since most reasonnable encodings of classical objects are computable from one another, we will often avoid specifying the encoding altogether. Usually, integers of $\mathbb{N}$ (resp. $\mathbb{Z}$) are represented as binary strings through their binary expansion; elements of $\mathbb{N}^m$ for arbitrary m as tuples of integers; and booleans $\{\top, \bot\}$ as the 1-bit strings $\{1, 0\}$.

The adjective computable also naturally applies to sets:

Definition 4.4 (Computable set)**.** *A set $S \subseteq X$ is* computable *if the function $1_S\colon X \to \{\top, \bot\}$ defined by $1_S(x) = 1$ if $x \in S$, and $1_S(x) = 0$ if $x \notin S$, is computable.*

4.1.3 Decision problems

From computability theory, we are mostly interested in the notion of *decision problems*, *i.e.* functions $f\colon \{0,1\}^* \to \{\top, \bot\}$ that can take two output values: true ($\top$) and false ($\bot$):

Definition 4.5 (Decision problem)**.** *A* decision problem PROBLEM *is a function $f\colon X \to \{\top, \bot\}$. An element $x \in X$ is a* positive instance *of f if $f(x) = \top$; and a* negative instance *if $f(x) = \bot$.*

In other words, a decision problem is a set. In this thesis, though, we will often define a decision problem PROBLEM as a yes-or-no question about some inputs:

> PROBLEM
> **Input:** some object $x \in X$;
> **Question:** does x verify some property $\mathcal{P}$?

to actually denote the set $\{x \in X\colon x \text{ verifies } \mathcal{P}\}$. From the previous definitions, a decision problem PROBLEM is *computable* (or *decidable*) if it defines a computable set; and *undecidable* otherwise.

We finally introduce the notion of (*many-one*) *reductions*, which allow to compare the difficulty of two decision problems:

Definition 4.6 ((Many-one reduction))**.** *Let $A \subseteq X$ and $B \subseteq X'$ be two decisions problems. We say that B is* harder *than A (or that A* reduces to *B), written $A \leq_m B$, if there exists a total computable function $f\colon X \to X'$ such that $x \in A$ if and only if $f(x) \in B$.*

Remark. ⚠ *Notice that, in the previous definition, f is not required to be surjective.*

Intuitively, reductions transform elements of the set X into elements of the set X' so that positive instances of A (resp. negative) are mapped to positive instances of B (resp. negative). In particular, if the set B is computable, and that A (many-one) reduces to B, then A is also computable.

4.1.4 Computably enumerable sets

Using the fact that computable functions can be partial, one defines semi-computable sets as the sets whose membership can be decided only for positive instances:

Definition 4.7 (Computably enumerable set)**.** *A set $S \subseteq X$ is* computably enumerable *if the partial function $1_S \colon \subseteq X \to \{\top, \bot\}$ defined as $1_S(x) = \top$ if $x \in S$ (and $1_S(x)$ is undefined otherwise) is computable.*
A set $S \subseteq X$ is co-computably enumerable *if $X \setminus S$ is computably enumerable.*

The most famous undecidable set/decision problem is the Halting problem[33], which is actually computably enumerable[34].

[33] Given a code $e \in \{0,1\}^*$, does the function φ_e halt on the empty input?

[34] To prove its semi-computability: begin the computation of $\varphi_e(\varepsilon)$, and return $\top$ if the computation halts.

4.2 Arithmetical hierarchy

Since we defined computable sets in the previous section, one could believe that computability theory has nothing to say about sets that are not computable. However, it turns out that computability allows to stratify subsets of $\{0,1\}^*$ depending on how undecidable they are.

4.2.1 Arithmetical hierarchy of sets

More precisely, the arithmetical hierarchy classifies the degree of uncomputability of a set $X \subseteq \{0,1\}^*$ depending on the number of *alternating quantifiers* required to obtain it from computable sets:

Definition 4.8 (Arithmetical hierarchy)**.** *For $n \in \mathbb{N}$, a set $X \subseteq \{0,1\}^*$ is said to be*

(i) Σ_n^0 *if there exists a computable relation $R \subseteq (\{0,1\}^*)^{n+1}$ such that:*

$$X = \{u \in \{0,1\}^* \colon \exists v_1, \forall v_2, \exists v_3 \dots \ R(v_1, \dots, v_n, u)\}.$$

(ii) Π_n^0 *if there exists a computable relation $R \subseteq (\{0,1\}^*)^{n+1}$ such that:*

$$X = \{u \in \{0,1\}^* \colon \forall v_1, \exists v_2, \forall v_3 \dots \ R(v_1, \dots, v_n, u)\}.$$

(iii) Δ_n^0 *if it is both Σ_n^0 and Π_n^0.*

Remark. *As mentioned above, in these formulas, the alternance of quantifiers is important: indeed, successive universal or successive existential quantifiers can, through encoding of tuples, be collapsed into a single quantifier of the same type.*

Example 4.9.

1. *A set $X \subseteq \{0,1\}^*$ is Σ_n^0 if and only if its complement $\{0,1\}^* \setminus X$ is Π_n^0.*
2. *A set $X \subseteq \{0,1\}^*$ is Σ_1^0 if and only if it is computably enumerable. Indeed, there exists a computable relation $R \subseteq (\{0,1\}^*)^2$ such that $X = \{u \in \{0,1\}^* \colon \exists v \in \{0,1\}^*, R(v,u)\}$. Then the program enumerating all $v \in \{0,1\}^*$ in parallel and checking whether $R(v,u)\downarrow = \top$ semi-decides whether $u \in X$.*

3. *A set $X \subseteq \{0,1\}^*$ is Δ^0_1 if and only if it is computable. Indeed, it is computably and co-computably enumerable, and one answers whether $u \in X$ by running the positive and the negative enumerations in parallel and checking which one answers.*

Proposition 4.10. *The sets of Σ^0_n and Π^0_n sets is stable by finite unions and intersections.*

Hint. Encode $(\{0,1\}^*)^2$ into $\{0,1\}^*$ to quantify on pairs of strings. □

Proposition 4.11. *For $n \in \mathbb{N}$, we have $\Sigma^0_n \subsetneq \Delta^0_{n+1}$ and $\Pi^0_n \subsetneq \Delta^0_{n+1}$.*

Hints. The inclusions follow from the definition. To prove that these inclusions are strict, one usually proves that the n^{th} Turing jump $\emptyset^{(n)}$ is Σ^0_n and cannot be Π^0_n, and encode the tuple $(\emptyset^{(n)}, \mathbb{N} \setminus \emptyset^{(n)})$ to obtain a Δ^0_{n+1} set that cannot be Σ^0_n or Π^0_n. For more details, see [MP22, Corollaire 5.5.6][35]. □

[MP22] Monin and Patey, *Calculabilité*.

[35] The book [MP22] is in French, but I am very partial to this book. In fact, the idea of defining computable functions without first introducing a detailed computational model is directly inspired by their approach of computability. Since an English version of the book is in preparation, in a few years from now English speakers should be able to follow this reference.

Since the arithmetical hierarchy defines infinitely many levels and does not collapse, we are interested in pinpointing the exact complexity of sets in this hierarchy. Relying on the notion of (many-one) *reductions* introduced earlier, consider:

Proposition 4.12. *Let $A \subseteq \{0,1\}^*$ and $B \subseteq \{0,1\}^*$ be two sets such that $A \leq_m B$. If B is Σ^0_n (resp. Π^0_n), then so is A.*

Proof. Let $f \colon \{0,1\}^* \to \{0,1\}^*$ be a reduction from A to B. If there exists a computable predicate $R \subseteq (\{0,1\}^*)^n \times \{0,1\}^*$ such that

$$B = \{u \in \{0,1\}^* \colon \exists v_1, \forall v_2, \dots \ R(v_1, \dots, v_n, u)\},$$

then, since $u \in A$ if and only if $f(u) \in B$, we have:

$$A = \{u \in \{0,1\}^* \colon \exists v_1, \forall v_2, \dots \ R(v_1, \dots, v_n, f(u))\}.$$

Since f is computable, the predicate $R(\cdots, f(\cdot))$ is also computable and provides a Σ^0_n presentation of A. The case Π^0_n is symmetric. □

Thus, we use many-one reductions to compare sets in the arithmetical hierarchy:

Definition 4.13 (Hardness)**.** *A set $X \subseteq \{0,1\}^*$ is Σ^0_n-hard (resp. Π^0_n-hard) if, for every Σ^0_n set $A \subseteq \{0,1\}^*$ (resp. Π^0_n), we have $A \leq_m X$.*

In other words, a Σ^0_n-hard set X is harder (for many-one reductions) than all Σ^0_n sets. Intuitively, this means that X is at level Σ^0_n or higher in the arithmetical hierarchy.

Definition 4.14 (Completeness)**.** *A set $X \subseteq \{0,1\}^*$ is Σ^0_n-complete (resp. Π^0_n-complete) if:*

- *X is Σ^0_n-hard (resp. Π^0_n-hard);*
- *and $X \in \Sigma^0_n$ (resp. $X \in \Pi^0_n$).*

Example 4.15. *Already mentioned earlier, the Halting problem* Halt *is actually Σ^0_1-complete.*

Proof. If $A \subseteq \{0,1\}^*$ is a Σ^0_n set, there exists a code e that enumerates the elements of A. Let f be the computable function that, on a given word $u \in \{0,1\}^*$, defines the program $e_u \in \{0,1\}^*$ that begins a computation f and stops if e ever enumerates u (or keeps waiting otherwise). Then $\varphi_{f(u)}(\varepsilon)$ halts if and only if $u \in A$, so that f is a reduction from A to Halt. □

Other examples of problems include FIN (given a code $e \in \{0,1\}^*$, does it halt on finitely many inputs?), which is Σ^0_2-complete; COFIN (given a code $e \in \{0,1\}^*$, does φ_e halt on all but finitely many inputs), which is Σ^0_3-complete...

Remark 4.16. *Though we formally defined the arithmetical hierarchy on sets of binary strings, these definitions extend through encodings to arbitrary encoded sets.*

4.2.2 Arithmetical hierarchy of real numbers

Real numbers can also be classified depending on their computational properties. For example:

Definition 4.17. A *real number $x \in \mathbb{R}$ is* computable *if there exists a computable function $f\colon \mathbb{N} \to \mathbb{Q}$ such that, for all $n \in \mathbb{N}$, we have $|x - f(n)| \leq 2^{-n}$.*

Since the set of computable functions is countable, only a countable subset of real numbers is actually computable. However, most "practical" real numbers turn out to be computable: this includes all rational numbers, usual operations on real numbers (addition, product, exponential, logarithm...), and many constants π, e, ...

The arithmetical hierarchy can also stratify real numbers depending on the difficulty of approximating them computably [ZW01]:

[ZW01] Zheng and Weihrauch, "The arithmetical hierarchy of real numbers".

Definition 4.18 (Arithmetical hierarchy on $\mathbb{R}$)**.** *For $n \in \mathbb{N}$, a real number $x \in \mathbb{R}$ is said to be:*

(i) Σ_n *if the set $\{r \in \mathbb{Q}\colon r \leq x\}$ is a Σ^0_n subset of $\mathbb{Q}$.*
(ii) Π_n *if the set $\{r \in \mathbb{Q}\colon r \leq x\}$ is a Π^0_n subset of $\mathbb{Q}$.*
(iii) Δ_n *if x is both Σ_n and Π_n.*

In the same way sets are classified in the arithmetical hierarchy depending on the number of alternating quantifiers in their definition, the arithmetical hierarchy classifies real numbers depending on the number of alternating limit operations required to define them[36]:

[36] This is how the hierarchy was originally introduced in [ZW01, Definition 7.1].

Proposition 4.19. *For $n \in \mathbb{N}$, a real number $x \in \mathbb{R}$ is:*

(i) Σ_n *if and only if there exists a computable function $f\colon \mathbb{N}^n \to \mathbb{Q}$ such that $x = \sup_{i_1} \inf_{i_2} \sup_{i_3} \dots f(i_1, \dots, i_n)$.*
(ii) Π_n *if and only if there exists a computable function $f\colon \mathbb{N}^n \to \mathbb{Q}$ such that $x = \inf_{i_1} \sup_{i_2} \inf_{i_3} \dots f(i_1, \dots, i_n)$.*

[37] i.e $r \in A \implies \forall r' \leq r, r' \in A$.

Proof. We prove by induction that if $A \subseteq \mathbb{Q}$ is a Σ^0_n and downwards closed[37] (resp. upwards closed) set described by a computable relation $R \subseteq \mathbb{N}^n \times \mathbb{Q}$, then there exists a function $f\colon \mathbb{N}^n \to \mathbb{Q}$, which is *computable from R*, such that $\sup A = \sup_{i_1} \inf_{i_2} \dots\ f(i_1, \dots, i_n)$ (resp. $\inf A = \inf_{i_1} \sup_{i_2} \dots\ f(i_1, \dots, i_n)$):

- $n = 1$: assume that $A \subseteq \mathbb{Q}$ is downwards closed (resp. upwards closed) and Σ^0_1 set: from a relation $R \subseteq \mathbb{N} \times Q$ describing A, we compute a surjective enumeration $f\colon \mathbb{N} \to A$ that verifies $\sup_{i_1} f(i_1) = \sup A$ (resp. $\inf_{i_1} f(i_1) = \inf A$).
- Assume that $A \subseteq \mathbb{Q}$ is downwards closed and Σ^0_{n+1} set. By definition, there exists a computable relation $R\colon \mathbb{N}^n \times \mathbb{Q}$ such that $r \in A$ if and only if $\exists i_1, \forall i_2, \dots\ R(i_1, \dots, i_n, r)$. Thus:

$$A = \bigcup_{i_1} \{r \in \mathbb{Q}\colon \forall i_2, \exists i_3, \dots\ R(i_1, i_2, \dots, i_n, r)\}.$$

The sets on the right-hand side of the equation are uniform Π_n sets, but not necessarily downwards closed: thus, we define a new relation $R' : \mathbb{N}^n \times \mathbb{Q} \to \{\top, \bot\}$ by $R'(i_1, \dots, i_n, r)$ if and only if, decoding $i_1 \in \mathbb{N}$ as a pair $(i_1', r') \in \mathbb{N} \times \mathbb{Q}$, we have $R(i_1', \dots, i_n, r')$ and $r \leq r'$. Denoting $A_{i_1} = \{r \in \mathbb{Q} : \forall i_2, \exists i_3, \dots\ R'(i_1, \dots, i_n, r)\}$, each A_{i_1} is downwards closed and we have

$$A = \bigcup_{i_1} A_{i_1}$$

since A was already downwards closed. Thus, A is a uniform union of downwards closed and Π^0_n sets. By induction hypothesis on the complements $\mathbb{Q} \setminus A_{i_1}$, there exists a computable function[38] $f : \mathbb{N}^n \to \mathbb{Q}$ such that, for all $i_1 \in \mathbb{N}$:

$$\inf_{i_2} \sup_{i_3} \dots\ f(i_1, i_2, \dots, i_n) = \inf(\mathbb{Q} \setminus A_{i_1}).$$

[38] Here, we use that the functions f_{i_1} obtained by induction hypothesis can all be computed from $R'(i_1, \cdots)$.

Since the A_{i_1} are downwards closed, notice that $\inf(\mathbb{Q} \setminus A_{i_1}) = \sup A_{i_1}$. Since $\sup A = \sup_{i_1} \sup A_{i_1}$, and that a similar reasonning applies to upwards-closed and Σ^0_{n+1} sets, this concludes the induction.

The converse statement is much easier to prove: let $f : \mathbb{N}^n \to \mathbb{Q}$ be a computable function such that $x = \sup_{i_1} \inf_{i_2} \sup_{i_3} \dots\ f(i_1, \dots, i_n)$. Then

$$\begin{aligned}
r \leq x &\iff r \leq \sup_{i_1} \inf_{i_2} \sup_{i_3} \dots\ f(i_1, \dots, i_n) \\
&\iff \exists i_1,\ r \leq \inf_{i_2} \sup_{i_3} \dots\ f(i_1, \dots, i_n) \\
&\iff \exists i_1, \forall i_2,\ r \leq \sup_{i_3} \dots f(i_1, \dots, i_n) \\
&\iff \dots \\
&\iff \exists i_1, \forall i_2, \exists i_3, \dots\ r \leq f(i_1, \dots, i_n);
\end{aligned}$$

so that $\{r \in \mathbb{Q} : r \leq x\}$ is a Σ^0_n set. □

Following the previous proposition, arithmetical real numbers are given by alternating limit operations on sequences of rational numbers. It is, however, possible to assume some natural monotonicity conditions on these sequences:

Proposition 4.20. *For $n \in \mathbb{N}$, a real number $x \in \mathbb{R}$ is:*

(i) *Σ_n if and only if there exists a computable function $f : \mathbb{N}^n \to \mathbb{Q}$ such that $x = \sup_{i_1} \inf_{i_2} \sup_{i_3} \dots\ f(i_1, \dots, i_n)$; and:*
the sequence $i_1 \mapsto \inf_{i_2} \sup_{i_3} \dots\ f(i_1, \dots, i_n)$ is increasing; the sequences $i_2 \mapsto \sup_{i_3} \inf_{i_4} \dots\ f(i_1, \dots, i_n)$ are decreasing for fixed i_1; the sequences $i_3 \mapsto \inf_{i_4} \sup_{i_4} \dots\ f(i_1, \dots, i_n)$ are increasing for fixed $(i_1, i_2) \in \mathbb{N}^2$; etc...

(ii) *Π_n if and only if there exists a computable function $f : \mathbb{N}^n \to Q$ such that $x = \inf_{i_1} \sup_{i_2} \inf_{i_3} \dots\ f(i_1, \dots, i_n)$; and:*
the sequence $i_1 \mapsto \inf_{i_2} \sup_{i_3} \dots\ f(i_1, \dots, i_n)$ is decreasing; the sequences $i_2 \mapsto \sup_{i_3} \inf_{i_4} \dots\ f(i_1, \dots, i_n)$ are increasing for fixed i_1; the sequences $i_3 \mapsto \inf_{i_4} \sup_{i_4} \dots\ f(i_1, \dots, i_n)$ are decreasing for fixed $(i_1, i_2) \in \mathbb{N}^2$; etc...

Proof. Let $x \in \mathbb{R}$ be a real number given by a function $f' : \mathbb{N}^n \to \mathbb{Q}$ such that $x = \dots \inf_{i_{n-2}} \sup_{i_{n-1}} \inf_{i_n}\ f'(i_1, \dots, i_n)$. We replace f' by the function f:

$$f(i_1, \dots, i_n) = \dots \min_{i'_{n-2} \leq i_{n-2}} \max_{i'_{n-1} \leq i_{n-1}} \min_{i'_n \leq i_n} f'(i'_1, \dots, i'_n).$$

Then $f : \mathbb{N}^n \to \mathbb{Q}$ is computable and verifies the desired monotonicity properties. We are left with cheecking that x is the limit of f: fixing the first

$(n-1)$ variables $i_1, \dots, i_{n-1} \in \mathbb{N}$, we have:

$$\begin{aligned}
\inf_{i_n} f(i_1, \dots, i_{n-1}, i_n) &= \inf_{i_n} \dots \max_{j_{n-1} \le i_{n-1}} \min_{j_n \le i_n} f'(j_1, \dots, j_n) \\
&= \lim_{i_n} \dots \max_{j_{n-1} \le i_{n-1}} \min_{j_n \le i_n} f'(j_1, \dots, j_n) \qquad \text{(By monotonicity, the limit exists.)} \\
&= \dots \max_{j_{n-1} \le i_{n-1}} \lim_{i_n} \min_{i'_n \le i_n} f'(j_1, \dots, j_n) \\
&= \dots \max_{j_{n-1} \le i_{n-1}} \inf_{i_n} f'(j_1, \dots, j_{n-1}, i_n).
\end{aligned}$$

By decreasing induction on the number of fixed variables, we conclude that $x = \dots \inf_{i_{n-2}} \sup_{i_{n-1}} \inf_{i_n} f(i_1, \dots, i_n)$. The other case being symmetric, this concludes the proof. □

4.3 The RAM model

Since the most common models (*e.g.* Turing machines, various Random Access Machines…) simulate each other with small overhead[39], complexity-related results often try to abstract themselves from the specifics of a given computational model. However, in this thesis, we will need (for example in Theorem 10.11) precise computation times, in which a polynomial overhead is not acceptable. Thus, we need to precisely state the computation model used: here, *Random Access Machines* with non-unit arithmetic operations.

[39] For example, a deterministic multitape Turing machine can simulate the execution of RAM programs up to a polynomial factor overhead in time complexity.

Random Access Machines (RAM) belong to the general class of *register machines*, which can be informally defined as a computational model that operates on multiple registers, each containing a single integer. These models were mostly developed during the 60s and early 70s, as alternatives to Turing machines that would better reflect the inner workings of modern computers: for example, let us mention the "pebbles in the ground" model [Mel61], various "counter machines" [Wan57; Min61], "Random Access Stored Program machines" [ER64; Har71] and "Random Access Machines" [CR73].

Two main competing models of complexity have been considered on Random Access Machines: the unit cost model (arithmetic operations are unitary constant time operations), and the logarithmic cost model (arithmetic operations happen in linear time relatively to the length of the integers involved). Very early in the history of these machines, it was noticed that unit cost and unbounded integers in registers could lead to somewhat unrealistic results: sorting an array in linear time [KR84], or solving PSPACE-complete problems in polynomial time [HS74]. Yet, most time complexity analysis is done with the unit criterion. There probably are two reasons for this: first, analysis of algorithms is easier with the unit cost criterion; second, reasonnable people will not cheat and will limit their RAM algorithms to "small integers".

The latter idea was formally introduced in the "Logarithmic Random Access Machines" model in [Sli78; Sli79; Sli81]: this model limits register sizes to $O(\log n)$, where n is the size of the input. Variations of this model also appeared in the litterature, for example "Random Access Computers" in [AV79, Appendix 1].

[Mel61] Melzak, "An informal arithmetical approach to computability and computation".

[Wan57] Wang, "A variant to Turing's theory of computing machines".

[Min61] Minsky, "Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines".

[ER64] Elgot and Robinson, "Random-access stored-program machines, an approach to programming languages".

[Har71] Hartmanis, "Computational complexity of random access stored program machines".

[CR73] Cook and Reckhow, "Time bounded random access machines".

[KR84] Kirkpatrick and Reisch, "Upper bounds for sorting integers on random access machines".

[HS74] Hartmanis and Simon, "On the power of multiplication in random access machines".

[Sli78] Slissenko, «Методы вычислений, основанные на адресной организации памяти».

[Sli79] Slissenko, «Сложностные задачи теории вычислений».

[Sli81] Slissenko, «Поиск периодичностей и идентификация полслов в реальное время».

[AV79] Angluin and Valiant, "Fast probabilistic algorithms for Hamiltonian circuits and matchings".

4.3.1 Random Access Machines

In the RAM *model*, registers (*i.e.* variables and memory cells) contain *binary words*, which are either read litteraly as strings of $\{0,1\}^*$ or as natural numbers of $\mathbb{N}$ (little-endian[40]).

[40] *i.e.* least significant digit first.

Definition 4.21. A Random Access Machine *is composed of*:

(*i*) A program, *which is a finite sequence of* instructions (*see below*) *indexed by integers*; *and an* instruction pointer $\texttt{IP} \in \mathbb{N}$, *which points to the instruction currently executed in the program*;
(*ii*) A *finite set of variables* V, *each containing a binary string*;
(*iii*) A memory *composed of infinitely many* memory cells $(M[j])_{j\in\mathbb{N}}$ *indexed* (*or "addressed"*) *by* $\mathbb{N}$, *each containing a binary string*;
(*iv*) *An* input array $(I[j])_{j\in\mathbb{N}}$ *and an* output array $(O[j])_{j\in\mathbb{N}}$, *each composed input cells* (*resp. output cells*) *containing binary strings and indexed by* $\mathbb{N}$.

A *program* in the RAM model is a finite sequence of instructions among the following set. Each instruction is given with its syntax and a high-level semantics. The names $\texttt{var}_0$, $\texttt{var}_1$ and $\texttt{var}_2$ denote arbitrary variables of V. At the end of each instruction (unless the instruction modified it, *e.g.* with `GOTO`), the instruction pointer `IP` is set to $\texttt{IP}+1$.

Memory management:

$\texttt{var}_0 \leftarrow \texttt{I[var}_1\texttt{]}$	Copy the word $I[\texttt{var}_1]$ from the input array into the variable $\texttt{var}_0$.
$\texttt{O[var}_0\texttt{]} \leftarrow \texttt{var}_1$	Outputs the word $\texttt{var}_1$ onto the cell $O[\texttt{var}_0]$ of the output array.
$\texttt{var}_0 \leftarrow \texttt{n}$	Set the variable $\texttt{var}_0$ to the constant $n \in \mathbb{N}$.
$\texttt{var}_0 \leftarrow \texttt{M[var}_1\texttt{]}$	Set the variable $\texttt{var}_0$ to the content of the memory cell $M[\texttt{var}_1]$.
$\texttt{var}_0 \leftarrow \texttt{NDET}$	Non-deterministically set the variable $\texttt{var}_0$ to any integer $n \in \mathbb{N}$.
$\texttt{M[var}_0\texttt{]} \leftarrow \texttt{var}_1$	Set the memory cell $M[\texttt{var}_0]$ to the value of the variable $\texttt{var}_1$.

Control instructions:

$\texttt{var}_0 \leftarrow \texttt{IP}$	Set the variable $\texttt{var}_0$ to the value of the instruction counter `IP`.
$\texttt{IF var}_0\texttt{=0, GOTO var}_1$	If $\texttt{var}_0$ is equal to 0, set `IP` to $\texttt{var}_1$; otherwise, set `IP` to $\texttt{IP}+1$.
`HALT`	Halt the execution in a valid state.
`CRASH`	Run the machine forever in an error state.

Operations:

$\texttt{var}_0 \leftarrow \texttt{var}_1 \texttt{ + var}_2$	Set $\texttt{var}_0$ to $\texttt{var}_1 + \texttt{var}_2$.
$\texttt{var}_0 \leftarrow \texttt{var}_1 \texttt{ - var}_2$	Set $\texttt{var}_0$ to $\texttt{var}_1 - \texttt{var}_2$ (truncated to 0 if negative).
$\texttt{var}_0 \leftarrow \texttt{var}_1 \texttt{ * var}_2$	Set $\texttt{var}_0$ to $\texttt{var}_1 \times \texttt{var}_2$.
$\texttt{var}_0 \leftarrow \texttt{var}_1 \texttt{ / var}_2$	Set $\texttt{var}_0$ to $\texttt{var}_1 / \texttt{var}_2$.
$\texttt{var}_0 \leftarrow \texttt{var}_1 \texttt{ \&\& var}_2$	Set $\texttt{var}_0$ to the bitwise AND of $\texttt{var}_1$ and $\texttt{var}_2$.
$\texttt{var}_0 \leftarrow \texttt{var}_1 \texttt{ \|\| var}_2$	Set $\texttt{var}_0$ to the bitwise OR of $\texttt{var}_1$ and $\texttt{var}_2$.
$\texttt{var}_0 \leftarrow \neg\texttt{var}_1$	Set $\texttt{var}_0$ to the bitwise NOT of $\texttt{var}_1$.
$\texttt{var}_0 \leftarrow \texttt{lshift var}_1$	Set $\texttt{var}_0$ to the left shift of $\texttt{var}_1$ (leftmost bit is removed).
$\texttt{var}_0 \leftarrow \texttt{rshift var}_1$	Set $\texttt{var}_0$ to the right shift of $\texttt{var}_1$ (padding with a 0).

The set of operations of our model is quite arbitrary, though one could argue it provides most functionalities modern assembler languages do. Most "missing instructions" (*e.g.* max and min, jumps based on more complex comparisons…) can be written as combinations of our given set of instructions.

Following a long tradition in the history of computer programming, the correct behavior of RAM programs in not enforced syntaxically. In other words, if an instruction has an undefined effect (for example, setting the instruction pointer `IP` outside of the program, dividing by zero, or reading a memory cell that has not been written yet), the machine is assumed to crash and catch fire[41].

[41] This is a bad situation. We do not want that.

4.3.2 Non-determinism, computability and time complexity

Unlike what had been defined in the previous sections, this RAM model has a few specificities that distinguish it from other computational models:

- RAM programs define *non-deterministic functions*, because the instruction `var ← NDET` allows to fill non-deterministically a given memory cell with an unknown integer;
- RAM programs define (non-deterministic) maps between arrays of binary strings $(\{0,1\}^*)^*$.

Thus, we revisit some of the previous definitions.

By encoding sequences of instructions into binary strings, we consider RAM programs as binary strings $e \in \{0,1\}^*$; thus, we still denote:

Definition 4.22. *Let $\varphi_e : \subseteq (\{0,1\}^*)^* \rightrightarrows (\{0,1\}^*)^*$ be the multifunction computed by the RAM program of code $e \in \{0,1\}^*$, i.e. the association of all inputs $I \in (\{0,1\}^*)^*$ and outputs $O \in (\{0,1\}^*)^*$ such that e admits a run on the input I that halts in a valid state and outputs O.*

We denote by $\varphi_e(I)\!\downarrow$ the set of all output arrays $O \in (\{0,1\}^*)^*$ such that there exists a run of the RAM program e on the input array I that halts in a valid state with output O. For a time $t \in \mathbb{N}$, we denote by $\varphi_e(I)\!\downarrow_{[t]}$ the set of all ouput arrays such that there exists a run of the RAM program e on the input array I that halts in a valid state with output O and has length $\leq t$.

Definition 4.23. *A multifunction $f : \subseteq (\{0,1\}^*)^* \rightrightarrows (\{0,1\}^*)^*$ is* computable *if there exists a code $e \in \{0,1\}^*$ such that $\varphi_e(I)\!\downarrow = f(I)$ for every $I \in \mathrm{Dom}(f)$, and $\varphi_e(I)\!\uparrow$ otherwise.*

In this case, we say that the code e *computes* the multifunction f.

As earlier, the computability of (multi)functions $(\{0,1\}^*)^* \rightrightarrows (\{0,1\}^*)^*$ extends from arrays of binary words $\{0,1\}^*$ to booleans $\{\top,\bot\}$, integers of $\mathbb{N}$, etc... with straightforward encodings. In fact, all previous computability notions (*e.g.* computability on sets) naturally extend to the non-deterministic setting.

We can now define the time complexity of non-deterministic functions:

Definition 4.24. *A multivalued function $f : (\{0,1\}^*)^* \rightrightarrows (\{0,1\}^*)^*$ has* time-complexity $t(s)$ *if there exists a code $e \in \{0,1\}^*$ that computes f and such that for all $I \in \mathrm{Dom}(f)$, we have $\varphi_e(I)\!\downarrow_{[t(s)]} = f(I)$, where $s = \sum_{j=1}^{+\infty} |I[j]|$ is the bit length of the array input I.*

Intuitively, the code e should compute the function f; and for all ouputs $O \in f(I)$, there should exist a run of e on input I that outputs O in less than $t(s)$ steps.

Having a well-defined computational model allows us to compute precise upper bounds on the complexity of any given problem. Since our set of instructions is actually rich enough to compute everything that is computable by a real-world programming language, and with identical running times, this thesis will *not*[42] contain any explicit word RAM program. Instead, it will only contain sketches of algorithms and a few pointers (very often, ideas of data structures) for their efficient RAM implementation.

[42] Much to everybody's joy and peace of mind.

4.3.3 log-Random Access Machines

The RAM model allows for arithmetic operations in constant time on integers of arbitrary sizes, which has unintended consequences – including

[HS74] Hartmanis and Simon, "On the power of multiplication in random access machines".

solving PSPACE-complete problems in polynomial time [HS74]. A classical restriction on its expressive power suggests to impose bounds on the word length of all variables and registers: however, since such a bound on the variables imply a bound on the memory cells that can be addressed, and thus accessed, a constant bound on the word length would result in such RAM machines being very shiny finite state automaton. Thus, we consider the log-RAM model:

Definition 4.25. A log-Random Acess Machine *is a* RAM *machine in which the word length of all variables and memory cells is bounded by* $O(\log(s))$*, for* s *the* bit size *of the input.*

As with other undefined operations, integer overflows (in case of addition, multiplication or bitwise manipulations) are not defined behaviors of log-RAM machines and programmers should be careful in *not* letting one happen.

Remark 4.26. *In Theorem 10.11, its proof and applications (i.e. for most of the second half of this thesis), results will be stated and proved on the* log-RAM *model. The remainder of this chapter applies on both* RAM *and* log-RAM *machines, which will be mentioned if applicable.*

4.3.4 Enumerations and basic transformations

Since a code is simply a binary string $e \in \{0,1\}^*$, it is actually possible for RAM programs to act on other RAM programs. In what follows, we consider a few classical results about such transformations.

Hardcoding input values The first result we mention is the possibility to computably hardcode the value of some input cells in the code of a program $e \in \{0,1\}^*$: this is the famous S_n^m theorem [Kle38].

[Kle38] Kleene, "On notation for ordinal numbers".

Proposition 4.27 (S_n^m theorem)**.** *For every* $m \in \mathbb{N}$*, there exists a computable total function* $S^m : \{0,1\}^* \times (\{0,1\}^*)^m \to \{0,1\}^*$ *such that, for every code* $e \in \{0,1\}^*$ *and every input array* $I \in (\{0,1\}^*)^*$*,*

$$\varphi_e(I) = \varphi_{S^m(e,I[0],\dots,I[m-1])}(I[m:]).$$

Furthermore, if e *had time-complexity* $t(s)$*, then* $S^m(e, I[0], \dots, I[m-1])$ *has time complexity* $O(t(s))$*.*

In other words, on RAM machines, the S_n^m theorem preserves running times up to a constant factor.

Sketch of proof. The function $S^m(e, x_1, \dots, x_m)$ performs the following transformations on the program e: it replaces every input reading instruction $\texttt{var}_0 \leftarrow \texttt{I[var}_1\texttt{]}$ by the following sequence of instructions:

- Check the value of the variable $\texttt{var}_1$.
- If it is smaller than m, then copy the constant x_m into $\texttt{var}_0$;
- If it is larger than m, then copy the input cell of index $\texttt{var}_1 - m$ into $\texttt{var}_0$;

Since we introduce several instructions for each previously existing single input reading instruction, do not forget to update all `GOTO` instructions from the original program e so that they point to the correct position in the new program $S^m(e, x_1, \dots, x_m)$. □

Remark 4.28. *If* e *is a* log-RAM *program, then* $S^m(e, x_1, \dots, x_m)$ *is also a* log-RAM *program.*

Recursion and fixpoint theorems Kleene's first recursion theorem (also from [Kle38]) states that total computable functions must have some programs whose behaviors are not modified:

Proposition 4.29. *For every computable total function $F\colon \{0,1\}^* \to \{0,1\}^*$, there exists $e \in \{0,1\}^*$ such that $\varphi_e = \varphi_{F(e)}$.*
Furthermore, if the code $F(e)$ has time-complexity $t(s)$ on valid input arrays $I \in \mathrm{Dom}(\varphi_{F(e)}) = \mathrm{Dom}(\varphi_e)$ of bit size s, then e has time complexity $O(t(s))$.

Proof. Let $b \in \{0,1\}^*$ be the following code: on a given array $I \in (\{0,1\}^*)^*$,

- Let $n = I[0]$ and $I' = I[1{:}]$, and compute $\varphi_n(n) \in (\{0,1\}^*)^*$;[43]
- If the previous step halts and $\varphi_n(n)\!\downarrow \in \{0,1\}^*$, compute $F(\varphi_n(n))$;
- Compute and return $\varphi_{F(\varphi_n(n))}(I') \in (\{0,1\}^*)^*$.

[43] If this step does not halt, the associated computation $\varphi_b(I)$ runs forever.

Applying the S_n^m theorem, there exists some computable total function $s\colon (\{0,1\}^*)^2 \to \{0,1\}^*$ that, in particular, verifies $\varphi_b(n : I) = \varphi_{s(b,n)}(I)$ for word $n \in \{0,1\}^*$ and input array $I \in (\{0,1\}^*)^*$.[44] But since s is a computable function, there exists a code $a \in \{0,1\}^*$ such that $\varphi_a(n) = s(b,n)$ for every $n \in \{0,1\}^*$; and since s is a total function, $\varphi_a\colon \{0,1\}^* \to \{0,1\}^*$ must be total too.

[44] Where $n : I$ denotes the concatenation of n with I.

Then $e = \varphi_a(a) \in \{0,1\}^*$ is well-defined, and forms a valid fixpoint. Indeed, for any $I \in (\{0,1\}^*)^*$, we have

$$\varphi_e(I) = \varphi_{s(b,a)}(I) = \varphi_b(a : I) = \varphi_{F(\varphi_a(a))}(I) = \varphi_{F(e)}(I),$$

since $\varphi_a(a)\!\downarrow \in \{0,1\}^*$ and that F is total.

What is the time complexity of e? By the S_n^m theorem, the time complexity of the code $e = s(b,a)$ has a constant-time factor overhead on the time complexity of b. Since the first two items of b (computing the codes $e = \varphi_a(a)$ and $f(e)$) take time $O(1)$, the time complexity of b essentially depends on the last step: computing $\varphi_{F(e)}(I[1{:}])$ for $I \in (\{0,1\}^*)^*$ the input of b, which can be performed with constant-time factor overhead on the original time-complexity of $F(e)$. □

Remark. *If $F(e)$ is a* log-RAM *program, then $s(b,a) = e$ is also a* log-RAM *program.*

From the first recursion theorem, we can prove Kleene's second recursion theorem (*a.k.a.* fixpoint theorem) [Kle38]: there exists programs that can access their own code[45]:

Proposition 4.30. *For every computable $f\colon \subseteq (\{0,1\}^*)^* \rightrightarrows (\{0,1\}^*)^*$, there exists a code $e \in \{0,1\}^*$ such that, for all input arrays $I \in (\{0,1\}^*)^*$:*

$$\varphi_e(I) = f(e : I).$$

Furthermore, if f has time complexity $t(s)$ on input arrays of bit size s, then e has time complexity $O(t(s))$.

[45] And, for example, print it: these are *quines*.

Proof. Let $e' \in \{0,1\}^*$ be a code for f, and let $s \in (\{0,1\}^*)^2 \to \{0,1\}^*$ be given by the S_n^m theorem: for every code $x \in \{0,1\}^*$ and input array $I \in (\{0,1\}^*)^*$, we have $\varphi_{s(e',x)}(I) = \varphi_{e'}(x : I) = f(x : I)$.

Let $F\colon \{0,1\}^* \to \{0,1\}^*$ be given by $F(x) = s(e',x)$. By the first recursion theorem, there exists some code $e \in \{0,1\}^*$ such that $\varphi_{F(e)} = \varphi_e$ and such that the program of code e only has constant overhead on the time complexity of $F(e)$ (thus, constant overhead on the time complexity of e', which computes f). Hence:

$$\varphi_e(I) = \varphi_{F(e)}(I) = \varphi_{s(e',e)}(I) = f(e : I).$$ □

Remark. *If $f(e : I)$ is implemented by a* log-RAM *program, then e is also a* log-RAM *program.*

4.3.5 Simulation by Turing machines

One may believe that non-deterministic log-RAM programs are inherently faster than non-deterministic Turing machines, since they can address memory cells in constant time without being constrained by the locality of head movements. The following proposition[46] disproves this intuition:

[46] This proposition is definitely folklore, whose first appearance seems to be [Wie83].

[Wie83] Wiedermann, "Deterministic and Nondeterministic Simulation of the RAM by the Turing Machine".

Proposition 4.31. *There exists a non-deterministic Turing machine $\mathcal{U}_{\text{RAM}}$ with finitely many tapes (the program tape, the memory tape, the input tape, the output tape, and finitely many work tapes) such that, for any* log-RAM *code $e \in \{0,1\}^*$ and any input array $I \in \{0,1\}^*$, the valid runs of the machine $\mathcal{U}_{\text{RAM}}$ when given e on its program tape and I on its input tape are in bijection with the valid runs of the* RAM *program e on the input array I.*
Furthermore, this bijection preserves the length of runs up to a polylogarithmic factor.

Sketch of proof. The machine $\mathcal{U}_{\text{RAM}}$ can easily simulate all operations on variables with polylogarithmic time overhead, since all operations of log-RAM machines operate on finitely many words of logarithmic sizes.

The only remaining part that we have to simulate are the memory accesses of the RAM machine: all operations involving the memory of the RAM will be simulated by the Turing machine using non-deterministic guesses, and these guesses will be checked afterwards. More precisely:

- When copying a value from a variable to a memory cell, the simulating Turing machine will write a record `(MEMORY,WRITE,time,address,w)` on its memory tape, where `time` is the elapsed number of computation steps, `address` is the address of the memory cell that is to be written, and w is the value of the variable that is to be copied;
- When reading a value from a memory cell to a variable, the simulating Turing machine will write a record `(MEMORY,READ,time,address,w)` on its memory tape, using the same notations.

At the end of the simulation (*i.e.* when the RAM program on the program tape orders to halt the execution), the machine $\mathcal{U}_{\text{RAM}}$ sorts the guesses that appear on its memory tape lexicographically by addresses and time, and checks the consistence of its guesses in linear time. It halts in a valid state if and only its guesses were correct and if RAM program did; otherwise, it crashes and runs forever.

Sorting the memory tape at the end of the computation adds a logarithmic factor to the time complexity, because sorting can be performed in quasi-linear time on a multitape Turing machine; and checking the consistency of the sorted tape can be performed in linear time. □

Remark. *As currently written, the logarithmic time overhead depends on the program p, because the machine $\mathcal{U}_{\text{RAM}}$ spends a lot of time reading its program tape by going back and forth. However, this dependency can be removed with same method: we can make $\mathcal{U}_{\text{RAM}}$ guess the current instruction, record the instruction counter and these guesses on a special tape, and at the end of the computations sort and check whether theses guesses were correct.*

Though we will not use Proposition 4.31 in this thesis, the idea of non-deterministically guessing memory values, recording these guesses and and sorting them later to check their consistency will be our main insight for proving Theorem 11.8.

5 Tools

This section introduces the family of Toeplitz subshifts, which often appear in proofs that encode real numbers as densities, or embed sequences of symbols in a shift-invariant object.

5.1 Toeplitz subshifts

5.1.1 The ruler sequence

Consider the word morphism $\tau\colon \mathbb{N}^* \to \mathbb{N}^*$ defined by $\tau(n) = 0 \cdot (n+1)$ for every $n \in \mathbb{N}$. Iterations from the word 0:

$$0 \xrightarrow{\tau} 01 \xrightarrow{\tau} 0102 \xrightarrow{\tau} 01020103 \xrightarrow{\tau} 0102010301020104\dots$$

define a unique infinite sequence $\mathcal{T} = \tau^\infty(0) \in \mathbb{N}^{\mathbb{N}}$ that is often called the *ruler sequence*[47]. We denote by $\mathcal{T}_n \in \mathbb{N}$ the n^{th} element of the sequence $\mathcal{T}$. On the infinite alphabet $\bar{\mathbb{N}} = \mathbb{N} \cup \{\infty\}$ (the compactification of $\mathbb{N}$), we define the closed and shift-invariant set

$$X_{\mathcal{T}} = \overline{\{x \in \mathbb{N}^{\mathbb{Z}} \colon \forall w \sqsubseteq x, w \sqsubseteq \mathcal{T}\}} \subseteq \bar{\mathbb{N}}^{\mathbb{Z}}.$$

The closure in the formula above adds to $X_{\mathcal{T}}$ configurations that may contain a single position with a ∞ symbol. For a given configuration $x \in X_{\mathcal{T}}$, and a position $i \in \mathbb{Z}$, we call *level* of i in x the value x_i. In particular, it follows from the definition that, for any configuration $x \in X_{\mathcal{T}}$, positions of level $\ell \in \mathbb{N}$ are $2^{\ell+1}$-periodic in x.

[47] See OEIS A007814.

Claim 5.1. *Let $t \in \bar{\mathbb{N}}^{[\![n]\!]}$ such that $t \sqsubseteq \mathcal{T}$ (i.e. $t \in \mathcal{L}_{[\![n]\!]}(X_{\mathcal{T}})$). Then t contains at least one position of every level $\ell < \lfloor \log(n-1) \rfloor$, and at most two positions of level $\ell \geq \lfloor \log(n-1) \rfloor$.*

Proof. Considering the sequence $\mathcal{T}$, notice that consecutive positions of level ℓ appear exactly at distance $2^{\ell+1}$ from one another, and that consecutive positions of level $\geq \ell$ appear exactly at distance 2^ℓ from one another.

Fix $t \in \mathcal{L}_{[\![n]\!]}(X_{\mathcal{T}})$.

- For any $\ell \in \mathbb{N}$ such that $2^{\ell+1} + 1 \leq |[\![n]\!]| = n$, at least one position of level ℓ must appear in w. Such levels verify in particular
$$\ell \leq \log(n-1) - 1;$$
- Reciprocally, consider $\ell \in \mathbb{N}$ such that t contains three positions of level $\geq \ell$: then $2 \cdot 2^\ell + 1 \leq |[\![n]\!]| = n$ and
$$\ell \leq \log \frac{n-1}{2} \leq \log(n-1) - 1. \qquad \square$$

5.1.2 Toeplitzification

For any alphabet $\mathcal{A}$ and sequence $u = (u_n)_{n\in\mathbb{N}} \in \mathcal{A}^{\mathbb{N}}$, define its *Toeplitzification* $\mathcal{T}(u) \in \mathcal{A}^{\mathbb{N}}$ as:

$$\mathcal{T}(u)\colon n \in \mathbb{N} \mapsto u_{\mathcal{T}_n} \in \mathcal{A}.$$

For example, the sequence defined by $u_n = 0$ if n is even, and $u_n = 1$ if n is odd, transforms into the *period-doubling sequence*[48] $\mathcal{T}(u) = 01000101010\ldots$.

[48] See OEIS A096268.

Similarly, given a sequence $u \in \mathcal{A}^{\mathbb{N}}$, the subshift $X_{\mathcal{T}(u)}$ is defined as:

$$X_{\mathcal{T}(u)} = \{x \in \mathcal{A}^{\mathbb{Z}} : \exists t \in X_{\mathcal{T}}, \forall i \in \mathbb{Z},\, t_i \in \mathbb{N} \implies x_i = u_{t_i}\}.$$

In other words, configurations of $X_{\mathcal{T}(u)}$ are made of arbitrarily long sequences of $\mathcal{T}(u)$, with the possible exception of at most one position of "infinite level" that can contain any symbol of $\mathcal{A}$.

Claim 5.2. *Let $w \in \mathcal{L}_{[\![n]\!]}(X_{\mathcal{T}(u)})$ for $u = (01)^\infty$ be a factor of the period-doubling sequence. For any two Toeplitz structures $t, t' \in X_{\mathcal{T}}$ such that $w_i = t_i \bmod 2$ and $w_i = t'_i \bmod 2$ for $i \in [\![n]\!]$ with $t_i, t'_i \neq \infty$, we have:*

$$\forall i \in \mathbb{Z},\, t_i \neq \infty \text{ and } t_i \leq (\log(n-1) - 2) \implies t_i = t'_i.$$

Proof. Let us denote $T(w)$ the set of Toeplitz structures compatible with w, *i.e.* $T(w) = \{t \in X_{\mathcal{T}} : \forall i \in [\![n]\!], t_i \neq \infty \implies w_i = t_i \bmod 2\}$. We prove by induction on ℓ that, for every $\ell \leq (\log n - 2)$, there exists $i_\ell \in \mathbb{N}$ such that for any $t \in T(w)$, $t_i \geq \ell$ if and only if $i \in i_\ell + 2^\ell\mathbb{Z}$:

- Base case: $\ell = 0$. The integer $i_0 = 0$ verifies the property above.
- Induction: $\ell \to \ell + 1$. Assume that there exists some i_ℓ such that positions of level $\geq \ell$ in any $t \in T(w)$ occur at positions $(i_\ell + 2^\ell\mathbb{Z})$.

  For any $t \in X_{\mathcal{T}}$ such that positions of level $\geq \ell$ appear at positions $(i_\ell + 2^\ell\mathbb{Z})$, two cases occur: either the positions of level ℓ exactly span $(i_\ell + 2^{\ell+1}\mathbb{Z})$ (and the positions of level $\geq \ell + 1$ are $(i_\ell + 2^\ell + 2^{\ell+1}\mathbb{Z})$), or the positions of level ℓ exactly span $(i_\ell + 2^\ell + 2^{\ell+1}\mathbb{Z})$ (in which case, the positions of level $\geq \ell + 1$ are $(i_\ell + 2^{\ell+1}\mathbb{Z})$).

  Furthermore, if $t \in T(w)$, the first cases implies that $w|_{(i_\ell + 2^{\ell+1}\mathbb{Z})}$ is constant and equals $(\ell \bmod 0)$, which the second implies that $w|_{(i_\ell + 2^\ell + 2^{\ell+1}\mathbb{Z})}$ is constant and equals $(\ell \bmod 0)$. Assuming that $\ell + 1 \leq \log(n-1) - 2$, only one of the aforementioned cases can occur among the Toeplitz structures of $T(w)$. Indeed, a position of level $\ell + 1$ must occur in $t|_{[\![n]\!]}$ for any $t \in \mathcal{T}$ by Claim 5.1: thus, there must exists some index $i \in [\![n]\!] \cap (i_\ell + 2^\ell\mathbb{Z})$ such that $w_i = (\ell + 1 \bmod 2) \neq (\ell \bmod 2)$. □

[49] In base 2: $\alpha = (.101000\ldots)_2$.

[50] Recall that, for $a \in \mathcal{A}$, $|w|_a$ is the number of occurences of a in $w \in \mathcal{A}^*$.

In particular, when considering a real number $\alpha \in [0, 1)$, one can associate to α the Toeplitzification $\mathcal{T}(\alpha) = \mathcal{T}((\alpha_n)_{n\in\mathbb{N}}) \in \{0, 1\}^{\mathbb{N}}$, where α_n is the n^{th} digit of the proper binary expansion of α. For example, for $\alpha = 5/8$, we obtain[49]: $\mathcal{T}(\alpha) = 10111010111010\ldots$. Considering the density[50] of symbols 1 in patterns of $\mathcal{T}(\alpha)$, one proves:

Claim 5.3. *For any $\alpha \in [0, 1)$, consider the associated Toeplitz sequence $\mathcal{T}(\alpha) \in \{0, 1\}^{\mathbb{N}}$. Then for any $w \sqsubseteq \mathcal{T}(\alpha)$, we have:*

$$|w|_1 = \alpha \cdot |w| + O(1).$$

Proof. Since positions of level ℓ are 2^ℓ-periodic in $\mathcal{T}(\alpha)$, we deduce that any subpattern $w \sqsubseteq \mathcal{T}(\alpha)$ of length $n \in \mathbb{N}$ must cover at least $\lfloor n/2 \rfloor$ positions of level 0, $\lfloor n/4 \rfloor$ positions of level 1...; and at most $\lceil n/2 \rceil$ positions of level 2, $\lceil n/4 \rceil$ positions of level 3.... In particular, denoting $(\alpha_n)_{n\in\mathbb{N}}$ the proper binary

expansion of α, we obtain:

$$\begin{aligned}
|w|_1 &\geq \sum_{i=0}^{\lfloor \log(n-1)-1 \rfloor} \left\lfloor \frac{n}{2^{i+1}} \right\rfloor \cdot \alpha_i \geq \sum_{i=0}^{\lfloor \log(n-1)-1 \rfloor} \left(\frac{n}{2^{i+1}} - 1 \right) \cdot \alpha_i \\
&\geq \sum_{i=1}^{\lfloor \log(n-1)-1 \rfloor} \frac{n}{2^i} \cdot \alpha_i + O(1) \\
&\geq n \cdot \left(\alpha - \sum_{i=\lfloor \log(n-1) \rfloor}^{+\infty} \frac{1}{2^i} \right) + O(1) \\
&\geq n \cdot \left(\alpha - O(1) \cdot \frac{1}{2^{\log n}} \right) + O(1) \\
&\geq n \cdot \alpha + O(1).
\end{aligned}$$

And similarly, using Claim 5.1:

$$|w|_1 \leq \sum_{i=1}^{\lfloor \log(n-1)-1 \rfloor} \left\lceil \frac{n}{2^i} \right\rceil \cdot \alpha_i + 2 \leq n \cdot \alpha + O(1). \qquad \square$$

5.1.3 Toeplitzification

Instead of considering sequences of $u \in \mathcal{A}^{\mathbb{N}}$, we also generalize to sets of sequences $U \subseteq \mathcal{A}^{\mathbb{N}}$ and define:

$$\mathcal{T}(U) = \{\mathcal{T}(u) \colon u \in U\}.$$

Proposition 5.4. *If $U \subseteq \mathcal{A}^{\mathbb{N}}$ is an effectively closed subset of $\mathcal{A}^{\mathbb{N}}$, then $\mathcal{T}(U)$ is also an effectively closed subset of $\mathcal{A}^{\mathbb{N}}$.*

Proof. Indeed, let $x \in \mathcal{A}^{\mathbb{N}}$. By definition, there exists a unique $s \in \mathcal{A}^{\mathbb{N}}$ such that $x_i = s_{\mathcal{T}_i}$ for every $i \in \mathbb{N}$: in particular, when given $x \in \mathcal{A}^{\mathbb{N}}$, the sequence $s \in \mathcal{A}^{\mathbb{N}}$ is computable, and x belongs in $\mathcal{T}(U)$ if and only if s belongs in U. Enumerating the prefixes $s|_{[\![n]\!]}$ for $n \in \mathbb{N}$, check in parallel the following condition: if there exists one such prefix $s|_{[\![n]\!]}$ such that $[s|_{[\![n]\!]}]_0 \cap U = \emptyset$, then reject x.

This procedure terminates in finite time and rejects $x \in \mathcal{A}^{\mathbb{N}}$ whenever $x \notin \mathcal{T}(U)$. This concludes the proof. $\square$

The subshift $X_{\mathcal{T}(U)}$ is similarly defined:

$$X_{\mathcal{T}(U)} = \overline{\bigcup_{u \in U} X_{\mathcal{T}(u)}}.$$

Claim 5.5. *For any alphabet $\mathcal{A}$, $|\mathcal{L}_{[\![n]\!]}(X_{\mathcal{T}(\mathcal{A}^{\mathbb{N}})})| \leq (2|\mathcal{A}|)^{\log n + O(1)}$.*

Proof. A pattern $w \in \mathcal{L}_{[\![n]\!]}(X_{\mathcal{T}(\mathcal{A}^{\mathbb{N}})})$ is entirely determined by its level structure and the symbol appearing at each level. Since at most $\log n + 2$ distinct Toeplitz levels can appear in such a pattern w, and that the position of each level is determined by its parity among positions of level $\ell - 1$,[51], the upper bound follows. $\square$

[51] Positions of level $\ell = 0$ cover one symbol out of two in w, and positions of level $\ell > 0$ cover the remaining positions. Positions of level $\ell = 1$ cover half of these remaining positions, etc...

Proposition 5.6. *If $U \subseteq \mathcal{A}^{\mathbb{N}}$ is an effectively closed subset of $\mathcal{A}^{\mathbb{N}}$, then $X_{\mathcal{T}(U)}$ is an effective subshift.*

Proof. Let us begin by defining another subshift: for $u \in \mathcal{A}^{\mathbb{N}}$, define

$$X'_{\mathcal{T}(u)} = \Big\{ x \in (\mathcal{A} \times \{0,1\})^{\mathbb{Z}} \colon \exists t \in X_{\mathcal{T}}, \\ \forall i \in \mathbb{Z},\ t_i \in \mathbb{N} \implies x_i = (u_{t_i}, t_i \bmod 2) \Big\};$$

and for $U \subseteq \mathcal{A}^{\mathbb{N}}$:

$$X'_{\mathcal{T}(U)} = \overline{\bigcup_{u \in U} X'_{\mathcal{T}(u)}}.$$

Since for any $U \subseteq \mathcal{A}^{\mathbb{N}}$, the subshift $X_{\mathcal{T}(U)}$ is a factor of $X'_{\mathcal{T}(U)}$ by the natural projection $\pi : \mathcal{A} \times \{0,1\} \to \mathcal{A}$, by Proposition 3.41 we can prove Proposition 5.6 on $X'_{\mathcal{T}(U)}$ instead.

This alternative version of Toeplitzification verifies the following property[52] by Claim 5.2: for any configuration $x \in X'_{\mathcal{T}(\mathcal{A}^{\mathbb{N}})}$, let $u \in \mathcal{A}^{\mathbb{N}}$ be a sequence and $w \in \mathcal{L}_{[\![n]\!]}(X'_{\mathcal{T}(u)})$ be a pattern such that $w \sqsubseteq x$; then for any sequence $v \in \mathcal{A}^{\mathbb{N}}$ such that $x \in X'_{\mathcal{T}(v)}$, we have $u|_{[\![\lfloor \log n - 2 \rfloor]\!]} = v|_{[\![\lfloor \log n - 2 \rfloor]\!]}$.

[52] Notice that it does not hold for $X_{\mathcal{T}(\mathcal{A}^{\mathbb{N}})}$. Indeed, when considering the configuration $x_i = i \bmod 2$, x can be obtained from either $u = 0111\ldots$ or $u = 1000\ldots$.

Let us now proceed with the proof. Fix an effectively closed set $U \subseteq \mathcal{A}^{\mathbb{N}}$. Let us first notice that:

$$X'_{\mathcal{T}(U)} = \bigcup_{u \in U} X_{\mathcal{T}(u)}.$$

The inclusion $\supseteq$ holds by definition of $X'_{\mathcal{T}(U)}$. For the converse inclusion $\subseteq$, consider $x \in X'_{\mathcal{T}(U)}$ and a sequence $u \in \mathcal{A}^{\mathbb{N}}$ such that $x \in X'_{\mathcal{T}(u)}$. By definition of $X'_{\mathcal{T}(U)}$, there exists sequences $u^{(n)} \in U \subseteq \mathcal{A}^{\mathbb{N}}$ and configurations $x^{(n)} \in \mathcal{A}^{\mathbb{Z}}$ such that $x^{(n)} \in X'_{\mathcal{T}(u^{(n)})}$ and $x = \lim_{n \to +\infty} x^{(n)}$. Because the $x^{(n)}$'s converge towards x, by Claim 5.2 we must have $u = \lim_{n \to +\infty} u^{(n)}$; and since U is closed, we conclude that $u \in U$.

To conclude, we prove that $X'_{\mathcal{T}(U)}$ is effective. Since U is effectively closed, the set of words $\{w \in \mathcal{A}^* : [w]_0 \cap U = \emptyset\}$ is computably enumerable. From it, define the following a set of forbidden patterns $\mathcal{F}$: denoting by $\mathcal{T}(v)' \in (\mathcal{A} \times \{0,1\})^{\mathbb{N}}$ the word defined as $\mathcal{T}(v)'_i = (v_{\mathcal{T}_i}, \mathcal{T}_i \bmod 2)$ for $v \in \mathcal{A}^{\mathbb{N}}$, define

$$\mathcal{F} = \bigcup_{n \in \mathbb{N}} \{\mathcal{T}(v)'|_{[\![2^{n+2}+1]\!]} : v \in \mathcal{A}^{\mathbb{N}}, [v|_{[\![n]\!]}]_0 \cap U = \emptyset\}.$$

We claim that $\mathcal{F}$ realizes $X_{\mathcal{T}(U)}$:

$\Longrightarrow$ Fix $x \in X'_{\mathcal{T}(U)}$: there exists $t \in X_{\mathcal{T}}$ and a sequence $u \in U$ such that $x_i = (u_{t_i}, t_i \bmod 2)$ for every position $i \in \mathbb{Z}$ such that $t_i \neq \infty$. For any forbidden pattern $f \in \mathcal{F}$, let $n \in \mathbb{N}$ be such that $f \in \mathcal{A}^{[\![2^{n+2}+1]\!]}$, and let us prove that f cannot appear in x. By definition of f, there exists a sequence $v \in \mathcal{A}^{\mathbb{N}}$ such that $f = \mathcal{T}(v)'|_{[\![2^{n+2}+1]\!]}$ and $[v|_{[\![n]\!]}]_0 \cap U = \emptyset$. By contradiction: if we had $f \sqsubseteq x$, then for any v' such that $x \in X'_{\mathcal{T}(v')}$ we would have $v'|_{[\![n]\!]} = v|_{[\![n]\!]}$ (Claim 5.2). In particular, we would have $u \notin U$: contradiction.

$\Longleftarrow$ Let $x \in X_{\mathcal{T}(\mathcal{A}^{\mathbb{N}})} \setminus X'_{\mathcal{T}(U)}$: there exists a sequence $u \in \mathcal{A}^{\mathbb{N}} \setminus U$ such that $x \in X_{\mathcal{T}(u)}$. Since $\mathcal{A}^{\mathbb{N}} \setminus U$ is open, there exists some $n \in \mathbb{N}$ such that $[u|_{[\![n]\!]}] \cap U = \emptyset$.

As every factor of $\mathcal{T}(u)'$ occurs infinitely often in x by definition of $X_{\mathcal{T}(u)}$, the pattern $\mathcal{T}(u)'|_{[\![2^{n+2}+1]\!]} \in \mathcal{F}$ occurs in x: thus, $x \notin X_{\mathcal{F}}$. □

Let us consider a particular case: for any real number $\alpha \in [0,1)$, denote by $U_\alpha = \{a \in \{0,1\}^{\mathbb{N}} : : \sum_{i=0}^{+\infty} a_i 2^{-(i+1)} \leq \alpha\}$ the set of binary expansions of real numbers $a \leq \alpha$. If $\alpha \in \Pi^0_1$, then U_α is an effectively closed subset of $\{0,1\}^{\mathbb{N}}$, and the subshift $X_{\mathcal{T}(U_\alpha)}$ (or $X_{\mathcal{T}(\leq \alpha)}$ for short) is effective.

Definition 5.7. *For any $\alpha \in [0,1] \cap \Pi^0_1$, the* density subshift $X_{\mathcal{T}(\leq \alpha)} \subseteq \{0,1\}^{\mathbb{Z}}$ *is the effective subshift defined as:*

$$X_{\mathcal{T}(\leq \alpha)} = \Big\{ x \in \{0,1\}^{\mathbb{Z}} : \exists t \in X_{\mathcal{T}}, \exists a \in \{0,1\}^{\mathbb{N}}, \sum_{i=0}^{+\infty} a_i 2^{-(i+1)} \leq \alpha \text{ and } \forall i \in \mathbb{Z},\ t_i \in \mathbb{N} \implies x_i = a_{t_i} \Big\}.$$

Context: soficity of subshifts

Introduction

Motivations As stated in the introduction, the main goal of this thesis is to study the following problem:

When is a $\mathbb{Z}^d$ subshift sofic?

To recognize where the difficulty lies when trying to determine the class of a subshift, let us take a detour through the theory of formal languages.

Classes of languages of finite words (regular, context-free, computable) are often defined computationally, *i.e.* by the various types of automata that recognize them (respectively finite automata, pushdown automata, Turing machines...). However, these definitions suffer from an unfortunate drawback: given a language as a set of words[53], it is very difficult to properly determine the classes this language might belong to. Indeed, this requires to either create an automaton that recognizes said language, which is often done on a case-by-case basis; or prove that none exists, which is often done by deriving a contradiction.

[53] *i.e.* not by an automaton, but as an explicit set $L \subseteq \mathcal{A}^*$.

Subshifts suffer from the same issue: to decide the soficity of a given subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$, one needs to either find a local subshift that projects onto X; or prove that none exists. The former may be very natural in the case of very "geometrical" subshifts, *e.g.* drawings of multidimensional grids, drawings of independant rectangles/squares, etc...; but not so much for "computational" subshifts, *e.g.* periodic lifts of arbitrary effective subshifts (Proposition 3.44). The latter is often very difficult, since most arguments of non-soficity rely on subshifts that have both *high complexity* (*e.g.* positive entropy) and many symmetries (*e.g.* embed a mirror).

Summary In Chapters 6 and 7, we dress a (probably incomplete) state of what is known about the class of sofic subshifts:

- Chapter 6 considers the case of $\mathbb{Z}$ subshifts, which is entirely solved: there exists a characterization of $\mathbb{Z}$ sofic subshifts inspired by tools from the formal language theory of finite words, namely the Myhill-Nerode theorem [RS59].
- Chapter 7 considers the case of $\mathbb{Z}^d$ sofic subshfits for $d \geq 2$. This class of multidimensional $\mathbb{Z}^d$ sofic subshifts turns out to be far more expressive than its one-dimensional conterpart. We also look at the currently known arguments of non-soficity.

[RS59] Rabin and Scott, "Finite automata and their decision problems".

6 Soficity of $\mathbb{Z}$ subshifts

This section considers the soficity of $\mathbb{Z}$ subshifts. Using the notion of extender sets, imported from the context of formal languages, we state the classical application of extender set in the context of $\mathbb{Z}$ subshifts: the number of extender sets determines their soficity (Proposition 6.5).

6.1 Syntactic monoid in formal languages

As mentioned in the introduction, sofic subshifts suffer the same drawbacks as languages of finite words: deciding whether a given language $L \subseteq \mathcal{A}^*$ is regular (resp. ...) requires to guess a suitable automata that recognizes it; or prove that non exists, which is always a difficult undertaking[54].

However, in the case of formal languages of finite words, the introduction of algebraic tools has been especially effective in solving this issue. The *syntactic monoid* of a language $L \subseteq \mathcal{A}^*$, often believed to have been introduced by [RS59] (the authors credit unpublished work by Myhill), is a monoid that is canonically induced by L. It relies on the notion of syntactic congruence, which intuitively defines any two words u and v in $\mathcal{A}^*$ to be equivalent if every occurence of u (as a factor) in the words of L can be replaced by v:

Definition 6.1 (Syntactic congruence). *Let $L \subseteq \mathcal{A}^*$ be a language. The* syntactic congruence *induced by L is the equivalence relation $\simeq_L$ (or $\simeq$ for short) over $\mathcal{A}^*$ defined by:*

$$u \simeq_L v \quad \iff \quad (\forall x, y \in \mathcal{A}^*,\ xuy \in L \iff xvy \in L).$$

The induced equivalences classes form the *syntatic monoid*:

Definition 6.2 (Syntactic monoid). *For a language $L \subseteq \mathcal{A}^*$, the* syntactic monoid *of L is the monoid $\mathcal{A}^*/\simeq_L$.*

The syntactic monoid provides a canonical algebraic object associated to a given language, which in particular does not depend on a given presentation (*i.e.* automaton) of said language. Furthermore, in the context of regular languages, Myhill-Nerode theorem answers our question: it states a necessary and sufficient condition for a language to be regular.

Proposition 6.3 (Myhill-Nerode theorem [RS59]). *A language $L \subseteq \mathcal{A}^*$ is regular if and only if its syntactic monoid is finite.*

In this section, we consider the generalization of syntactic congruence from finite words to bi-infinite words: namely, we consider the notion of *extender sets*. Similarly to its variations (*predecessor* and *follower sets*), it entirely characterizes soficity among $\mathbb{Z}$ subshifts.

In other words, the soficity of $\mathbb{Z}$ subshifts is entirely resolved: to determine whether a $\mathbb{Z}$ subshift $X \subseteq \mathcal{A}^{\mathbb{Z}}$ is sofic, one only needs to count its extender sets (see Proposition 6.5). This should not come as a surprise: similarly to regular languages, which are morphic images of the so-called *local languages*[55], $\mathbb{Z}$ sofic subshifts are morphic images of local subshifts.

[54] Restraining ourselves to proofs based on automata, the pumping lemma can sometimes be used to prove that a language is not regular; but it famously fails to apply to some non-regular languages.

[RS59] Rabin and Scott, "Finite automata and their decision problems".

[55] A language $L \subseteq \mathcal{A}^*$ is *local* if there exists sets of prefixes $P \subseteq \mathcal{A}$, suffixes $S \subseteq \mathcal{A}$ and forbidden 2-factors $F \subseteq \mathcal{A}^2$ such that $L = (P\mathcal{A}^* \cap \mathcal{A}^*S) \setminus (\mathcal{A}^*F\mathcal{A}^*)$.

6.2 Definitions

[KM13] Kass and Madden, "A sufficient condition for non-soficness of higher-dimensional subshifts".

[Fre16b] French, "Follower and extender sets in symbolic dynamics".

The notion of extender sets has been introduced in [KM13], and studied extensively in the context of $\mathbb{Z}$ subshifts in [Fre16b]. Yet, a very similar notion of *follower sets*[56] was already used to characterize soficity among $\mathbb{Z}$ subshifts in [LM95, Chapter 3].

[56] Follower sets are a generalization of the so-called *residuals* from languages of finite words. They are well-defined on $\mathbb{Z}$ subshifts, not-so-much on $\mathbb{Z}^d$ subshifts.

[LM95] Lind and Marcus, *An introduction to symbolic dynamics and coding*.

Given a subshift $X \subseteq \mathcal{A}^{\mathbb{Z}}$, the extender set of a pattern $w \in \mathcal{A}^*$ is the set of valid completions of w into full configurations in X:

Definition 6.4. *Given a $\mathbb{Z}$ subshift $X \subseteq \mathcal{A}^{\mathbb{Z}}$ and a pattern $w \in \mathcal{A}^*$, the* extender set *of w in X is defined by:*

$$E_X(w) = \{(x, y) \in \mathcal{A}^{-\mathbb{N}} \times \mathcal{A}^{\mathbb{N}} : xwy \in X\}.$$

Let us consider the extender sets of a few example subshifts:

- Extender sets of the *full shift*: let $X = \mathcal{A}^{\mathbb{Z}}$ be the full shift on some finite alphabet $\mathcal{A}$. Since every configuration is allowed in the full shift, there is only one extender set: indeed, for all patterns $w \in \mathcal{A}^*$, we have:
$$E_X(w) = \mathcal{A}^{-\mathbb{N}} \times \mathcal{A}^{\mathbb{N}}.$$
- Extender sets of the *sunny-side-up*: let $\mathcal{A} = \{\square, \blacksquare\}$, and denote by $X = \{x \in \mathcal{A}^{\mathbb{Z}} : \forall i, j \in \mathbb{Z},\ x_i = x_j = \blacksquare \implies i = j\}$ the sunny-side-up subshift[57]. There are two extender sets in X: if there exists $i \in \mathrm{dom}(w)$ such that $w_i = \blacksquare$, then
$$E_X(w) = \{(\square^{-\mathbb{N}}, \square^{\mathbb{N}})\};$$
and if $w = \square^{\mathrm{dom}(w)}$ otherwise, we have:
$$E_X(w) = \{(\square^{-\mathbb{N}}, \square^{\mathbb{N}})\} \cup \{\square\}^{-\mathbb{N}} \times (\square^* \blacksquare \square^{\mathbb{N}})\} \cup (\square^{-\mathbb{N}} \blacksquare \square^*) \times \{\square\}^{\mathbb{N}}.$$
- Extender sets of the mirrored counter: let $\mathcal{A} = \{a, b, @, c, d\}$ and X the closure of the configurations $\{a^{-\mathbb{N}} b a^n @ c^n d c^{\mathbb{N}} : n \in \mathbb{N}\}$. There are infinitely distinct extender sets for patterns of X: indeed, denoting $w_k = ba^k@ \in \mathcal{A}^{k+2}$, we have
$$E_X(w_k) = \{(a^{-\mathbb{N}}, c^k d c^{\mathbb{N}})\}.$$
In particular, $E_X(w_k) = E_X(w_{k'})$ if and only if $k = k'$.
- Extender sets of the non-mirrored counter: let $\mathcal{A} = \{a, b, @, c, d\}$ and X the closure of the configurations $\{a^{-\mathbb{N}} b a^n @ c^{n'} d c^{\mathbb{N}} : n \neq n'\}$. There are infinitely many distinct extender sets for patterns of X: indeed, denoting $w_k = ba^k@ \in \mathcal{A}^{k+2}$, we have
$$E_X(w_k) = \{(a^{-\mathbb{N}}, c^{k'} d c^{\mathbb{N}}) : k' \neq k\}.$$
In particular, $E_X(w_k) = E_X(w_{k'})$ if and only if $k = k'$.
- Extender sets and periodic configurations: let $\mathcal{A} = \{\square, \blacksquare, *\}$ and $X = \{x \in \mathcal{A}^{\mathbb{Z}} : \forall i, j \in \mathbb{Z},\ x_i = * \text{ and } x_j = * \implies x \text{ is } (i-j)\text{-periodic}\}$.[58] For every $n \in \mathbb{N}$, $\{E_X(w) : w \in \mathcal{A}^n\}$ contains at least 2^n extender sets. Indeed, for any two distinct words $w, w' \in \{\square, \blacksquare\}^n$, the periodic configuration $x = {}^{\infty}(w*)^{\infty}$ is valid in X and verifies $x|_{[\![n]\!]} = w$; but $w' \ltimes_{[\![n]\!]} x$ is not valid in X because it contains several (in fact: infinitely many) symbols $*$ but is aperiodic. Thus, $E_X(w) \neq E_X(w')$.

[57] It consists of configurations containing at most a single yellow square $\blacksquare$ over a $\square$ background.

[58] If a configuration of X contains two symbols $*$ at distance, say, $k \in \mathbb{N}$, then it is actually k-periodic.

Even though the definition is not strictly needed in this manuscript, one could proceed as in languages of finite words and define the *syntactic congruence* induced by a $\mathbb{Z}$ subshift $X \subseteq \mathcal{A}^{\mathbb{Z}}$ as $u \simeq_X v$ if $E_X(u) = E_X(v)$. The syntactic monoid $\mathcal{A}^*/\simeq_X$ is then directly in bijection with the extender sets $\{E_X(w) : w \in \mathcal{A}^*\}$ of X.

6.3 Extender sets and soficity of $\mathbb{Z}$ subshifts

6.3.1 Characterization of soficity in terms of extender sets

As the syntactic monoid can be used to prove that a language of finite words is regular, one can use extender sets to characterize soficity among $\mathbb{Z}$ subshifts. In fact, we understand the following characterization of sofic subshifts as a rephrasing of Myhill-Nerode theorem:

Proposition 6.5. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}}$ be a $\mathbb{Z}$ subshift. The following are equivalent:*

(i) *X is a sofic subshift.*
(ii) *$\{E_X(w) : w \in \mathcal{A}^*\}$ is finite.*
(iii) *There exists M such that $|\{E_X(w) : w \in \mathcal{A}^n\}| \leq M$ for any $n \in \mathbb{N}$.*
(iv) *$\{F_X(l) : l \in \mathcal{A}^{-\mathbb{N}}\}$ is finite.*[59]
(v) *$\{F_X(w) : w \in \mathcal{A}^*\}$ is finite.*
(vi) *There exists M such that $|\{F_X(w) : w \in \mathcal{A}^n\}| \leq M$ for any $n \in \mathbb{N}$.*

[59] For $w \in \mathcal{A}^*$, the *follower set* of w is $F_X(w) = \{y \in \mathcal{A}^{\mathbb{N}} : \exists x \in \mathcal{A}^{-\mathbb{N}},\ xwy \in X\}$. This naturally extends to infinite words $l \in \mathcal{A}^{-\mathbb{N}}$ with $F_X(l) = \{r \in \mathcal{A}^{\mathbb{N}} : lr \in X\}$.

Proof. We mostly follow the proof of [OP16, Lemma 3.4].

$(i) \implies (ii)$ Assume that $X \subseteq \mathcal{A}^{\mathbb{Z}}$ is sofic: by definition, there exists a local subshift $X' \subseteq \mathcal{B}^{\mathbb{Z}}$ and a projection $\pi : \mathcal{B} \to \mathcal{A}$ such that $\pi(X') = X$. Since X' is local, the extender set of any pattern $w' \in \mathcal{B}^n$ in X' only depends on its first and last symbols w'_0 and w'_{n-1}. Thus, the extender set of a pattern $w \in \mathcal{A}^n$ is determined by the first and last symbols of its preimages: in particular, there are less than $2^{|\mathcal{B}^2|}$ extender sets in X.

$(ii) \implies (iii)$ Tautologically, $|\{E_X(w) : w \in \mathcal{A}^*\}| \geq |\{E_X(w) : w \in \mathcal{A}^n\}|$.

$(iii) \implies (iv)$ By contraposition, assume that $\{F_X(l) : l \in \mathcal{A}^{-\mathbb{N}}\}$ is infinite. Fix $k \in \mathbb{N}$ and some semi-infinite words $l_0, \dots, l_k \in \mathcal{A}^{-\mathbb{N}}$ such that $F_X(l_i) \neq F_X(l_j)$ for $i \neq j$: by definition, for every pair (i, j) such that $i \neq j$, there exists $r_{i,j} \in \mathcal{A}^{\mathbb{N}}$ such that $l_i r_{i,j} \in X$ and $l_j r_{i,j} \notin X$ or vice-versa.

Assume the former: then for all $n \in \mathbb{N}$, we have $r_{i,j} \in F_X(l_i|_{[-n..0]})$; but by compactness, there exists some $N_{i,j}$ such that for any $n \geq N_{i,j}$, we have $r_{i,j} \notin F_X(l_j|_{[-n..0]})$. If the latter held, then $r_{i,j} \in F_X(l_j|_{[-n..0]})$; but there exists $N_{i,j}$ such that for $n \geq N_{i,j}$, we have $r_{i,j} \notin F_X(l_i|_{[-n..0]})$.

By taking $n = \max N_{i,j}$, we obtain that the words $l_i|_{[-n..0]}$ have distinct extender sets, so that $|\{E_X(w) : w \in \mathcal{A}^{n+1}\}| \geq k$: since k is arbitrary, the sets $\{E_X(w) : w \in \mathcal{A}^n\}$ do not have bounded size as n increases.

$(iv) \implies (i)$ Assume that $F = \{F_X(l) : l \in \mathcal{A}^{-\mathbb{N}} \text{ and } F_X(w) \neq \emptyset\}$ is finite. Then consider the SFT X' defined on the alphabet $F \times \mathcal{A}$ whose forbidden patterns are: $\{(F_X(l), a)(F_X(l'), b) \in (F \times \mathcal{A})^2 : F_X(l') \neq F_X(lb)\}$. For $\pi : F \times \mathcal{A} \to \mathcal{A}$ the natural projection, we claim that $\pi(X') = X$:

$X \subseteq \pi(X')$: For $x \in X$, consider the configuration $x' \in (F \times \mathcal{A})^{\mathbb{Z}}$ defined by $x_i = (F_X(x_{(-\infty..i]}), x_i)$: it is admissible in X' by definition of X', and verifies $\pi(x') = x$.

$\pi(X') \subseteq X$: For $x' \in X'$ and $n \in \mathbb{N}$, denote $x'_i = (f_i, x_i) \in F \times \mathcal{A}$ and $l \in \mathcal{A}^{-\mathbb{N}}$ a word such that $f_{-n-1} = F_X(l)$. By definition of X', we know that $F_X(lx_{-n}) = f_{-n}$ is non-empty; and inductively, we obtain that $F_X(lx_{-n} \dots x_i) = f_i$ is non-empty. In particular, $lx_{-n} \dots x_n$ (and thus, $x_{-n} \dots x_n$) is locally admissible in X: by compactness, the configuration $(x_i)_{i \in \mathbb{Z}}$ belongs in X.

Finally, the equivalence between items (iv), (v) and (vi) follows from the equalities:

$$F_X(w) = \bigcup_{l \in \mathcal{A}^{-\mathbb{N}}} F_X(lw) \quad \text{and} \quad F_X(l) = \bigcap_{n \in \mathbb{N}} F_X(l|_{[-n..0]}).$$

This concludes the proof. □

[OP16] Ormes and Pavlov, "Extender sets and multidimensional subshifts".

This result can be improved in the following sense: if X is not sofic, then $|\{E_X(w) : w \in \mathcal{L}_{[\![n]\!]}(X)\}|$ cannot be less than n. These results are reminiscent of the famous Morse-Hedlund theorem [MH38, Theorem 7.3], and of the existence of Sturmian words:

[MH38] Morse and Hedlund, "Symbolic dynamics".

[OP16] Ormes and Pavlov, "Extender sets and multidimensional subshifts".

Proposition 6.6 ([OP16, Theorem 1.1])**.** *Let X be a $\mathbb{Z}$ subshift. If there exists $n \in \mathbb{N}$ such that $|\{E_X(w) : w \in \mathcal{L}_{[\![n]\!]}(X)\}| \leq n$, then X is sofic.*

Proposition 6.7 ([OP16, Theorem 1.4])**.** *There exists a non-sofic $\mathbb{Z}$ subshift X such that $|\{E_X(w) : w \in \mathcal{L}_{[\![n]\!]}(X)\}| = n + 1$ for all $n \in \mathbb{N}$.*

Proof. Consider $X \subseteq \{0,1\}^{\mathbb{Z}}$ a Sturmian subshift on $\mathbb{Z}$. Since Sturmian subshifts have pattern complexity $N(n) = n + 1$, we immediately obtain that $|\{E_X(w) : w \in \mathcal{L}_{[\![n]\!]}(X)\}| \leq n + 1$. On the other hand, since X is not sofic [Pyt02, Corollary 6.1.11], we cannot have $|\{E_X(w) : w \in \mathcal{L}_{[\![n]\!]}(X)\}| \leq n$. □

[Pyt02] Pythéas Fogg, *Substitutions in dynamics, arithmetics and combinatorics*.

6.3.2 Examples

Going back to the previous examples:

- The full shift $X = \mathcal{A}^{\mathbb{N}}$ is sofic: it has a single extender set.
- The sunny-side-up $X = \{x \in \mathcal{A}^{\mathbb{Z}} : \forall i, j \in \mathbb{Z},\ x_i = x_j = \square \implies i = j\}$ is sofic: it has two extender sets (depending on whether the given pattern contains the symbol $\square$ or not).
- The mirrored counter (defined as the closure of the configurations $\{a^{-\mathbb{N}} b a^n @ c^n d c^{\mathbb{N}} : n \in \mathbb{N}\}$) is not sofic, since it has infinitely many extender sets (the patterns $w_k = b a^k @$ yield distinct extender sets).

6.3.3 Final word

Among $\mathbb{Z}$ subshifts, soficity is entirely solved and determined by the number of extender sets. Indeed, these results provide us with a systematic method to determine the soficity of a $\mathbb{Z}$ subshift X: looking at the configurations of X, either $\{E_X(w) : w \in \mathcal{A}^*\}$ is finite, which means that X is sofic; or the sequence $(|\{E_X(w) : w \in \mathcal{A}^n\}|)_{n \in \mathbb{N}}$ is not bounded and X is not sofic.

Soficity of multidimensional subshifts

7

As opposed to the one-dimensional case, $\mathbb{Z}^d$ sofic subshifts have yet to be characterized for $d \geq 2$. In this section, we provide some examples illustrating the difficulty of determining whether a multidimensional subshift is sofic. We also recall the classic "counting argument" of non-soficity, and intuit that $[\![n]\!]^d$ square patterns in $\mathbb{Z}^d$ sofic subshifts can only communicate information through their borders, which are of size $O(n^{d-1})$.

7.1 A rich class of subshifts

The rule of thumb that governs $\mathbb{Z}$ sofic subshifts is that a subshift $X \subseteq \mathcal{A}^{\mathbb{Z}}$ is sofic if only a *finite amount of information* is enough to determine whether, given a pattern $w \in \mathcal{A}^{[\![n]\!]}$ and a configuration $x \in \mathcal{A}^{\mathbb{Z}}$, we have[60] $w \ltimes_{[\![n]\!]} x \in X$ (Proposition 6.5). The same intuition governs regular languages of finite words, which are often described as the languages whose computations involve finite amounts of memory. In other words, sofic subshifts on $\mathbb{Z}$ are often thought as having "simple" configurations.

On $\mathbb{Z}^d$, the soficity of multidimensional sofic subshifts has yet to be mathematically determined. The one-dimensional intuition still applies: as sofic subshifts are projections of *local* subshifts, the compatibility of a partial configurations and of d-dimensional square patterns of domain $[\![n]\!]^d$ is **constrained by their borders, which are of size** $O(n^{d-1})$.

However, if $O(n^{d-1})$ is finite when $d = 1$, and that having finite memory is a well-understood computational setting[61], the possibilities yielded by having communication $O(n^{d-1})$ in dimensions $d \geq 2$ are harder to grasp. In fact, many $\mathbb{Z}^d$ effective subshifts with complicated structures and computationally complex configurations have surprisingly turned out to be sofic, including for example:

- Substitution-based subshifts are a class of subshifts defined by an iterated *substitution rule*, which maps letters of the alphabet to, say, hyperrectangular finite patches[62]. Many substitution-based tilings are actually sofic in the multidimensional setting by the famous results of [Moz89]. See [AS14] for a generalization to S-adic systems.

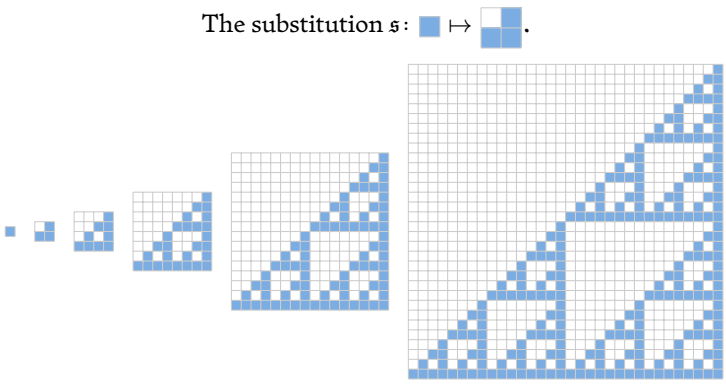


Figure 7.1: A few iterations of a 2-dimensional substitution.

[60] Recall that $w \ltimes_F x$ is the pattern with symbols from w on F, and x outside of F.

[61] This makes $\mathbb{Z}$ sofic subshifts unable to simulate unbounded counters, for example.

[62] Substitution rules can actually be more general, at the cost of dealing with gluing rules for patterns of more arbitrary shapes [JK12].

[Moz89] Mozes, "Tilings, substitution systems and dynamical systems generated by them".

[AS14] Aubrun and Sablik, "Multidimensional effective S-adic subshifts are sofic".

[JK12] Jolivet and Kari, "Consistency of multidimensional combinatorial substitutions".

- The *even shift* and the *odd shift* are the subshifts of $\{\square, \blacksquare\}^{\mathbb{Z}^2}$ such that every finite $\blacksquare$-connected component has respectively even and odd cardinality. The even shift (folklore) and the odd shift ([Cas10, unpublished]) are both sofic.

[Cas10] Cassaigne, *Odd shift*.

- All periodic lifts of effective subshifts (see Proposition 3.44, originally initiated in [Hoc09] and improved in [AS13; DRS10]).

[Hoc09] Hochman, "On the dynamics and recursive properties of multidimensional symbolic systems".

[AS13] Aubrun and Sablik, "Simulation of effective subshifts by two-dimensional subshifts of finite type".

[DRS10] Durand, Romashchenko, and Shen, "Effective closed subshifts in 1D can be implemented in 2D".

- The *square shift* is the subshift $X_s \subseteq \{\square, \blacksquare\}^{\mathbb{Z}^2}$ such that every $\blacksquare$-connected component is a square. Many subshifts of X_s have actually turned out to be sofic, for example those whose square sizes are restricted to prime numbers, or even arbitrary Π_1^0 subsets of $\mathbb{N}$ [Wes17].

[Wes17] Westrick, "Seas of squares with sizes from a Π_1^0 set".

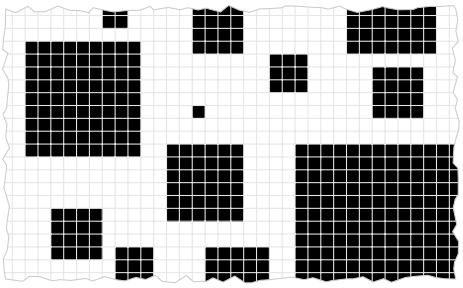


Figure 7.2: A "sea of squares" configuration.

- The *α-density shift* is the subshift $X_\alpha \subseteq \{\square, \blacksquare\}^{\mathbb{Z}^2}$ such that the number of symbols $\blacksquare$ in any $n \times n$ pattern is $O(n^\alpha)$. If $\alpha < 1$ is any rational (or even Π_1-computable) real number, then X_α is a sofic subshift [Des21, Theorem 4].

[Des21] Destombes, "Algorithmic complexity and soficness of shifts in dimension two".

These examples illustrate that multidimensional sofic subshifts can enforce very complex structures in their configurations, both geometrically – by restricting their configurations to complex substitutive shapes – and computationally – by computing arbitrary Π_1^0 subsets of $\mathbb{N}$ –, thus painting a very different behavior from their one-dimensional counterparts.

7.2 Proving soficity

How can we prove the soficity of a multidimensional $\mathbb{Z}^d$ subshift? The answer is, unfortunately, that no general method exists, and that reasonnings must be done on a case by case basis. Yet, some ideas recurringly appear in the litteratures, and the tools of the trade have considerably expanded over the last few decades to prove the soficity of increasingly complex subshifts. We mention here a few of these common methods used to build a local cover for a given set of configurations.

Geometry Some subshifts, whose definitions have a geometrical flavor, are often proved sofic by drawing construction lines. These include, for example, drawings of rectangles, squares, regular grids...

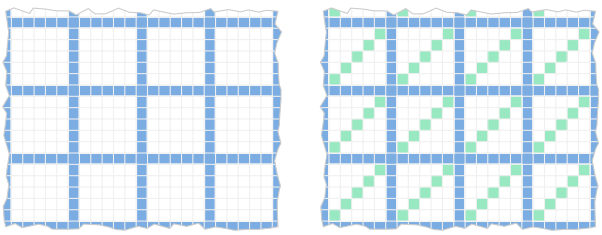


Figure 7.3: A grid configuration (on the left), and a local cover (on the right). In the cover, $\square$-diagonals between crosses make the grid regular.

For more involved geometrical constructions, Mozes' theorem [Moz89] shows that substitutive subshifts are sofic. These include, for example, Toeplitz-like structures and their generalization to a multidimensional setting; and other fractal-like systems.

Computations Subshifts whose definitions are more computationally flavored (counting the number of occurences of a given pattern, restrictions on distances between said occurences, or embedding arbitrary computations to decide a set of integers $X \subseteq \mathbb{N}$) are trickier.

The simplest method of embedding arbitrary Turing machine computations consists in drawing the space-time diagram of the associated machine on $\mathbb{Z}^2$, since these are defined by local constraints. However, this method generates a lot of "meaningless" configurations (configurations that do not contain any computation, computations without any Turing machine head…) and does not behave well with compactness; thus, embeddings of Turing machine computations are usually implemented using a Robinson-like structure[63] [Rob71].

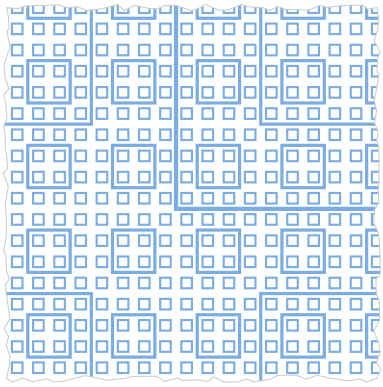


Figure 7.6: Nested squares as obtained by Robinson tilings: as their sizes increase, squares embed more computation steps.

More recently, the so-called fixpoint construction has been applied to sofically realize a large variety of constructions [DRS12]. While some subshifts have been proved sofic by building additional layers upon the original construction (*e.g.* the seas of square shifts from [Wes17], see Theorem 14.2), Proposition 3.44 is today one of the easiest ways to prove some subshift's soficity: since lifts of effective subshifts are sofic, it is possible to sofically realize some $\mathbb{Z}^d$ subshifts by building an effective $\mathbb{Z}^{d-1}$ subshift whose periodic lift can be used, and draw upon this periodic lift some additional construction lines/geometrical gadgets.

Sufficient conditions for soficity Unfortunately, very few sufficient conditions for multidimensional soficity have ever been found: we should mention [OP16, Theorem 1.1], which is a characterization in dimension $d = 1$ but becomes weaker as the dimension increases.

All in all, proving the soficity of multidimensional subshifts often relies on the author's intuition and some folklore cookbook recipes rather than systematic theorems.

7.3 Disproving soficity

The converse task – namely, disproving the soficity of a given multidimensional subshift – is no trivial task either. The non-soficity of some example

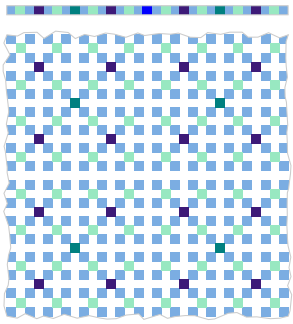


Figure 7.4: One and two-dimensional Toeplitz structures.

[Moz89] Mozes, "Tilings, substitution systems and dynamical systems generated by them".

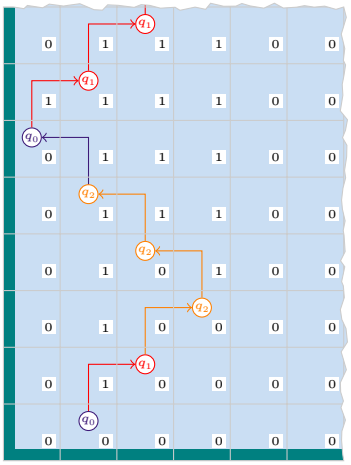


Figure 7.5: Drawing space-time diagrams of Turing machines in tilings.

[63] This method remains the most classical proof of undecidability for the Domino problem.

[Rob71] Robinson, "Undecidability and nonperiodicity for tilings of the plane".

[DRS12] Durand, Romashchenko, and Shen, "Fixed-point tile sets and their applications".

[OP16] Ormes and Pavlov, "Extender sets and multidimensional subshifts".

subshifts, the most famous being probably the so-called "mirror subshift", has been folklore knowledge and widely known within the community. The argument, which we describe below, consists in enforcing the symmetry of two half planes on $\mathbb{Z}^2$ within the configurations of the subshift: this requires to remember $O(n^2)$ bits of information per pattern of domain $[\![n]\!]^2$, which cannot be enforced sofically.

A few sufficient condition for multidimensional non-soficity have been formalized in the litterature, including [Pav13; KM13; DR22]. As noted in [Wes17], the main intuition that governs all multidimensional sofic subshifts is about the information that can be contained within patterns: "*all examples known to the author of effectively closed shifts which are not sofic were obtained by in some sense allowing elements to pack too much important information into a small area*".

In this section, we briefly illustrate this intuition of "packing too much important information" with the example of the *mirror subshift*.

[Pav13] Pavlov, "A class of nonsofic multidimensional shift spaces".

[KM13] Kass and Madden, "A sufficient condition for non-soficness of higher-dimensional subshifts".

[DR22] Destombes and Romashchenko, "Resource-bounded Kolmogorov complexity provides an obstacle to soficness of multidimensional shifts".

[Wes17] Westrick, "Seas of squares with sizes from a Π^0_1 set".

7.3.1 Packing too much information and the counting argument

The most classical example of a $\mathbb{Z}^2$ effective but non-sofic subshift is probably the *mirror subshift*, as its non-soficity is quite elementary to prove.

The case of the mirror subshift

Let $X_{\text{mirror}} \subseteq \{\square, \blacksquare, \blacksquare\}^{\mathbb{Z}^2}$ be the $\mathbb{Z}^2$ subshift defined as follows:

$$X_{\text{mirror}} = \{\square, \blacksquare\}^{\mathbb{Z}^2} \cup \bigcup_{j_0 \in \mathbb{Z}} \Big\{ x \in \{\square, \blacksquare, \blacksquare\}^{\mathbb{Z}^2} : \forall i, j \in \mathbb{Z},\ x_{i,j} = \blacksquare \iff j = j_0 \text{ and } \forall i, j' \in \mathbb{Z},\ x_{i,j_0+j'} = x_{i,j_0-j'} \Big\}$$

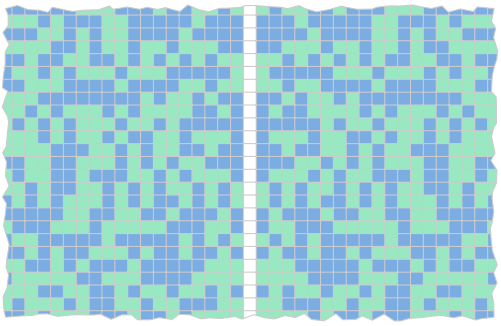


Figure 7.7: A configuration of the mirror subshift.

The subshift X_{mirror} is obviously effective, by simply forbidding the patterns containing a symbol $\blacksquare$ that do not respect the symmetry condition. However, it is also famously non-sofic:

Proposition 7.1. *The $\mathbb{Z}^2$ mirror shift is not sofic.*

To prove the non-soficity of the $\mathbb{Z}^2$ subshift X_{mirror}, or of any subshift $Y \subseteq \mathcal{A}^{\mathbb{Z}^d}$, one needs to prove that there can be no SFT cover for it. Up to my knowledge, most arguments of non-soficity revolve around the same *counting argument* which goes as follows:

Proof. By contradiction, assume that X_{mirror} is a $\mathbb{Z}^2$ sofic subshift. There would exist a local subshift $X' \subseteq \mathcal{A}^{\mathbb{Z}^2}$ and a projection $\pi : \mathcal{A} \to \{\square, \blacksquare, \blacksquare\}$ such that $\pi(X') = X_{\text{mirror}}$.

Let us consider the set of configurations $C \subseteq X_{\text{mirror}}$ having the mirror line just before the origin, *i.e.* $C = \{x \in X_{\text{mirror}} : x|_{\mathbb{Z}\times\{-1\}} = \blacksquare^{\mathbb{Z}\times\{-1\}}\}$.

For any $n \in \mathbb{N}$, notice that the number of valid patterns of $\partial([\![n]\!]^2)$ in the local subshift X' is bounded by $|\mathcal{A}|^{4n}$. However, let us also notice that all 2^{n^2} patterns of $\{\square, \blacksquare\}^{[\![n]\!]^2}$ occur as $x|_{[\![n]\!]^2}$ for some $x \in C$. Applying the socket-drawer principle, there exists two distinct patterns $u, v \in \{\square, \blacksquare\}^{[\![n]\!]^2}$ and two configurations $x', y' \in X'$ such that

- $\pi(x')|_{[\![n]\!]^2} = u$ and $\pi(y')|_{[\![n]\!]^2} = v$ (x' and y' respectively project on $[\![n]\!]^2$ to u and v);
- $x'|_{\partial([\![n]\!]^2)} = y'|_{\partial([\![n]\!]^2)}$ (x' and y' are equal on the border $\partial([\![n]\!]^2)$).

Since X' is a local cover, we can exchange the patterns of $[\![n]\!]^2$ in x' and y' to obtain that the configuration $z' = y' \ltimes_{[\![n]\!]^2} x'$ is a valid configuration of X'. This is a contradiction: we obtain a configuration $\pi(z') \in C$ such that $u \ltimes_{[\![n]\!]^2} \pi(z') \in X_{\text{mirror}}$ and $v \ltimes_{[\![n]\!]^2} \pi(z') \in X_{\text{mirror}}$, breaking the symmetry condition because $u \neq v$. □

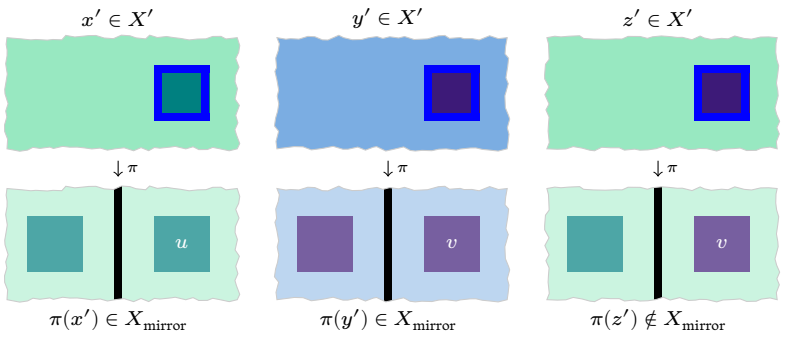


Figure 7.8: Configurations $x', y', z' \in X'$ and their projections.

Analysis of the counting argument

This counting argument is, in fact, a *fooling set*[64]/*fooling pair*[65] argument. It exhibits an unbounded family of pairs of $[\![n]\!]^2$ patterns $(u_i, v_i)_{i\in I}$ such that the pair (u_i, v_j) is valid if and only if $i = j$. In fact, a subshift that admits too many fooling pairs like the mirror subshift cannot be sofic.

Unfortunately, this argument is very restricted in its applications:

- The subshift $Y \subseteq \mathcal{A}^{\mathbb{Z}^d}$ whose soficness is supposed to be disproved needs to have *high pattern complexity*, such as having infinite *surface entropy*[66]. This limits the argument to high complexity subshifts, even thoguh – as noted in [DR22] – it is instructive to realize that even some non-effective subshifts can admit very low pattern complexity.
- The second limitation comes from fooling pair arguments, as these arguments only apply if swapping patterns inside a configuration do introduce an error in the tiling. Admitting fooling pairs seems closely related to admitting increasing sequences of extender sets [KM13], but there exists subshifts that do not verify these conditions: for example, Question 15.26 describes a subshift in which this second requirement fails, as any pattern in said subshift is compatible with all partial configurations *except one*.

Despite these weaknesses, the "mirror subshift" example is a very good illustration of how quantifying the amount of information that appears within patterns of a subshift can prove their non-soficity: if said amount of information is too high, then a subshift cannot be sofic.

[64] From formal language theory.

[65] From communication complexity

[66] The surface entropy of a given subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is defined as:

$$h_{d-1}(X) = \limsup_{n\to+\infty} \frac{\log|N_X([\![n]\!]^d)|}{n^{d-1}}.$$

7.3.2 Limits on ressources

We briefly mention another argument of non-soficity that, for one, does not rely on such counting arguments, but on a bound on the computational ressources on the patterns of domain $[\![n]\!]^d$: [DR22, Corollary 1] provides the first known example of an effective but non-sofic $\mathbb{Z}^2$ subshift with only polynomial pattern complexity.

[DR22] Destombes and Romashchenko, "Resource-bounded Kolmogorov complexity provides an obstacle to soficness of multidimensional shifts".

This example is not obtained by packing a lot of important information within a pattern of limited size, but rather by packing information that requires a lot of computational power to verify. More precisely, the authors' build a subshift whose patterns have low Kolmogorov complexity (*e.g.* $O(\log n)$ bits for patterns of size $[\![n]\!]^2$, hence the polynomial pattern complexity), but high time-bounded Kolmogorov complexity (*e.g.* $\Omega(n^{1.5})$ for patterns of size $[\![n]\!]^2$, when given time limitations $T(n) = 2^{n^3}$), which prevents the subshift from being sofic.

7.3.3 Final word

When is a multidimensional subshift sofic? Answering this question is surprisingly difficult, and only partial conditions are known at the moment, all revolving around some kind of counting/information-theoretic argument.

In the case of the mirror subshift, having to synchronize $O(n^2)$ cells colored with either □ or ■ on both sides of the mirror line would require to transmit $O(n^2)$ bits of information accross borders of size $O(n)$, which is not possible in a local (and thus, in a sofic) subshift.

However incomplete it is, this counting argument highlights the main intuition that will guide us through the second part "Soficity and small representations" of this thesis: the compability of square patterns of domain $[\![n]\!]^d$ and of partial configurations is, in a d-dimensional sofic subshift, constrained by the size of their border: thus, at most $O(n^{d-1})$ bits of communication can be exchanged. And, following the intuition of [DR22], these communications must have limited computational power.

Multidimensional extender sets

Summary

This whole chapter is joint work with Léo Paviet Salomon and Pascal Vanier [CPV25].

[CPV25] Callard, Paviet Salomon, and Vanier, "Computability of extender sets in multidimensional subshifts".

Motivations Motivated by the role they play in the soficity of $\mathbb{Z}$ subshifts, we study in the first part of this thesis the generalization of extender sets to higher-dimensional subshifts, as defined in [KM13]: informally, the extender set of a pattern p in a subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is the set $E_X(p)$ of configurations with a p-shaped hole that correctly extend p into valid configurations of X.

[KM13] Kass and Madden, "A sufficient condition for non-soficness of higher-dimensional subshifts".

Similarly to the case of $\mathbb{Z}$ subshifts, we are interested in the number of extender sets $\{E_X(w) : w \in \mathcal{A}^{[\![n_1,\dots,n_d]\!]}\}$ through several questions: if X is an SFT, a sofic or an effective subshifts, how many extender sets can there be? From a computational point of view, how hard is it to count the number of extender sets of a given subshift X? Can dynamical restrictions (minimality, mixingness...) change these numbers?

In particular, we are interested in *extender entropies*, which are a dynamical invariant introduced in [FP19] as the *asymptotic growth rate* of the number of extender sets.

[FP19] French and Pavlov, "Follower, predecessor, and extender entropies".

Structure of the chapters This part is naturally divided in two chapters:

- Chapter 8 considers extender sets defined by multidimensional subshifts. Defining multidimensional extender sets as in [OP16], we adapt the extender entropies from [FP19] into the multidimensional setting. We prove some elementary properties on extender sets and extender entropies, and look at their (un)computability.
- Chapter 9 characterizes the set of values achieved by well-known classes of subshifts, both generically and under some dynamical/computational restrictions.

[OP16] Ormes and Pavlov, "Extender sets and multidimensional subshifts".

Summary of results We prove that there exists a clear distinction between the extender sets of $\mathbb{Z}$ and $\mathbb{Z}^d$ subshifts for $d \geq 2$: while counting the number of extender sets entirely determines the soficity of $\mathbb{Z}$ subshifts, in higher dimension sofic and effective subshifts can have arbitrarily high number of extender sets.

In Theorem 9.13 and Theorem 9.2, we characterize extender entropies of $\mathbb{Z}^d$ sofic and effective subshifts as the non-negative Π_3-computable real numbers of the arithmetical hierarchy. In particular, they span a dense subset of $\mathbb{R}_+$ (so that the number of extender sets can grow arbitrarily fast among sofic $\mathbb{Z}^d$ subshifts if $d \geq 2$). We also consider the behaviors of extender sets and extender entropies under various computational (Theorem 9.20) and dynamical (Proposition 9.25 and Theorem 9.28) restrictions.

The picture of multidimensional extender entropies looks as follows:

		$\mathbb{Z}$	$\mathbb{Z}^d$ $(d \geq 2)$
Generic	SFT	$\{0\}$ (Proposition 8.8)	
	Sofic	$\{0\}$ (Proposition 9.12)	Π_3 (Theorem 9.13)
	Effective	Π_3 (Theorem 9.2)	
Computable	Sofic	$\{0\}$ (Proposition 9.12)	Π_2 (Theorem 9.20)
	Effective	Π_2 (Theorem 9.20)	
Minimal	Sofic	$\{0\}$ (Proposition 9.24)	
	Effective	Π_1 (Proposition 9.25)	
1-block-gluing	Sofic	$\{0\}$ (Proposition 9.12)	Π_3 (Theorem 9.28)
	Effective	Π_3 (Theorem 9.26)	

Figure 7.9: Possible extender entropies of various classes of subshifts (new results are highlighted). All sets should be intersected with $\mathbb{R}_+$.

Final word In terms of soficity, this means in particular that the number of extender sets of a multidimensional subshift cannot be used to distinguish between sofic and effective subshifts. It had been conjectured that sofic $\mathbb{Z}^d$ subshift would always have extender entropy zero in [KM13], which was later disproved (see for example Example 8.5 from [DR22]): by proving that the extender entropies of sofic and effective subshifts span the same set of values, we definitely conclude that the number of extender sets of a subshift cannot be used to prove or disprove its soficity.

[KM13] Kass and Madden, "A sufficient condition for non-soficness of higher-dimensional subshifts".

[DR22] Destombes and Romashchenko, "Resource-bounded Kolmogorov complexity provides an obstacle to soficness of multidimensional shifts".

8 Extender sets of multidimensional subshifts

In this section, we consider the generalization of extender sets of subshifts to the multidimensional setting from [KM13], and the associated *extender entropy* from [FP19], which is a dynamical invariant obtained as the asymptotic growth rate of the number of extender sets.

[FP19] French and Pavlov, "Follower, predecessor, and extender entropies".

8.1 Extender sets

[KM13] defines multidimensional extender sets as follows:

Definition 8.1. *Given a $\mathbb{Z}^d$ subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ and a pattern $w \in \mathcal{A}^{\circledast d}$, the* extender set *of w in X is defined by*

$$E_X(w) = \{x|_{\mathbb{Z}^d \setminus \mathrm{dom}(w)} \in \mathcal{A}^{\mathbb{Z}^d \setminus \mathrm{dom}(w)} : x \in X,\ x|_{\mathrm{dom}(w)} = w\}.$$

For two (potentially infinite) patterns $u, v \in \mathcal{A}^{\circledast d}$, and a set of positions $D \subseteq \mathrm{dom}(u)$, recall that $u \ltimes_D v \in \mathcal{A}^{\mathrm{dom}(u) \cup \mathrm{dom}(v)}$ denotes the pattern of domain $\mathrm{dom}(u) \cup \mathrm{dom}(v)$ defined as $(u \ltimes_D v)_{\mathbf{i}} = u_{\mathbf{i}}$ if $\mathbf{i} \in D$, and $(u \ltimes_D v)_{\mathbf{i}} = v_{\mathbf{i}}$ otherwise. If $\mathrm{dom}(u)$ and $\mathrm{dom}(v)$ are disjoint, we also denote $u \sqcup v$ their union. With these notations, the extender set of $w \in \mathcal{A}^{\circledast d}$ in $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is:

$$E_X(w) = \{x \in \mathcal{A}^{\mathbb{Z}^d \setminus \mathrm{dom}(w)} : x \sqcup w \in X\}.$$

In a subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$, two patterns u and v in $\mathcal{A}^{\circledast d}$ verify $E_X(u) = E_X(v)$ if any occurence of u in the configurations of X can be swapped with v without introducing a forbidden pattern.

One can think of extender sets (and, in particular, their number) as a way of measuring the complexity of a subshift that is somewhat orthogonal to the more traditional pattern complexity: the full-shift possesses a very easy description but has maximal entropy. Extender sets measure the complexity of patterns relatively to their completions into full configurations, and how much information is needed to ensure that a pattern $w \in \mathcal{A}^{\circledast d}$ is compatible with a partial configuration $x \in \mathcal{A}^{\mathbb{Z}^d \setminus \mathrm{dom}(w)}$.

Remark. *At this point, I should probably address the following conflict of notations: on $\mathbb{Z}$ subshifts, the extender set of a pattern $w \in A^*$ in $X \subseteq \mathcal{A}^{\mathbb{Z}}$ has been defined to be both*

$$E_X(w) = \{(x, y) \in \mathcal{A}^{-\mathbb{N}} \times \mathcal{A}^{\mathbb{N}} : xwy \in X\}$$
$$\text{and } E_X(w) = \{x \in \mathcal{A}^{\mathbb{Z} \setminus \mathrm{dom}(w)} : x \sqcup w \in X\}.$$

These definitions differ because string concatenation on $\mathbb{Z}$ allowed to define extender sets independently of the size of the pattern. Yet, from this point onward we will exclusively use the newer definition: while it makes the assertion "a $\mathbb{Z}$ subshift is sofic if and only if it defines finitely many extender sets" meaningless[67]*, the other characterization $(i) \iff (iii)$ in Proposition 6.5 still determines the soficity of $\mathbb{Z}$ subshifts.*

[67] By definition, patterns of different sizes have distinct domains, and thus distinct extender sets.

8.2 Extender sets in example subshifts

Let us consider the opposite point of view: how does one create a subshift that has some restrictions on the number of extender sets? In this section, we consider a few examples of subshifts and describe their extender sets.

Classical subshifts

Example 8.2 (Full shifts)**.** *Let $\mathcal{A}$ be an alphabet, and $X = \mathcal{A}^{\mathbb{Z}^d}$ the associated full shift. As in the one-dimensional case, for every domain $D \subseteq \mathbb{Z}^d$, there exists a single extender set for patterns of $\mathcal{A}^D$: for $w \in \mathcal{A}^D$, we have*

$$E_X(w) = \{x \in \mathcal{A}^{\mathbb{Z}^d \setminus \mathrm{dom}(w)}\}.$$

Example 8.3 (Sunny-side-up)**.** *On the alphabet $\{\square, \blacksquare\}$, let X be the* sunny-side-up *subshift composed of at most a single $\blacksquare$ cell over a $\square$ background:*

$$X = \{x \in \{\square, \blacksquare\}^{\mathbb{Z}^d} : |x|_{\blacksquare} \leq 1\}.$$

Then for very domain $D \subseteq \mathbb{Z}^d$, there exists two extender sets for the valid patterns of $w \in \{\square, \blacksquare\}^D$: either there exists $\mathbf{i} \in D$ such that $w_{\mathbf{i}} = \blacksquare$, in which case

$$E_X(w) = \{\square^{\mathbb{Z}^d \setminus D}\};$$

or $w = \square^D$, in which case any partial configuration of X belongs to its extender set:

$$E_X(w) = \{x \in \{\square, \blacksquare\}^{\mathbb{Z}^d \setminus D} : |x|_{\blacksquare} \leq 1\}.$$

Figure 8.1: A configuration of the sunny-side-up.

Mirror, mirror

Example 8.4 (Mirror subshift)**.** *On the alphabet $\{\square, \blacksquare, \blacksquare\}$, let X be the* mirror *subshift: at most a single hyperplane $\blacksquare^{\mathbb{Z}^{d-1}}$ reflects $\square$ and $\blacksquare$ cells on both its sides, acting as a mirror*

$$X = \{\square, \blacksquare\}^{\mathbb{Z}^d} \cup \bigcup_{i_0 \in \mathbb{Z}} \{x \in \{\square, \blacksquare, \blacksquare\}^{\mathbb{Z}^d} : \forall i \in \mathbb{Z}, \mathbf{j} \in \mathbb{Z}^{d-1},\ x_{i,\mathbf{j}} = \blacksquare \iff i = i_0 \text{ and } \forall i \in \mathbb{Z}, \mathbf{j} \in \mathbb{Z}^{d-1},\ x_{i_0+i,\mathbf{j}} = x_{i_0-i,\mathbf{j}}\}$$

Then the patterns of domain $[\![n_1, \dots, n_d]\!]$ admit at least $2^{n_1 \cdots n_d}$ extender sets. Indeed, let $C = \{x \in X : x|_{n_1 \times \mathbb{Z}^{d-1}} = \blacksquare^{\mathbb{Z}^{d-1}}\}$ be the set of configurations in which the mirror is placed at the hyperplane $\{n_1\} \times \mathbb{Z}^{d-1}$. For any two distinct words $w, w' \in \{\square, \blacksquare\}^{[\![n_1,\dots,n_d]\!]}$, let us consider a configuration $x \in C$ such that $x|_{[\![n_1,\dots,n_d]\!]} = w$. By the symmetry condition on X, $w' \ltimes_{[\![n_1,\dots,n_d]\!]} x$ is not a valid configuration of X; thus, $E_X(w) \neq E_X(w')$.

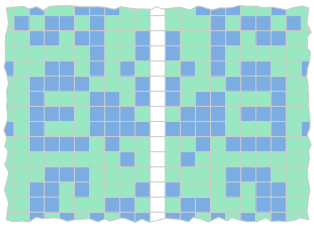


Figure 8.2: A configuration of the mirror subshift.

Unfortunately, the mirror subshift is famously non-sofic (as seen on $\mathbb{Z}^2$ in Proposition 7.1). However, it is actually very easy to create a very similar subshift, with the same number of extender sets, but that is actually sofic:

Example 8.5 (Non-deterministic mirror)**.** *Let us slightly alter the mirror subshift: instead of mirroring a whole half-space of symbols $\{\square, \blacksquare\}$, let us mirror at most one cell instead: on the alphabet $\{\square, \blacksquare, \blacksquare, \textvisiblespace\}$, we define X as the completion of the following set of configurations:*

$$\bigcup_{i_0 \in \mathbb{Z}} \{x \in \{\square, \blacksquare, \blacksquare, \textvisiblespace\}^{\mathbb{Z}^d} : \forall i \in \mathbb{Z}, \mathbf{j} \in \mathbb{Z}^{d-1},\ x_{i,\mathbf{j}} = \blacksquare \iff i = i_0$$
$$i < i_0 \implies x_{i,\mathbf{j}} \in \{\square, \blacksquare\},\ i > i_0 \implies x_{i,\mathbf{j}} \in \{\textvisiblespace, \square, \blacksquare\}$$
*and there exists at most a single $(i, \mathbf{j}) \in \mathbb{Z}^d$ such that $x_{i_0+i,\mathbf{j}} \in \{\square, \blacksquare\}$, in which case $x_{i_0+i,\mathbf{j}} = x_{i_0-i,\mathbf{j}}$}\}.$$

We prove that patterns of domain $[\![n_1, \dots, n_d]\!]$ yield at least $2^{n_1 \cdots n_d}$ extender sets. To do so, fix two distinct words $w, w' \in \{\square, \blacksquare\}^{[\![n_1,\dots,n_d]\!]}$. By definition, there exists some $i \in [1 \,..\, n_1]$ and $j \in [\![n_2, \dots, n_d]\!]$ such that $w_{n_1-i,j} \neq w'_{n_1-i,j}$. Let $x \in X$ be a configuration such that $x|_{[\![n_1,\dots,n_d]\!]} = w$, $x|_{\{n_1\} \times \mathbb{Z}^{d-1}} = \blacksquare^{\mathbb{Z}^{d-1}}$ and $x_{n_1+i,j} \in \{\square, \blacksquare\}$ reflects the cell $x_{n_1-i,j}$. Then $w' \ltimes_{[\![n_1,\dots,n_d]\!]} x$ is not a valid configuration of X, because it breaks the symmetry condition: this proves that $E_X(w) \neq E_X(w')$.

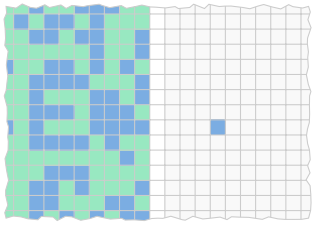


Figure 8.3: A configuration of the non-deterministic mirror subshift.

Lifts

Example 8.6 (Periodic lifts). *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift, and let us consider its periodic lift*

$$X^{\Uparrow} = \{x' \in \mathcal{A}^{\mathbb{Z}^{d+1}} : \exists x \in X, \forall i \in \mathbb{Z}, x'|_{\{i\}\times\mathbb{Z}^d} = x\}.$$

Because of periodicity in the configurations of $X^{\Uparrow}$, distinct finite patterns have distinct extender sets. More precisely, for $w' \in \mathcal{A}^{[\![n_1,\dots,n_{d+1}]\!]}$ a valid pattern in $X^{\Uparrow}$, there exists $w \in \mathcal{A}^{[\![n_2,\dots,n_{d+1}]\!]}$ such that $w'|_{\{i\}\times[\![n_2,\dots,n_{d+1}]\!]} = w$ for every $i \in [\![n_1]\!]$. Then,

$$E_{X^{\Uparrow}}(w') = \{x' \in \mathcal{A}^{\mathbb{Z}^d\setminus[\![n_1,\dots,n_{d+1}]\!]} : \exists x \in E_X(w), x'|_{\{i\}\times\mathbb{Z}^d} = w \sqcup x \text{ if } i \notin [\![n_1]\!] \text{ and } x'|_{\{i\}\times(\mathbb{Z}^d\setminus[\![n_2,\dots,n_d]\!])} = x \text{ otherwise}\}.$$

Example 8.7 (Free lifts). *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift, and let us consider its free lift*

$$X^{\rightleftharpoons} = \{x' \in \mathcal{A}^{\mathbb{Z}^{d+1}} : \forall i \in \mathbb{Z}, x'|_{\{i\}\times\mathbb{Z}^d} \in X\}.$$

Then the extenders of $X^{\rightleftharpoons}$ are products of extenders of X: more precisely, for $w \in \mathcal{A}^{[\![n_1,\dots,n_{d+1}]\!]}$, we have:

$$E_{X^{\rightleftharpoons}}(w') = \{x' \in \mathcal{A}^{\mathbb{Z}^d\setminus[\![n_1,\dots,n_{d+1}]\!]} : \exists (x_0,\dots,x_{n_1-1}) \in \prod_{i=0}^{n_1-1} E_X(w'|_{\{i\}\times[\![n_2,\dots,n_{d+1}]\!]}), x'|_{\{i\}\times\mathbb{Z}^d} \in X \text{ if } i \notin [\![n_1]\!] \text{ and } x'|_{\{i\}\times[\![n_2,\dots,n_{d+1}]\!]} = x_i \text{ otherwise}\}.$$

8.3 Properties of extender sets

Extender sets regularly appear in the litterature under various disguises. Most frequently, they appear in the context of swappable pairs of patterns[68] in subshifts of finite type.

[68] For a given domain $D \subseteq \mathbb{Z}^d$, two patterns $u, v \in \mathcal{A}^D$ are *swappable* if any occurence of u in a configuration of X can be replaced by v without introducing a forbidden pattern.

8.3.1 Subshifts of finite type

In subshifts of finite type, the border of a pattern entirely determines its extender set. More precisely, for $S \subseteq \mathbb{Z}^d$ a subset of positions, we denote by $\partial_l(S)$ the border of S of width l, formally defined as[69]:

$$\partial_l(S) = \{\mathbf{s} \in S : d(\mathbf{s}, S^c) \leq l\}.$$

[69] We denote by d the L_1 (or. "Manhattan") distance on $\mathbb{Z}^d$.

The *interior* of S is then defined as $\mathcal{I}_l(S) = S \setminus \partial_l(S)$, and the *diameter* of S as $\operatorname{diam}(S) = \max_{i,j\in S} d(i,j)$.

Proposition 8.8. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift of finite type and $\mathcal{F}$ a family of forbidden patterns that defines X. Denote $l = \max_{f\in\mathcal{F}} \operatorname{diam}(\operatorname{dom}(f))$. Then for $w \in \mathcal{A}^{\circledast d}$, the extender set of w is entirely determined by $w|_{\partial_l(\operatorname{dom}(w))}$.*

In other words: for $D \subseteq \mathbb{Z}^d$ a finite domain and $w, w' \in \mathcal{A}^D$ two patterns such that $w, w' \in \mathcal{L}(X)$, having $w|_{\partial_l(D)} = w'|_{\partial_l(D)}$ implies that $E_X(w) = E_X(w')$.

Proof. Fix $D \subseteq \mathbb{Z}^d$ and two finite patterns $w, w' \in \mathcal{A}^D$ such that $w, w' \in \mathcal{L}(X)$ and $w|_{\partial_l(D)} = w'|_{\partial_l(D)}$; and consider $x \in E_X(w)$ (so that $x \sqcup w \in X$).

Since $w|_{\partial_l(D)} = w'|_{\partial_l(D)}$, no forbidden pattern appears in the partial configuration $x \sqcup w'|_{\partial_l(D)}$. Since $w' \in \mathcal{L}(X)$, no forbidden pattern appears in w'. Since no forbidden pattern can overlap between $\operatorname{dom}(x)$ and the interior $\mathcal{I}_l(\operatorname{dom}(w))$ by definition of l, the configuration $x \sqcup w'$ is valid in X. □

A counting argument is very often used to prove that two patterns in an SFT have the same extender set. The following argument exists in multiple variations [Sch95; Hoc10, ...], the most simple being:

[Sch95] Schmidt, "The cohomology of higher-dimensional shifts of finite type".

[Hoc10] Hochman, "On the automorphism groups of multidimensional shifts of finite type".

Example 8.9. *Let X be an SFT such that $h(X) > 0$. Then there exists two distinct finite patterns $u, v \in \mathcal{L}(X)$ such that $E_X(u) = E_X(v)$.*

Proof. Fix $\mathcal{F}$ a family of forbidden patterns defining X, and denote by l the diameter of this family: $l = \max_{f \in \mathcal{F}} \mathrm{diam}(\mathrm{dom}(f))$.

By definition of $h(X)$, the number of square patterns of domain $[\![n]\!]^d$ is $|\mathcal{L}_{[\![n]\!]^d}(X)| = 2^{h(X) \cdot n^d + o(n^d)}$. However, the number of $\partial_l([\![n]\!]^d)$ patterns is bounded by $2^{l \cdot O(n^{d-1})}$. By the socket-drawer principle, there exists $n \in \mathbb{N}$ and two distinct patterns $w, w' \in \mathcal{L}_{[\![n]\!]^d}(X)$ such that $w|_{\partial_l([\![n]\!]^d)} = w'|_{\partial_l([\![n]\!]^d)}$. By the previous proposition, this implies that $E_X(w) = E_X(w')$. □

In the previous example, the finite patterns u and v can actually have the same *marker* on their borders[70] [Hoc10, Corollary 13]; in particular, their occurrences in X cannot intersect, which removes issues like Example 8.12.

[70] For $F = \partial_l([\![n]\!]^d)$, a pattern $w \in \mathcal{A}^F$ is a *marker* if, for any configuration $x \in X$ and any $p, p' \in \mathbb{Z}^d$ such that w occurs in x at positions p and p', either $p = p'$ or $|p - p'| > m + l$.

[Hoc10] Hochman, "On the automorphism groups of multidimensional shifts of finite type".

[71] Where a configuration $x \in \mathcal{A}^{\mathbb{Z}^d}$ is *aperiodic* if for every $p \in \mathbb{Z}^d$, there exists a *breaking position* $i \in \mathbb{Z}^d$ such that $x_i \neq x_{i+p}$.

[Sim15] Simpson, "Symbolic dynamics: entropy = dimension = complexity".

8.3.2 Aperiodicity

Exchanging patterns is very often used to prove the existence of aperiodic configurations[71]: indeed, subshifts of finite type contain aperiodic configurations, which can be proved using Kolmogorov complexity [Sim15], measures of maximal entropy, or by combining Example 8.9 with the following geometrical argument:

Proposition 8.10. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift and $w, w' \in \mathcal{L}(X)$ be two distinct finite patterns such that $E_x(w) = E_X(w')$. There exists an aperiodic configuration $x \in X$.*

Proof. Let $x \in X$ be a configuration such that $x|_{\mathrm{dom}(w)} = w$. Assume that x is periodic of period $p \in \mathbb{Z}^d$, and since $E_X(w) = E_X(w')$ let us denote x' a configuration in which an occurence of w in x has been replaced by w'. We claim that x' is aperiodic.

Indeed, let $q \in \mathbb{Z}^d$ be a vector. If q was broken in x at position $i \in \mathbb{Z}^d$, by p-periodicity of x it was broken at all positions $i + \mathbb{Z} \cdot p$, such that infinitely many breaks of q still exist in x'. If q was a period in x, then since a single occurence of w has been replaced by w' in x', q cannot be a period of x'. □

One should remark, however, that sofic subshifts of positive entropy also contain aperiodic configurations, even though we cannot guarantee that the two considered patterns w, w' verify $E_X(w) = E_X(w')$: they can be exchanged in *some configurations*, but not necessarily all of them.

8.3.3 Iterated replacements

By definition, the extender sets of two patterns w and w' are equal if and only if any occurence of w in a configuration can be replaced by w'. This process can be (carefully) iterated to obtain the following lemma:

Proposition 8.11. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift and $w, w' \in \mathcal{L}(X)$ be two finite patterns such that $E_X(w) = E_X(w')$. There exists a configuration $x \in X$ such that w' does not appear in x.*

The intuition would be to swap any occurence of w' with an occurence of w (using the fact that $E_X(w) = E_X(w')$). Unfortunately, replacing w' by w might introduce other occurences of w' in the considered configuration; for example, consider the following case:

Example 8.12. *Let $\mathcal{A} = \{\square, \blacksquare\}$ and let X be the subshift defined by the forbidden patterns $\mathcal{F} = \{\blacksquare\square\}$.*
Let $x \in \mathcal{A}^{\mathbb{Z}}$ be the configuration defined as $x_i = \square$ if $i < 0$, and $x_i = \blacksquare$ otherwise, and pick $w' = \square\blacksquare\blacksquare\blacksquare$ and $w = \square\square\blacksquare\blacksquare$. It is clear that $E_X(w) = E_X(w')$, and w' does occur only once in x; however, replacing w' by w introduces another occurence of w' in x (shifted by one cell). Thus, removing w' from x requires infitely many replacements.

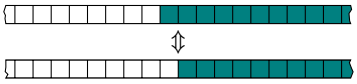


Figure 8.4: Replacing w' in a configuration of X.

We prove Proposition 8.11 sequentially[72]: as in the previous example, we iteratively remove all occurences of w'.

Proof. Consider $w, w' \in \mathcal{L}(X)$ two distinct finite patterns with the same extender set $E_X(w) = E_X(w')$; and denote $D = \mathrm{dom}(w) = \mathrm{dom}(w')$.

Denoting $\leq_{\text{lex}}$ the usual lexicographic ordering on $\mathbb{Z}^d$, let us fix some arbitrary ordering $\preceq$ on $\mathcal{A}$ and extend it to (partial) lexicographic orderings $\preceq_{\text{lex}}$ on $\mathcal{A}^D$ for $D \subseteq \mathbb{Z}^d$: for $u, v \in \mathcal{A}^D$, we have $u \preceq_{\text{lex}} v$ if either $u = v$, or if there exists some $\mathbf{i} \in D$ such that $u_{\mathbf{i}} \prec v_{\mathbf{i}}$ and $u_{\mathbf{j}} = v_{\mathbf{j}}$ for all $\mathbf{i} <_{\text{lex}} \mathbf{j}$.

Without loss of generality, assume that $w \prec_{\text{lex}} w'$ (otherwise, change the ordering $\preceq$ on $\mathcal{A}$). Let us fix y a configuration in which w' appears. We prove that there exists a configuration $y^{(n)}$ such that w' does not appear in the ball $[-n \,..\, n]^d$. This will conclude the proof by compactness.

Consider $y|_{[-n..n]^d}$. If w' does not appear in $y|_{[-n..n]^d}$, we can take $y^{(n)} = y$. Otherwise, let us pick an occurence of w' in $y|_{[-n..n]^d}$. As $E_X(w') = E_X(w)$, we can replace this occurence of w' with an occurence of w: this results in a configuration y' that belongs in X, but such that $y'|_{[-n..n]^d} \prec_{\text{lex}} y|_{[-n..n]^d}$. Since finitely many patterns color $[-n \,..\, n]^d$, repeating these replacements must eventually terminate in a configuration $y^{(n)}$ such that w' does not appear in $y^{(n)}|_{[-n..n]^d}$. □

[72] The first occurence of this argument that I know of appeared in the proof of [QT00, Lemma 2.2].

[QT00] Quas and Trow, "Subshifts of multi-dimensional shifts of finite type".

8.3.4 Minimality

An application of Proposition 8.11 is the following: in a minimal subshift, there are no interchangeable pairs. In other words:

Proposition 8.13. *In a minimal subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$, each extender set corresponds to a single pattern: in other words, for two finite patterns $w, w' \in \mathcal{L}(X)$, $E_X(w) = E_X(w')$ if and only if $w = w'$.*

Proof. By contraposition, assume there exists two distinct finite patterns $w, w' \in \mathcal{L}(X)$ such that $E_X(w) = E_X(w')$. By Proposition 8.11, there exists a configuration x in which w' does not appear and X is not minimal. □

8.4 Extender entropy

From now on, we will count the number of extender sets of a given size in subshifts. For a subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$, we slightly abuse notations and denote by $E_X(\llbracket n_1, \dots, n_d \rrbracket)$ all the extender sets of patterns of domain $\llbracket n_1, \dots, n_d \rrbracket$, *i.e.*

$$E_X(\llbracket n_1, \dots, n_d \rrbracket) = \{E_X(w) : w \in \mathcal{A}^{\llbracket n_1, \dots, n_d \rrbracket}\}.$$[73]

As is usual with subshifts and complexity measures, the exact behavior of the sequence $(|E_X(\llbracket n_1, \dots, n_d \rrbracket)|)_{n_1,\dots,n_d \in \mathbb{N}^d}$ can greatly vary. Unfortunately, the sequence is not monotonic[74] [OP16, Example 3.5[75]], and even in the case of $\mathbb{Z}$ sofic subshifts, $(|E_X(\vec{n})|)_{\vec{n} \in \mathbb{N}}$ can be almost any eventually periodic sequence [Fre16a, Theorem 1.3].

Our study of the sequence $(|E_X(\llbracket n_1, \dots, n_d \rrbracket)|)_{n_1,\dots,n_d \in \mathbb{N}^d}$ mostly focuses on its asymptotic growth, which was introduced in [FP19] under the name of

[73] Note that $E_X(\llbracket n_1, \dots, n_d \rrbracket)$ is a set of extender sets, and not an extender set itself. In particular, it does *not* count partial configurations: it is the number of *equivalence classes* of patterns of a given size.

[74] To be precise: given $(\vec{n}^{(k)})_{k \in \mathbb{N}}$ for $\vec{n}^{(k)} \in \mathbb{N}^d$ an increasing sequence, $(|E_X(\vec{n}^{(k)})|)_{k \in \mathbb{N}}$ is not monotonic.

[OP16] Ormes and Pavlov, "Extender sets and multidimensional subshifts".

[75] (Attributed to Martin Delacourt) There exists a $\mathbb{Z}$ subshift $Y \subseteq \{a, b, c\}^{\mathbb{Z}}$ with $N_X(\llbracket 2n \rrbracket) = 46$ and $N_X(\llbracket 2n+1 \rrbracket) = 44$.

[Fre16a] French, "Characterizing follower and extender set sequences".

[FP19] French and Pavlov, "Follower, predecessor, and extender entropies".

extender entropy.

Definition 8.14. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift. The* extender entropy *of X, denoted $h_E(X)$, is defined as*[76]*:*

$$h_E(X) = \lim_{n_1,\dots,n_d \in \mathbb{N}^d} \frac{\log |E_X(\llbracket n_1, \dots, n_d \rrbracket)|}{n_1 \cdots n_d}.$$

[76] See Section 2.2, Definition 2.1 for a brief explanation about multivariate limits. Informally: the limit is taken when all the n_i's grow to $+\infty$.

Proposition 8.15. *The limit in the previous definition exists and is well-defined. Furthermore,*

$$\lim_{n_1,\dots,n_d \in \mathbb{N}^d} \frac{\log |E_X(\llbracket n_1, \dots, n_d \rrbracket)|}{n_1 \cdots n_d} = \inf_{n_1,\dots,n_d \in \mathbb{N}^d} \frac{\log |E_X(\llbracket n_1, \dots, n_d \rrbracket)|}{n_1 \cdots n_d}.$$

Proof. We prove that the function $n_1, \dots, n_d \mapsto |E_X(\llbracket n_1, \dots, n_d \rrbracket)|$ is submultiplicative. The proposition will then follow from the multivariate version of the subadditive lemma (see Lemma 2.3).

More formally, fix some $1 \leq i \leq d$, $n_1, \dots, n_d \in \mathbb{N}^d$ and $m_i \in \mathbb{N}$. We want to prove that

$$|E_X(\llbracket n_1, \dots, n_i + m_i, \dots, n_d \rrbracket)| \leq |E_X(\llbracket n_1, \dots, n_i, \dots, n_d \rrbracket)| \cdot |E_X(\llbracket n_1, \dots, m_i, \dots, n_d \rrbracket)|.$$

To do so, consider the map

$$f \colon \mathcal{A}^{\llbracket n_1,\dots,n_i+m_i,\dots,n_d \rrbracket} \to E_X(\llbracket n_1, \dots, n_i, \dots, n_d \rrbracket) \times E_X(\llbracket n_1, \dots, m_i, \dots, n_d \rrbracket)$$

defined as follows: given an arbitrary pattern $w \in \mathcal{A}^{\llbracket n_1,\dots,n_i+m_i,\dots,n_d \rrbracket}$, we decompose w into $w = u \sqcup v$ with $u \in \mathcal{A}^{\llbracket n_1,\dots,n_i,\dots,n_d \rrbracket}$ and $v \in \mathcal{A}^{\llbracket n_1,\dots,m_i,\dots,n_d \rrbracket}$; then, we define $f(w)$ to be $(E_X(u), E_X(v))$.

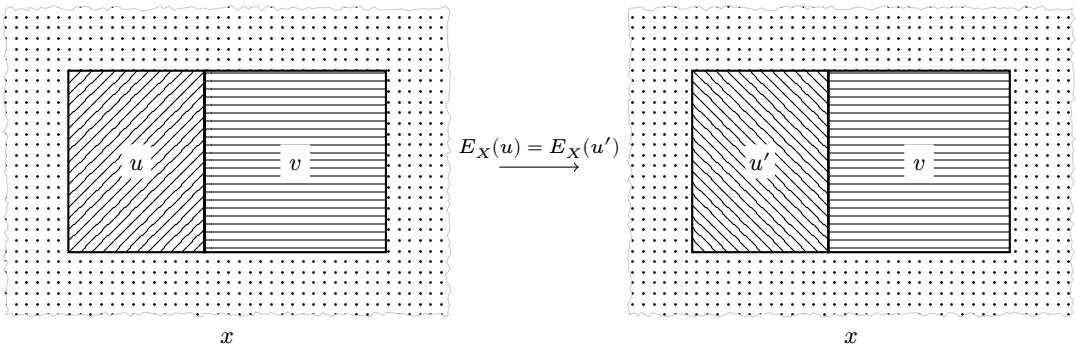


Figure 8.5: Replacing u by u' in the configuration $x \sqcup w = x \sqcup (u \sqcup v)$.

Let us prove that if any two patterns $w, w' \in \mathcal{A}^{\llbracket n_1,\dots,n_i+m_i,\dots,n_d \rrbracket}$ verify $f(w) = f(w')$, then $E_X(w) = E_X(w')$. Indeed, denoting $w = u \sqcup v$ and $w' = u' \sqcup w'$ as above, we have by definition of f that $E_X(u) = E_X(u')$ and $E_X(v) = E_X(v')$. Considering an arbitrary $x \in \mathcal{A}^{\mathbb{Z}^d \setminus \mathrm{dom}(w)}$, we do successive replacements and obtain:

$$\begin{aligned} x \sqcup w \in X &\iff x \sqcup u \sqcup v \in X \\ &\iff x \sqcup u' \sqcup v \in X \quad \text{Since } E_X(u) = E_X(u'). \\ &\iff x \sqcup u' \sqcup v' \in X \quad \text{Since } E_X(v) = E_X(v'). \\ &\iff x \sqcup w' \in X. \end{aligned}$$

Thus, $E_X(w) = E_X(w')$. This implies that $|E_X(\llbracket n_1, \dots, n_i + m_i, \dots, n_d \rrbracket)| \leq |E_X(\llbracket n_1, \dots, n_i, \dots, n_d \rrbracket)| \cdot |E_X(\llbracket n_1, \dots, m_i, \dots, n_d \rrbracket)|$ and concludes the proof. □

By definition of this limit, the extender entropy of X does not depend on the sequence of hyperrectangles used to compute it: in other words,

Proposition 8.16. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift. For any sequence of rectangular domains $([\![n_1^{(k)}, \dots, n_d^{(k)}]\!])_{k\in\mathbb{N}}$ such that $\lim_{k\to+\infty} n_i^{(k)} = +\infty$ for every $1 \le i \le d$:*

$$\frac{\log |E_X([\![n_1^{(k)}, \dots, n_d^{(k)}]\!])|}{n_1^{(k)} \cdots n_d^{(k)}} \xrightarrow[k\to+\infty]{} h_E(X).$$

In particular, it is enough to consider the extender sets over the domains $[\![n]\!]^d$:

$$\frac{\log |E_X([\![n]\!]^d)|}{n^d} \xrightarrow[n\to+\infty]{} h_E(X).$$

Proof. Let $\varepsilon > 0$. By definition of $h_E(X)$, there exists $(N_1, \dots, N_d) \in \mathbb{N}^d$ such that, for all $(n_1, \dots, n_d) \in \mathbb{N}^d$:

$$(n_1, \dots, n_d) \ge (N_1, \dots, N_d) \implies \left| \frac{\log |E_X([\![n_1, \dots, n_d]\!])|}{n_1 \cdots n_d} - h_E(X) \right| \le \varepsilon.$$

Let $(n_1^{(k)}, \dots, n_d^{(k)})_{k\in\mathbb{N}}$ be a sequence such that $\lim_{k\to+\infty} n_i^{(k)} = +\infty$ for every $1 \le i \le d$. In particular, there exists $K \in \mathbb{N}$ such that $n_i^{(k)} \ge N_i$ for every $1 \le i \le d$ and $k \ge K$. Thus, for all $k \in \mathbb{N}$:

$$k \ge K \implies \left| \frac{\log |E_X([\![n_1^{(k)}, \dots, n_d^{(k)}]\!])|}{n_1^{(k)} \cdots n_d^{(k)}} - h_E(X) \right| \le \varepsilon. \qquad \square$$

8.5 Examples

Let us go back to the examples of Section 8.2 and compute their extender entropies:

Example 8.17 (Full shift)**.** *Let $X = \mathcal{A}^{\mathbb{Z}^d}$ be a full shift. Then $h_E(X) = 0$.*

Example 8.18 (Sunny-side-up)**.** *Let $X \subseteq \{\square, \blacksquare\}^{\mathbb{Z}^d}$ be the sunny-side-up subshift. Then $h_E(X) = 0$.*

Example 8.19 (Mirror subshifts)**.** *Let $X \subseteq \{\square, \blacksquare, \blacksquare\}^{\mathbb{Z}^d}$ be the mirror subshift and $X' \subseteq \{\square, \blacksquare, \blacksquare, \text{␣}\}^{\mathbb{Z}^d}$ be the non-deterministic mirror subshift. Then $h_E(X) = h_E(X') = 2$.*

Proof. We have proved that $E_X([\![n]\!]^d) \ge 2^{n^2}$. Since $|\mathcal{L}_{[\![n]\!]^d}(X)| \le (n+1) \cdot 2^{n^2}$ (by considering all positions at which the mirror hyperplane can appear), we conclude by Proposition 8.22 that $h_E(X) = 2$. The proof is similar for $h_E(X')$. $\square$

Example 8.20 (Periodic and free lifts)**.** *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be any subshift. Then $h_E(X^{\Uparrow}) = 0$ and $h_E(X^{\rightleftharpoons}) = h_E(X)$.*

Proof. Since $h(X^{\Uparrow}) = 0$, we obtain by Proposition 8.22 that $h_E(X^{\Uparrow}) = 0$. In the case of the free lift, we already noticed that

$$|E_{X^{\rightleftharpoons}}([\![n_1, \dots, n_{d+1}]\!])| = |E_X([\![n_2, \dots, n_{d+1}]\!])|^{n_1}.$$

Thus, by taking logarithm and dividing by $n_1 \cdots n_{d+1}$, we conclude that $h_E(X^{\rightleftharpoons}) = h_E(X)$. $\square$

The latter example will be used in the next chapter to lift properties/constructions from dimensions $d = 1$ and $d = 2$ to higher dimensions $d' \ge d$:

Corollary 8.21. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift that is effective (resp. sofic, resp. SFT). Then $X^{\rightleftharpoons^{d'}} \subseteq \mathcal{A}^{\mathbb{Z}^{d'}}$ is a $\mathbb{Z}^{d'}$ effective (resp. sofic, resp. SFT) subshift such that $h_E(X^{\rightleftharpoons^{d'}}) = h_E(X)$.*

8.6 Properties of extender entropies

8.6.1 Dynamical properties

Proposition 8.22. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift. Then $h_E(X) \leq h(X)$.*

Proof. The function $f\colon w \in \mathcal{L}_{[\![n_1,\dots,n_d]\!]}(X) \mapsto E_X(w) \in E_X([\![n_1,\dots,n_d]\!])$ is surjective, so that $|E_X([\![n_1,\dots,n_d]\!])| \leq |\mathcal{L}_{[\![n_1,\dots,n_d]\!]}(X)|$. Going to the limit on the n_i's, we obtain $h_E(X) \leq h(X)$. □

Proposition 8.23. *Let $X, X' \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be two subshifts. Then $h_E(X \times X') = h_E(X) + h_E(X')$.*

Proof. Let $w, w' \in \mathcal{A}^{*d}$ be such that $\mathrm{dom}(w) = \mathrm{dom}(w')$, and consider two configurations $x, x' \in \mathcal{A}^{\mathbb{Z}^d \setminus \mathrm{dom}(w)}$. By definition of the Cartesian product, we have:

$$(x, x') \sqcup (w, w') \in X \times X' \quad \iff \quad (x \sqcup w, x' \sqcup w') \in X \times X'.$$

Thus, we deduce that $E_{X\times X'}(w, w') = E_X(w) \times E_{X'}(w')$, which results in $|E_{X\times X'}([\![n_1,\dots,n_d]\!])| = |E_X([\![n_1,\dots,n_d]\!])| \cdot |E_{X'}([\![n_1,\dots,n_d]\!])|$. □

Proposition 8.24. *The extender entropy is a conjugacy invariant of $\mathbb{Z}^d$ subshifts.*

The classical topological entropy can be proved to be a conjugacy invariant, since topological entropy is (weakly) decreasing under factor map. Unfortunaly, extender entropy is not monotonic under factor maps, as noticed in [FP19, Theorem 3.7]. Thus, we proceed differently.

[FP19] French and Pavlov, "Follower, predecessor, and extender entropies".

Proof. Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ and $Y \subseteq \mathcal{B}^{\mathbb{Z}^d}$ be two conjugated subshifts by a bijective factor map $\varphi : X \to Y$ of biradius r. We prove that $h_E(X) = h_E(Y)$.

Claim. *For $D \subseteq \mathbb{Z}^d$ a finite domain and any two patterns $w, w' \in \mathcal{A}^D$ such that $w|_{\partial_{4r}(D)} = w'|_{\partial_{4r}(D)}$ and $E_X(w|_{\mathcal{I}_{2r}(D)}) = E_X(w'|_{\mathcal{I}_{2r}(D)})$. We claim that $E_Y(\varphi(w)) = E_Y(\varphi(w'))$.*[77]

Indeed, let y be a configuration in Y such that $y|_{\mathcal{I}_r(D)} = \varphi(w)$. Since φ^{-1} is bijective, there exists some x such that $\varphi(x) = y$: since φ^{-1} has radius r, we deduce that $x|_{\mathcal{I}_{2r}(D)} = w|_{\mathcal{I}_{2r}(D)}$. Since $E_X(w|_{\mathcal{I}_{2r}(D)}) = E_X(w'|_{\mathcal{I}_{2r}(D)})$, we know that the configuration x' defined by $x'_i = x_i$ if $i \notin \mathcal{I}_{2r}(D)$ and $x'_i = w'_i$ otherwise is valid in X. Finally, let us consider $y' = \varphi(x') \in Y$:

- Since φ has radius r and that $x|_{\mathbb{Z}^d\setminus\mathcal{I}_{4r}(D)} = x'|_{\mathbb{Z}^d\setminus\mathcal{I}_{4r}(D)}$, we have

  $$y'|_{\mathbb{Z}^d\setminus\mathcal{I}_{3r}(D)} = y|_{\mathbb{Z}^d\setminus\mathcal{I}_{3r}(D)}.$$

  In particular,

  $$\begin{aligned} y'|_{\mathbb{Z}^d\setminus\mathcal{I}_r(D)} &= y|_{\mathbb{Z}^d\setminus\mathcal{I}_r(D)} \\ y'|_{\mathcal{I}_r(D)\setminus\mathcal{I}_{3r}(D)} &= y|_{\mathcal{I}_r(D)\setminus\mathcal{I}_{3r}(D)} = \varphi(w)|_{\mathcal{I}_r(D)\setminus\mathcal{I}_{3r}(D)} \\ &= \varphi(w')|_{\mathcal{I}_r(D)\setminus\mathcal{I}_{3r}(D)}. \end{aligned}$$

  (the latter holding because φ has radius r and $w|_{\partial_{4r}(D)} = w'|_{\partial_{4r}(D)}$.)
- And since $x'|_{\mathcal{I}_{2r}(D)} = w'|_{\mathcal{I}_{2r}(D)}$, we have

  $$y'|_{\mathcal{I}_{3r}(D)} = \varphi(w')|_{\mathcal{I}_{3r}(D)}.$$

[77] Since φ has radius r, we denote by $\varphi(w)$ the pattern of domain $\mathcal{I}_r(D)$ that is the image of w by φ.

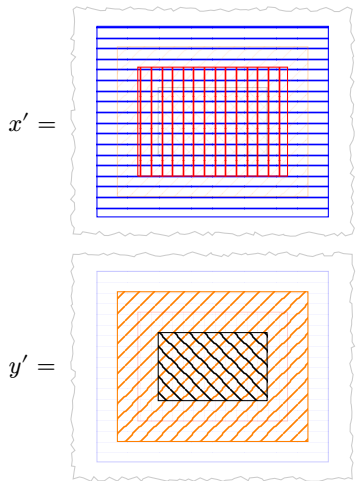


Figure 8.6: This figure draws the areas $D \supseteq \mathcal{I}_r(D) \supseteq \mathcal{I}_{2r}(D) \supseteq \mathcal{I}_{3r}(D)$. In x', we draw $D = \mathrm{dom}(w) = \mathrm{dom}(w')$ with ≡ and $\mathcal{I}_{2r}(D) = \mathrm{dom}(\varphi^{-1}(\varphi(w)))$ with |||. In y', we draw $\mathcal{I}_r(D) = \mathrm{dom}(\varphi(w))$ with // and $\mathcal{I}_{3r}(D) = \mathrm{dom}(\varphi(w'|_{\mathcal{I}_{2r}(D)}))$ with \\.

[78] In an hyperrectangle of dimension d and size $n_1 \cdots n_d$, there are $2d$ facets of dimension $d-1$; and each facet of area $\prod_{j\neq i} n_j$ occurs twice.

In particular, we have $y'|_{\mathbb{Z}^d \setminus \mathcal{I}_r(D)} = y|_{\mathbb{Z}^d \setminus \mathcal{I}_r(D)}$, and $y'|_{\mathcal{I}_r(D)} = \varphi(w')$. Thus $E_Y(\varphi(w)) \subseteq E_Y(\varphi(w'))$ and by symmetry $E_Y(\varphi(w)) = E_Y(\varphi(w'))$.

End of the proof. From the previous claim, we deduce that[78]:

$$|E_Y([\![n_1 - 2r, \dots, n_d - 2r]\!])| \leq 2^{\sum_{i=1}^{d} 2 \cdot 4r \cdot \prod_{j \neq i} n_j} \cdot |E_X([\![n_1 - 4r, \dots, n_d - 4r]\!])|;$$

and the symmetric bound exists when exchanging X and Y. Completing the computations, we conclude that $h_E(X) = h_E(Y)$. □

8.6.2 Computational complexity

Finally, we relate the computational complexity of the extender entropy of a subshift with the complexity of its language in the two following propositions.

Proposition 8.25. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift such that $\mathcal{L}(X)$ is a Π^0_n (resp. Σ^0_n) language. Then, when given some finite $D \subseteq \mathbb{Z}^d$ and two patterns $u, v \in \mathcal{A}^D$, deciding whether $E_X(u) \subseteq E_X(v)$ is a Π^0_{n+1} problem. In particular, deciding whether $E_X(u) = E_X(v)$ is a Π^0_{n+1} problem.*

Sketch of proof. Notice that $E_X(u) \subseteq E_X(v)$ if and only if:

$$\forall n \in \mathbb{N}, \forall w \in \mathcal{A}^{[\![n]\!]^d},\ u \sqsubseteq w \implies (w \notin \mathcal{L}(X)) \vee (w|_{\mathrm{dom}(w) \setminus D} \sqcup v \in \mathcal{L}(X)).$$

Since the righthand side of the implication is a disjunction between a Σ^0_n and a Π^0_n problem, and that the only infinite quantification before it is "$\forall n \in \mathbb{N}$", deciding the inclusion of extender sets is indeed a Π^0_{n+1} problem. □

We proved in [CPV25, Proposition 10] that, in the context of $\mathbb{Z}$ effective subshifts, the inclusion of extender sets is in fact Π^0_2-complete.

[CPV25] Callard, Paviet Salomon, and Vanier, "Computability of extender sets in multidimensional subshifts".

From the previous proposition, we deduce the computational complexity of counting the number of extender sets:

Proposition 8.26. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift such that $\mathcal{L}(X)$ is a Π^0_n (resp. Σ^0_n) language. Then, when given $k \in \mathbb{N}$ and $(n_1, \dots, n_d) \in \mathbb{N}^d$, determining whether "$k \leq |E_X([\![n_1, \dots, n_d]\!])|$" is a Σ^0_{n+1} problem.*

Proof. Indeed, $k \leq |E_X([\![n_1, \dots, n_d]\!])|$ if and only if:

$$\exists u_1, \dots, u_k \in \mathcal{L}_{[\![n_1, \dots, n_d]\!]}(X),\ \bigwedge_{1 \leq i < j \leq k} E_X(v_i) \neq E_X(v_j).$$

Which we rewrite into:

$$\exists u_1, \dots, u_k \in \mathcal{A}^{[\![n_1, \dots, n_d]\!]},\ \bigwedge_{1 \leq i \leq d} u_i \in \mathcal{L}(X) \ \wedge \bigwedge_{1 \leq i < j \leq k} E_X(u_i) \neq E_X(u_j).$$

Since the quantification "$\exists u_1, \dots, u_k \in \mathcal{A}^{[\![n_1, \dots, n_d]\!]}$" is finite, and that Σ^0_{n+1} is stable by finite conjuctions, it follows from Proposition 8.25 that determining whether $k \leq |E_X([\![n_1, \dots, n_d]\!])|$ is a Σ^0_{n+1} problem. □

Characterizations of extender entropies

9

This section studies the set of possible values for extender entropies of various classes of subshifts (effective, sofic, SFTs) with computational and dynamical properties. We completely characterize extender entropies in the arithmetical hierarchy of real numbers, the main result being Theorem 9.13: extender entropies of $\mathbb{Z}^2$ sofic subshifts exactly span the (non-negative) Π_3-computable real numbers.

Most of these results were published in [CPV25].

[CPV25] Callard, Paviet Salomon, and Vanier, "Computability of extender sets in multidimensional subshifts".

9.1 Subshifts of finite type

Theorem 9.1. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift of finite type. Then $h_E(X) = 0$.*

Proof. Let $\mathcal{F}$ be a finite family of forbidden patterns defining X, and let us denote $l \in \mathbb{N}$ the diameter $l = \max_{f \in \mathcal{F}} \operatorname{diam}(\operatorname{dom}(f))$. From Proposition 8.8, the number of extender sets of domain $[\![n]\!]^d$ is bounded by the number of possible patterns of domain $\partial_l([\![n]\!]^d)$. In particular,

$$\log |E_X([\![n]\!]^d)| \leq 2dl \cdot \log |\mathcal{A}| \cdot n^{d-1} = O(n^{d-1}).$$

Taking the limit, we obtain $h_E(X) = 0$. □

9.2 Effective subshifts

Theorem 9.2. *For $d \geq 1$, the set of extender entropies of $\mathbb{Z}^d$ effective subshifts is exactly $[0, +\infty) \cap \Pi_3$.*

This theorem is composed of two statements:

Lemma 9.3. *Let $Z \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be an effective subshift. Then $h_E(Z) \in [0, +\infty) \cap \Pi_3$.*

Proof. Since Z is effective, the finite patterns $\mathcal{L}(Z) \cap \mathcal{A}^{*d}$ form a Π_1^0 set. By Proposition 8.26, we deduce that the sets $\{k \in \mathbb{N} \mid k \leq |E_Z([\![0,n]\!]^d)|$ are Σ_2^0 sets for $n \in \mathbb{N}$; so that all $\{r \in \mathbb{Q} \mid r \leq \frac{\log |E_Z([\![0,n]\!]^d)|}{n^d}\}$ are Σ_2^0 sets too.

And since

$$h_E(Z) = \lim_{n \to +\infty} \frac{\log |E_Z([\![0,n]\!]^d)|}{n^d} = \inf_{n \in \mathbb{N}} \frac{\log |E_Z([\![0,n]\!]^d)|}{n^d},$$

we conclude that $h_E(Z) \in \Pi_3$ as the infimum of a uniform sequence of Σ_2 real numbers. □

The rest of this section focuses on the converse statement. By Corollary 8.21, we reduce to the one-dimensional case and are left with proving:

Lemma 9.4. *For any real number $\alpha \in [0, +\infty) \cap \Pi_3$, there exists an effective $\mathbb{Z}$ subshift Z_α such that $h_E(Z_\alpha) = \alpha$.*

The following sections are dedicated to proving Lemma 9.4 by providing an explicit construction of such a Z effective subshift Z_α. Let us fix $\alpha \in \Pi_3$ (we reduce to the case $0 \leq \alpha \leq 1$, since extender entropy is additive under

cartesian product by Proposition 8.23), and fix a presentation $(\alpha_{i,j})_{i,j\in\mathbb{N}^2}$ of α, that is a recursively enumerable family of Π_1 real numbers such that $\alpha = \inf_{i\in\mathbb{N}} \sup_{j\in\mathbb{N}} \alpha_{i,j}$. We assume that $0 \leq \alpha_{i,j} \leq 1$ for all $i, j \in \mathbb{N}$, and by Proposition 4.20, we assume that the family $(\alpha_{i,j})_{i,j\in\mathbb{N}^2}$ satisfies some monotonicity properties: for all $i \in \mathbb{N}$, $(a_{i,j})_{j\in\mathbb{N}}$ is weakly increasing towards some $\alpha_i \in [0,1] \cap \Sigma_2$; and the sequence $(\alpha_i)_{i\in\mathbb{N}}$ is weakly decreasing towards α.

9.2.1 Encoding integers in configurations

The first step of this proof consists in fixing a method for the encoding of integers within $\mathbb{Z}$-configurations: to encode an integer $i \in \mathbb{N}$, we use configurations in which a symbol $*$ is i-periodic over a blank background.

More precisely, consider the alphabet $\mathcal{A}_* = \{*, \sqcup\}$. We define $\langle i \rangle_{k_1}$ as the following i-periodic configuration of $\mathcal{A}_*^{\mathbb{Z}}$

$$\langle i \rangle_{k_1} = \sigma^{k_1}(\dots \sqcup \underbrace{* \sqcup \dots \sqcup *}_{i+1 \text{ symbols}} \sqcup \dots \sqcup * \sqcup \dots),$$

or more formally $(\langle i \rangle_{k_1})_p = *$ if $p = k_1 \bmod i$ and $(\langle i \rangle_{k_1})_p = \sqcup$ otherwise. A configuration $\langle i \rangle_{k_1}$ is said to *encode* the integer $i \in \mathbb{N}$.

For i ranging in $\mathbb{N}$ and $k_1 \in [\![i]\!]$, all the configurations $\langle i \rangle_{k_1}$ generate a subshift that we denote

$$X_* = \bigcup_{i\in\mathbb{N}} \{\langle i \rangle_{k_1} \in \mathcal{A}_*^{\mathbb{Z}} : 0 \leq k_1 < i\} \cup \langle \infty \rangle$$

where $\langle \infty \rangle = \{x \in \mathcal{A}_*^{\mathbb{Z}} : |x|_* \leq 1\}$ is the set of configurations containing at most a single symbol $*$. The configurations of $\langle \infty \rangle$ are said to be *degenerate*, and appear in X_* as limit configurations of all the $\langle i \rangle_{k_1}$.

9.2.2 Auxiliary subshift Z'_α

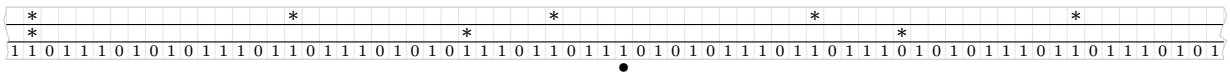


Figure 9.1: A *proper* configuration $(\langle 15 \rangle_{11}, \langle 25 \rangle_{16}, z^{(d)})$ where $z^{(d)}$ is obtained from the Toeplitz encoding $\mathcal{T}(\beta)$ for $\beta = \frac{5}{8} = 0.101_{(2)}$.

We first create an auxiliary subshift Z'_α on the following three layers[79]:

1. **First layer** L_1: We set $L_1 = X_*$ to encode integers $i \in \mathbb{N}$. Intuitively, $i \in \mathbb{N}$ will denote which Σ_2 number α_i is approximated in the configuration.
2. **Second layer** L_2: We also set $L_2 = X_*$ to encode integers $j \in \mathbb{N}$. Intuitively, $j \in \mathbb{N}$ will denote which Π_1 number $\alpha_{i,j}$ is approximated in the configuration.
3. **Density layer** L_d: We define the density layer $L_d = \{0,1\}^{\mathbb{Z}}$. Whenever the first two layers L_1, L_2 are non-degenerate and repectively encode integers $i \in \mathbb{N}$ and $j \geq i$, this layer will be restricted to densities $\lesssim \alpha_{i,j}$. Since the real numbers $\alpha_{i,j}$ are a computably enumerable family of Π_1 real numbers, the sets $D_{i,j} = \{a \in \{0,1\}^{\mathbb{N}} : \sum_{k\in\mathbb{N}} a_k 2^{-(k+1)} \leq \alpha_{i,j}\}$ are effectively closed, and so are the $\mathcal{T}(D_{i,j})$.

such that Z'_α is defined as:

$$Z'_\alpha = \left\{(z^{(1)}, z^{(2)}, z^{(d)}) \in L_1 \times L_2 \times L_d : z^{(2)} \in \langle \infty \rangle \right\}$$
$$\cup \left\{(z^{(1)}, z^{(2)}, z^{(d)}) \in L_1 \times L_2 \times L_d : \exists i \in \mathbb{N}, \exists j \geq i, \exists k_1, k_2 \in \mathbb{N}, \right.$$
$$\left. z^{(1)} = \langle i \rangle_{k_1},\ z^{(2)} = \langle j \rangle_{k_2} \text{ and } \exists \beta \leq \alpha_{i,j},\ z^{(d)}_{p+k_1} = \mathcal{T}(\beta)_{p \bmod i} \right\}.$$

[79] Constructions with subshifts are often defined in terms of "layers" $L1, \dots, L_i$: by this, we mean that the resulting shift is a subshift of the cartesian product of $\prod_{i\in I} L_i$. In this context, we freely use the isomorphism $(\mathcal{A}_1 \times \mathcal{A}_2)^{\mathbb{Z}} \simeq \mathcal{A}_1^{\mathbb{Z}} \times \mathcal{A}_2^{\mathbb{Z}}$ to denote configurations as either tuples of configurations or configurations on tuples of symbols. See Definition 3.10.

Claim 9.5. *Since $\alpha \in \Pi_3$, the subshift Z'_α is an effective $\mathbb{Z}$ subshift.*

Proof. Since the subshift X_* is effective, the conditions on the first two layers are straightforward to enforce[80]. To obtain the subshift Z'_α, we additionally forbid patterns $w = (w^{(1)}, w^{(2)}, w^{(d)})$ of every support $[\![n]\!]$ such that:

- $w^{(1)}$ is a restriction of $\langle i \rangle_{k_1}$ to $[\![n]\!]$ containing at least two symbols $*$;
- $w^{(2)}$ is a restriction of $\langle j \rangle_{k_2}$ to $[\![n]\!]$ (for some $j \geq i$) containing at least two symbols $*$;
- Either $w^{(d)}$ is not i-periodic, or we have $[w^{(d)}|_{[\![i-1]\!]}]_0 \cap \mathcal{T}(D_{i,j}) = \emptyset$, where $D_{i,j} = \{a \in \{0,1\}^{\mathbb{N}} : \sum_{k\in\mathbb{N}} a_k 2^{-(k+1)} \leq \alpha_{i,j}\}$.

Recall that, since $(\alpha_{i,j})_{i,j\in\mathbb{N}^2}$ are a computably enumerable family of Π_1 real numbers, the sets $D_{i,j}$ (and thus $\mathcal{T}(D_{i,j})$ by Proposition 5.4) are effectively closed. In particular, the aforementioned patterns form a computably enumerable family of forbidden patterns. □

[80] Namely: there can be at most a single $*$ symbol on L_2 between two $*$ symbols on L_1.

Let us call *proper* the configurations $z = (z^{(1)}, z^{(2)}, \cdot) \in Z'_\alpha$ such that $z^{(1)} = \langle i \rangle_{k_1}$ and $z^{(2)} = \langle j \rangle_{k_2}$ for some $i, j \in \mathbb{N}$. Conversely, let us call *degenerate* the configurations $z = (z^{(1)}, z^{(2)}, \cdot) \in Z'_\alpha$ such that $z^{(2)} \in \langle \infty \rangle$. From this separation, we distinguish patterns based on the configurations in which they appear: patterns $w \in \mathcal{L}(Z'_\alpha)$ that only appear in degenerate configurations are called *degenerate*; while patterns $w \in \mathcal{L}(Z'_\alpha)$ that can appear in a proper configuration are called *proper*.[81]

The distinction is motivated by the two following claims:

[81] The distinction between proper and degenerate patterns is, unfortunately, a bit complex to clarify. Patterns in which two symbols $*$ appear in the second layer must be proper, since only proper configurations can contain several symbols $*$ on their second (and thus, first) layer. However, the pattern $z|_{[\![n]\!]}$ for a proper configuration z that encodes very large integers $i \in \mathbb{N}$, $j \geq i$ with $i \geq n$ is also proper. Thus, most of the proof will reason on proper configurations $z \in Z'_\alpha$ (and their restrictions to $[\![n]\!]$) instead of patterns.

Claim 9.6. *Denote $D_E(n) = \{E_{Z'_\alpha}(w) : w \in \mathcal{L}_{[\![n]\!]}(Z'_\alpha)$ degenerate$\}$ the extender sets of degenerate patterns of support $[\![n]\!]$. Then $|D_E(n)| = O(n^3)$.*

Proof. Let $u, v \in \mathcal{L}_{[\![n]\!]}(Z'_\alpha)$ be two degenerate patterns such that $u^{(1)} = v^{(1)}$ and $u^{(2)} = v^{(2)}$: then u and v must verify $E_{Z'_\alpha}(u) = E_{Z'_\alpha}(v)$ because there is no restriction on their density layer. Since at most a single $*$ symbol can appear on the second layer of degenerate patterns, and that their first layer is either periodic or contains a single symbol $*$, we obtain $D_E(n) = O(n^3)$. □

Claim 9.7. *For any two distinct proper patterns $u, v \in \mathcal{L}_{[\![n]\!]}(Z'_\alpha)$, we have $E_{Z'_\alpha}(u) \neq E_{Z'_\alpha}(v)$.*

Proof. Let $u \in \mathcal{L}_{[\![n]\!]}(Z'_\alpha)$ be a proper pattern. By definition, u can be extended into some proper configuration $z = (\langle i \rangle_{k_1}, \langle j \rangle_{k_2}, z^{(d)})$ such that $z|_{[\![n]\!]} = u$. By definition, z must be periodic and $i \cdot j$ is a period: thus, $z|_{[\![n]\!]}$ is entirely determined by $z|_{\mathbb{Z}\setminus[\![n]\!]}$: in particular, only u can complete $z|_{\mathbb{Z}\setminus[\![n]\!]}$. □

Yet, there are only polynomially many distinct proper patterns of a given support $[\![n]\!]$ in Z'_α. The next section will nevertheless create a subshift Z_α with the correct (exponential) number of proper patterns, thanks to:

Claim 9.8.

(i) Let $i \in \mathbb{N}$ be an integer and $z \in Z'_\alpha$ be a proper configuration such that $z^{(1)} = \langle i \rangle_{k_1}$. An i-period of the density layer $z^{(d)}$ contains at most $\alpha_i \cdot i + O(1)$ symbols 1.

(ii) Let $i \in \mathbb{N}$ be an integer and $z \in Z'_\alpha$ be a proper configuration such that $z^{(1)} = \langle i \rangle_{k_1}$. Then for any $n \leq i$, a factor of length n of the density layer $z^{(d)}$ contains at most $\alpha_n \cdot n + O(1)$ symbols 1.

Proof. Point (i) follows directly from Claim 5.3. For point (ii), let $z \in Z'_\alpha$ be a proper configuration such that $z^{(1)} = \langle i \rangle_{k_1}$ for some $i \geq n$ and $0 \leq k_1 \leq i$: denoting $w = z|_{[\![n]\!]}$, we might need to apply Claim 5.3 on two patterns u, v

such that $w^{(d)} = u \cdot v$ (depending on the position of $k_1 \in [\![i]\!]$) to obtain that the number of symbols 1 in $w^{(d)}$ is bounded by $\alpha_i \cdot n + O(1)$. We conclude by monotonicity of the sequence $(\alpha_i)_{i\in\mathbb{N}}$. □

9.2.3 Free bits and the subshift Z_α

To create the desired exponential number of extender sets, we create a subshift Z_α by adding *free bits* $\{b, d\}$ on top of the symbols 1 of the density layer of Z'_α. Informally, if there were $\beta \cdot i + O(1)$ symbols 1 in an i-period of the density layer in Z'_α, adding free bits on top of the symbols 1 will yield $2^{\beta\cdot i+O(1)}$ patterns in Z_α. Thus, we add the following fourth layer:

4. **Free layer** L_f: We define the free layer $L_f = \{\text{␣}, b, d\}^{\mathbb{Z}}$. Given the synchronizing map $\pi_s \colon \{\text{␣}, b, d\} \to \{0, 1\}$ defined as $\pi_s(b) = \pi_s(d) = 1$, and $\pi_s(\text{␣}) = 0$, we say that two configurations $z^{(d)} \in L_d$ and $z^{(f)}$ in L_f are synchronized if $\pi_s(z^{(f)}) = z^{(d)}$.

so that

$$Z_\alpha = \left\{ (z^{(1)}, z^{(2)}, z^{(d)}, z^{(f)}) \in L_1 \times L_2 \times L_d \times L_f \colon z^{(2)} \in \langle\infty\rangle \right\}$$
$$\cup \left\{ (z^{(1)}, z^{(2)}, z^{(d)}, z^{(f)}) \in L_1 \times L_2 \times L_d \times L_f \colon \exists i \in \mathbb{N}, \exists j \geq i, \exists k_1, k_2 \in \mathbb{N}, \right.$$
$$z^{(1)} = \langle i \rangle_{k_1},\ z^{(2)} = \langle j \rangle_{k_2},\ \pi_s(z^{(f)}) = z^{(d)},$$
$$\left. \exists \beta \leq \alpha_{i,j},\ {z^{(d)}}_{p+k_1} = \mathcal{T}(\beta)_{p \bmod i} \text{ and } z^{(f)} \text{ is } i\text{-periodic} \right\}.$$

Claim 9.9. *Since $\alpha \in \Pi_3$, the $\mathbb{Z}$ subshift Z_α is effective.*

Proof. This follows from Z'_α being effective: on $Z'_\alpha \times L_f$, we enforce the i-periodicity of the free layer and synchronize it with the density layer whenever the second layer contains at least two symbols $*$. □

Extending the terminology from Z'_α to Z_α, we call *proper* the configurations $z = (z^{(1)}, z^{(2)}, \cdot, \cdot) \in Z_\alpha$ such that $z^{(1)} = \langle i \rangle_{k_1}$ and $z^{(2)} = \langle j \rangle_{k_2}$ for some $i, j \in \mathbb{N}$ with $j \geq i$; and *degenerate* the configuration $(z^{(1)}, z^{(2)}, \cdot, \cdot) \in Z_\alpha$ such that $z^{(2)} \in \langle\infty\rangle$. Similarly, a pattern is *proper* if it can be extended into a proper configuration, and *degenerate* if it only extends to degenerate configurations.

Since the free layer is i-periodic only in the case of proper configurations, Claims 9.6 and 9.7 extends from Z'_α to Z_α by the very same arguments:

Claim 9.10.

(i) *Denote $D_E(n) = \{E_{Z_\alpha}(w) \colon w \in \mathcal{L}_{[\![n]\!]}(Z_\alpha)$ degenerate$\}$. Then we have $|D_E(n)| = O(n^3)$.*
(ii) *Let $u, v \in \mathcal{L}(Z_\alpha)$ be two distinct proper patterns in Z_α. Then we have $E_{Z_\alpha}(u) \neq E_{Z_\alpha}(v)$.*

Thus, we are left with counting the number of proper patterns in Z_α. For $z \in Z_\alpha$ a proper configuration such that $z^{(1)} = \langle i \rangle_{k_1}$, the configuration z is i-periodic; thus, the number of possible assignations of the free layer only depends on the numbers of symbols 1 in an i-period $z^{(d)}|_{[\![i]\!]}$. With this in mind, we can prove the following bounds:

Lemma 9.11. *Let $P(n) = \{w \in \mathcal{L}_{[\![n]\!]}(Z_\alpha) \colon w$ is proper$\}$. Then:*

$$2^{n\cdot\alpha_n+O(1)} \leq P(n) \leq \text{poly}(n) \cdot \sum_{i=1}^{n} 2^{\alpha_i \cdot i+O(1)}.$$

Proof: lower bound. For $n \in \mathbb{N}$ and $j \geq n$, let us consider the configuration $z'_j = (\langle n \rangle_0, \langle j \rangle_0, T(\alpha_{n,j})_{\bmod n}) \in Z'_\alpha$. By Claim 9.8, the number of symbols 1 in the density layer $z'^{(d)}_j|_{[\![n]\!]}$ is at least $\alpha_{n,j} \cdot n + O(1)$; and since $\alpha_{n,j} \to \alpha_n$, there exists some $J \geq n$ such that $z'^{(d)}_J|_{[\![n]\!]}$ contains at least $\alpha_n \cdot n + O(1)$ symbols 1.

Now, consider the patterns $W = \{z|_{[\![n]\!]} : z \in Z_\alpha, (z^{(1)}, z^{(2)}, z^{(d)}) = z'_J\}$. These are all proper patterns, and by definition of the free layer (every symbol 1 in the n-period of $z'^{(d)}_J$ yields two distinct configurations in Z_α) we have:

$$|W| = 2^{\alpha_n \cdot n + O(1)}.$$ □

Proof: upper bound. For $n \in \mathbb{N}$, we overestimate the number of proper patterns $|P(n)|$ by considering restrictions $z'|_{[\![n]\!]}$ for z' ranging in the proper configurations of Z'_α. For fixed $\langle i \rangle_{k_1}$, $\langle j \rangle_{k_2}$ and n-factor of $z'^{(d)}$, we bound the number of symbols 1 in $z'^{(d)}$ by Claim 9.8:

- If $i \leq n$, the density layer of $z'^{(d)}$ is i-periodic, and in an i-period there are less than $\alpha_i \cdot i + O(1)$ symbols 1.
- If $i > n$, $z'|_{[\![n]\!]}$ is a factor of length n of z', so that there are less than $\alpha_n \cdot n + O(1)$ symbols 1 in its density layer $z'^{(d)}|_{[\![n]\!]}$.

Summing over all cases of $\langle i \rangle_{k_1}$ and $\langle j \rangle_{k_2}$ over the domain $[\![n]\!]$, and over all possibilites of $z'^{(d)}|_{[\![n]\!]}$ by Claim 5.5, we obtain:

$$\begin{aligned} P(n) &\leq \sum_{i=1}^{n} \sum_{k_1=0}^{i} \sum_{j=i}^{n} \sum_{k_2=0}^{j} O(i^2) \cdot 2^{\alpha_i \cdot i + O(1)} + \sum_{k_1=0}^{n} \sum_{k_2=0}^{n} O(n^2) \cdot 2^{\alpha_n \cdot n + O(1)} \\ &\leq \text{poly}(n) \cdot \sum_{i=1}^{n} 2^{\alpha_i \cdot i + O(1)}. \end{aligned}$$ □

Combining Lemma 9.11 with Claim 9.10, we deduce that $h_E(Z_\alpha) = \alpha$ by taking the limit over $n \to +\infty$. This concludes the proof.

9.3 Sofic subshifts

In the case of sofic $\mathbb{Z}^d$ subshifts, the possible set of extender entropies depends of the dimension $d \in \mathbb{N}$. In the case $d = 1$, Proposition 6.5 shows that sofic $\mathbb{Z}$ subshifts have finitely many extender sets; thus:

Proposition 9.12. *For $d = 1$, a sofic subshift $Y \subseteq \mathcal{A}^{\mathbb{Z}}$ verifies $h_E(Y) = 0$.*

In dimension $d \geq 2$, we prove a very different result: namely, sofic $\mathbb{Z}^d$ subshifts and effective $\mathbb{Z}^d$ subshifts have the same expressive power in terms of extender entropies.

Theorem 9.13. *For $d \geq 2$, the set of extender entropies of $\mathbb{Z}^d$ sofic subshifts is exactly $[0, +\infty) \cap \Pi_3$.*

As earlier, this statement is a double inclusion. Since sofic subshifts are effective (see Proposition 3.40), Lemma 9.3 yields:

Lemma 9.14. *Let $Y \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a sofic subshift. Then $h_E(Y) \in [0, +\infty) \cap \Pi_3$.*

The rest of this section is dedicated to the converse statement. By Corollary 8.21, we reduce to the case of $\mathbb{Z}^2$ subshifts and are left with proving the following:

Lemma 9.15. *For any real number $\alpha \in [0, +\infty) \cap \Pi_3$, there exists a sofic $\mathbb{Z}^2$ subshift Y_α such that $h_E(Y_\alpha) = \alpha$.*

The following sections are dedicated to proving Lemma 9.15 by providing an explicit construction of such a $\mathbb{Z}^2$ sofic subshift Y_α. Similarly to the effective case on $\mathbb{Z}$, let us fix $\alpha \in \Pi_3$ (we reduce to the case $0 \leq \alpha \leq 1$, since extender entropy is additive under cartesian product by Proposition 8.23), and fix a presentation $(\alpha_{i,j})_{i,j\in\mathbb{N}^2}$ of α, that is a recursively enumerable family of Π_1 real numbers such that $\alpha = \inf_{i\in\mathbb{N}} \sup_{j\in\mathbb{N}} \alpha_{i,j}$. We assume that $0 \leq \alpha_{i,j} \leq 1$ for all $i, j \in \mathbb{N}$, and by Proposition 4.20, we assume that the family $(\alpha_{i,j})_{i,j\in\mathbb{N}^2}$ satisfies some monotonicity properties: for all $i \in \mathbb{N}$, $(a_{i,j})_{j\in\mathbb{N}}$ is weakly increasing towards some $\alpha_i \in [0,1] \cap \Sigma_2$; and the sequence $(\alpha_i)_{i\in\mathbb{N}}$ is weakly decreasing towards α.

9.3.1 The subshift Y_α^f: lifting the previous construction

Before we begin the construction itself, it is interesting to consider the naive generalization of our $\mathbb{Z}$ construction to $\mathbb{Z}^2$. By lifting the subshift Z'_α of the previous proof periodically from $\mathbb{Z}$ to $\mathbb{Z}^2$ and adding free bits, we obtain the subshift[82]:

$$Y_\alpha^f = \{(y^{(1)\Uparrow}, y^{(2)\Uparrow}, y^{(d)\Uparrow}, y^{(f)}) \in L_1^{\Uparrow} \times L_2^{\Uparrow} \times L_d^{\Uparrow} \times \{\text{␣}, b, d\}^{\mathbb{Z}^2} : \pi_s(y^{(f)}) = y^{(d)\Uparrow}\}.$$

[82] We abuse notations and consider $Z'_\alpha \times \{0,1\}^{\mathbb{Z}^2}$ as a subshift on four layers by the classical isomorphism $(S_1 \times \cdots \times S_i) \times S_{i+1} \simeq S_1 \times \cdots \times S_{i+1}$.

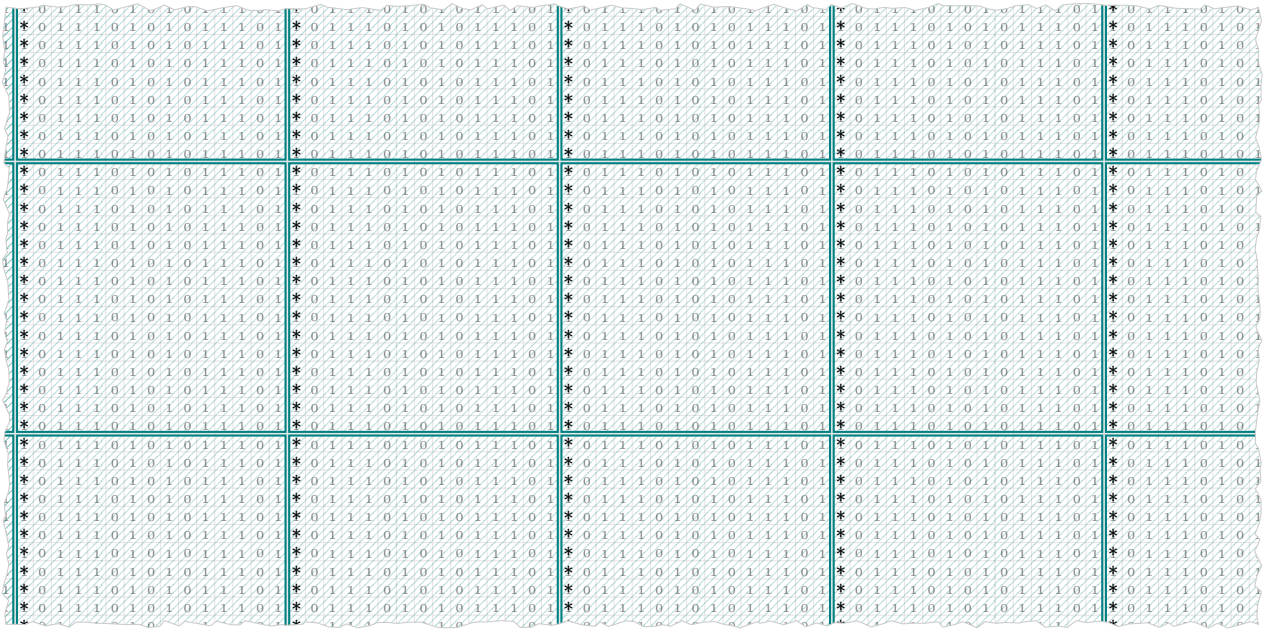


Figure 9.2: Sketch of a *proper* configuration of Y_α^f: $z^{(1)} = \langle 15 \rangle_{11}$ and $z^{(d)}$ is obtained from the Toeplitz encoding $\mathcal{T}(\beta)$ for $\beta = \frac{5}{8} = 0.101_{(2)}$. We highlight with colored hatched squares the areas of fundamental domain $[\![15]\!]^2$ which periodize the free bits.

The arguments of the previous proof generalize so that enforcing proper configurations to have a strongly periodic free layer

$$\{y \in Y_\alpha^f : y^{(1)} = \langle i \rangle_{k_1} \wedge y^{(2)} = \langle j \rangle_{k_2} \implies y^{(f)} \text{ is } [\![i]\!]^2\text{-biperiodic}\}$$

creates a $\mathbb{Z}^2$ subshift of extender entropy α. Unfortunately, such a subshift cannot be sofic if $\alpha > 0$ by the counting argument presented in Proposition 7.1: among proper configurations, there exists $2^{\alpha \cdot i^2 + O(i)}$ distinct patterns of domain $[\![i]\!]^2$ and only $2^{O(i)}$ borders of such patterns; yet, exchanging patterns is impossible because of the periodicity of the free layer.

Thus, let us consider once again a proper configuration $y' \in Y_\alpha^f$ such that $y^{(1)} = \langle i \rangle_{k_1}$. Instead of relying on the periodicity of the free layer, we introduce in the next section another construction to generate distinct extender sets for distinct possible sets of free bits completing y' over the domain $[\![i]\!]^2$.

9.3.2 Marking bits and positions in configurations

To solve the aforementioned issue, we borrow the main idea of Example 8.5: instead of periodizing the whole square of domain $[\![i]\!]^2$ to enforce distinct extender sets for proper configurations $y \in Y_\alpha^f$ such that $y^{(1)} = \langle i \rangle_{k_1}$, it is enough to non-deterministically pick and periodize *one position* inside the square $[\![i]\!]^2$.

More formally, consider the alphabet $\{\text{␣}, \diamond\}$. We denote by $[m_1, m_2]$ the configuration containing a single symbol $\diamond$ at position $(m_1, m_2) \in \mathbb{Z}^2$, and symbols ␣ at every other position $p \neq (m_1, m_2)$. We say that the symbol $\diamond$ is the *marker* of the configuration.

Given a configuration $x = [m_1, m_2]$ and an integer $i \in \mathbb{N}$, we say that a position $p \in \mathbb{Z}^2$ is *marked* if $p \in (m_1 + i\mathbb{Z}, m_2 + i\mathbb{Z})$. Notice, in particular, that many marked positions $p \in \mathbb{Z}^2$ satisfy $x_p = \text{␣}$.

The subshift generated by all configurations $[m_1, m_2]$ is:

$$Y_\diamond = \{[m_1, m_2] : (m_1, m_2) \in \mathbb{Z}^2\} \cup [\infty],$$

where $[\infty] = \text{␣}^{\mathbb{Z}^2}$ is the whole blank configuration, which appears in $Y_\diamond$ as the limit configuration of all the $[m_1, m_2]$.

Figure 9.3: A typical configuration of $Y_\diamond$.

9.3.3 The subshift Y_α

Considering the following five layers:

- **Lifted layers**: we define the first, second and density layers to be respectively $L_1^{\Uparrow}$, $L_2^{\Uparrow}$ and $L_d^{\Uparrow}$, where L_1, L_2 and L_d are the first, second and density layers of the subshift Z'_α defined in the previous proof;
- **Marker layer**: we set $L_m = Y_\diamond$ to mark positions $p \in (m_1 + i\mathbb{Z}, m_2 + i\mathbb{Z})$;
- **Free layer**: we define $L_f = \{\text{␣}, b, d\}^{\mathbb{Z}^2}$ and still use the synchronizing map $\pi_s : \{\text{␣}, b, d\} \to \{0, 1\}$ of the previous proof;

we define the subshift Y_α as:

$$Y_\alpha = \left\{ (z^{(1)\Uparrow}, z^{(2)\Uparrow}, z^{(d)\Uparrow}, y^{(m)}, y^{(f)}) \in L_1^{\Uparrow} \times L_2^{\Uparrow} \times L_d^{\Uparrow} \times Y_\diamond \times \{\text{␣}, b, d\}^{\mathbb{Z}^2} : \right.$$
$$\left. z^{(2)} \in \langle \infty \rangle,\ \pi_s(y^{(f)}) = z^{(d)\Uparrow} \right\}$$
$$\cup \left\{ (z^{(1)\Uparrow}, z^{(2)\Uparrow}, z^{(d)\Uparrow}, y^{(m)}, y^{(f)}) \in L_1^{\Uparrow} \times L_2^{\Uparrow} \times L_d^{\Uparrow} \times Y_\diamond \times \{\text{␣}, b, d\}^{\mathbb{Z}^2} : \exists i \in \mathbb{N}, \exists j \geq i, \exists k_1, k_2 \right.$$
$$z^{(1)} = \langle i \rangle_{k_1},\ z^{(2)} = \langle j \rangle_{k_2},\ \pi_s(y^{(f)}) = z^{(d)\Uparrow},\ \exists \beta \leq \alpha_{i,j} :\ {z^{(d)}}_{p+k_1} = T(\beta)_{p \bmod i}$$
$$\left. \text{and } \forall (m_1, m_2) \in \mathbb{Z}^2, (y^{(m)} = [m_1, m_2] \iff y^{(f)}|_{(m_1+i\mathbb{Z})\times(m_2+i\mathbb{Z})} \text{ is constant}) \right\}.$$

The main difference between Y_α^f and Y_α is that Y_α does not periodize whole $[\![i]\!]^2$ squares, but only free bits at positions marked by the marker layer.

Once again, we call *proper* the configurations $y = (z^{(1)\Uparrow}, z^{(2)\Uparrow}, \cdot, \cdot, \cdot) \in Y_\alpha$ such that $z^{(1)} = \langle i \rangle_{k_1}$ and $z^{(2)} = \langle j \rangle_{k_2}$ for some $i, j \in \mathbb{N}$ with $j \geq i$; and *degenerate* the configurations $(z^{(1)\Uparrow}, z^{(2)\Uparrow}, \cdot, \cdot, \cdot) \in Y_\alpha$ such that $y^{(2)} \in \langle \infty \rangle$. Among patterns of $\mathcal{L}(Y_\alpha)$, we call *proper* the patterns that can be extended into a proper configurations; and *degenerate* the patterns that can only be extended into degenerate configurations. Finally, we say that two patterns $w, w' \in \mathcal{L}_D(Y_\alpha)$ over the same domain $D \subseteq \mathbb{Z}^2$ are *similar* if they are equal on their first four layers, *i.e.* $\pi_{L_1^{\Uparrow} \times L_2^{\Uparrow} \times L_d^{\Uparrow} \times L_m}(w) = \pi_{L_1^{\Uparrow} \times L_2^{\Uparrow} \times L_d^{\Uparrow} \times L_m}(w')$.

Claim 9.16. *Denote $D_E(n_1, n_2) = \{E_{Y_\alpha}(w) : w \in \mathcal{L}_{[\![n_1,n_2]\!]}(Y_\alpha)$ degenerate$\}$ the extender sets of degenerate patterns of support $[\![n_1, n_2]\!]$. Then, we have $|D_E(n_1, n_2)| = O(n_1^3 \cdot n_2^2)$.*

Proof. As there is no restriction on the density (nor the free) layer of degenerate configurations, any two similar degenerate patterns $w, w' \in \mathcal{L}_{[\![n_1,n_2]\!]}(Y_\alpha)$ verify $E_{Y_\alpha}(w) = E_{Y_\alpha}(w')$. We are left with counting the number of possibilites for the first, second, and marker layer of such patterns. □

Unfortunately, an analog of Claim 9.10 does not hold here, since not all proper patterns generate distinct extender sets. Instead, we have:

Claim 9.17. *Two similar proper patterns $w, w' \in \mathcal{L}_{[\![n_1,n_2]\!]}(Y_\alpha)$ have distinct extender sets if and only if there exists a proper configuration y that extends w and whose marker layer $y^{(m)}$ marks a position $p \in [\![n_1, n_2]\!]$ such that $w_p^{(f)} \neq w_p'^{(f)}$.*

Proof. Let $w \in \mathcal{L}_{[\![n_1,n_2]\!]}(Y_\alpha)$ be a proper pattern, and let y be a proper configuration that extends w with $y^{(m)} = [m_1, m_2]$ for $m_1, m_2 \in \mathbb{Z}$. Let us denote $y^{(1)} = \langle i \rangle_{k_1}^{\Uparrow}$. Since $y^{(f)}|_{(m_1+i\mathbb{Z})\times(m_2+i\mathbb{Z})}$ is constant, the value of $w^{(f)}$ in positions marked by $y^{(m)}$ is entirely determined by $y|_{\mathbb{Z}^2\setminus[\![n_1,n_2]\!]}$: in particular, any pattern w' such that $y|_{\mathbb{Z}^2\setminus[\![n_1,n_2]\!]} \in E_{Y_\alpha}(w')$ must verify $w_p'^{(f)} = w_p^{(f)}$ if p is marked in $y^{(m)}$. □

We will not explicitely state the conditions on which a position can be marked in an extending configuration of a given proper pattern (such conditions can be quite complicated: for example, if two symbols $*$ appear on the first layer, then i is known and marked positions are among the i-periodic free bits...).

Instead, we will just mention the following fact: in a given proper configuration y such that $y^{(1)} = \langle i \rangle_{k_1}^{\Uparrow}$ and $y^{(m)} \neq [\infty]$, exactly one position per $i \times i$ square is marked. Thus, when fixing the first four layers of a configuration and ranging in the possible assigments of its free layer, there are i^2 distinct sets of marked positions, and the number of extender sets they generate depends only on the number of symbols 1 in $y^{(d)}|_{[\![i\times i]\!]}$.

With this in mind, we can prove the following bounds (that share a striking resemblance with the effective $\mathbb{Z}$ case):

Lemma 9.18. *Let $P_E(n_1, n_2) = \{E_{Y_\alpha}(w) : w \in \mathcal{L}_{[\![n_1,n_2]\!]}(Y_\alpha)$ proper$\}$. Then:*[83]

$$2^{\alpha_{\max(n_1,n_2)}\cdot n_1 n_2 + O(n_2)} \leq P_E(n_1, n_2) \leq \text{poly}(n_1, n_2) \cdot \sum_{i=1}^{n_1} 2^{\alpha_i \cdot i n_2 + O(n_2)}$$

[83] With our defintions, if $p = (p_1, p_2) \in \mathbb{Z}^2$, then p_1 is the abscissa and p_2 the ordinate of p, *i.e.* its horizontal and vertical coordinates respectively.

Proof: lower bound. Let us fix $n_1, n_2 \in \mathbb{N}$, and denote $N = \max(n_1, n_2)$. Following the proof of the effective $\mathbb{Z}$ case, there exists a configuration $z' = (\langle N \rangle_0, z'^{(2)}, z'^{(d)}) \in Z'_\alpha$ such that the number of symbols 1 in the density layer $z'^{(d)}|_{[\![n_1]\!]}$ is at least $\alpha_N \cdot n_1 + O(1)$. Let us consider the patterns:

$$W = \Big\{ y|_{[\![n_1,n_2]\!]} : y \in Y_\alpha,\ (y^{(1)}, y^{(2)}, y^{(d)}) = (z'^{(1)}, z'^{(2)}, z'^{(d)})^{\Uparrow} \\ \text{and } y^{(m)} = [m_1, m_2] \text{ for } (m_1, m_2) \notin [\![n_1, n_2]\!] \Big\} :$$

these are all similar proper patterns. For $w, w' \in W$, there exists some position $p = (p_1, p_2) \in [\![n_1, n_2]\!]$ such that $w_p^{(f)} \neq w_p'^{(f)}$. Since the configuration $y_\diamond = [N + p1, N + p2]$ marks (p_1, p_2) (and only[84] (p_1, p_2) in $[\![n_1, n_2]\!]$) and verifies $y_\diamond|_{[\![n_1,n_2]\!]} = \text{␣}^{[\![n_1,n_2]\!]}$, there exists a proper configuration that marks p and extends w: thus, w and w' have distinct extender sets. □

[84] It is actually important that only (p_1, p_2) is marked in $[\![n_1, n_2]\!]$. Indeed, if $(m_1 + N\mathbb{Z}, m_2 + n\mathbb{Z})$ intersects $[\![n_1, n_2]\!]$ in two positions, then these positions can only be marked if they bear the same free bit in $w^{(f)}$.

Proof: upper bound. We proceed as in Lemma 9.11: for a fixed domain $[\![n_1, n_2]\!]$, we overestimate the number of extenders of proper patterns by considering restrictions $y|_{[\![n_1,n_2]\!]}$ for y ranging the in the proper configurations of Y_α. For fixed $y^{(1)} = \langle i \rangle_{k_1}^{\Uparrow}$, $y^{(2)}$ and $y^{(d)}$, let us bound the number of extender sets generated by all the assigments of free bits.

As mentioned above, these only depend on the number of symbols 1 in $y^{(d)}|_{[\![i,i]\!]}$. Thus, we need to consider how $[\![i, i]\!]$ and $[\![n_1, n_2]\!]$ intersect. If $[\![n_1, n_2]\!] \subseteq [\![i, i]\!]$, then every position could be marked by a well-chosen extending configuration. If $[\![i, i]\!]$ is contained within $[\![n_1, n_2]\!]$, only a subset of positions of $[\![i, i]\!]$ could realistically be marked[85]. Instead of pursuing this

[85] Among many other conditions, marked positions must be among the period positions of $y^{(f)}|_{[\![n_1,n_2]\!]}$...

case disjunction any further, we avoid the difficulty entirely by generously overestimating which positions can be marked: we will consider that all the free bits assignments over $[\![i, n_2]\!]$ generate distinct extender sets. Since by Claim 9.7,

- If $i \leq n_1$, the restriction of the density layer $y^{(d)}$ to the rectangle $[\![i, n_1]\!]$ contains less than $\alpha_i \cdot i n_2 + O(n_2)$ symbols 1.
- If $i \geq n_1$, the restriction of the density layer $y^{(d)}$ to the rectangle $[\![n_1, n_2]\!]$ contains less than $\alpha_{n_1} \cdot n_1 n_2 + O(n_2)$ symbols 1;

when summing over all cases of $\langle i \rangle_{k_1}^{\Uparrow}$, $\langle j \rangle_{k_2}^{\Uparrow}$, all possibilities of $y^{(d)}$ by Claim 5.5, not forgetting to account for the possible presence of $\diamond$ in $y^{(d)}|_{[\![n_1,n_2]\!]}$, and generously overestimating which positions can be marked, we conclude that:

$$\begin{aligned} P_E(n_1, n_2) &\leq \sum_{i=1}^{n_1} \sum_{k_1=0}^{i} \sum_{j=i}^{n_1} \sum_{k_2=0}^{j} O(i^2) \cdot (O(n_1 n_2) + 2^{\alpha_i \cdot i n_2 + O(n_2)}) \\ &\qquad + \sum_{k_1=0}^{n_1} \sum_{k_2=0}^{n_1} O(n^2) \cdot (O(n_1, n_2) + 2^{\alpha_{n_1} \cdot n_1 n_2 + O(n_2)}) \\ &\leq \text{poly}(n_1, n_2) \sum_{i=1}^{n_1} 2^{\alpha_i \cdot i n_2 + O(n_2)} \end{aligned}$$

□

Combining Lemma 9.18 with Claims 9.16 and 9.17, we deduce that $h_E(Y_\alpha) = \alpha$ by taking the limit over $n_1 = n_2 = n \to +\infty$. Thus, we are left with proving:

Claim 9.19. *The $\mathbb{Z}^2$ subshift Y_α is a sofic subshift.*

The proof is very standard and unsurprising. We include here a sketch of the argument for the sake of completeness:

Sketch of proof. Lifting the $\mathbb{Z}$ effective subshift[86] Z'_α to $\mathbb{Z}^2$ and following Proposition 3.44, we nearly obtain that Y_α is a sofic subshift. The only remaining hiccup lies in periodizing free bits in proper configurations, and in proper configurations *only*.

To periodize free bits at coordinates $(m_1 + i\mathbb{Z}, m_2 + i\mathbb{Z})$, we introduce the $\mathbb{Z}^2$ *grid subshift* Y_{grid} on the alphabet $\{\boxplus, \boxminus, \boxvert\}$, which is defined as the closure of all the square grid configurations (see Figure 9.4). The subshift Y_{grid} is sofic[87]. To synchronise it with the first layer $L_1^{\Uparrow}$, we define $Y_{\text{grid},*} \subseteq L_1^{\Uparrow} \times Y_{\text{grid}}$ the set of configurations $(x^{(1),\Uparrow}, x^{(g)})$ such that $x^{(1)} = \langle i \rangle_{k_1}$ if and only if $x^{(g)}$ has mesh $i \times i$ (see Figure 9.5). The subshift $Y_{\text{grid},*}$ is also sofic[88].

Now, let us prove that Y_α is a sofic subshift. We begin by adding a **Proper Layer** to the $\mathbb{Z}$ effective subshift Z'_α, which is set to be $\mathsf{p}^{\mathbb{Z}}$ in *proper configurations* of Z'_α, and can be either $\mathsf{d}^{\mathbb{Z}}$ or $\mathsf{p}^{\mathbb{Z}}$ in *degenerate configurations*. Denoting by Z''_α the resulting $\mathbb{Z}$ effective subshift, we use Proposition 3.44 to lift it sofically to $\mathbb{Z}^2$ and define:

$$\begin{aligned} Y'_\alpha = \Big\{ &(z^{(1)\Uparrow}, z^{(2)\Uparrow}, z^{(d)\Uparrow}, z^{(p)\Uparrow}, y^{(\diamond)}, y^{(g)}, y^{(f)}) \in Z''^{\Uparrow}_\alpha \times Y_\diamond \times Y_{\text{grid}} \times \{0, 1, \sqcup\}^{\mathbb{Z}^2} : \\ &(z^{(1)\Uparrow}, y^{(g)}) \in Y_{\text{grid},*},\ \pi_s(y^{(f)}) = z^{(d)\Uparrow}, \\ &\forall \mathbf{p} \in \mathbb{Z}^2,\ y^{(\diamond)}_{\mathbf{p}} = \diamond \implies y^{(g)}_{\mathbf{p}} = \boxplus, \\ &\text{and } \exists b \in \{0, 1, \sqcup\},\ \forall \mathbf{p} \in \mathbb{Z}^2,\ z^{(p)} = \mathsf{p}^{\mathbb{Z}} \wedge y^{(g)}_{\mathbf{p}} = \boxplus \implies y^{(f)}_{\mathbf{p}} = b \Big\} \end{aligned}$$

In other words, in Y'_α, the *marker* of the marker layer must appear on top of a cross $\boxplus$ of the grid layer; and free bits along the grid are periodized *only when* the proper layer contains the symbol p.

We then claim that Y_α is a factor of Y'_α by just erasing the unecessary layers: indeed, we just have to check the periodicity of the free bits. In Y'_α, both

[86] The subshift Z'_α defined in the proof of Lemma 9.4.

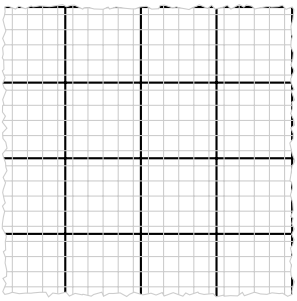


Figure 9.4: A typical configuration of the square grid subshift Y_{grid}.

[87] The tiles $\boxplus$, $\boxminus$ and $\boxvert$ define, by enforcing the continuity of black lines, a local subshift whose configurations are irregular grids; to enforce regular square grids, we make each cross $\boxplus$ send draw diagonals in an SFT cover (since diagonals can only cross black lines through other crosses $\boxplus$, the grids become regular).

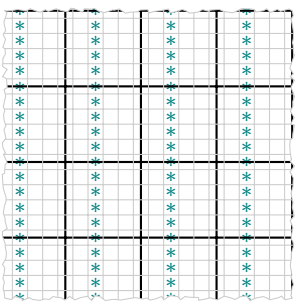


Figure 9.5: A typical configuration of the subshift $Y_{\text{grid},*}$.

[88] Indeed, force that exactly a single symbol $*$ can appear between two $\boxvert$ columns.

cases $z^{(p)} = \mathsf{p}^{\mathbb{Z}}$ and $z^{(p)} = \mathsf{d}^{\mathbb{Z}}$ are possible when $z^{(2)} \in \langle\infty\rangle$, so that, when projecting, the periodicity condition enforced on some free bits of $y^{(f)}$ is lost. On the other hand, if $z^1 = \langle i\rangle_{k_1}$ and $z^{(2)} = \langle j\rangle_{k_2}$, then $y^{(p)}$ is forced to be $\mathsf{p}^{\mathbb{Z}}$, so that the periodicity condition is kept when projecting. Since Y'_α is sofic, this concludes the proof. □

9.4 Computable subshifts

Theorem 9.20. *For $d \geq 1$, the set of extender entropies of $\mathbb{Z}^d$ computable subshifts is exactly $[0, +\infty) \cap \Pi_2$.*

Since computable subshifts have Π_0^0 language by definition, we obtain the first inclusion:

Lemma 9.21. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be computable. Then $h_E(X) \in [0, +\infty) \cap \Pi_2$.*

Proof. By definition, X being computable implies that $\mathcal{L}(X)$ is a Π_0^0 set. Following the argument of Lemma 9.3, we conclude that $h_E(X) \in \Pi_2$. □

Lemma 9.22. *Let $\alpha \in [0, +\infty) \cap \Pi_2$. There exists a $\mathbb{Z}^d$ computable subshift X_α such that $h_E(X_\alpha) = \alpha$. If $d \geq 2$, X_α can even be taken sofic.*

Proof. Fix $\alpha \in [0, +\infty) \cap \Pi_2$ defined by $\alpha = \inf_{i\in\mathbb{N}} \sup_{j\in\mathbb{N}} \alpha_{i,j}$ for $(\alpha_{i,j})_{(i,j)\in\mathbb{N}^2}$ a computable sequence of rational numbers. Considering the subshifts X_α and Y_α that were respectively constructed in the proofs of Lemma 9.4 and Lemma 9.15, we claim that, since $\alpha \in \Pi_2$, the subshifts Z_α and Y_α are computable subshifts.

Indeed, the density layer of Z_α (resp. Y_α) is made of factors of Toeplitz words $T(\beta)$ for $\beta \leq \alpha_{i,j}$: if $\alpha_{i,j}$ is rational, the sets $\{T(\beta) : \beta \leq \alpha_{i,j}\}$ are all uniformly decidable, so that the density layers of Z_α and Y_α (and thus, the subshifts themselves) are in turn computable. □

9.5 Minimal subshifts

In the case of minimal subshift, Proposition 8.13 shows that distinct patterns yield distinct extender sets. Thus:

Lemma 9.23. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a minimal subshift. Then $h_E(X) = h(X)$.*

Proof. By Proposition 8.13, we know that there is exactly one extender set per pattern in X, *i.e* that $|E_X([\![n]\!]^d)| = |\mathcal{L}_{[\![n]\!]^d}(X)|$. By taking the limit, we conclude that $h_E(X) = h(X)$. □

In terms of extender entropy of minimal subshifts, we deduce:

Proposition 9.24. *Let $Y \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a minimal sofic subshift. Then $h_E(Y) = 0$.*

Proposition 9.25. *For $d \geq 1$, the set of extender entropies of minimal $\mathbb{Z}^d$ effective subshifts is $[0, +\infty) \cap \Pi_1$.*

Indeed:

- Minimal sofic subshifts have entropy zero (Proposition 3.49).
- Effective subshifts have Π_1 entropy; conversely, for every real number $\alpha \in [0, +\infty) \cap \Pi_1$, there exists a minimal effective subshift Z_α such that $h(Z_\alpha) = \alpha$: for example, take the subshift from the proof of [Kůr03, Theorem 4.77] with a computable sequence of integers $(k_n)_{n\in\mathbb{N}}$.

[Kůr03] Kůrka, *Topological and symbolic dynamics*.

9.6 Mixing subshifts

All our previous examples rely on very non-mixing conditions to generate exponentially many extender sets. We prove that this is, in fact, not a restriction.

Theorem 9.26. *For $d \geq 1$, the set of extender entropies of 1-block-gluing effective $\mathbb{Z}^d$ subshifts is $[0, +\infty) \cap \Pi_3$.*

This is a consequence of the following lemma: adding a safe-symbol # to the alphabet $\mathcal{A}$ creates mixingness and preserves extender entropies. More precisely, let $\mathcal{R}^{\circledast d}$ denote the set of (potentially infinite) hyperrectangles R such that $R \subseteq \mathbb{Z}^d$. Given a subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$, for $\mathcal{A}_\# = \mathcal{A} \cup \{\#\}$ we define the subshift $X_\# \subseteq \mathcal{A}_\#^{\mathbb{Z}^d}$ as the set of (potentially infinite) non-contiguous rectangular patterns of X over a background of symbols #:

$$X_\# = \{x \in \mathcal{A}_\#^{\mathbb{Z}^d} : \exists J, \exists (R_j)_{j\in J} \in (\mathcal{R}^{\circledast d})^J,\ d(R_{j_1}, R_{j_2}) \geq 1 \text{ if } j_1 \neq j_2,$$
$$\bigsqcup_{j\in J} R_j = \{\mathbf{i} \in \mathbb{Z}^d : x_\mathbf{i} \neq \#\} \text{ and } \forall j \in J,\ x|_{R_j} \in \mathcal{L}(X)\}.$$

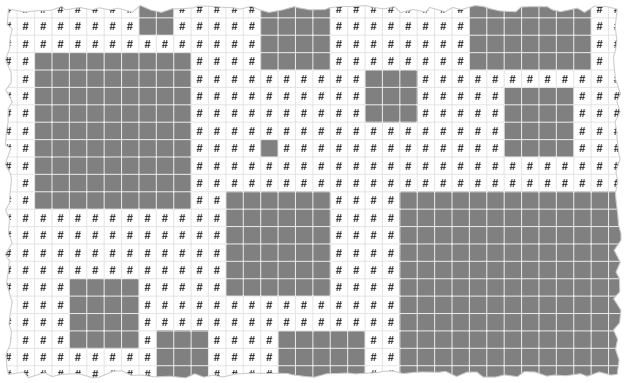


Figure 9.6: Layout of a configuration in $X_\#$: independant rectangles float over a background of # symbols.

Lemma 9.27. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be an effective subshift. The subshift $X_\# \subseteq \mathcal{A}_\#^{\mathbb{Z}^d}$ defined above is 1-block-gluing, effective, and verifies $h_E(X_\#) = h_E(X)$.*

Proof. Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$. The subshift $X_\#$ is effective since the set of finite patterns in $\mathcal{L}(X)$ is Π_1^0 (see Proposition 3.45). It is 1-block-gluing, since for any two rectangles $R, R' \in \mathcal{R}^{\circledast d}$ such that $d(R, R') \geq 1$ and any two patterns $w \in \mathcal{L}_R(X_\#)$, $w' \in \mathcal{L}_{R'}(X_\#)$, the configuration $x \in \mathcal{A}_\#^{\mathbb{Z}^d}$ defined as follows is valid in $X_\#$: $x_\mathbf{i} = w_\mathbf{i}$ if $\mathbf{i} \in R$, $x_\mathbf{i} = w'_\mathbf{i}$ if $i \in R'$, or $x_\mathbf{i} = \#$ otherwise. To complete this proof, we are left with computing the extender sets of $X_\#$.

For a fixed domain $D \subseteq \mathbb{Z}^d$, assume that two patterns $w, w' \in \mathcal{A}^D$ verify $E_X(w) \neq E_X(w')$: by definition, there exists some partial configuration $x \in \mathcal{A}^{\mathbb{Z}^d \setminus D}$ that extends either w or w' but not the other. Since the same configuration still exists in $X_\#$, and still extends only one pattern among w and w', we obtain $E_{E_\#}(w) \neq E_X(E_\#)$. In particular, $h_E(X) \leq h_E(X_\#)$.

Reciprocally, let us consider the extender sets of patterns in $X_\#$. Given a domain $[\![n]\!]^d$ and a pattern $w \in \mathcal{L}_{[\![n]\!]^d}(X_\#)$, we define the *geometry* of w (denoted $G(w)$) as the set of maximal non-adjacent hyperrectangles $R \in \mathcal{R}^{\circledast d}$ (*i.e.* $d(R, R') \geq 1$ for distinct $R, R' \in G(w)$) that cover the non-# symbols of w (*i.e.* $\sqcup_{R\in G(w)} R = \{\mathbf{i} \in [\![n]\!]^d : w_\mathbf{i} \neq \#\}$). Considering the *border geometry* $\partial G(w)$ defined by

$$\partial G(w) = \{R \in G(w) : R \cap \partial_1([\![n]\!]^d) \neq \emptyset\},$$

the extender set $E_{X_\#}(w)$ in $X_\#$ is entirely determined by the border geometry $\partial G(w)$ and the extender sets $\{E_X(w|_R) \colon R \in \partial G(w)\}$ in X.

From these considerations, we bound the number of extender sets in $X_\#$. By definition of the extender entropy of X, for $\varepsilon > 0$, there exists some $N \in \mathbb{N}^d$ such that, for all $n \in \mathbb{N}$,

$$(n_1, \dots, n_d) \geq (N_1, \dots, N_d) \implies \log |E_X(\llbracket n_1, \dots, n_d \rrbracket)| \leq (h(X) + \varepsilon)\, n_1 \cdots n_d.$$

Thus, for any hyperrectangle $R \subseteq \llbracket n \rrbracket^d$ of size $r_1 \times \cdots \times r_d$ (and denoting $\|R\| = r_1 \cdots r_d$ its volume), two cases arise:

- Either $r_i \geq N_i$ for all $1 \leq i \leq d$ (we denote $\mathcal{R}_L$ this set of *large hyperrectangles*): in this case, we bound the number of extender sets as follows: $\log |E_X(R)| \leq (h(X) + \varepsilon)\, \|R\|$.
- Or there exists some $1 \leq i \leq d$ such that $r_i < N_i$ (we denote $\mathcal{R}_S$ this set of *small hyperrectangles*): in this case, we bound $\log |E_X|(R)$ with $\log |E_X|(R) \leq \log |\mathcal{A}_\#| \cdot \|R\|$.

For a fixed border geometry ∂G, we have:

$$\begin{aligned} \prod_{R \in \partial G \cap \mathcal{R}_S} |E_\#(R)| &\leq |\mathcal{A}_\#|^{\sum_{R \in \partial G \cap \mathcal{R}_L} \|R\|} \\ &\leq |\mathcal{A}_\#|^{\sum_{i=1}^d 2 \cdot n^{d-1} \cdot N_i}; \end{aligned}$$

$$\begin{aligned} \prod_{R \in \partial G \cap \mathcal{R}_L} |E_\#(R)| &\leq 2^{(h(X)+\varepsilon) \cdot \sum_{R \in \partial G \cap \mathcal{R}_L} \|R\|} \\ &\leq 2^{(h(X)+\varepsilon) \cdot n^d}. \end{aligned}$$

Summing over all geometries, we obtain:

$$\begin{aligned} |E_X(\llbracket n_1, \dots, n_d \rrbracket)| &\leq \sum_{\substack{\partial G = \partial G(w) \colon \\ w \in \mathcal{L}_{\llbracket n_1, \dots, n_d \rrbracket}(X_\#)}} \prod_{R \in \partial G \cap \mathcal{R}_S} |E_\#(R)| \cdot \prod_{R \in \partial G \cap \mathcal{R}_L} |E_\#(R)| \\ &\leq \sum_{\substack{\partial G = \partial G(w) \colon \\ w \in \mathcal{L}_{\llbracket n_1, \dots, n_d \rrbracket}(X_\#)}} |\mathcal{A}_\#|^{\sum_{i=1}^d 2 \cdot n^{d-1} \cdot N_i} \cdot 2^{(h(X)+\varepsilon) \cdot n^d}. \end{aligned}$$

For a given pattern $w \in \mathcal{L}_{\llbracket n \rrbracket^d}(X_\#)$, the number of rectangles in $\partial G(w)$ is bounded by the hypersurface of $\llbracket n \rrbracket^d$, which is $2d \cdot n^{d-1}$. Furthermore, a d-dimensional hyperrectangle in $\llbracket n \rrbracket^d$ is entirely determined by two points. Thus, the number of possible border geometries is bounded by:

$$|\{\partial G(w) \colon w \in \mathcal{L}_{\llbracket n \rrbracket^d}(X_\#)\}| \leq (O(n^{2d}))^{2d \cdot n^{d-1}} \leq 2^{O(n^{d-1} \cdot \log n)}.$$

Plugging all these equations together leads to:

$$\log |E_{X_\#}(\llbracket n \rrbracket^d)| \leq (h(X) + \varepsilon) \cdot n^d + O(n^{d-1} \cdot \log n)$$

By Proposition 8.16, $h_E(X_\#)$ can be computed using any sequence of hyperrectangles. In particular, dividing by n^d in the previous inequality and taking the limit on n, we obtain that $h_E(X_\#) \leq h(X) + \varepsilon$. Since ε was arbitrary, we conclude the proof:

$$h_E(X_\#) \leq h(X). \qquad \square$$

With a slightly more involved construction, one can prove that 1-block-gluing $\mathbb{Z}^d$ sofic subshifts realize all the sofic extender entropies:

Theorem 9.28. *For $d \geq 2$, the set of extender entropies of 1-block-gluing $\mathbb{Z}^d$ sofic subshifts is $[0, +\infty) \cap \Pi_3$.*

Sketch of proof. Let us fix $\alpha \in [0, +\infty) \cap \Pi_3$, and denote by $Y = Y_\alpha \subseteq \mathcal{A}^{\mathbb{Z}^2}$ the subshift that was built in the proof of Lemma 9.15, but restricted to

[89] Y is still sofic, has the same extender entropy since it can be computed along any sequence of hyperrectangles; but now has a reduced "density" of information.

configurations $\langle i \rangle_{k_1}$ and $\langle j \rangle_{k_2}$ that are powers of 2.[89] Denoting $\mathcal{C}^{*d} \subseteq \mathcal{R}^{\circledast d}$ the set of finite hypercubes $C \subseteq \mathbb{Z}^d$, define:

$$Y'_{\#} = \{x \in \mathcal{A}_{\#}^{\mathbb{Z}^2} : \exists J, \exists (R_j)_{j\in J} \in (\mathcal{R}^{\circledast 2})^J,\ d(R_{j_1}, R_{j_2}) \geq 1 \text{ if } j_1 \neq j_2,$$
$$\bigsqcup_{j\in J} R_j = \{\mathbf{i} \in \mathbb{Z}^d : x_{\mathbf{i}} \neq \#\} \text{ and } \forall j \in J,\ R_j \in \mathcal{C}^{*2} \implies x|_{R_j} \in \mathcal{L}(Y)\}.$$

Intuitively, non $\mathcal{A}$-hyperrectangles in $Y'_{\#}$ can be filled with any pattern on $\mathcal{A}$; but $\mathcal{A}$-hypercubes in $Y'_{\#}$ can only be filled with valid patterns from Y.

Following the same proof, $Y'_{\#}$ is 1-block-gluing and $h_E(Y'_{\#}) = h_E(Y)$:

- Since hyperrectangles $R \in \mathcal{R}^{\circledast d} \setminus \mathcal{C}^{*d}$ that are *not* cubic can now contain every pattern of $\mathcal{A}^R$, all these new patterns belong in the same extender class and the upper bound $h_E(Y'_{\#}) \leq h_E(Y)$ still holds.
- However, since $\mathcal{A}^{\mathbb{Z}^d} \subseteq Y'_{\#}$, the inequality $h_E(Y) \leq h_E(Y'_{\#})$ needs to be proved with finite hypercubic patterns of $\mathcal{L}(X)$ instead of complete configurations: if $y^{(0)} \in Y$ distinguishes the extender classes of two patterns w, w' in Y, then there exists some $n \in \mathbb{N}$ such that the configuration $y \in Y'_{\#}$ defined as $y_{\mathbf{i}} = y^{(0)}_{\mathbf{i}}$ if $\mathbf{i} \in [\![-n, n]\!]^d$, and $y_{\mathbf{i}} = \#$ otherwise, distinguishes w from w' in $Y'_{\#}$.

We are left with proving that $Y'_{\#}$ is sofic. To prove this, we apply Theorem 10.11 based on the following main ideas: fixing a square domain $[\![n]\!]^2$, values of $\langle i \rangle$ and $\langle j \rangle$ being powers of 2 implies that the square $[\![n]\!]$ can only cover polylog(n) many distinct pairs (i, j). Since this is the most complicated application of Theorem 10.11 in this thesis, we recommend the reader to warm-up to these methods by skimming through some examples in Chapter 14 before doing this one. □

Sketch of proof: soficity of $Y'_{\#}$. The proof is written on $\mathbb{Z}^2$, but the generalization to arbitrary $\mathbb{Z}^d$ is straightforward.

Representing the subshift Y Considering the definition of the subshift $Y = Y_\alpha$ from Section 9.3.3, we represent patterns w of domain $[\![n_1, n_2]\!]$ as tuples composed of the following information:

- About the first layer $w^{(1)}$:[90] if no symbol $*$ appears on the layer $w^{(1)}$, this field is empty. If a single column of $*$ symbols appears on $w^{(1)}$, this field contains its *shift* $k_1 \in [\![n_1]\!]$, defined as its horizontal position inside $w^{(1)}$. If at least two columns of $*$ symbols appear on $w^{(1)}$, this field contains a *shift* $k_1 \in [\![n_1]\!]$ and a period $i \in [\![n_1]\!]$, respectively defined as the horizontal position of the leftmost column of symbols $*$, and the shortest distance between two columns of symbols $*$.

  If i is not a power of 2, or if $w^{(1)}$ is not horizontally periodic of period i, we actually do not define a representation for w.
- About the second layer $w^{(2)}$: same ideas as in the first layer $w^{(1)}$, while enforcing that $j \geq i$;
- About the density layer $w^{(d)}$: as the lift of a Toeplitz word, we follow the ideas of Example 10.8. More precisely:
  - If two symbols $*$ appear in $w^{(1)}$, this field contains the prefix $u' \in \{0,1\}^{\log i}$ that appears (in a Toeplitz way) between two $*$ columns of $w^{(1)}$;
  - If at most one $*$ column appears in $w^{(1)}$, this field contains (all non-deterministically guessed) a sequence of bits $b \in \{0,1\}^{\log n_1}$ such that b_ℓ denotes the parity of the positions of level ℓ among the positions of level $\ell - 1$; a prefix $u' \in \{0,1\}^{\log n_1}$ of the sequence embedded in the Toeplitz, consistently with $w^{(d)}$ and b; and the at most three positions and symbols of $[\![n_1]\!]$ that are not covered by the positions of level $\ell \leq \log n_1$ as determined by b;

[90] This representation is inspired by the represention of X_* defined in Section 14.3.2.

  - (Or, if such Toeplitz structure is clearly not possible in $w^{(d)}$, this field can be kept empty on the condition that $w^{(2)}$ contains at most a single $*$ column)
- About the marker layer $w^{(m)}$: if $w^{(m)}$ contains a symbol $\diamond$, this field contains its position, denoted $\mathbf{m}_\diamond$. Otherwise, it is empty.
- About the free layer $w^{(f)}$:
  - If $\mathbf{m}_\diamond$ is well-defined, this field should contain the value of the free bit $w^{(f)}_{\mathbf{p}_f}$.

    If i and j are both well-defined, all free bits in $w^{(f)}$ that appear on the grid centered on $\mathbf{m}_\diamond$ and of mesh (i, i) should have this value; if j is not defined, then we should remember an additional bit telling whether all free bits in $w^{(f)}$ that appear on the grid centered on $\mathbf{m}_\diamond$ and of mesh (i, i) have the same value or not;
  - If $\mathbf{m}_\diamond$ is not defined, we should non-deterministically guess $O(1)$ positions, and remember these positions and the value of the free bits from $w^{(f)}$ at these positions.

Then, there exists an induction that has time complexity $t(s) = O(s)$ that results in the subshift Y_α. To sketch the ideas, this induction should merge information about the first two layers (as for X_* in Section 14.3.2), merge information about the density layer (as in Example 10.8), and merge information about the free layer (to ensure that either j is undefined and the free bits on the aforementioned grid are allowed to differ, or ensure that they are all equal if j is discovered "later"). Finally, if i and j are well-defiend, it should check the prefix u' about the density layer against an approximation of $\alpha_{i,j} \in \Pi_1$ obtained after $\log \log n_1$ computation steps.

Representing the subshift $Y'_\#$ To define a representation of $Y'_\#$, remember that the configurations drawing rectangles of $\mathcal{A}$ symbols over a # background define a sofic subshift: by embeding the symbols of a local cover inside the representation of patterns of domain $[\![1]\!]^2$ and the first induction step[91], we can assume – in the rest of this proof – that all patterns which are to be represented and computed upon have the correct structure.

[91] See Theorem 10.10

Then, we proceed as in Theorem 14.2. We define a representation function $\mathcal{R}$ that associates to patterns w of domain $[\![n]\!]^2$ a tuple composed of the following information:

- The size $n \in \mathbb{N}$ of the domain;
- A *pair list* of maps $(i, j) \mapsto u'$ for $i \in [\![n]\!]$, $j \in [\![n]\!]$, and $u' \in \{0, 1\}^{O(\log n)}$ such that: if a square appears completely in w, then i, j and u' are respectively the period of the first layer, the period of the second layer, and the prefix of the density layer, as defined by the representation function of $Y = Y_\alpha$ defined above;[92]
- A *corner list* containing the following information about squares appearing partially inside w:
  - Either a square has a single corner appearing in w, in which case we remember the position and the orientation of this corner, and a model of the associated pattern of Y;
  - Or a square has two corners appearing in w, and the distance between these two corners is greater than $\frac{n}{8}$: in which case we remember the position and the orientation of these corners, and guess a possible model of the "complete square" (that straddles over the border of w).
  - Or a square has two corners appearing in w, the distance between these two corners is greater than $\frac{n}{8}$, and we guess that the partially appearing square is actually a rectangle: in which case we

[92] The *pair list* can, and will, contain other elements.

remember the position and the orientation of these corners, and remember a boolean to remember that these corners are supposed to represent a rectangle, and prevent them from being merged into a square.

The induction function $\mathcal{I}$ should then:

- Update the size to $2n$;
- Merge the *pair lists* into a single map: if a pair of maps that we want to merge define different prefixes for the same input pair (i, j), we keep the maximal prefix (which corresponds to the binary expansion of the larger real number);
- Merge the *corner lists*. If two corners list define corners that match into a rectangle, then:
  - If both subrepresentations had simulatenously guessed "rectangle", and that these corners indeed define a rectangle that is not a square, then delete these corners;
  - If both sides had simulatenously guessed "square", and that these corners indeed define a square, check that they guessed identical representations of the complete square;
  - Otherwise, reject the computation.
- Delete the small squares from the *corner list*: if two corners at distance less than $\frac{2n}{8}$ define a square in the corner list, consider the model of the "complete square" that was guessed; merge the association $(i, j) \mapsto u'$ from this model with the *pair list* defined above (once again, only keep the maximal prefix); and delete these corners from the *corner list*;
- Finally, for each pair $(i, j) \mapsto u'$ in the *pair list*, compute $\log \log n$ steps of an approximation of $\alpha_{i,j} \in \Pi_1$, and check that u' defines a rational number that is smaller than this approximation of $\alpha_{i,j}$.
- If any of these checks failed, reject the computation. Otherwise, return the new model.

We then claim that the pair $(\mathcal{R}, \mathcal{I})$ defines the subshift $Y'_{\#}$.[93]. Since $\mathcal{R}$ defines representations of length $\log^2(n)$ on patterns of domain $[\![n]\!]$;[94], and that $\mathcal{I}$ has time complexity $t(s) = O(s \cdot \log s)$;[95] we conclude by Theorem 10.11 that $Y'_{\#}$ is sofic. □

[93] This claim is straightforward: the only trick of this proof is the deletion of corners at distance less than $\frac{n}{8}$, since these are checked by adjacent inductions induced by the definition of inductive validity. See Theorem 14.2 for more details.

[94] Since i and j are powers of 2, only $\log^2(n)$ distinct pairs (i, j) can appear in the pair lists of such patterns; and the corner list has length $O(1)$, and contains $\log n$ bits of information per item.

[95] Use suitable data structures for the pair and corner lists, such as balanced binary trees.

Soficity and small representations

Summary

This whole chapter is joint work with Léo Paviet Salomon and Pascal Vanier.

Motivations We believe that the number of extender sets fails to characterize soficity in dimension $d \geq 2$ because they quantify the *deterministic* information communicated between patterns and their partial complementary configurations. Indeed, in a sofic subshift, the configuration $w \ltimes_{[\![n]\!]^d} x$ is valid if and only if $w|_{[\![n]\!]^d}$ and $x|_{\mathbb{Z}^d \setminus [\![n]\!]^d}$ can *non-deterministically* agree on a pair of compatible borders in a local cover, which is $O(n^{d-1})$ bits of information. However, extender sets in sofic subshifts summarize the whole set of possible borders of the covers of $w|_{[\![n]\!]^d}$, which turns out to be much larger[96] (by the previous half of this thesis, sofic subshifts can have $2^{\Theta(n^d)}$ distinct extender sets, *i.e.* require $\Theta(n^d)$ bits of deterministic communication).

In the second part of this thesis, we explore the soficity of multidimensional subshifts in a non-deterministic setting. Considering patterns of fundamental domain $[\![n]\!]^d$ in configurations as entitites in a non-deterministic communication setting, we ask how much non-deterministic information needs to be communicated between adjacent $[\![n]\!]^d$ patterns to verify the validity of a configuration.

[96] One should notice that, on $\mathbb{Z}$, the naive bound $2^{2^{O(1)}}$ on the number of sets of borders in a cover is still a constant independant of n: which makes the distinction between deterministic and non-deterministic communication only relevant in higher dimension.

Summary of results We introduce the notion of *inductive representations* in Definition 10.1, which quantify the amount of information communicated between adjacent patterns of increasingly larger sizes that have limited computation ressources (Definition 10.2) inductively: we first have a round of communication between adjacent patterns of domain $[\![1]\!]^d$, then adjacent patterns of domain $[\![2]\!]^d$, ...

- In Theorem 10.10, we prove that validity in $\mathbb{Z}^d$ sofic subshifts can be checked by only having patterns of domain $[\![1]\!]^d$ do the verification: communications of patterns of larger domains is not needed.
- In Theorem 10.11, we prove that if the configurations of a $\mathbb{Z}^d$ subshift can be verified in rounds of communications such that their patterns of domain $[\![n]\!]^d$ only need to communicate $O(n^\alpha)$ bits under time constraints $t(s) = O(s^\beta)$ with their neighboors (for $\alpha < d-1$ and $\alpha \cdot \beta < d-1$), then it is actually sofic.

Structure of the chapters These chapters are organized as follows:

- In Chapter 10, we introduce the notion of *inductive representations* along with several examples, and relate it to sofic multidimensional subshifts. We also state our main result Theorem 10.11, whose proof is postponed to Chapter 13.
- In Chapter 11, we consider a higher-dimensional and parallel model of computation called "mesh-connected multicomputers". Relating this model with classical RAM computations, we prove a time-compressing simulation (Theorem 11.8) that allows d-dimensional multicomputer with n^d processors to simulate, in n steps of computations, n^d steps of RAM computations.
- Chapter 12 and Chapter 13 are dedicated to the proof of Theorem 10.11, which is built upon the "fixpoint construction" from [DRS12]. In Chapter 14, we apply this result to both recover existing soficity proofs and prove the new-found soficity of some effective subshifts.
- Finally, in Chapter 15, we relate multidimensional soficity and communication complexity. In particular, we explore example subshifts inspired by classical communication complexity, and develop some perspectives and future questions to investigate.

[DRS12] Durand, Romashchenko, and Shen, "Fixed-point tile sets and their applications".

[Wes17] Westrick, "Seas of squares with sizes from a Π^0_1 set".

[Des21] Destombes, "Algorithmic complexity and soficness of shifts in dimension two".

[DR22] Destombes and Romashchenko, "Resource-bounded Kolmogorov complexity provides an obstacle to soficness of multidimensional shifts".

Final word Theorem 10.11 is the main result of these chapters: inspired by [Wes17] and [Des21, Theorem 4], it generalizes these results to an abstract sufficient condition for soficity based on the amount of information contained and communicated between adjacent patterns in the configurations of subshifts. As such, it also provides a partial answer to [DR22, Open problem].

Our results unfortunately cannot answer when the information bound is tight: assuming that a $\mathbb{Z}^d$ subshift has inductive representations of size $\Theta(n^{d-1})$ for its $[\![n]\!]^d$ patterns, is it actually sofic? This motivated the examples we introduced in Chapter 15, but many of these perspectives are only sketched and would deserve proper exploration.

In other words: the soficity of multidimensional subshifts is not solved yet!

Soficity and inductive representations

10

We introduce the notions of *representations* (Definition 10.1) and *inductive representations* (Definition 10.2), which formalize the intuition of communicating information between adjacent patterns to verify the validity of their concatenation.

In Theorem 10.10, we prove that sofic $\mathbb{Z}^d$ subshifts admit inductive representations of size $O(1)$. In Theorem 10.11, we state the main result of these chapters: if a $\mathbb{Z}^d$ subshift admits inductive representations of size $O(n^\alpha)$ with time constraints $t(s) = O(s^\beta)$ (for $\alpha < d-1$ and $\alpha \cdot \beta < d-1$), then it is actually sofic.

10.1 Recursive representations

10.1.1 Representations

What does it mean to consider the information contained inside a pattern? While Kolmogorov complexity[97] has traditionally been the go-to theory for these questions, the notion we need to formalize our intuitions on patterns is somewhat orthogonal. For example, inside a full shift $\mathcal{A}^{\mathbb{Z}^d}$, most patterns have maximal Kolmogorov complexity; but since all patterns are valid in $\mathcal{A}^{\mathbb{Z}^d}$, we do not need any information about patterns to decide their validity.

[97] Informally, the Kolmogorov complexity of a pattern $w \in \mathcal{A}^{*d}$ is the length of the shortest program that prints w on its output.

Thus, we suggest the following straightforward definition: to each pattern, we associate a set of strings that describe it. We call these strings *representations*:

Definition 10.1. *Given a dimension $d \in \mathbb{N}$ and a finite alphabet $\mathcal{A}$, a* representation function *is a multivalued function $\mathcal{R} : \mathcal{A}^{*d} \rightrightarrows \{0,1\}^*$. For a pattern $w \in \mathcal{A}^{*d}$, the elements of $\mathcal{R}(w) \subseteq \{0,1\}^*$ are called* representations.

How do representation functions relate to subshifts? For a given representation function $\mathcal{R} : \mathcal{A}^{*d} \rightrightarrows \{0,1\}^*$, we associate a subshift by only allowing patterns that posses a valid representation, *i.e.* we consider:

$$\{x \in \mathcal{A}^{\mathbb{Z}^d} : \forall w \sqsubseteq x,\ \mathcal{R}(w) \neq \emptyset\}.$$

As mentioned in the introduction, we use representations to determine whether a given pattern is *valid*. For example, the representation function that maps every pattern $w \in \mathcal{A}^{*d}$ to an encoding of w as a binary string defines the full-shift $\mathcal{A}^{\mathbb{Z}^d}$, as does the representation function that associates every pattern $w \in \mathcal{A}^{*d}$ to the single empty representation $\{\varepsilon\}$. However, one is much shorter than the other!

The definition of representation functions is very general, perhaps too much: for example, given a subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$, one could define a representation function $\mathcal{R}$ as $\mathcal{R}(w) = \{\varepsilon\}$ if $w \in \mathcal{L}(X)$, and $\mathcal{R}(w) = \emptyset$ otherwise. To effectively quantify the information needed to decide the validity of a pattern, we will, from now on, only consider *inductive representation functions*.

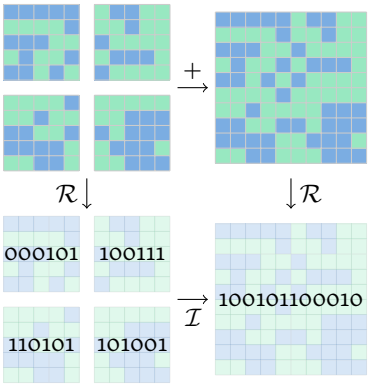


Figure 10.1: Induction on $\mathbb{Z}^2$: four patterns, their concatenations and the associated representations.

10.1.2 Inductions

We define inductive representation functions as the functions for which the representations of a pattern can be computed from the representations of its subpatterns:

Definition 10.2. *Given $d \in \mathbb{N}$, a finite alphabet $\mathcal{A}$ and a representation function $\mathcal{R} \colon \mathcal{A}^{*d} \rightrightarrows \{0,1\}^*$, an* induction *for $\mathcal{R}$ is a predicate $\mathcal{I} \subseteq (\{0,1\}^*)^{2^d} \times \{0,1\}^*$ such that:*

$$\forall w \in \mathcal{A}^{[\![2n]\!]^d}, \forall r \in \{0,1\}^*, \forall (r_{\mathbf{i}})_{\mathbf{i}\in[\![2]\!]^d} \in (\{0,1\}^*)^{2^d},$$
$$\Big((\forall \mathbf{i} \in [\![2]\!]^d,\ r_{\mathbf{i}} \in \mathcal{R}(w|_{C_{\mathbf{i}}})) \wedge \mathcal{I}((r_{\mathbf{i}})_{\mathbf{i}\in[\![2]\!]^d}, r)\Big) \implies r \in \mathcal{R}(w),$$

where the cubes $(C_{\mathbf{i}})_{\mathbf{i}\in[\![2]\!]^d}$ form the partition $[\![2n]\!]^d = \sqcup_{\mathbf{i}\in[\![2]\!]^d} C_{\mathbf{i}}$ of $[\![2n]\!]^d$ into 2^d cubes of size $[\![n]\!]^d$.

In other words, for a given representation function $\mathcal{R} \colon \mathcal{A}^{*d} \rightrightarrows \{0,1\}^*$, an *induction* $\mathcal{I} \subseteq (\{0,1\}^*)^{2^d} \times \{0,1\}^*$ allows to verify the representations of larger patterns from representations of its subpatterns. Even though inductions are formally defined as predicates $\mathcal{I} \subseteq (\{0,1\}^*)^{2^d} \times \{0,1\}^*$, we will often think of them as non-deterministic functions that return a representation $r \in \{0,1\}^*$ when given 2^d subrepresentations $(r_{\mathbf{i}})_{\mathbf{i}\in[\![2]\!]^d} \in (\{0,1\}^*)^{2^d}$.

⚠ Notice that no "surjectivivity condition" is required in this definition: it is *not* necessary that, for every $r \in \mathcal{R}(w)$, there exists $(r_{\mathbf{i}}) \in (\{0,1\}^*)^{2^d}$ such that $r_{\mathbf{i}} \in \mathcal{R}(w|_{C_{\mathbf{i}}})$ and $\mathcal{I}((r_{\mathbf{i}})_{\mathbf{i}\in[\![2]\!]^d}, r)$.

Remark 10.3. *To be precise, the definition above defines 2-inductions (since the induction associates descriptions of $[\![2n]\!]^d$ hypercubes with descriptions of $[\![n]\!]^d$ hypercubes). However, this is not much of a restriction: any decomposition along a hyperrectangle of $[\![n]\!]^d$ hypercubes can be simulated by a 2-induction.*

10.1.3 Representations and inductive validity

Since no surjectivity is required, a representation function $\mathcal{R}$ might define representations for a pattern $w \in \mathcal{A}^{[\![2^n]\!]^d}$ even though the actual set of representations which can be obtained when iterating $\mathcal{I}$ from the "pixels" of w may be empty, or a strict subset of $\mathcal{R}(w)$.

Thus, we define the notion of *inductively valid* patterns, which are patterns $w \in \mathcal{A}^{[\![2^n]\!]^d}$ which can be inductively computed through n steps of $\mathcal{I}$ (thus, that admit valid representations at every level from pixels of domain $\simeq [\![1]\!]^d$ to the whole pattern of domain $[\![2^n]\!]^d$). For the following definition, we partition $[\![2^n]\!]^d$ into nested hypercubes of sizes $[\![2^{n-k}]\!]^2$ for every $0 \le k \le n$:

Note 10.4. *For $n \in \mathbb{N}$ and $0 \le k \le n$, we denote by $C^k_{\mathbf{i}} \subseteq [\![2^n]\!]^d$ the hypercubes of size $[\![2^k]\!]$ forming the partition of $[\![2^n]\!]^d$ into $2^{(n-k)\cdot d}$ elements:*

$$[\![2^n]\!]^d = \bigsqcup_{\mathbf{i}\in[\![2^{n-k}]\!]^d} C^k_{\mathbf{i}}.$$

We denote by $C_{\mathbf{i}} = C^{n-1}_{\mathbf{i}}$. Notice that these partitions are nested: the hypercube $C^{k+1}_{\mathbf{i}}$ is the union of 2^d elements of the partition $(C^k_{\mathbf{i}'})_{\mathbf{i}'\in[\![2^{n-k}]\!]^d}$.[98]

[98] More precisely, for $\mathbf{i} \in [\![2^{n-k-1}]\!]^d$, the set $C^{k+1}_{\mathbf{i}}$ is the union of the sets $C^k_{\mathbf{i}'}$ for $i' \in 2\cdot\mathbf{i} + [\![2]\!]^d$.

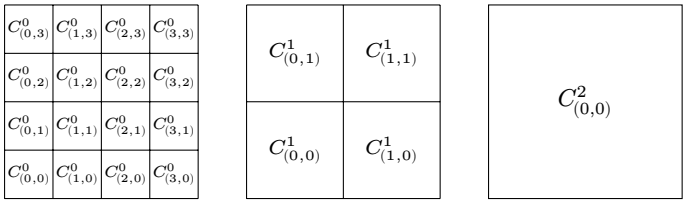


Figure 10.2: Three levels of nested partitions in dimension $d = 2$.

Definition 10.5. *A pattern $w \in \mathcal{A}^{[\![2^n]\!]^d}$ is said to be* inductively valid *if there exists representations $r^k_{\mathbf{i}} \in \{0,1\}^*$ for $0 \leq k \leq n$ and $\mathbf{i} \in [\![2^{n-k}]\!]^d$ such that:*

(i) For every $0 \leq k \leq n$ and $\mathbf{i} \in [\![2^{n-k}]\!]^d$, we have $r^k_{\mathbf{i}} \in \mathcal{R}(w|_{C_{k,\mathbf{i}}})$;

(ii) For every $0 \leq k \leq n$ and $\mathbf{i} \in [\![2^{n-k-1}]\!]^d$, we have $\mathcal{I}((r^k_{\mathbf{i}'})_{\mathbf{i}' \in I}, r^{k+1}_{\mathbf{i}})$, where $I = 2 \cdot \mathbf{i} + [\![2]\!]^d \subseteq [\![2^{n-k}]\!]^d$ is the set of 2^d adjacent hypercubes such that $\sqcup_{\mathbf{i}' \in I} C^k_{\mathbf{i}'} = C_{k+1,\mathbf{i}}$.

(iii) For every $0 \leq k < n$ and every hypercube $I = \mathbf{i} + [\![2]\!]^d \subseteq [\![2^{n-k}]\!]^d$ indexing 2^d adjacent hypercubes $(C^k_{\mathbf{i}})_{\mathbf{i} \in I}$, there exists $r \in \{0,1\}^*$ such that $\mathcal{I}((r^k_{\mathbf{i}})_{\mathbf{i} \in I}, r)$.

The representation $r^n_{\mathbf{0}}$ is called a final representation *of w.*

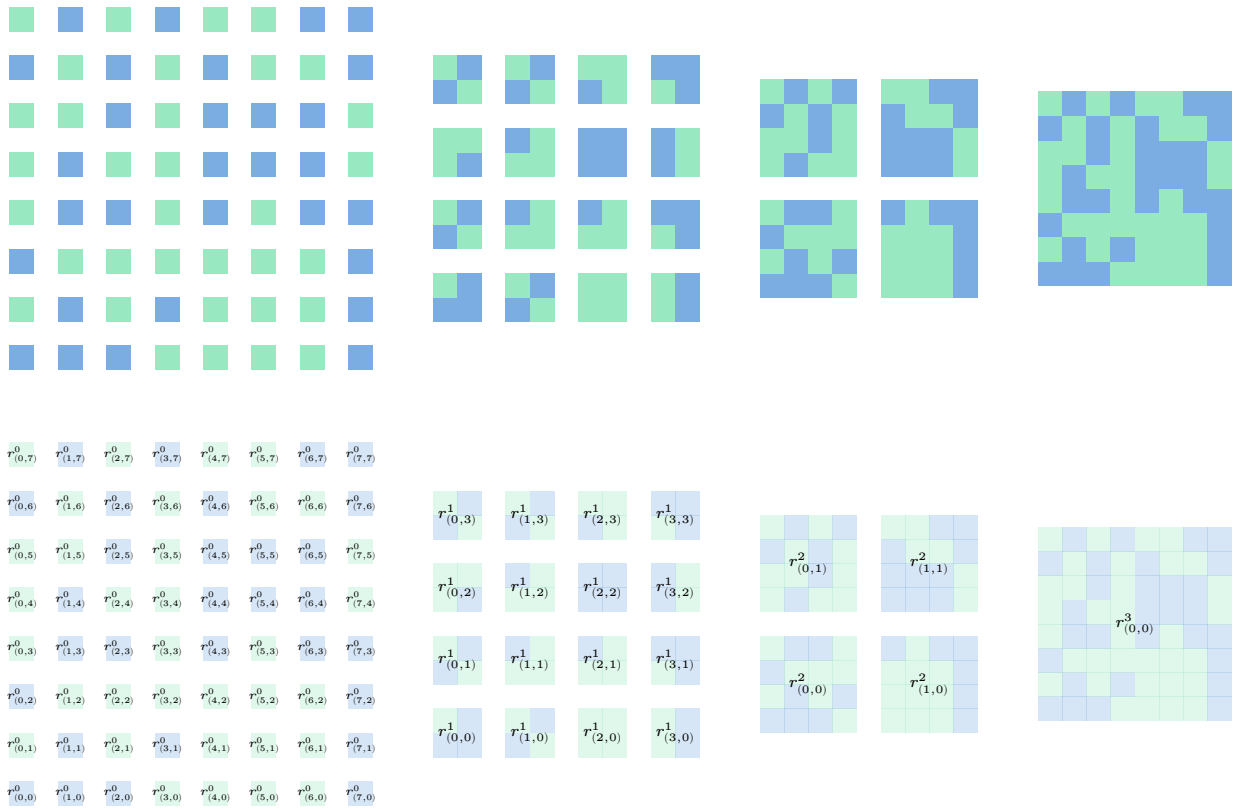


Figure 10.3: A pattern of domain $[\![2^3]\!]^d$ and its inductive representations (condition (i)). By condition (ii), representations $r^1_{(0,3)}$, $r^1_{(1,3)}$, $r^1_{(0,2)}$ and $r^1_{(1,2)}$ are given to the induction $\mathcal{I}$ to compute the representation $r^2_{(0,1)}$; representations $r^1_{(2,3)}$, $r^1_{(3,3)}$, $r^1_{(2,2)}$ and $r^1_{(3,2)}$ is given to the induction $\mathcal{I}$ to compute the representation $r^2_{(0,1)}$... but by condition (iii), every group of 2×2 adjacent representations, such as $r^1_{(1,2)}$, $r^1_{(2,3)}$, $r^1_{(1,2)}$ and $r^1_{(2,2)}$ will compute an induction step.

Conditions (i) and (ii) are straightforward, and define a hierarchy of representations that are inductively computed using the nesting induced by the partitions $((C^k_{\mathbf{i}})_{\mathbf{i} \in [\![2^{n-k}]\!]^d})_{0 \leq k \leq n}$. Condition (iii) is slightly more technical: due to the nesting induced by said partitions, some naughty patterns could avoid detection by aligning themselves with the nesting; by checking every set of 2^d adjacent hypercubes in (iii), we ensure that bad alignment does not prevent patterns from being checked by the induction.

Inductive representations can be used to define subshifts by only allowing patterns that are *inductively valid*:

$$X_{\mathcal{R},\mathcal{I}} = \Big\{ x \in \mathcal{A}^{\mathbb{Z}^d} : \forall w \sqsubseteq x, \exists n \in \mathbb{N}, \exists w' \in \mathcal{A}^{[\![2^n]\!]^d}, \\ w \sqsubseteq w' \sqsubseteq x \text{ and } w' \text{ is inductively valid} \Big\}.$$

10.2 Examples

Let us consider a few basic examples of inductive representations for well-chosen subshifts.

Example 10.6. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be any subshift. For any injective encoding $\langle\cdot\rangle : \mathcal{A}^{*d} \to \{0,1\}^*$ of patterns into binary strings, the representation function $\mathcal{R}$ defined by $\mathcal{R}(w) = \{\langle w\rangle\}$ if $w \in \mathcal{L}(X)$, and $\mathcal{R}(w) = \emptyset$ otherwise, is inductive (for $\mathcal{I}$ the predicate that computes the concatenation of the 2^d subpatterns, and checks whether the resulting pattern is valid) and verifies $X_{\mathcal{R},\mathcal{I}} = X$.*

In the previous example, the induction $\mathcal{I}$ is not necessarily a computable predicate. However, in the case of effective subshifts, such representations can be effectively computed. We present here a trick that will often be repeated in later examples: **we embed the computations of the forbidden patterns into the induction $\mathcal{I}$ and check for their existence in patterns via their representations**.

Example 10.7. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be an effective subshift. Let us fix a time-efficient encoding $\langle\cdot\rangle : \mathcal{A}^{*d} \to \{0,1\}^*$ of patterns into binary strings (for example, encode patterns as d-dimensional arrays), let us define:*

- *The representation $\mathcal{R}$ deterministically associates to a pattern $w \in \mathcal{A}^{*d}$ the single representation $\langle w\rangle$;*
- *When given 2^d subrepresentations $\langle w_i\rangle$ for $i \in [\![2]\!]^2$, the induction $\mathcal{I}$ computes $\langle w\rangle$, where w is the concatenation of 2^d patterns $(w_i)_{i\in[\![2]\!]^d}$. Denoting s the total bit size of its input, $\mathcal{I}$ then computes $O(\log s)$ computation steps of an enumeration of the forbidden patterns of X and checks $\langle w\rangle$: if any of these forbidden patterns appears in w, then $\mathcal{I}$ rejects the computation; otherwise, it accepts and returns $\langle w\rangle$.*
  Then $\mathcal{I}$ has time complexity $t(s) = O(s)$, since it only performs pattern concatenation and checks for subpattern occurences of size $O(\log s)$.

Thus X admits inductive representations $\mathcal{R}$ of size $O(n^d)$ on $[\![n]\!]^d$ patterns with computable inductions $\mathcal{I}$ of time complexity $t(s) = O(s)$.

From the previous example, all effective subshifts admit inductive representations of size at most $O(n^d)$. However, due to the geometrical configurations of many example subshifts, the information contained within patterns of domain $[\![n]\!]^d$ can sometimes be compressed into representations much shorter than $O(n^d)$.

Example 10.8. *Let $U \subseteq \mathcal{A}^{\mathbb{N}}$ be an effectively closed set, and $X = X_{\mathcal{T}(U)} \subseteq \mathcal{A}^{\mathbb{Z}}$ the associated Toeplitz subshift. We prove the existence of inductive representations of size $O(\log n)$ and associated time complexity $t(s) = O(s)$.*

Recall that Claim 5.1 showed that for every valid pattern $w \in \mathcal{L}_{[\![n]\!]}(X)$, there exists an element $u \in U$ such that:

- *One position out of two in w contains the symbol u_0. These are called the* positions of level 0*;*
- *Among the remaining positions, one out of two contains the symbol u_1. These are called the* positions of level 1*;*
- *Etc... until level $\log n$;*
- *And at most three positions in w contain symbols u_j for $j \geq \log n$.*

Notice that the positions occupied by levels $\ell \leq \log n - 1$ are entirely determined by their parity among the positions of level $\ell - 1$. Unfortunately, it is not possible to determine the level of a given position $i \in [\![n]\!]$: for example, the sequence $0^{\mathbb{N}}$ is Toeplitzified into the sequence $0^{\mathbb{Z}}$.

Yet, following this structure, we define a non-deterministic representation $\mathcal{R}$ to contain, for $w \in \mathcal{A}^n$ a valid pattern in X:

- *The size $n \in \mathbb{N}$;*
- *A non-deterministic guess of a sequence of bits $b \in \{0,1\}^{\log n}$ such that b_ℓ denotes the parity of the positions of level ℓ among the positions of level $\ell - 1$;*
- *A non-deterministic guess of a* prefix *$u' \in \mathcal{A}^{\log n}$; it should be consistent with w and the parity sequence $b \in \{0,1\}^{\log n}$, i.e. all positions of level ℓ (as determined by b) should contain the symbol u'_ℓ in w;*
- *The (at most) three remaining positions of $[\![n]\!]$ that are not covered by positions of level $\ell \leq \log n$ (as determined by the parity sequence b), and the symbols of $\mathcal{A}$ that each contains.*

As a result, representations of patterns of domain $[\![n]\!]$ are all of size $O(\log n)$.

As is often the case in our examples, the induction $\mathcal{I}$ follows directly from the representations of $\mathcal{R}$. Here, the induction $\mathcal{I}$ should check the consistency of the two input subrepresentations m_1, m_2, and guess the next letter of the U-prefix:

- *Check that the* prefixes *of m_1 and m_2 are the same; if not, reject the computation;*
- *Check that the parity sequence of m_2 is the parity sequence of m_1, but shifted by n; if not, reject the computation;*
- *Guess a new prefix u' of length $1 + \log n$ that agrees with the prefixes of m_1 and m_2, and guess a new parity sequence of length $1 + \log n$ that agrees with the parity sequence of m_1. The new symbol of the prefix and the new bit of parity should be consistent with the (at most) six positions of m_1 and m_2 that were not covered by levels $\ell \leq \log n$. If not, reject the computation;*
- *Remember the (up to) three positions not covered by the new parity sequence and their symbols in m_1 and m_2.*
- *Since U is effectively closed, compute $\log \log n$ steps of an enumeration of the cylinders of U^c. This enumerates $O(\log \log n)$ words of length $O(\log \log n)$: if u' is suffix of any of them, reject the computation; otherwise, return the newly computed representation.*

In other words, X admits representations of size $O(\log n)$ on $[\![n]\!]$ patterns with inductions that are computable in linear time $t(s) = O(s)$.

In the previous example, the subshift $X_{\mathcal{T}(U)}$ has polynomial pattern complexity. Thus, the astute reader may not be surprised that patterns of domain $[\![n]\!]$ admit representations of size $O(\log n)$. Of course, any full shift will admit inductive representations of size $O(1)$; but more interestingly, taking the alphabet $\mathcal{A} = \{0,1\}$ and adding free bits on top of all symbols 1 leads to a new "Toeplitz-like" subshift – this time of exponential pattern complexity – on which the very same representations and inductions still apply.

Remark 10.9. *The Toeplitz example perfectly illustrates our main intuition about inductive representations: they should either*

- *Outright* reject a pattern *for geometrical/basic structural reasons (e.g. if it does not follow a Toeplitz structure);*
- *Or* contain enough information *about the underlying pattern to allow checking it for the presence of forbidden patterns (e.g. in the Toeplitzifications of an effectively closed set U, check whether the forbidden patterns do not appear Toeplitzified in the configuration).*[99]

In particular, the information representations summarize from patterns is not their pattern complexity, but rather relates to the amount of information one needs to know about a pattern to reject it if it is invalid.

More examples of inductive representations are developed in Chapter 14.

[99] Of course, this distinction is purely conceptual: while it guides most of our examples and constructions below, the intuition it provides has little to do with sofic subshifts and is probably limited. Maybe some parallels could be drawn with the distinction difference between locally and globally valid patterns in a local/sofic subshift; but intuitions only take us this far.

10.3 Necessary and sufficient conditions for soficity

10.3.1 Recursive representations of sofic subshifts

Theorem 10.10. *For $d \in \mathbb{N}$ and a finite alphabet $\mathcal{A}$, let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a sofic subshift. There exists a representation function $\mathcal{R} : \mathcal{A}^{*d} \rightrightarrows \{0,1\}^*$ and an induction $\mathcal{I} \subseteq (\{0,1\}^*)^{2^d} \times \{0,1\}^*$ for $\mathcal{R}$ such that:*

(i) For any $w \in \mathcal{A}^{[\![n]\!]^d}$ and for any representation $r \in \mathcal{R}(w)$, the size of r verifies $|r| = O(1)$;

(ii) $\mathcal{I}$ is computable in time $t(s) = O(1)$ in the log-RAM *model; and*

(iii) $X = X_{\mathcal{R},\mathcal{I}} = \Big\{ x \in \mathcal{A}^{\mathbb{Z}^d} : \forall w \sqsubseteq x, \exists n \in \mathbb{N}, \exists w' \in \mathcal{A}^{[\![2^n]\!]^d}, w \sqsubseteq w' \sqsubseteq x \text{ and } w' \text{ is inductively valid} \Big\}$.

The proof directly follows from the definition of soficity: representations of patterns of domain $[\![1]\!]^d$ contain a symbol from the alphabet of a local cover; and representations of patterns of domain $[\![n]\!]^d$ for $n \geq 2$ are just empty strings.

Proof. For $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ a sofic subshift, let $Y \subseteq \mathcal{B}^{\mathbb{Z}^d}$ be a local cover of X and denote $\pi : \mathcal{B} \to \mathcal{A}$ a projection from Y to X. For $n \in \mathbb{N}$ and a pattern $w \in \mathcal{A}^{[\![n]\!]^d}$, we define $\mathcal{R}(w)$ to be the following set of representations:

- If $n = 1$, every symbol $b \in \mathcal{B}$ such that $\pi(b) = w$;
- If $n \geq 2$, the empty string ε;

and we define the induction $\mathcal{I}((r_{\mathbf{i}})_{\mathbf{i} \in [\![2]\!]^d}, r)$ as follows:

- If the $(r_{\mathbf{i}})_{\mathbf{i} \in [\![2]\!]^d}$ are representations of patterns of domain $[\![1]\!]^d$, each $r_{\mathbf{i}}$ is a symbol of $\mathcal{B}$. Check that their concatenation into a hypercube of domain $[\![2]\!]^d$ is locally admissible in the local cover Y: if the check fails, reject the computation. Otherwise, accept and return the empty string ε as a valid representation;
- Otherwise, when given 2^d empty strings, accept and return the empty string.

Then $\mathcal{R}$ and $\mathcal{I}$ verify items (i) and (ii) of the theorem. Since inductively valid patterns $w \in \mathcal{A}^{[\![2^n]\!]^d}$ are exactly the patterns that admit a locally admissible preimage in the local cover Y, we conclude that $X = X_{\mathcal{R},\mathcal{I}}$. □

In other words, the configurations of a $\mathbb{Z}^d$ sofic subshift can be check valid with just the first step of inductive validity. This agrees with the definitions of sofic subshfits as projections of local subshifts: the validity of all patterns can be non-deterministically checked by adjacency constraints.

10.3.2 Soficity of subshifts with small inductive representations

As mentioned in the introduction, the intuition about sofic subshifts is that the amount of information a pattern of domain $[\![n]\!]^d$ can communicate to its exterior is bounded by the size of its border, *i.e.* $O(n^{d-1})$.

In a partial converse statement, we now prove the main theorem of these chapters: if X is a subshift that admits inductive representations of size $O(n^\alpha)$ for $\alpha < d - 1$ on its patterns of domain $[\![n]\!]^d$, then – under some time constraints – X is actually sofic. In other words: if the validity of the configurations of a given subshift X only require $O(n^\alpha)$ bits of communication for their n^{th} step of inductive validity, and that said steps can be implemented efficiently in space $[\![n]\!]^d$, then X is sofic:

Theorem 10.11. *Fix $d \in \mathbb{N}$ and a finite alphabet $\mathcal{A}$. Let $\mathcal{R} \colon \mathcal{A}^{*d} \rightrightarrows \{0,1\}^*$ be a representation function and $\mathcal{I} \subseteq (\{0,1\}^*)^{2^d} \times \{0,1\}^*$ be an induction for $\mathcal{R}$ such that:*

(i) There exists $\alpha \in \mathbb{R}_+$ with $\alpha < d-1$ such that, for any $w \in \mathcal{A}^{[\![n]\!]^d}$ and every representation $r \in \mathcal{R}(w)$, the size of r verifies $|r| = O(n^\alpha)$;

(ii) There exists $\beta \in \mathbb{R}_+$ with $\alpha \cdot \beta < d-1$ such that $\mathcal{I}$ is computable in time $t(s) \leq O(s^\beta)$ in the log-RAM *model;*

Then

$$X_{\mathcal{R},\mathcal{I}} = \Big\{ x \in \mathcal{A}^{\mathbb{Z}^d} : \forall w \sqsubseteq x, \exists n \in \mathbb{N}, \exists w' \in \mathcal{A}^{[\![2^n]\!]^d}, \\ w \sqsubseteq w' \sqsubseteq x \text{ and } w' \text{ is inductively valid} \Big\}$$

is a sofic subshift.

This theorem can be used to prove the soficity of a wide variety of subshifts (see Chapter 14 for examples) by quantifying the amount of information shared between patterns. We argue in Chapter 15 that our arguments are an instance of ressource-bounded communication complexity.

As for the proof itself, it is based on the expanding simulating tileset construction from [DRS12], directly inspired by previous results from L. Westrick[100] [Wes17] and J. Destombes [Des21]. Based on a geometrical arrangement to compute inductive representations among the macro-tiles of the aforementioned simulating tilesets, it also implements fast computations by borrowing ideas from a higher-dimensional and parallel model of computation: namely, mesh-connected multiprocessors (see Chapter 11). As the next chapters are busy introducing the tools that we have not yet covered, the proof itself is postponed to Chapter 13.

Remark 10.12. *We could actually lessen the gap between $O(n^\alpha)$ for $\alpha < d-1$ and arbitrary functions $\alpha \colon \mathbb{N} \to \mathbb{N}$ such that $\alpha(n) = o(n^{d-1})$. Indeed, the only overhead induced by our proof of Theorem 10.11 is logarithmic, so that some functions $\alpha \colon \mathbb{N} \to \mathbb{N}$ and $\beta \colon \mathbb{N} \to \mathbb{N}$ respecting $\beta \circ \alpha(n) = o(\frac{n^{d-1}}{\log n})$ could actually result in a sofic subshift.*

However, additional properties (which hold for power functions) are required in the proof: monotonicity, submultiplicativity, $\alpha(L_\ell) \ll N_\ell$ for a sequence of zoom factors[101] $(N_\ell)_{\ell \in \mathbb{N}}$... Which altogether would probably make a general statement difficult to parse.

[DRS12] Durand, Romashchenko, and Shen, "Fixed-point tile sets and their applications".

[100] The main argument in [Wes17] shows that the "seas of squares" subshift admits representations of size $\tilde{O}(n^{2/3})$ and an induction computable in time $t(s) = O(s)$.

[Wes17] Westrick, "Seas of squares with sizes from a Π^0_1 set".

[Des21] Destombes, "Algorithmic complexity and soficness of shifts in dimension two".

[101] See Section 12.4 for definitions.

Mesh-Connected MultiComputers 11

To implement arbitrary computations in patterns of domain $[\![n]\!]^2$, one usually draws space-time diagrams of Turing machines: n tape cells on each line, with n lines for the n steps of computations. We have, however, rarely seen any consideration being given to the computational models available for higher-dimensional tilings.

In this chapter, we consider the parallel model of *Mesh-Connected Multicomputers* [Akl85]. In Theorem 11.8, we prove that a $(d-1)$-dimensional MCMC of size N^{d-1} can simulate, in time $O(N)$, at least N^{d-1} computation steps of a RAM program: this makes MCMCs suitable to implement the computations of Theorem 10.11, which operate on inputs of size $O(N^{d-1})$, inside patterns of domain $[\![N]\!]^d$.

[Akl85] Akl, *Parallel sorting algorithms*.

11.1 Mesh-Connected MultiComputers and algorithms

A Mesh-Connected MultiComputer describes a d-dimensional array of processors with bounded memory arranged in a regular grid, each processor only communicating with its immediate neighbors [Akl85, Chapter 5]. This representation is also known under many different names, including Mesh-Connected Multiprocessors, Mesh-Connected Parallel Computers, Mesh-Connected (Parallel) Array Processors... and is both used in the SIMD[102] and MIMD[103] settings.

Mesh-Connected Multicomputers seem to have arisen from a mix of emerging real-world parallel computers[104] and algorithmic complexity at the end of 60s/early 70s. For an overview of the parallel algorithmic problems discussed within the context of MCMCs and other parallel architectures (sorting, routing, linear algebra...), we refer the reader to [Lei92].

[102] Single Instruction Multiple Data: many processors perform the same instruction synchronously on different data points.

[103] Multiple Instructions Multiple Data: processors execute distinct instructions on different data points.

[104] Such as the ILLIAC IV (original article in 1968).

[Lei92] Leighton, *Introduction to parallel algorithms and architectures*.

11.1.1 Mesh-Connected MultiComputers

Definition 11.1. A *d-dimensional* Mesh-Connected MultiComputer (MCMC) *of size N^d is a d-dimensional array of N^d processors arranged in a regular grid. Each processor possesses $O(\log N)$ bits of memory, and can communicate with its $2d$ immediately adjacent neighbors*[105].

[105] Processors might have less than $2d$ neighbors, for example if they sit on the border of the grid.

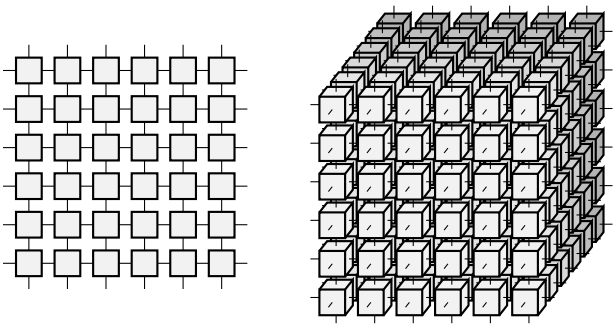


Figure 11.1: Drawing of a 2D and a 3D array of processors.

The precise specifications of the processors involved in an MCMC may considerably vary in the litterature, ranging from unbounded computational power to finite state machines with various sets of instructions. Non-deterministic computations does not seem to have been investigated in MCMCs, maybe because of motivations grounded in CPU architecture rather than theoretical computational complexity[106].

[106] For example, sending information from one processor to another can be done in $O(1)$ steps in a non-deterministic setting: a chain of processors can simulate-nously "guess" the message, and then simultaneously check that these guesses are consistent.

Since time complexity and simulations will be important in the proof of our main Theorem 10.11, and that tilings are a very natural setting for non-determinism, we settle on the following definition for processors:

Definition 11.2. A processor *in an N^d MCMC is a non-deterministic* RAM *machine whose memory is made of finitely many registers of length $O(\log N)$ bits. During a computation step, each processor may either read one register from an adjacent neighboor, or perform one computation step of a non-deterministic* RAM *program.*

11.1.2 Sorting in MCMCs

Designing programs for mesh-connected multicomputers can be quite challenging, but also leads to unexpected results. For example, consider the sorting problem: every processor in an N^d MCMC is given an integer, and the MCMC should permute these integers so that they end up sorted according to a pre-defined indexing. Surprisingly, many sorting problems in MCMCs can actually be solved in time $O(N)$.

Boustrophedon ordering The *Boustrophedon ordering* is a geometrical ordering of d-dimensional arrays that generalizes classical orderings on the line. Formally, let us consider $(O, \leq)$ a totally ordered set.

Definition 11.3. A *2-dimensional array $a \in O^{N\times N}$ is said to be* sorted in Boustrophedon order[107] *if, for every $(i,j) \leq_{\text{lex}} (i',j')$, we have:*

- *If $i < i'$, then $a_{i,j} \leq a_{i',j'}$;*
- *If $i = i'$ and i is even, then $a_{i,j} \leq a_{i',j'}$;*
- *If $i = i'$ and i is odd, then $a_{i,j} \geq a_{i',j'}$.*

[107] Boustrophedon is a style of writing in which lines alternate being written from left to right and right to left. The term comes from greek βουστροφηδόν, "in the way an ox turns [while plowing]".

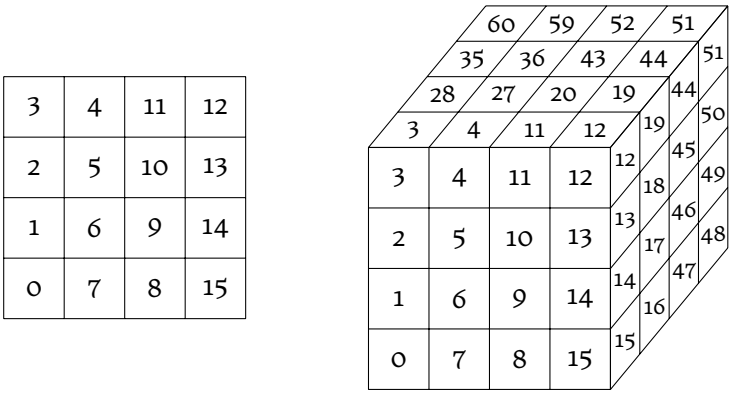


Figure 11.2: 2D and 3D arrays of integers in Boustrophedon order.

The Boustrophedon ordering is also sometimes called the *snake ordering*, or the *shear ordering*[108]. It generalizes to dimension $d \geq 3$ by induction: for a given d-dimensional array, its $(d-1)$-dimensional subarrays are sorted alternatively in increasing or decreasing Boustrophedon order. The d-dimensional Boustrophedon ordering of the hypercube $[\![n]\!]^d$ also generates a simple reflected n-ary Gray code.

[108] Computer scientists have an unbridled imagination for agricultural metaphors.

Introductory sorting algorithms on MCMCs A nearly-optimal algorithm for Boustrophedon ordering on 2-dimensional MCMCs was introduced in [SSS86]. While it has a logarithmic overhead from the optimal complexity, it is very simple to implement, and illustrates how the higher-dimensional geometry of MCMCs can be used to sort elements faster than traditional array sorting:

[SSS86] Sen, Scherson, and Shamir, "Shear Sort: A True Two-Dimensional Sorting Techniques for VLSI Networks".

function SHEAR-SORT(a) ▷ *a is 2-dimensional array*
  loop
    Sort even-indexed rows of a in parallel from left to right;
    Sort odd-indexed rows of a in parallel from right to left;
    Sort all columns of a in parallel from top to bottom.

Lemma 11.4 ([SSS86])**.** *Let $a \in O^{N\times N}$ be a matrix of size $N \times N$ given as input to* SHEAR-SORT*. After $\log N$ iterations of the main loop, the matrix a is sorted by Boustrophedon ordering.*

It was further generalized to d-dimensional MCMCs in [CS92]:

[CS92] Corbett and Scherson, "Sorting in Mesh Connected Multiprocessors".

Lemma 11.5 ([CS92])**.** *For $d \in \mathbb{N}$, the algorithm* MESH-SORT *for Boustrophedon sorting on MCMCs of size N^d has time complexity $O(d^2 \cdot N \log N)$.*

Optimal sorting algorithms on MCMCs While these algorithms are easy to implement, their complexities are suboptimal. The *bitonic sort*[109], introduced in [Bat68], was adapted to several optimal algorithms on higher-dimensional mesh-connected multicomputers for various indexings in [TK77; NS79, ...]. In particular,

[109] This is a parallel sorting algorithm for 1-dimensional array that has parallel complexity $O(\log^2(n))$.

[Bat68] Batcher, "Sorting networks and their applications".

[TK77] Thompson and Kung, "Sorting on a Mesh-Connected Parallel Computer".

[NS79] Nassimi and Sahni, "Bitonic sort on a Mesh-Connected Parallel Computer".

Lemma 11.6 ([NS79])**.** *For $d \in \mathbb{N}$, there exists an algorithm for Boustrophedon sorting on MCMCs of size N^d with time complexity $O(d^2 \cdot N)$.*

Remark 11.7. *Since the litterature about mesh-connected multicomputers is not interested in the one-dimensional case, we should clearly state the the result still holds in dimension $d = 1$ by implementing a parallel merge sort*[110].

[110] Since each halves of the array are recursively sorting in parallel, and that merging takes $O(N)$ steps on arrays of size N, the recurrence on the time complexity $T(N)$ is of the form $T(N) = T(N/2) + O(N)$, so that $T(N) = O(N)$.

11.2 Simulating RAM programs with MCMCs

To avoid the task of designing parallel algorithms for mesh-connected multicomputers, we prove that traditional log-RAM programs can be efficiently simulated by MCMCs. These ideas follow the proof of Proposition 4.31: using non-determinism, memory calls are "guessed" during the simulation, and these guesses are written on a special memory array. At the end of the simulation, the computation device checks whether these guesses were correct by sorting this memory array and verifying its consistency.

More precisely, we prove the following result:

Theorem 11.8. *Let e be a* RAM *program running in time $t(n)$. For any $d \in \mathbb{N}$, there exists an* MCMC *program that, if run on a mesh-connected grid of N^d processors for $N = \sqrt[d]{t(n)}$, simulates e in time $O(N)$.*

In other words: d-dimensional MCMCs can simulate, in time $O(N)$, at least N^d steps of computations of any RAM program. We had never heard of such simulation results before, though they turn out to be useful in the context of higher-dimensional tilings (see Section 11.3); however, we would not be surprised if – as most simulations results – this was well-known to some people and considered to be "folklore".

Proof. Let e be a RAM program. As in Proposition 4.31, the associated MCMC program will be divided in two parts: a *simulation part*, which runs the computations of e; and a *validation part*, which checks whether the run was actually valid.

Simulation part ($O(1)$ steps) Following the Boustrophedon indexing of the N^d processors of the MCMC, we make the i^{th} processor of the MCMC simulate the i^{th} step of a valid run of the program e. More precisely:

1. We store the program e in each processor ($O(1)$ bits);
2. Each processor "guesses" a value for the instruction pointer, a value for each variable of the log-RAM model, and stores the resulting values in a first set of memory cells ($O(1) + O(\log N)$ bits[111]);
3. Each processor "simulates" the operation corresponding to the instruction pointer it guessed at the previous step as follows:
   - **Input readings** `var`$_0$ ← `I[var`$_1$`]`**:** non-deterministically, the processor "guesses" a binary word $w \in \{0,1\}^{O(\log N)}$ and stores the record `(INPUT,READ,i,w)` in its memory, where `i` is the current value of `var`$_1$ ($O(\log N)$ bits);
   - **Memory readings** `var`$_0$ ← `M[var`$_1$`]`**:** non-deterministically, the processor "guesses" a binary word $w \in \{0,1\}^{O(\log N)}$ and stores the record `(MEMORY,READ,time,address,w)` in its memory, where `time` is the time step simulated[112] `address` is the current value of `var`$_1$ ($O(\log N)$ bits).
   - **Memory writings** `M[var`$_0$`]` ← `var`$_1$**:** the processor stores a record `(MEMORY,WRITE,time,address,w)` in its memory, where `time` is the time step simulated[113], `address` is the current value of `var`$_0$ and `w` is the current value of `var`$_1$ ($O(\log N)$ bits).
   - In all cases, after guessing memory readings if needed, the processor executes its instruction on its set of variables. In a second set of memory cells, it stores the updated values of the variables and of the instruction pointer ($O(\log N)$ bits).
4. All processors check in parallel the consistency of their computation step with the neighbor that precedes them in Boustrophedon order: more precisely, they check the consistency of the instruction pointer and the consistency of the variables.

[111] Indeed, the input is of size at most $O(N^d)$, so that $\log N^d = O(\log N)$.

[112] Which corresponds to the index of the processor in Boustrophedon ordering.

[113] *idem*.

Validation part ($O(N)$ steps) During the *simulation part*, all the memory calls (*i.e.* reading the input, or reading) had to be guessed. We are left with checking that these guesses were actually correct.

Since each processor is given one word of the input array, processors should keep them intact during the $O(1)$ steps of simulation. At the end of the simulation part, each processor may contain:

- An input initial record `(INPUT,WRITE,i,value)` where i is an index of size $O(\log N)$ bits, and w is the i^{th} binary word of length $O(\log N)$ of the input array I, as seen by the processor;
- An input reading record `(INPUT,READ,i,value)` if the computation step simulated was an input reading (where `time` and `i` are integers and `b` is a bit);
- A memory reading record `(MEMORY,READ,time,address,value)` or a memory writing record `(MEMORY,WRITE,time,address,value)` if the computation step simulated was a memory reading or a memory writing (where `time` is an integer, and `address` and `value` are binary words of length $O(\log N)$).

During the validation part, the MCMC performs the following steps:

1. Using Lemma 11.6, we independently sort `INPUT` records and `MEMORY` records by lexicographic order on their addresses and time steps[114] ($O(N)$ steps).
2. Adjacent processors check in parallel with the neighbor that precedes them in Boustrophedon order that, if they contain records operating on the same memory value, these records are chronologically consistent[115]; and crash the computation if they are not ($O(1)$ steps).

If the *validation part* did not crash, then the MCMC accepts or rejects depending on the final state of the simulated run of the program e. □

[114] With the notations above, the sorting is done by increasing values of `address`, and for equal values of `address` by increasing values of `time`.

[115] Two successive memory readings should return the same value; an unitialized memory cell should not be read; a memory reading following a memory writing should read the correct value...

11.3 Space-time diagrams of MCMCs

Configurations Since a processor in a mesh-connected multicomputer of size N^d consists of a RAM program and finitely many variables of size $O(\log N)$, we define the *state* of an MCMC processor as the tuple containing:

- The value of all its variables ($O(1) \cdot O(\log N)$ bits);
- The RAM program it runs ($O(1)$ bits);
- And the value of its instruction pointer ($O(1)$ bits).

Such a mesh-connected multicomputer being made of N^d processors sitting on a d-dimensional grid, we call *configurations* (*i.e.* the possible global states of the MCMC) the d-dimensional arrays of domain $[\![N]\!]^d$ that map a position $\mathbf{i} \in [\![N]\!]^d$ to the state of the processor of index $\mathbf{i}$.

We now consider a graphical representation of the computation of mesh-connected multicomputers as *space-time diagrams*:

Definition 11.9 (Space-time diagram)**.** *For a given run of a mesh-connected multicomputer of size N^d during T computation steps, the associated* space-time diagram *is a $(d+1)$-dimensional array of size $[\![T]\!] \times [\![N]\!]^d$ such that each subarray $\{t\} \times [\![N]\!]$ contains the configuration of the* MCMC *at the t^{th} step of computation.*

Locality One noticeable property of mesh-connected multicomputers is their well-suitedness for tiling implementations[116]:

Note 11.10 (Locality)**.** *Space-time diagrams of mesh-connected multicomputers are defined by* local rules.

[116] Up to the fact that a processor state contains $O(\log N)$ bits of information, which – unlike Turing machines, whose cells all contain $O(1)$ bits of information – is not constant as we consider multicomputers of increasing sizes.

Indeed, the validity of a space-time diagram only depends on checking the validity of adjacent positions: for a given processor $\mathbf{i} \in [\![N]\!]^d$ and a time step $t \in [\![T]\!]$, validity can be enforced by only checking that the state of the processor $\mathbf{i}$ at time t is consistent with the state it was in at the time step $t-1$, and – in case of a memory reading – that it correctly reads the memory of its neighboor of index $\mathbf{i} + \mathbf{e}_j$ for some $1 \leq j \leq d$.

Distribution to subarrays Using the fact that space-time diagrams of mesh-connected multicomputers are governed by local rules, we will sometimes distribute their computations to subarrays of size $[\![M]\!]^d$ for $M < N$:

Definition 11.11. *For a given mesh-connected multicomputer of size N^d, a* subarray *of size M^{d+1} is a cubic piece of size M^{d+1} from a space-time diagram.*

A grid of $(N/M)^{d+1}$ subarrays of size M^{d+1} can, together, hold any space-time diagram composed of N computation steps from an MCMC of size N^d. Furthermore, assuming that each subarray is a valid piece of computation, the validity of the whole grid of subarray can be checked by only exchanging $O(M^d)$ processor states between all pairs of adjacent subarrays[117].

[117] Since space-time diagrams are local, local validity between subarray only requires to communicate the processor states appearing on their border.

12 The expanding simulation framework

Our proof of Theorem 10.11 relies on the so-called "fixpoint construction" of subshifts, which was published in a series of articles ranging from [DRS08] to [DRS12]. In this chapter, we present this construction and the expanding simulating tilesets it builds. The reader already familiar with the methods involved should feel free to quickly check the vocabulary used and then skip to the next chapter.

[DRS08] Durand, Romashchenko, and Shen, "Fixed point and aperiodic tilings".

[DRS12] Durand, Romashchenko, and Shen, "Fixed-point tile sets and their applications".

12.1 Tilesets

Since this construction is formulated in terms of Wang tiles, we briefly recall Wang tiles as a special case of local subshift: Wang tiles are square tiles whose edges are colored, and the local validity rule of the resulting subshift enforces that neighboring tiles share the same color on their common edge.

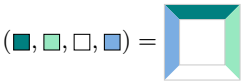


Figure 12.1: A Wang tile on $\mathbb{Z}^2$.

Definition 12.1. *Let C be a finite set of colors. On $\mathbb{Z}^2$, a* Wang tile *is an element $(c_{\text{North}}, c_{\text{East}}, c_{\text{South}}, c_{\text{West}}) \in C^4$, which is understood as a square tile whose edges are colored by colors of C.*

Wang tiles generalize to arbitrary dimensions $d \in \mathbb{N}$: instead of being 4-tuples of colors $c \in C$, they are 2^d-tuples of such colors.

Definition 12.2. *For C a finite set of colors, a* tileset *$T \subseteq C^{2^d}$ is a set of Wang tiles. Furthermore, a tileset $T \subseteq C^{2^d}$ defines a local subshift $X_T \subseteq T^{\mathbb{Z}^d}$, whose configurations are colorings $x \in T^{\mathbb{Z}^d}$ such that any two adjacent Wang tiles have the same color on their common edge.*

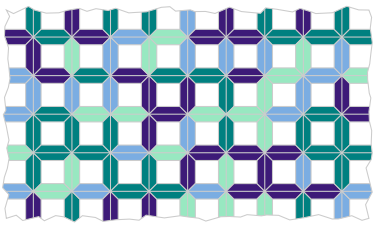


Figure 12.2: A configuration of Wang tiles.

Wang tiles appeared in [Wan61, Section 4.1] in order to reduce fragments of logic to geometrical $\mathbb{Z}^2$ tilings. It conjectured the decidability of the Domino problem (given a tileset T, is the resulting subshift non-empty?), and that all tilesets allow strongly periodic configurations. These were later disproved in [Ber64], which coined the term "Wang tiles" and proved the Domino problem undecidable.

[Wan61] Wang, "Proving theorems by pattern recognition – II".

[Ber64] Berger, "The undecidability of the Domino problem".

We mentioned in Proposition 3.42 that subshifts of finite type are conjugated to local subshifts. In fact, both are conjugated to Wang tilings:

Proposition 12.3. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a local subshift. There exists a tileset T on colors $C = \mathcal{A}^2$ such that X_T and X and conjugate.*

Sketch of proof. We devise the tiles of T to bear pairs of symbols $(a, b) \in \mathcal{A} \times \mathcal{A}$ encoding the local rule of X on their $(d-1)$-dimensional facets:

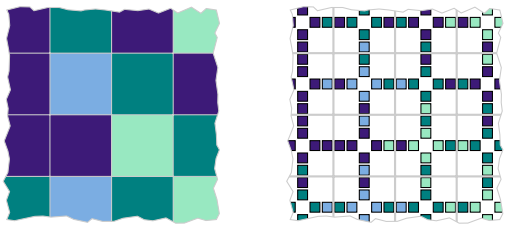


Figure 12.3: Transformation of configurations from X to X_T. □

12.2 Simulation

As mentioned in the introduction, the expanding simulation framework is based on a "fixpoint construction" introduced in a series of articles including [DRS08], [DRS10] and [DRS12]; themselves based on earlier work on self-simulating cellular automata [Gác86; Gác01]. For a more complete history on this fascinating construction, we refer to the extensive survey [Tör21].

[DRS08] Durand, Romashchenko, and Shen, "Fixed point and aperiodic tilings".

[DRS10] Durand, Romashchenko, and Shen, "Effective closed subshifts in 1D can be implemented in 2D".

[DRS12] Durand, Romashchenko, and Shen, "Fixed-point tile sets and their applications".

[Gác86] Gács, "Reliable computation with cellular automata".

[Gác01] Gács, "Reliable cellular automata with self-organization".

[Tör21] Törmä, "Fixed point constructions in tilings and cellular automata".

The whole fixpoint construction is built upon the notion *simulations* between tilesets:

Definition 12.4. *Let T and S be two tilesets.* A simulation *of T by S with zoom factor N is an injective map $\phi : T \to S^{N\times N}$, whose image consists of locally valid S-patterns of size $N \times N$ called* macro-tiles, *such that:*

- *For any two tiles $t_1, t_2 \in T$, the horizontal (resp. vertical) concatenation $t_1 t_2$ is locally valid in T if and only if the horizontal (resp. vertical) concatenation $\phi(t_1)\phi(t_2)$ is locally valid in S.*
- *For every valid S-tiling $s \in S^{\mathbb{Z}^2}$, there exists a unique decomposition of s in an infinite grid of $N \times N$* macro-tiles *$s|_{(Ni,Nj)+[\![N]\!]^2} \in \phi(T)$.*

Intuitively, a simulation of T by S is a bijection between T-tilings and S-tilings (up to a zoom factor N) which maps the tiles of T to valid patches of $N \times N$ patches of S-tiles called *macro-tiles*.

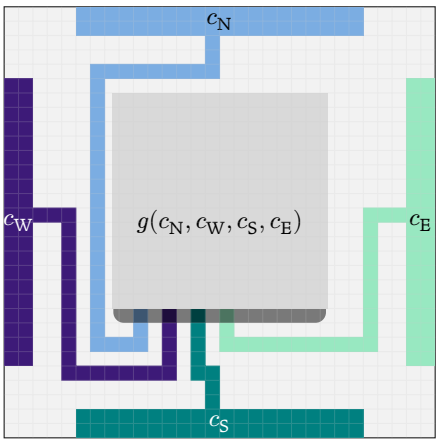
12.3 Overview of the construction

As we will build upon the fixpoint construction in this thesis, this chapter recalls from [DRS08] how it can be used to build an expanding sequence of simulating tilesets. Before we begin the proof, we briefly summarize the main principles involved:

Simulating tiles from a fixed tileset For a tileset T with colors $\mathcal{C} \subseteq \{0,1\}^*$, we introduce the function $g\colon (\{0,1\}^*)^4 \to \{\top, \bot\}$ that *recognizes it*:

Definition 12.5. *A function $g\colon (\{0,1\}^*)^4 \to \{\top, \bot\}$ is said to* recognize *the tileset $T \subseteq (\{0,1\}^*)^4$ if $g(c_N, c_W, c_S, c_E) = \top$ if and only if $(c_N, c_W, c_S, c_E) \in T$.*

The key idea of the proof will be, for a given tileset T, to build a tileset S whose macro-tiles will implement colors on their borders; and embed the computations of the function recognizing the tileset T that we want to simulate on the colors that appear on their borders, as to ensure that each macro-tile really simulates a tile of T.



On the borders of the macro-tile, areas called "macro-colors" contain the colors of a tile $(c_N, c_W, c_S, c_E) \in (\{0,1\}^*)^4$. Wires transport the colors to the middle area, which embeds the computations of $g(c_N, c_W, c_S, c_E)$ and verifies that $(c_N, c_W, c_S, c_E) \in T$.

Figure 12.4: Schematics of a macro-tile.

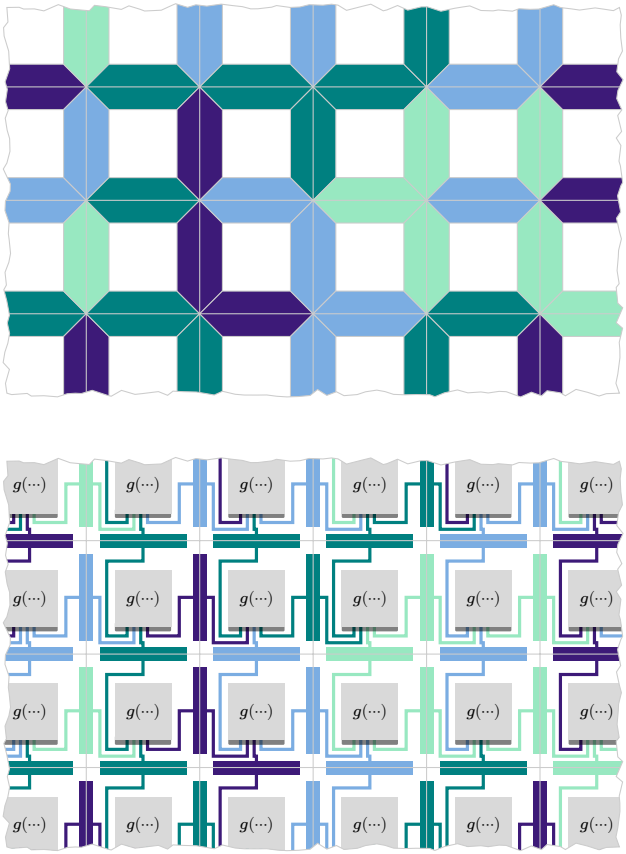


Figure 12.5: A configuration of X_T, and the corresponding configuration of macro-tiles (up to zoom factor N).

The zoom factor N with which S simulates T will depend on the time complexity of running the computations of g; but in the end, for a given T, there will exist some $N \in \mathbb{N}$ such that the tileset S we sketched above simulates T with zoom factor N.

A sequence of tilesets that simulate each other To build a sequence of tilesets $(T_\ell)_{\ell\in\mathbb{N}}$ such that each tileset simulates $T_{\ell+1}$, notice that the method above goes in the wrong direction: we build the simulating tileset S from the tileset T, and not the other way around[118].

[118] In other words, we "build T_ℓ from the tiles of $T_{\ell+1}$".

The solution consists in working with the functions that recognizes these tilesets instead of the tilesets themselves. More precisely, let $e \in \{0,1\}^*$ be a code describing a function $\varphi_e : \mathbb{N} \times (\{0,1\}^*)^4 \to \{\top, \bot\}$; and consider the tileset $\{(c_N, c_W, c_S, c_E) \in (\{0,1\}^*)^4 : \varphi_e(\ell+1, c_N, c_W, c_S, c_E) = \top$; which, by happenstance, we denote $T'_{\ell+1}$. In the previous paragraphs, we sketched a tileset T'_ℓ that simulates $T'_{\ell+1}$ with some zoom factor N_ℓ; and we denote $c_N, c_W, c_S, c_E \mapsto f(e, \ell, c_N, c_W, c_S, c_E)$ the function that recognizes the tileset T'_ℓ: indeed, this tileset only depends of N_ℓ – which we assume is computed from $\ell \in \mathbb{N}$ – and of the code e describing $T'_{\ell+1}$.

With these notations, we can actually apply the **fixpoint theorem** (Proposition 4.30): there exists some code $e \in \{0,1\}^*$ such that $f(e, \ell, \cdots) = \varphi_e(\ell, \cdots)$. But then, for well-behaving sequences of zoom factors $(N_\ell)_{\ell\in\mathbb{N}}$, the tilesets $T_\ell = \{c_N, c_W, c_S, c_E \in (\{0,1\}^*)^4 : \varphi_e(\ell, c_N, c_W, c_S, c_E) = \top\}$ will follow [DRS08] and form a sequence of tilesets $(T_\ell)_{\ell\in\mathbb{N}}$ such that each T_ℓ simulates $T_{\ell+1}$ with zoom factor N_ℓ once ℓ is large enough. In particular, there exists some ℓ large enough such that, for all $\ell' \geq \ell$, the tileset T_ℓ will simulate $T_{\ell'}$ with zoom factor $N = \prod_{k=\ell}^{\ell'-1} N_k$.

12.4 Building expanding simulating tilesets

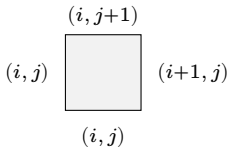
We now formally develop the proof sketch of the previous section to build sequences of simulating tilesets with expanding zoom factors. In what follows, we fix an increasing sequence of integers $(N_\ell)_{\ell\in\mathbb{N}}$. Restrictions on the growth of $(N_\ell)_{\ell\in\mathbb{N}}$ will be added along the proof.

Additionally, all the colors that we consider for these tilesets will be binary strings of $\{0,1\}^*$, and represent records made of various data fields: for convenience, these fields have been named instead of numbered.

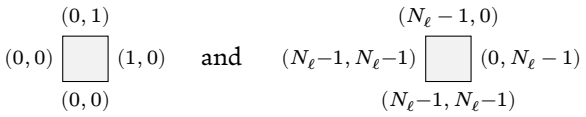
12.4.1 Macro-tiles

By definition of a simulation, any tileset T_ℓ that simulates another $T_{\ell+1}$ with zoom factor N_ℓ must verify the following property: each valid T_ℓ must have a unique decomposition into a regular grid of $N_\ell \times N_\ell$ *macro-tiles*, *i.e.* $N_\ell \times N_\ell$ locally valid blocks of T_ℓ-tiles.

To ensure this, we make each tile of T_ℓ contain its coordinates (modulo $N_\ell \times N_\ell$) in its colors, so that adjacent T_ℓ-tiles must organize themselves into $N_\ell \times N_\ell$ blocks. More precisely, we add a *position field* in the colors of T_ℓ-tiles that contains a pair of integers $(i,j) \in [\![N_\ell]\!] \times [\![N_\ell]\!]$. Furthermore, we force the tiles of T_ℓ to have the following form (where all coordinates are taken modulo (N_ℓ, N_ℓ)):



so that valid T_ℓ configurations are made of $N_\ell \times N_\ell$ blocks organized neatly into a regular grid, each having respectively tiles



at their South-West and North-East corners. In the whole proof, these blocks are called *macro-tiles*.

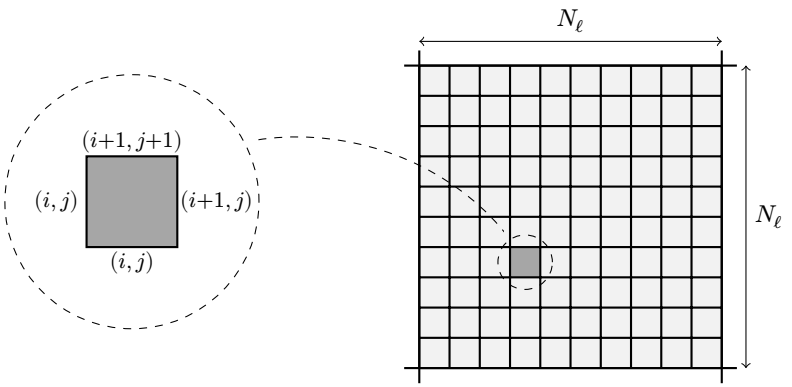


Figure 12.6: Structure of a macro-tile.

12.4.2 Macro-colors

To make macro-tiles actually simulate the tiles of $T_{\ell+1}$ instead of blank tiles, we now add *macro-colors* to the macro-tiles formed by groups of T_ℓ-tiles. Recall that all colors are assumed to be (through encoding) binary strings.

Since any tileset $T_{\ell+1}$ must be finite, the tiles of $T_{\ell+1}$ involve only finitely many colors. Assuming that these colors are binary strings of length, say, $\mathrm{len}_{\ell+1} \in \mathbb{N}$, we choose $\mathrm{len}_{\ell+1}$ tiles of T_ℓ along each of the four edges of a macro-tile to bear an additional *color field* consisting of a a single bit: along any edge of a macro-tile, the concatenation of the color field of these color fields will form a string of $\{0,1\}^{\mathrm{len}_{\ell+1}}$ called a *macro-color*.

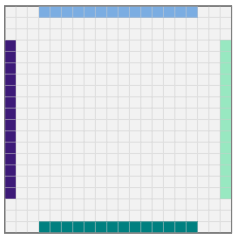


Figure 12.7: Areas containing the macro-colors in a macro-tile.

Remark. *We are very informal in this description*[119]*, as we did not specify the positions of these macro-colors nor their length* $\mathrm{len}_{\ell+1} \in \mathbb{N}$*. One important point is that whether the "color field" of a* T_ℓ*-tile is blank or actually contains a bit of information should only depend on its position in the macro-tile (for one, it must be on the border of a macro-tile; one could choose, for example, to pick the* len_ℓ *middle tiles along an edge). See Remark "Computable layout".*

[119] Somehow, though the reader may disagree, we found that full formalism does not help making the exposition clearer.

By having the bits of macro-colors facing the outer side of the macro-tiles, we ensure that adjacent macro-tiles must have the same macro-color along their common edge. In particular, at this point of the construction, T_ℓ simulates the tileset $(\{0,1\}^{\mathrm{len}_{\ell+1}})^4$ with zoom factor N_ℓ.

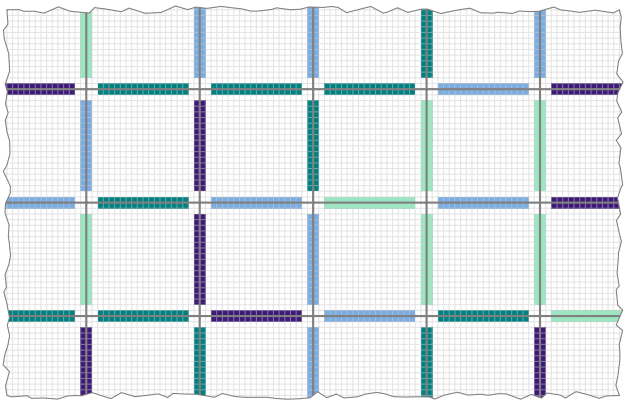
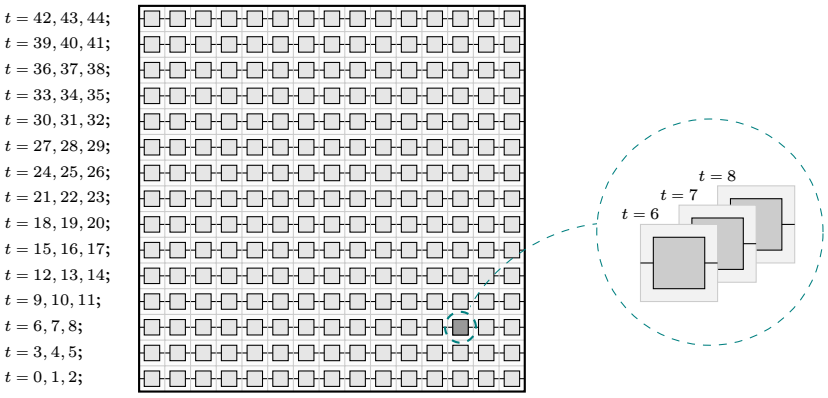


Figure 12.8: Adjacent macro-tiles with their macro-colors.

12.4.3 Computation layer

In order to restrict the tuples of macro-colors that can appear on the edges of the macro-tiles to a given subset of $(\{0,1\}^{\mathrm{len}_\ell})^4$, we provide these macro-tiles with some embedded computations, which are implemented by adding another field in the colors of T_ℓ.

Consider the mesh-connected computer described in Theorem 11.8: with a grid of N^{d-1} processors, it simulates N^{d-1} steps of RAM computation with a constant factor of time overhead. Following Section 11.3, we embed its space-time diagrams into macro-tiles of size $N_\ell \times N_\ell$.



3 time steps are packed on each line, resulting in $3 \cdot N_\ell$ steps of MCMC computations being contained inside this space-time diagram of size $[\![N_\ell]\!]^2$.

Figure 12.9: The computation layer of a macro-tile. In dimension $d = 2$, it draws the space-time diagram of a one-dimensional MCMC of size N_ℓ for $3 \cdot N_\ell$ steps of computation.

More precisely, let us consider the MCMC program $e_{\text{RAM}} \in \{0,1\}^*$ that simulates log-RAM programs with constant factor overhead as in Theorem 11.8. The processors of the associated MCMC of size N_ℓ contain:

- The RAM program $e_{\text{RAM}} \in \{0,1\}^*$, which is the code executed by the processor;
- A special variable `pos` that stores the position of the processor in the MCMC array;
- A special variable `program` that contains the code of a log-RAM program $e \in \{0,1\}^*$ that is to be simulated by the whole MCMC;
- A special variable `input` that is interpreted as the $\texttt{pos}^{\text{th}}$ input word of a log-RAM input array;
- Finitely many variables $\texttt{var}_i \in \{0,1\}^*$ of length at most $O(\log N_\ell)$.

To embed the space-time diagram of a mesh-connected computer operating on N_ℓ processors in time $O(N_\ell)$, we add to each of the $N_\ell \times N_\ell$ tiles in a macro-tile a *processor field* containing $O(1)$ consecutive states of an MCMC processor[120] that operate on words of size $O(\log N_\ell)$. All the processor fields should, together, draw a space-time diagram of N_ℓ MCMC processors on $O(1) \cdot N_\ell$ time steps.

[120] Why not just a single processor state? Because the simulation of log-RAM programs from Theorem 11.8 induces a $O(1)$-time overhead, so that we need to "linearly compress" the space-time diagram along the time direction.

At this point of the proof, macro-tiles can embed N_ℓ computation steps of an arbitrary log-RAM program. For these computations to correctly simulate a given tileset, let e be a log-RAM program such that $\varphi_e : (\{0,1\}^*)^* \to \{\top, \bot\}$ is a function recognizing a tileset T; and let us write e in the variable `program` of the $O(1)$ processor states appearing in each tile.

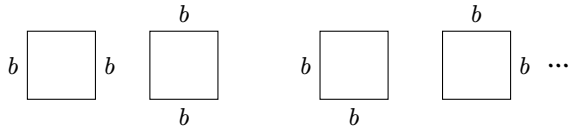
Since T is finite, there exists some time bound $b \in \mathbb{N}$ such that that e terminates on every $t \in T$ in time less than b. If N_ℓ is larger than b, so that every computation of e on the tiles of T has enough time to accept: then the MCMC admits an accepting run on a given macro-color $(c_{\text{N}}, c_{\text{W}}, c_{\text{S}}, c_{\text{E}})$ if and only if it $(c_{\text{N}}, c_{\text{W}}, c_{\text{S}}, c_{\text{E}})$ is accepted by a run of the log-RAM program e, *i.e.* if and only if $(c_{\text{N}}, c_{\text{W}}, c_{\text{S}}, c_{\text{E}}) \in T$.

12.4.4 Wiring the macro-colors to the computation layer

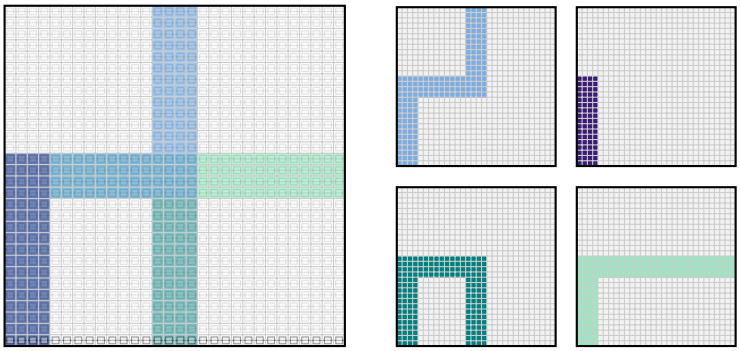
To make macro-tiles and their computation layer correctly simulate other tilesets, we need to ensure that the input array of their computation layer contains an exact copy of their macro-colors; in other words, the macro-colors need to be carried from the edges of the macro-tiles to the ir computation layer's input array.

To proceed, we build a system of *wires* into the tiles of T_ℓ by adding finitely many *color fields* to their colors, one per wire that goes through the tile:

- Each wire carries a single bit $b \in \{0,1\}$ from the border of the macro-tile to an MCMC state processor of time $t = 0$. In a cell, the wire can carry this bit in a straight line or taking a turn:



- Together, a set of contiguous wires forms a *cable*. A cable carries a whole macro-color from the corresponding edge of a macro-tile to the input array (time $t = 0$) of its computation layer.



The i^{th} bit of a macro-color is dispatched to the i^{th} processor of the MCMC. In particular, on $\mathbb{Z}^2$, each processor receives four bits: one per macro-color.

Figure 12.10: The wires in a macro-tile (altogether, and individually).

Remark. *Once again, we only sketch the wire layouts and do not formally specify the disposition of the wires in a macro-tile. The important point is that whether the* color fields *of a T_ℓ-tile are blank or carry a bit of information, and if they do, the directions from and to which they carry this bit, only depend of the position of the tile into the macro-tile. Once again, see Remark "Computable layout".*

12.4.5 Recursion theorem[121]

Summary of the construction At this point of the construction, for every level $\ell \in \mathbb{N}$, the colors used in the tiles of T_ℓ contain the following fields[122]:

- A *position field*, consisting of $2 \log N_\ell$ bits;
- $O(1)$ *wire fields*[123], consisting of 1 bits;
- $O(1)$ *processor fields*[124], consisting of $O(\log N_\ell)$ bits.

We denote by $\text{len}_\ell \in \mathbb{N}$ the total bit length of these fields.

Since we want T_ℓ to simulate $T_{\ell+1}$, we need to consider the possible values for the sequence $(N_\ell)_{\ell \in \mathbb{N}}$. Mainly, the colors of $T_{\ell+1}$ need to fit in the computation zone of T_ℓ tiles, which implies that $\text{len}_{\ell+1} = \log N_{\ell+1} \ll N_\ell$. For this proof, $N_\ell = 2^\ell$ is enough; though later proofs will need a faster-growing sequence. (We will also need the computations of the function recognizing $T_{\ell+1}$ to fit inside a macro-tile of size $N_\ell \times N_\ell$, see below).

Remark (Computable layout)**.** *The geometric distribution of wires and macro-colors in macro-tiles has not been described explicitely. Such a wire layout should verify some constraints: for example, only finitely many wires should be allowed to cross in any given cell.*

In what follows, we fix a wire layout that is computable in the following sense: given ℓ and $(i,j) \in [\![N_\ell]\!] \times [\![N_\ell]\!]$, the number of wires and their directions in the tile at position (i,j) in the macro-tile should be computable in time $\text{polylog}(N_\ell)$.

[121] Giving a clear and concise explanation of the fixpoint construction is a difficult writing exercise. After much reading, I decided to follow the exposition of [Wes17].

[Wes17] Westrick, "Seas of squares with sizes from a Π^0_1 set".

[122] And this, for every level $\ell \in \mathbb{N}$. Half the magic of the construction is happening here: we described a whole sequence of tilesets $(T_\ell)_{\ell \in \mathbb{N}}$ that are nearly entirely determined – thus, almost ready for simulation – up to the content of the `program` variables of their MCMC processors.

[123] We use the wire field for both transporting macro-colors from the edges to the input tape of a macro tile, and writing the macro-colors on the edges of a macro-tile.

[124] Containing $O(1)$ consecutive states of an MCMC processor in an array of size N_ℓ.

For exposition purposes, the rest of this chapter will assume that space is horizontal and time is vertical in space-time diagrams.

Recognizing the tilesets $(T_\ell)_{\ell\in\mathbb{N}}$ The tilesets $(T_\ell)_\ell$ that were built in the previous paragraphs can be recognized by a single function $f(e,\ell,c_N,c_W,c_S,c_E)$ uniformly, whose program acts as follows:

1. Compute N_ℓ;
2. Check that each c_N, c_W,... is a string of length len_ℓ;
3. Decode $(i,j)\in[\![N_\ell]\!]\times[\![N_\ell]\!]$ from the *position fields* of (c_N,c_W,c_S,c_E), *i.e.* compute the position of the tile in the macro-tile;
4. Check the *processor fields* of (c_N,c_W,c_S,c_E) to correctly contain $O(1)$ consecutive processor states (with variables of length $O(\log N_\ell)$), and check the content of their `pos` variables (which should contain the position i) and of their `program` variables (which should contain the code $e\in\{0,1\}^*$). Finally, check that the respective processor fields of the four colors (c_N,c_W,c_S,c_E) draw a valid symbol of MCMC space-time diagram;
5. Check the *wire* fields of (c_N,c_W,c_S,c_E) to obey the wire layout for the tile at position (i,j) in a macro-tile of dimension $N_\ell\times N_\ell$;
6. (*Checking the input array*) If the tile (i,j) contains an initial processor state for time $t=0$, check the variables appearing in the *processor fields*: if `pos` $=0$, check that `input` has value $\ell+1$; if `pos` >0, check that `input` is consistent with the four bits appearing in the four *wire fields*.
7. If all checks were valid, accept. Otherwise, reject.

Notice that, for any fixed code $e\in\{0,1\}^*$, taking $N_\ell=2^\ell$ allows to define a log-RAM algorithm[125] for f that runs in time $\mathrm{polylog}(N_\ell)$ on inputs (e,ℓ,c_N,c_W,c_S,c_E).

[125] Since we operate with log-RAM algorithms, should briefly mention how the input is distributed into an input array: we write e in $I[0]$, ℓ in $I[1]$, and the argument (c_N,c_W,c_S,c_E) is written on all $I[j]$ for $j\geq 2$, four bits at a time (on $\mathbb{Z}^2$: one per component of (c_N,c_W,c_S,c_E)).

Fixpoint theorem As mentioned in the overview of the whole expanding tileset construction, we make T_ℓ simulate $T_{\ell+1}$ by applying the fixpoint theorem Proposition 4.30: there exists the code of a log-RAM program $e\in\{0,1\}^*$ such that:

- $\varphi_e(\ell,c_N,c_W,c_S,c_E)=f(e,\ell,c_N,c_W,c_S,c_E)$ for all ℓ,c_N,c_W,c_S,c_E;
- Executions of φ_e and $f(e,\cdot,\cdot)$ have the same running time (up to a constant factor).

By defininining

$$T_\ell=\{(c_N,c_W,c_S,c_E)\in(\{0,1\}^*)^4\colon \varphi_e(\ell,c_N,c_W,c_S,c_E)=\top\},$$

one sees that T_ℓ simulates $T_{\ell+1}$ with zoom factor N_ℓ once the running time of $\varphi_e(\ell+1,c_N,c_W,c_S,c_E)$ (which is $\mathrm{polylog}(N_{\ell+1})$) is strictly smaller than N_ℓ, which – by our choice of N_ℓ – eventually holds for every ℓ that is larger than some, say, ℓ_0.

Final word

For ℓ large enough, the tilings of T_ℓ follow an inductive structure: the tiles of T_ℓ organize themselves into *macro-tiles* in T_ℓ-configurations, these macro-tiles corresponding to $T_{\ell+1}$-tiles that, themselves, organize themselves into *macro-macro-tiles*...

In what follows, we define *macro-tiles of level* $\ell\in\mathbb{N}$ any set of T_{ℓ_0}-tiles that is the image of a T_ℓ-tile by the iterated simulation map. By construction, and for any $\ell\geq\ell_0$, macro-tiles of level $\ell+1$ are composed of $N_\ell\times N_\ell$ macro-tiles of level ℓ, thus making each valid T_{ℓ_0}-configuration define a hierarchy of nested macro-tiles.

At this point, the tileset T_{ℓ_0} has little interest, apart from building aperiodic configurations[126]. However, there is a lot of available "free time" inside the computation layer of the macro-tiles: e runs in time polylog$(N_{\ell+1})$ inside macro-tiles of level ℓ, but the latter contains the computations of an MCMC of size N_ℓ that can simulate up to N_ℓ steps of RAM computations. By embedding more involved computations, the expanding simulation framework allows for many applications (see *e.g.* [DRS12; Wes17, ...]). In particular, in the next Chapter 13, we use the available time and space to implement all the computations of representations and inductions required in Theorem 10.11.

Remark 12.6 (Generalization to $\mathbb{Z}^d$)**.** *We suggest a straightforward adaptation of the previous construction to tilings of $\mathbb{Z}^d$: since d-dimensional macro-tiles have $2d$ adjacent neighboors, they will have $2d$ macro-colors that form a tuple which we denote* **c***. The macro-colors of level $\ell+1$ are strings of length at most N_ℓ^{d-1}, as they need to fit on a $(d-1)$-dimensional facet; and macro-tiles of level ℓ embed, in all their processor fields, $O(N_\ell)$ steps of computations from a mesh-connected computer of size N_ℓ^{d-1}.*

[126] All T_{ℓ_0}-configurations are aperiodic because any period would need to shift any ℓ-level macro-tile to another ℓ-level macro-tile, which requires shifts of unbounded lengths.

[DRS12] Durand, Romashchenko, and Shen, "Fixed-point tile sets and their applications".

[Wes17] Westrick, "Seas of squares with sizes from a Π^0_1 set".

13 Proof of Theorem 10.11

This chapter contains the proof of Theorem 10.11. This proof is developed within the expanding tileset setting from Section 12.4, with *computation layers* implementing mesh-connected multicomputers from Chapter 11.

Essentially, for a given representation function $\mathcal{R}$, we make all macro-tiles contain a representation of the pattern that appears below it, and inductively check the consistency of these representations by computing the associated induction $\mathcal{I}$ from children to parent macro-tiles.

In this chapter, we prove Theorem 10.11. Let us fix a representation function $\mathcal{R} : \mathcal{A}^{*d} \to \{0,1\}^*$ and an induction $\mathcal{I} \subseteq (\{0,1\}^*)^{2^d} \times \{0,1\}^*$ for $\mathcal{R}$. We also fix $\langle\mathcal{I}\rangle \in \{0,1\}^*$ a log-RAM algorithm for $\mathcal{I}$, and $\alpha \in \mathbb{Q}_+$ and $\beta \in \mathbb{Q}_+$ verifying the hypotheses of Theorem 10.11[127] on $\langle\mathcal{I}\rangle$: in particular, we have $\alpha < d-1$ and $\alpha \cdot \beta < d-1$. In the proof, we also denote $\gamma = \frac{\alpha\cdot\beta}{d-1} < 1$.

[127] In Theorem 10.11, α and β were real numbers. Up to taking α and β slightly larger, we can assume they are rational.

We will build a tileset T whose valid tilings form an SFT that factors onto $X_{\mathcal{R},\mathcal{I}}$. The tiles of T will be designed as in the expanding tileset construction, but we considerably extend the macro-colors so that all macro-tiles will check the inductive validity of the "pixel pattern" on which they sit.

To fix notations, the tileset T_ℓ will simulate $T_{\ell+1}$ with zoom factor N_ℓ for every $\ell \in \mathbb{N}$.[128] We also denote $L_\ell = \prod_{j=0}^{\ell-1} N_j$ the *pixel zoom factor*, *i.e.* the zoom factor at which T_0 simulates T_ℓ.

[128] Setting $N_\ell = 2^{2^{2^\ell}}$ will satisfy all later requirements.

13.1 Overview of the construction

(For clearer exposition, the following paragraphs focus on the case $d = 2$. The construction is then written in full generality on arbitrary $\mathbb{Z}^d$.)

Reprensenting patterns in macro-tiles Before proving Theorem 10.11, let us briefly sketch the proof on $\mathbb{Z}^2$ to clarify the ideas involved. Mainly, our construction will consists in:

1. Superimposing the configurations of $\mathcal{A}^{\mathbb{Z}^2}$ with the configurations of a tileset T_0 obtained by the expanding simulation construction (*i.e.* that simulates some tileset $T_1, T_2, \ldots$);
2. Make the macro-tiles of level ℓ contain a representation of the pattern of size $L_\ell \times L_\ell$ (and on the alphabet $\mathcal{A}$) upon which they sit.

By computing the representations inductively from children to parent macro-tiles, we will ensure that only inductively valid patterns of $\mathcal{A}^{*2}$ can appear below a macro-tile. This will result in a subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^2} \times T_0^{\mathbb{Z}^2}$ that is local, because T_0 is a tileset; and whose natural projection to $\mathcal{A}^{\mathbb{Z}^2}$ yields the desired subshift $X_{\mathcal{R},\mathcal{I}}$.

The main obstacle in implementing this proof sketch appears in the inductive computations of the representations: since a parent macro-tile is composed of $N_\ell \times N_\ell$ children macro-tiles, but the induction $\mathcal{I}$ only computes representations for 2×2 adjacent subrepresentations, we actually have to embed several consecutive levels of induction inside a parent macro-tile,

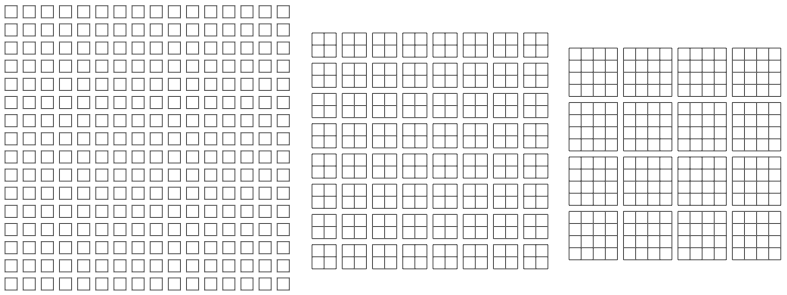


Figure 13.1: A parent macro-tile (drawn as $N_\ell \times N_\ell$ children) and its division into groups of 2×2 and 4×4 children macro-tiles.

while the time-complexity of a single induction step is already nearing the computation capacities of the macro-tiles.

A hierarchy of nested computations By taking N_ℓ to be a power of 2, a natural recursive structure appears to inductively compute a representation of an $L_{\ell+1} \times L_{\ell+1}$ pattern from $N_\ell \times N_\ell$ representations of its $L_\ell \times L_\ell$ subpatterns[129]:

- Divide the parent macro-tile into $\frac{N_\ell}{2} \times \frac{N_\ell}{2}$ groups of 2×2 children macro-tiles. Since each of these children macro-tile possesses one representation from a pattern of size $L_\ell \times L_\ell$, each of these 2×2 groups of tiles can compute a step from the induction $\mathcal{I}$. As a result, the parent macro-tile possesses $\frac{N_\ell}{2} \times \frac{N_\ell}{2}$ representations of patterns of size $2L_\ell \times 2L_\ell$, which we call "intermediate representations".
- Then, divide the parent macro-tile into $\frac{N_\ell}{4} \times \frac{N_\ell}{4}$ groups of 4×4 children macro-tiles. Since each of these 4×4 tiles cover 2×2 intermediate representations from the previous item, each of these 4×4 groups of tiles can compute a step from the induction $\mathcal{I}$. As a result, the parent macro-tile possesses $\frac{N_\ell}{4} \times \frac{N_\ell}{4}$ representations of patterns of size $4L_\ell \times 4L_\ell$, which form newer and larger "intermediate representations".
- ...
- After $\log N_\ell$ divisions of the parent macro-tile into successive groups of $2^k \times 2^k$ children macro-tiles, the $N_\ell \times N_\ell$ children macro-tile collectively hold a representation from a pattern of size[130] $L_{\ell+1} \times L_{\ell+1}$. By definition of the induction $\mathcal{I}$, this representation is a representation of the pattern covered by the parent macro-tile, and should be communicated to its computation layer.

[129] For clarity, recall that L_ℓ is the pixel size of a macro-tile, and that $L_{\ell+1} = L_\ell \cdot N_\ell$.

[130] Indeed, $2^{\log N_\ell} \cdot L_\ell = L_{\ell+1}$.

Inductively validity The previous paragraphs will correctly check that patterns covered by macro-tiles admit a valid representation. To complete the proof, we will need to consider condition (iii) from the definition of "inductive validity" (Definition 10.5):

- *For every* $0 \leq k < n$ *and every square* $I = \mathbf{i} + [\![2]\!]^2 \subseteq [\![2^{n-k}]\!]^2$ *indexing* 2×2 *adjacent squares* $(C^k_{\mathbf{i}})_{\mathbf{i} \in I}$*, there exists* $r \in \{0,1\}^*$ *such that* $\mathcal{I}((r^k_{\mathbf{i}})_{\mathbf{i} \in I}, r)$.

where $C^k_{\mathbf{i}}$ is the element at position $\mathbf{i} \in [\![2^{n-k}]\!]$ in the partitionning of $[\![2^n]\!]^2$ into squares of size $2^k \times 2^k$. Intuitively, this condition checks the validity of arbitrary squares of 2×2 adjacent intermediate representations $r^k_{\mathbf{i}}$.

To implement this condition in our construction, we will make every set of 2×2 groups of $2^k \times 2^k$ children macro-tiles from the previous paragraphs perform *a single* step of induction $\mathcal{I}$ on the intermediate representations they hold.

The chapter implements this proof sketch on arbitrary $\mathbb{Z}^d$ for $d \geq 2$.

13.2 Distributed computations and subarray computation layers

In the expanding tileset construction described in Chapter 12, a parent macro-tile embedded in its *computation layer* the whole space-time diagram of a mesh-connected multicomputer of size N_ℓ^{d-1} for $O(N_\ell)$ steps of computation. In this section, we provide an implementation of *distributed MCMC computations* accross several macro-tiles: these new *computation layers* will contain a subarray of MCMC computations, and corresponding fields will be added to the macro-colors to ensure the consistency of these computations accross adjacent macro-tiles.

13.2.1 Subarrays of MCMC computations

Let us fix some $M_{\ell+1} \in \mathbb{N}$ (whose value will be chosen later[131]). Among the $[\![N_\ell]\!]^d$ children macro-tiles composing a parent macro-tile of level $\ell+1$, we pick a block of $[\![M_{\ell+1}]\!]^d$ adjacent children macro-tiles to contain *subarray computation layers*. More precisely:

- Each children tile will contain $2\log N_{\ell+1}$ *processor fields* in their macro-colors[132], in order to form together as many *subarray computation layers* of the parent macro-tile.
- Each *processor field* contains the states of an MCMC processor during $O(1)$ consecutive steps of computations. In the *processor field* of index $1 \le k \le \log N_{\ell+1}$, the MCMC processor operates on variables of word length $O(\log(2^k \cdot M_{\ell+1}))$, in accordance with the size of the mesh-connected computers whose space-time diagrams will be distributed.

[131] Since such a subarray will need to contain at least a complete computation of the induction $\mathcal{I}$ on the representations of patterns of size $[\![L_\ell]\!]^d$, picking $M_\ell = O(L_\ell^\gamma)$ will be enough.

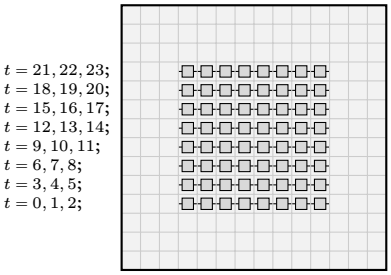


Figure 13.2: An MCMC subarray computation of size $M_\ell \times M_\ell$ inside a macro-tile.

[132] The precise number $2\log N_{\ell+1}$ will be justified later.

13.2.2 Wiring subarrays of adjacent macro-tiles

To continue the computations of the mesh-connected computers seamlessly accross the subarrays of adjacent macro-tiles, we draw a system of wires from the borders of the subarrays to new macro-color fields of the parent macro-tile. More precisely:

- For each subarray computation layer of index $1 \le k \le \log N_{\ell+1}$, we add a corresponding *processor wire field* of size $O(\log(2^k \cdot M_{\ell+1}))$ that can transport $O(1)$ processor states.

  Together, these processor wire fields should draw cables from the borders of the subarrays to the macro-colors of the parent macro-tile.
- For each subarray computation layer of index $1 \le k \le \log N_{\ell+1}$, we add a corresponding *subarray computation field* to the parent macro-tile. It contains $O(M_{\ell+1}^{d-1})$ processor states, which is the amount of processor states appearing on a facet[133] of the subarray of size $[\![M_{\ell+1}]\!]^d$.

[133] As noticed in Note 11.10, space-time diagrams of mesh-connected computers are local.

Remark. *Be mindful that the two items operate on distinct levels of macro-colors: the wires and the processors appear in the macro-colors of the* children *macro-tiles (of level ℓ), and together draw the abstract* subarray computation layers *of the parent macro-tile. On the other hand, the* subarray computation fields *appear in the macro-colors of the* parent *macro-tile*[134] *(of level $\ell+1$).*

[134] Since we define all levels of macro-tiles simulataneously, this paragraph only means that macro-tiles of level ℓ only contain $2\log N_\ell$ such fields, and not $2\log N_{\ell+1}$.

Note 13.1. *In the cases where a subarray appears on the border of the mesh-connected multicomputer, the continuity of the computation should not be enforced in the corresponding direction.*
As it is useful to have the borders of a macro-tile's subarrays appear in its macro-colors (so that its main computation layer can use it in its computations, for example), we actually make each subarray computation field *contain the borders of both adjacent subarrays, so that continuity of the computation is not enforced by default (and will, thus, be left to the discretion of the macro-tile's computation layer). See Paragraph 'Clustering and induction computations' on page 120.*

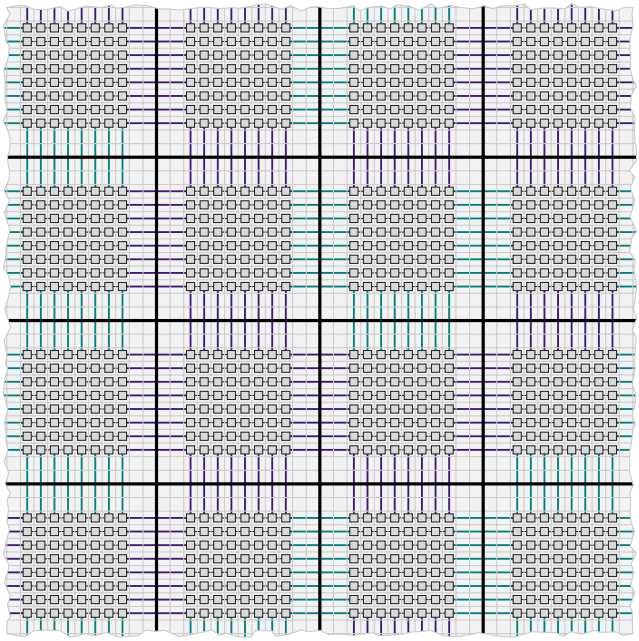


Figure 13.3: Distributing the computations of a mesh-connected multicomputer accross the subarrays of adjacent macro-tiles.

At this point of the proof, each children macro-tile (of level ℓ) contains $2 \log N_\ell$ *subarray computation layers* of indices $k \in [1 \,..\, 2 \log N_\ell]$. Conceptually, we group all the subarrays of the same layer $k \in [1 \,..\, 2 \log N_\ell]$ together and say that the reunion of all the subarray computation fields of index k from all the children macro-tiles forms the k^{th} *distributed computation layer* of their parent macro-tile.

13.3 Implementation of inductive representations in macro-tiles

Now that macro-tiles possess additional computation capacities, we can implement the computations checking the inductive validity of the pattern that will be covered by the macro-tiles.

13.3.1 Adding representations to the input arrays

Since we want macro-tiles of level ℓ to contain a representation of the pattern of size $[\![L_\ell]\!]^d$ they cover, we begin by adding a new parameter $r \in \{0,1\}^*$ to the function computed by the macro-tiles of level ℓ. In other words, a well-formed input array should now encode a tuple of the form

$$(e, \ell, r, \mathbf{c}).^{135}$$

[135] Where $\mathbf{c}$ is a tuple of $2d$ macro-tiles.

As said parameter $r \in \{0,1\}^*$ is supposed to contain a representation of a pattern of size $[\![L_\ell]\!]^d$, by our hypothesis on the lengths of the representations given by the representation function $\mathcal{R}$, we can actually bound the length of r by $O(L_\ell^\alpha)$.

At this point of the construction, a parent macro-tile (of level $\ell + 1$) is composed of N_ℓ^d children macro-tiles (of level ℓ) that all contain a representation. The rest of the construction essentially consists in ensuring that the representation of a parent macro-tile is consistent with the representations of its children by implementing $\log N_\ell$ successive levels of inductions.

13.3.2 Addressing scheme

In order to describe the implementation of these $\log N_\ell$ successive levels of induction, we need to define a computable addressing scheme of the children macro-tiles inside their parent.

Layering In the overview of the construction, we sketched that parent macro-tiles (of level $\ell+1$) will need to implement $\log N_\ell$ successive levels of induction computations $\mathcal{I}$. To this end, we use $\log N_\ell$ levels of *layers* from the parent macro-tiles.

More precisely, recall that we call *distributed computation layer* of index $1 \leq k \leq 2\log N_\ell$ the reunion of the *subarray computation layers* of index k from all the children macro-tiles. We will implement the k^{th} level of inductions by using the *distributed computation layer* of index k.

Clustering We also sketched in this overview that the k^{th} level of inductions should group children macro-tiles into groups of $(2^k)^d$ adjacent tiles; and that the next level of inductions should merge 2^d adjacent groups of the previous level together[136], to form groups of $(2^{k+1})^d$ children macro-tiles.

To this end, we define an addressing scheme that we call *clustering*, which consists in addressing the groups of children macro-tiles, and addressing the position of said children macro-tiles inside their groups. This definition is reminiscent of Note 10.4, because it implements the very same idea[137]:

[136] Which corresponds to the subrepresentation inputs of the induction $\mathcal{I}$.

[137] Thus, we naturally use the same notations.

Definition 13.2. *Consider a children macro-tile (of level ℓ) at position $\mathbf{i} \in [\![N_\ell]\!]^d$ inside its parent. For an index $0 \leq k \leq \log N_\ell$, we say that:*

- *The children tile $\mathbf{i}$ belongs to the cluster $C^k_{\mathbf{i}'}$ of index $\mathbf{i}' = \left\lfloor \frac{\mathbf{i}}{2^k} \right\rfloor$;*
- *The position of the children tile $\mathbf{i}$ in its cluster $C^k_{\mathbf{i}'}$ is $\mathbf{i} - 2^k \cdot \mathbf{i}'$.*

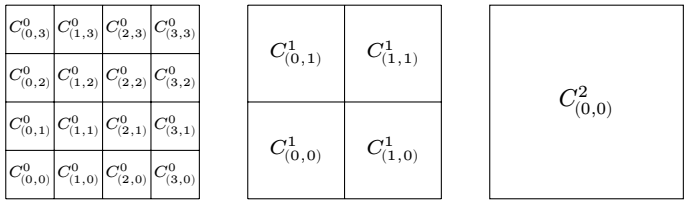


Figure 13.4: Clustering of the first three levels of computations.

This clustering system provides the correct partitionning of the distributed computation layers into nested groups of $(2^k)^d$ adjacent children macro-tiles:

Claim. *Fix a distributed computation layer of index $1 \leq k \leq \log N_\ell$:*

(i) A cluster $C^k_{\mathbf{i}'}$ is composed of $[\![2^k]\!]^d$ children macro-tiles;

(ii) Inside this layer, cluster indices range in $\left[\!\left[\frac{N_\ell}{2^k}\right]\!\right]^d$;

(iii) The cluster $C^k_{\mathbf{i}'}$ is the reunion of 2^d clusters from the layer of index $k-1$:

$$C^k_{\mathbf{i}'} = \bigcup_{\mathbf{j} \in [\![2]\!]^d} C^{k-1}_{2\cdot\mathbf{i}'+\mathbf{j}}.$$

13.3.3 Computing successive induction steps

We denote by $r_{\mathbf{i}}$ the representation that appears on the input array of the children macro-tile of index $\mathbf{i} \in [\![N_\ell]\!]^d$; and by $r \in \{0,1\}^*$ the representation that appears on the input array of the parent macro-tile. we want to ensure that r can be obtained by applying $\log N_\ell$ steps of the induction $\mathcal{I}$ on the representations $(r_{\mathbf{i}})_{\mathbf{i} \in [\![N_\ell]\!]^d}$ of the children macro-tiles.

Clustering and induction computations The clustering scheme divides the distributed computation layer of index $1 \leq k \leq \log N_\ell$ into blocks of $[\![2^k]\!]^d$ MCMC space-time subarrays. We implement the computations of the induction $\mathcal{I}$ into the children macro-tiles of these clusters as follows:

- Continuity of the space-time diagram is enforced accross adjacent children macro-tiles' *subarray computation layers* of index $1 \leq k \leq \log N_\ell$ if and only if they belong in the same cluster $C^k_{\mathbf{i}'}$.

  In other words, the clusters of the distributed computation layer of index $1 \leq k \leq \log N_\ell$ now define blocks of $[\![2^k]\!]^d$ subarrays forming the distributed space-time diagrams of mesh-connected multicomputers of size $[\![2^k \cdot M_\ell]\!]^{d-1}$ during $O(2^k \cdot M_\ell)$ computation steps.
- The processors implement the MCMC program $e_{\text{RAM}} \in \{0,1\}^*$ that simulates log-RAM programs with constant factor overhead from Theorem 11.8.
- Since each children macro-tile can compute its position $\mathbf{i} - 2^k \cdot \mathbf{i}'$ in its cluster, the children macro-tiles covering the time step $t = 0$ in the distributed space-time diagram check, in their *subarray computation field*, the value of the variable `pos`; and that the variable `program` contains the log-RAM program $\langle \mathcal{I} \rangle$.

At this point of the construction, the children macro-tiles in a cluster $C^k_{\mathbf{i}'}$ form, with their subarray computation layers of index k, the space-time diagram of size $[\![2^k M_\ell]\!]^d$ of a mesh-connected computer that simulates the induction $\mathcal{I}$ for $(2^k M_\ell)^{d-1}$ computation steps.

Communication between adjacent layers The nested structure of clusters between adjacent layers (any cluster from the k^{th} layer is the reunion of 2^d clusters from the $(k-1)^{\text{th}}$ layer[138]) follows the structure imposed by the induction $\mathcal{I}$: $(\{0,1\}^*)^{2^d} \times \{0,1\}^* \to \{\top, \bot\}$ (which is a predicate that associates representations of larger patterns from 2^d given representations of their subpatterns).

To ensure that the $\log N_\ell$ layers of distributed computations perform the desired $\log N_\ell$ levels of successive inductions, we wire the representation validated by each mesh-connected multicomputer of the $(k-1)^{\text{th}}$ distributed computation layer at position, say, $2 \cdot \mathbf{i}' + \mathbf{j}$ for $\mathbf{j} \in [\![2]\!]^d$, to the corresponding input argument of the MCMC of index $\mathbf{i}'$ in the k^{th} layer.

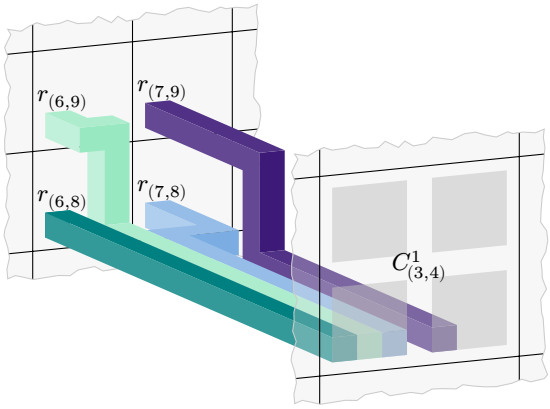


Figure 13.5: Wiring between children macro-tiles and the first layer.

More precisely, for each layer of index $1 \leq k \leq \log N_\ell$, we draw a system of wires that maps input arrays of a cluster's MCMC on the $(k-1)^{\text{th}}$ distributed computation layer to the corresponding argument in the input array of the parent cluster's MCMC on the k^{th} layer. These wires are drawn using a corresponding *subarray wire field* (one per layer of index $1 \leq k \leq \log N_\ell$), which will appear in some children macro-tiles[139].

[138] More precisely, we have:

$$C^k_{\mathbf{i}'} = \bigcup_{\mathbf{j} \in [\![2]\!]^d} C^{k-1}_{2 \cdot \mathbf{i}' + \mathbf{j}}$$

for $\mathbf{i}' \in \left[\!\left[\frac{N_\ell}{2^k}\right]\!\right]^d$.

Of course, in the actual figure in dimension $d = 2$, the computation layer of the children macro-tiles and the clusters of the first layer are actually superimposed. The view here is exploded to make the reading easier.

[139] As wires have already been used inside the expanding tileset construction, the implementation of these new wires should hopefully be clear.

Remark. *The representations given as input (resp. validated) in the clusters of the $(k-1)^{th}$ layer all have length $O(2^{k-1} \cdot L_\ell)^\alpha$ (resp. $O(2^k \cdot L_\ell)^\alpha$) by our hypothesis on the lengths of the representation function $\mathcal{R}$. In particular, the starting and ending position of the associated cables are computable, and there exists efficiently computable wiring layouts for all the associated $\log N_\ell$ layers.*

For a fixed cluster $C^k_{\mathbf{i}'}$, the children macro-tiles composing $C^k_{\mathbf{i}'}$ contain, on their k^{th} subarray computation layer, the distributed space-time diagram of an MCMC that validates a representation, which we denote $r^{(k)}_{\mathbf{i}'}$. With these *subarray wire fields*, we just ensured that this representation is obtained by k iterated computations of the induction $\mathcal{I}$ on all their representations $r_{\mathbf{i}}$ for $\mathbf{i} \in 2^k \cdot \mathbf{i}' + [\![2^k]\!]^d$.

Communication to the parent macro-tile The last distributed computation layer of index $k = \log N_\ell$ contains, by construction, the space-time diagram of a mesh-connected multicomputer of size $[\![N_\ell \cdot M_\ell]\!]^d$ that computes the induction $\mathcal{I}$, distributed among all the $[\![N_\ell]\!]^d$ children macro-tiles.

To ensure that the representation r of the parent macro-tile is obtained from the representations $(r_{\mathbf{i}})_{\mathbf{i} \in [\![N_\ell]\!]^d}$ of the children macro-tiles, we add wires between the input array of the parent macro-tile and the input array of this mesh-connected multicomputer appearing in the distributed computation layer of index $k = \log N_\ell$. As before, we implement these wires by adding a *subarray-to-parent wire field* to some[140] of the children macro-tiles.

[140] The sizes of all the arguments appearing in the input array of the parent macro-tile are known, as are the sizes of arguments in the input array of the distributed MCMC. This allows to easily compute the starting and ending position of these wires, leading to an efficiently computable wire layout.

Remark. *These wires take as input pieces of the representation r when distributed among in the* distributed computation layer *of the parent macro-tile, i.e. into pieces of size $[\![M_\ell]\!]^d$; and outputs pieces of the representation r when distributed to some processors of the* computation layer *of the parent macro-tile, i.e. into pieces of size $O(\log N_\ell)$. Thus, these wires slightly differ from our previous wires in that they also "split" to distribute blocks of size $[\![M_\ell]\!]^d$ into blocks of size $O(\log N_\ell)$.*

13.3.4 Inductive validity of the representations

To verify condition (iii) of Definition 10.5, every block of 2^d intermediate representations $r^{(k)}_{\mathbf{i}'}$ validated by the inductions $\mathcal{I}$ from all the first $\log N_\ell$ layers of distributed computations must be able to validate an additional step of induction $\mathcal{I} \colon (\{0,1\}^*)^{2^d} \times \{0,1\}^* \to \{\top, \bot\}$.

To implement this condition, we use the remaining $\log N_\ell$ *distributed computation layers* and add another $\log N_\ell$ layers of *subarray wires*. For an index $0 \leq k < \log N_\ell$:

- Continuity of the space-time diagram on the *distributed computation layer* of index $1 + \log N_\ell + k$ is enforced accross adjacent children macro-tiles' *subarray computation layers* of index k if and only if they belong in the same cluster $C^k_{\mathbf{i}'}$.

  In other words, the clusters of the distributed computation layer of index $1 + \log N_\ell + k$ now define blocks of $[\![2^k]\!]^d$ subarrays forming distributed space-time diagrams of mesh-connected multicomputers.
- These new MCMCs implement the program $e_{\text{RAM}} \in \{0,1\}^*$ in their processors and simulate the log-RAM program $\langle \mathcal{I} \rangle$.
- The new *subarray wires* on the layer of index $1 + \log N_\ell + k$ map the arguments of the MCMC appearing in the cluster $C^k_{\mathbf{i}'}$ of the distributed computation layer of index $1 + \log N_\ell + k$ with the representations $r^{(k)}_{\mathbf{i}''}$ validated by the adjacent clusters $C^k_{\mathbf{i}''}$ (for $\mathbf{i}'' \in \mathbf{i} + [\![2]\!]^d$, *including accross the border of the parent macro-tile*[141]) on their *distributed computation layer* of index k.[142]

[141] The children macro-tiles directly communicate with each other through these new *subarray wire fields* on their macro-colors, irrelevantly of their respective parent macro-tile.

[142] Or, if $k = 0$, the representations $r_{\mathbf{i}}$ directly contained in the children macro-tiles.

In other words, all the intermediar representations $r^{(k)}_{\mathbf{i}''}$ for $\mathbf{i}'' \in \mathbf{i}' + [\![2^k]\!]^d$ appear as the input arguments of an additional induction step $\mathcal{I}$. It is computed within the MCMC space-time diagram that is distributed on the *subarray computation layer* of index $1 + \log N_\ell + k$ of the children macro-tiles composing the cluster $C^k_{\mathbf{i}'}$.

13.4 Final considerations and fixpoint theorem

Finally, let us denote by $K \in \mathbb{N}$ an integer whose value will be determined later (it will be used to manage constants resulting from $O(\cdot)$ considerations). A well-formed input tape in a macro-tile contains (in this order) a tuple:

$$(e, K, \ell, m, \mathbf{c})$$

where $e \in \{0,1\}^*$ is a log-RAM program, $K \in \mathbb{N}$ is a fixed constant, $\ell \in \mathbb{N}$ is an integer representing the level of the macro-tile, $m \in \{0,1\}^*$ is a string of size $O(L_\ell^\alpha)$ and $\mathbf{c}$ is a tuple of $2d$ binary strings containing the following fields:

- A *position field*[143] of length $2 \log N_\ell$;
- A *computation field*[144] of length $O(\log N_\ell)$ bits; and $2 \log N_{\ell+1}$ *processor fields*, each containing $O(1)$ consecutive processor states of size $O(\log(N_{\ell+1} \cdot M_{\ell+1}))$ bits;
- A *wire field*[145] of length $O(1)$ bits; and $2 \log N_{\ell+1}$ *processor wire fields*, each of length $O(\log(N_{\ell+1} \cdot M_{\ell+1}))$ bits;
- $2 \log N_\ell$ *subarray computation fields*[146], each containing $O(M_\ell^{d-1})$ processor states of length $O(\log(N_\ell \cdot M_\ell))$ bits;
- $2 \log N_\ell$ *subarray wire fields*, each of length $O(M_\ell^{d-1} \cdot \log(N_\ell \cdot M_\ell))$ bits;
- A *subarray-to-parent wire field* of length $O(M_\ell^{d-1} \cdot \log(N_\ell \cdot M_\ell))$.

In what follows, we set:

$$N_\ell = 2^{2^{2^\ell} + K} \quad \text{and} \quad M_\ell = K \cdot L_\ell^\gamma.$$

for K the constant mentioned above. Notice that, when given K and ℓ, N_ℓ and L_ℓ can be computed in time $\mathrm{polylog}(N_\ell)$. By padding the input fields with blank symbols if necessary, we assume that each field has a fixed length that only depends on ℓ, and that can be computed in time $\mathrm{polylog}(N_\ell)$. (In other words: the constants associated to each $O(\cdot)$ are known and hardcoded.)

Let us now describe the log-RAM program appearing on the program tape. Let $f(e, K, \ell, r, \mathbf{c})$ be the following algorithm[147]:

1. *Layout of parent macro-tiles:*
   (i) Given K and ℓ, compute N_ℓ, L_ℓ, $N_{\ell+1}$, $L_{\ell+1}$, $L_{\ell+2}$ and M_ℓ;
   Time: $\mathrm{polylog}(N_\ell)$
   (ii) Check that each color $c_\cdot$ is a string containing all the aforementioned fields, and that each field is of the correct length;
   Time: $O(M_\ell^{d-1} \cdot \log L_{\ell+1})$
   (iii) Decode $i \in [\![N_\ell]\!]^d$ from the *position field* of $(c_{\mathrm{N}}, c_{\mathrm{W}}, c_{\mathrm{S}}, c_{\mathrm{E}})$, *i.e.* compute the position of this tile in the parent macro-tile;
   Time: $\mathrm{polylog}(N_\ell)$
   (iv) Check the *computation field*: the *computation field* should encode $O(1)$ consecutive computation step of an MCMC processor operating on variables of word length $O(\log N_\ell)$, running the MCMC program $e_{\mathrm{RAM}} \in \{0,1\}^*$ (from Theorem 11.8), and have a variable `program` containing the RAM program $e \in \{0,1\}^*$ and a variable `pos` containing the position $i \in [\![N_\ell]\!]^d$ decoded in the previous item;
   Time: $\mathrm{polylog}(N_\ell)$

[143] The position in the parent macro-tile.

[144] One processor of a mesh-connected multicomputer of size $O(N_\ell^d)$.

[145] Carrying one bit from the parent's macro-colors to one children macro-tile from the input area of the MCMC.

[146] That contain the borders of MCMC subarrays of size $[\![M_\ell]\!]^d$.

[147] To make f a log-RAM program, we should briefly mention how the input tuple is distributed to the RAM input array: e is written in $I[0]$, ℓ is written in $I[1]$, $r \in \{0,1\}^{O(L_\ell^\alpha)}$ is written into the next consecutive memory cells as chunks of length $O(\log N_\ell)$, and $\mathbf{c}$ is written on the next consecutive memory cells $2d$ bits at time.

(v) Considering the `input` variable of the processor from the *computation field*, check – depending on the position $\mathbf{i}$ – that it contains K (if the tile has position $\mathbf{i} = (0, 0 \dots)$), $\ell + 1$ (if $\mathbf{i} = (1, 0 \dots)$), or the correct slice[148] of r;
Time: polylog(N_ℓ)

(vi) Check the *wire field*: compute the wire layout for macro-tiles of size N_ℓ^d and check that the *wire field* is consistent with the position $i \in [\![N_\ell]\!]^d$ of said layout;
Time: polylog(N_ℓ)

(vii) Check the *processor* and *processor wire fields*: in particular, check that each of the $O(\log N_{\ell+1})$ *free computation fields* correctly encode one computation step of a processor operating on variables of word length $O(\log L_{\ell+2})$; and compute the $O(\log N_{\ell+1})$ free wire layouts and check that that the *free wire fields* correctly encode wires at position $i \in [\![N_\ell]\!]^d$ from said layouts;
Time: polylog(N_ℓ)

2. *Checking the distributed computations of the induction* $\mathcal{I}$

(i) For every $0 \leq k \leq \log N_\ell$, compute all the clusters and cluster positions $(C^k_{\mathbf{i}'}, \mathbf{i}'')$ depending on the position of the children macro-tile $\mathbf{i}$;
Time: polylog(N_ℓ)

(ii) Check the *subarray computation fields* of $\mathbf{c}$: they should contain $O(M_\ell^{d-1})$ processor states from the border of an MCMC space-time subarray. These processors should run the MCMC program $e_{\text{RAM}} \in \{0,1\}^*$ and operate on variables of word length $O(\log(2^k \cdot M_\ell))$. Depending on the position $\mathbf{i}''$ of the processor in its cluster on the k^{th} cluster, continuity of the computations should be enforced accordingly, as well as the `pos` variable of each processor. The `program` variable should contain the log-RAM program $\langle I \rangle \in \{0,1\}^*$;
Time: $2 \log N_\ell \cdot O(M_\ell^{d-1} \cdot \log L_{\ell+1})$

(iii) Check all *subarray wire fields*: compute all the wire layouts between clusters and check that the *subarray wire fields* of the tile $\mathbf{c}$ correctly follow these layouts at position $i \in [\![N_\ell]\!]^d$ in the parent macro-tile. If $\mathbf{i}''$ is an input or output position for a wire on the k^{th} layer, check the consistency of the k^{th} *subarray wire field* with the corresponding *subarray computation field* to ensure that correct data is copied on the wires. For layers 1 and $1 + \log N_\ell$,[149] check that the argument representation r is correctly written on the wires.
Time: $2 \log N_\ell \cdot O(M_\ell^{d-1} \cdot \log L_{\ell+1})$

(iv) Check the *subarray-to-parent field*: compute the wire layout of the subarray-to-parent wires and check that the *subarray-to-parent field* of the tile $(\mathbf{c})$ is consistent this layout at position $i \in [\![N_\ell]\!]^d$ in the parent macro-tile. If $\mathbf{i}$ is an input (resp. output) position for these wires, check the consistency of the *subarray-to-parent field* with the *subarray computation field* (resp. *computation field*);
Time: $O(M_\ell^{d-1} \cdot \log L_{\ell+1})$

3. If one check fails, reject the computation. Otherwise, accept.

We complete the construction as in Chapter 12, by making the function f (which recognizes the children tiles) run on its own code $e \in \{0,1\}^*$ (so that parent macro-tiles will compute the same recognizing function).

More precisely, we apply the runtime-preserving fixpoint theorem of log-RAM programs Proposition 4.30 on $f(e, K, \ell, r, \mathbf{c})$: there exists a log-RAM program $e \in \{0,1\}^*$ such that $f(e, \dots) = \varphi_e(\dots)$, and $f(e, \dots)$ and $\varphi_e(\dots)$ have the same running time (up to a constant factor).

[148] Since r has fixed length $O(L_\ell^\alpha)$, the tile at position $\mathbf{i}$ which variable of the RAM input array it is supposed to see.

[149] Which directly take as input the children macro-tiles.

In order to address considerations involving time overheads from $O(\cdot)$, let $K \in \mathbb{N}$ be a constant such that, by defining $N_\ell = 2^{2^{2^\ell}+K}$, $M_\ell = K \cdot L_\ell^\gamma$ and hardcoding all $O(\cdot)$ factors in the algorithm f, we verify all the following constraints for every $\ell \in \mathbb{N}$:

- *Parent macro-colors fit inside parent macro-tiles:* the size of the macro-colors of level $\ell+1$ verifies $O(M_{\ell+1}^{d-1} \cdot \log L_{\ell+2}) < N_\ell^{d-1}$;
- *Computations of $T_{\ell+1}$-tiles fit inside macro-tiles of level $\ell+1$:* the algorithm e has time complexity $O(M_{\ell+1}^{d-1} \cdot \log(L_{\ell+2})) < N_\ell^{d-1}$;
- *Clusters are large enough to contain the corresponding induction computations:* the induction $\mathcal{I}$ on patterns of size $[\![2^k \cdot L_\ell]\!]^d$ is computed in time $O((2^k \cdot L_\ell)^{\alpha\cdot\beta}) < (2^k \cdot M_\ell)^{d-1}$;

and all these constraints can be satisfied because, as $\gamma < 1$, we asymptotically have $L_{\ell+1}^\gamma \cdot \log L_{\ell+2} \ll N_\ell$.

If we were to define

$$T_\ell = \{\mathbf{c} \in (\{0,1\}^*)^{2d} \colon \exists r \in \{0,1\}^*,\ \varphi_e(K, \ell, r, \mathbf{c}) = \top\},$$

the attentive reader might notice that T_ℓ does *not* simulate $T_{\ell+1}$ (at least, for the definition of simulation we introduced earlier):

- *Validity of cluster computations:* we should actually restrict T_ℓ to macro-colors $\mathbf{c}$ whose *subarray computation fields* are actually borders of *valid* space-time subarrays of MCMC computations.
  We denote by $V \colon (K, \ell, \mathbf{c}) \mapsto \{\top, \bot\}$ the function checking whether the *cluster computation fields* of a macro-tile $\mathbf{c}$ can actually be filled with correct MCMC space-time subarrays.
- *Recursive validity of representations:* the other reason is that this consists actually aims at proving Theorem 10.11. Indeed, T_ℓ-tilings simulate the $T_{\ell+1}$-tilings whose tiles contain inductively valid representations for an additional $\log N_\ell$ steps of induction.

In other words, let us actually define

$$\begin{aligned} T_\ell = \{\mathbf{c} \in (\{0,1\}^*)^4 \colon\ & \exists m \in \{0,1\}^*, \\ & \exists w \in \mathcal{A}^{[\![L_\ell]\!]^d} \text{ inductively valid with final representation } r, \\ & V(K, \ell, \mathbf{c}) = \top \text{ and } \varphi_e(K, \ell, r, \mathbf{c}) = \top\}. \end{aligned}$$

Proposition 13.3. *For every $\ell \in \mathbb{N}$, T_0 simulates T_ℓ with zoom factor L_ℓ.*

Sketch of proof. Reasonning inductively, let us assume that T_0 simulates T_ℓ. By construction, the tiles of T_ℓ organize themselves into blocks of $[\![N_\ell]\!]^d$ called *macro-tiles* of level $\ell+1$. Let $\mathbf{c}$ be the macro-colors appearing on the borders of a such a macro-tile of level $\ell+1$ in a T_0 tiling:

- All *subarray computation fields* in the macro-colors of $\mathbf{c}$ are borders of valid MCMC space-time subarrays by definition of the *processor/processor wire fields* of the T_ℓ-tiles; thus, we have $V(K, \ell+1, \mathbf{c}) = \top$;
- Since $O(M_{\ell+1}^{d-1} \cdot \log(L_{\ell+2})) < N_\ell^{d-1}$, all valid computations of the recognizing algorithm $\varphi_e(K, \ell+1, m, \mathbf{c})$ fit in any such macro-tile;
- Reciprocally, such a macro-tile can only admit macro-colors $\mathbf{c}$ that admit a valid computation of $\varphi_e(K, \ell+1, r, \mathbf{c})$;

On the one hand, the final representation of an inductively valid pattern of domain $[\![L_{\ell+1}]\!]^d$ can be embedded into such a macro-tile of level $\ell+1$ by embedding its intermediate representations in the corresponding macro-tiles/distributed space-time diagrams. Reciprocally, since such macro-tiles implement the last $\log N_\ell$ steps of checking the inductive validity of a pattern, and that tiles of T_ℓ contain final representations of inductively valid patterns of domain $[\![L_\ell]\!]^d$, we deduce[150] that macro-tiles of level $\ell+1$ embed final representations of inductively valid patterns of domain $[\![L_\ell \cdot N_\ell]\!]^d = [\![L_{\ell+1}]\!]^d$. Thus, we conclude that T_0 simulates $T_{\ell+1}$ with zoom factor $L_{\ell+1}$. □

[150] Actually, verifying condition (iii) of Definition 10.5 requires careful considerations of the induction steps computed at levels $\ell' \leq \ell$ and *accross* macro-tiles of level ℓ. Which works as intended, because the wires described in Section 13.3.4 join all hypercubes of 2^d macro-tiles of level ℓ', *even if they have distinct parent macro-tiles*.

13.5 Resulting tileset

Let T_0 be the tileset resulting from the construction above. We now define an SFT X that factors onto $X_{\mathcal{R},\mathcal{I}} \subseteq \mathcal{A}^{\mathbb{Z}^d}$.

Let us define $\mathcal{A}_0 \subseteq \mathcal{A} \times T_0$ as follows: $\mathcal{A}_0$ is defined as the set of all pairs (a,t) of a symbol $a \in \mathcal{A}$ and a tile $t \in T_0$ for which there exists a representation $r \in \mathcal{R}(a)$ verifying $\varphi_e(K,0,r,t) = \top$. Then, define X as the set of all the configurations of $\mathcal{A}_0^{\mathbb{Z}^d}$ whose projection to T_0 form valid T_0-tilings.

We are left with proving that:

Lemma 13.4. *Let us denote $\pi : \mathcal{A}_0 \to \mathcal{A}$ the natural projection from $\mathcal{A}_0$ to $\mathcal{A}$. Then*

$$\pi(X) = X_{\mathcal{R},\mathcal{I}}.$$

We reason by double inclusions.

Part 1: $\subseteq$. Let $x \in X$ be a valid configuration of X, and let $w \sqsubseteq x$. Without loss of generality (up to shifting x and taking a bigger pattern w), we assume that $\mathrm{dom}(w) = [\![n]\!]^d$ for some $n \in \mathbb{N}$.

Since T_0 simulates any tileset T_ℓ with zoom factor L_ℓ by the previous lemma, every T_0-configuration can be inductively decomposed into nested arrays of *macro-tiles* of size $[\![L_\ell]\!]^d$. While we cannot ensure that a macro-tile of level ℓ entirely contains the domain $[\![n]\!]^d$, there exists a hypercube of 2^d macro-tiles of level ℓ that entirely covers the domain $[\![n]\!]^d$. We denote by $D \subseteq \mathbb{Z}^d$ the subset of $\mathbb{Z}^d$ that is covered by these macro-tiles of level ℓ, and such that $[\![n]\!]^d \subseteq D$.

By definition, we have $\pi(w) \sqsubseteq \pi(x|_D)$; and by construction of the tileset T_0, the pattern $\pi(x|_D)$ is an inductively valid pattern, since the projection of $x|_D$ to T_0 forms a hierarchy of nested macro-tiles that all verify the inductive validity of the patterns they cover. Since w was arbitrary, we conclude that $\pi(x) \in X_{\mathcal{R},\mathcal{I}}$. □

The converse inclusion should be straightforward: given an inductively valid pattern for $(\mathcal{R},\mathcal{I})$, we want to place a macro-tile on top of it. It is, however, unecessarily complicated by a small hiccup: in the definition of T_ℓ, all *cluster computation fields* are supposed to be fillable with valid subarray computations.

Part 2: $\supseteq$. Let $x \in X_{\mathcal{R},\mathcal{I}}$ be a valid configuration, and let $u \sqsubseteq x$ be an arbitrary pattern. We enlarge u by considering $\ell \in \mathbb{N}$ and $u' \sqsubseteq x$ such that $u \sqsubseteq u'$ and, *up to translation*: $\mathrm{dom}(u) \subseteq \frac{N_\ell}{2} + [\![L_\ell]\!]^d$ and $[\![L_{\ell+1}]\!]^d \subseteq \mathrm{dom}(u')$.

By definition of $X_{\mathcal{R},\mathcal{I}}$, there exists $v \sqsubseteq x$ such that $u' \sqsubseteq v$ and such that v is *inductively valid*. Without loss of generality (up to shifting x), we assume that v has domain $[\![2^n]\!]$ for some $n \in \mathbb{N}$.

Parent hypercubes: Since $[\![L_{\ell+1}]\!]^d \subseteq \mathrm{dom}(v)$, and that the pattern u fits (up to translation) in a box of size $[\![L_{\ell+1}]\!]^d$, we know that by partitionning $[\![2^n]\!]^d$ into hypercubes of fundamental domain $[\![L_{\ell+1}]\!]$, at most 2^d such hypercubes are enough to cover u in x'. Let us call these the *parent hypercubes*.

Children hypercubes: Actually, the pattern u fits (up to translation) in a box of size $[\![L_\ell]\!]^d$. Thus, when partitionning all the parent hypercubes into hypercubes of fundemantal domain $[\![L_\ell]\!]$, 2^d such hypercubes are enough to cover u in x'. Let us call these the *children hypercubes*.

Children macro-tiles: We can now define 2^d macro-tiles that will strictly cover the pattern u. Indeed, since all parent hypercubes cover patterns that are inductively valid, we define a T_ℓ tiling of the 2^d children hypercubes as follows:

- Fill the *position field* of the macro-tiles of level ℓ with the position of the corresponding children hypercubes in their parent hypercube; and, recursively, fill the *position field* of the macro-tiles of level $\ell' < \ell$;
- Fill the *subarray computation* and *subarray wire fields* of the macro-tiles of level $\ell' \leq \ell$ with valid MCMC space-time subarrays of the induction function $\mathcal{I}$ on the representations obtained by inductive validity of v;[151]
- The *computation*, *wire*, *processor* and *processor wire fields* of macro-tiles of level $\ell' < \ell$ should be completely determined; at level ℓ, these should be filled with plausible data (consistent between the 2^d children macro-tiles).

[151] This explains why we go down from $[\![L_{\ell+1}]\!]^d$ to $[\![2L_\ell]\!]^d$: the only justification we found to explain why the *distributed computation layers* of the 2^d macro-tiles of level ℓ could be filled by valid subarrays required to consider the representations of the parent hypercubes obtained by the inductive validity of v.

Thus, we obtain a T_0-valid pattern w^0 over the domain $[\![2L_\ell]\!]^d$.

On the other hand, denote by w the pattern of domain $[\![2L_\ell]\!]^d$ that is equal, *up to shift*, to the pattern of x covered by all the children hypercubes. By construction, (w, w^0) is a locally valid pattern in X; and since $u \sqsubseteq w$, this proves that u is locally valid in $\pi(X)$. □

14 Applications of Theorem 10.11

In this chapter, we consider several applications of Theorem 10.11 to show how representations and inductions can prove soficity of subshifts in classical (Theorems 14.1 and 14.2) and novel examples (Theorem 14.3, ...).

These examples also illustrate various intuitions and ideas about the possibilities of this construction: embedding effective forbidden pattern computations inside the induction function, "forgetting" data inside representations once it has been checked by the induction function...

14.1 Right-computable densities

A huge inspiration for these chapters has been the Ph.D thesis of Juline Destombes [Des21, Theorem 4 & 5], which proves that any effective subshifts on the alphabet $\{\square, \blacksquare\}$ whose patterns contain a sublinear amount of symbols $\blacksquare$ is actually sofic. We generalize this statement to arbitrary dimensions $d \geq 2$:

[Des21] Destombes, "Algorithmic complexity and soficness of shifts in dimension two".

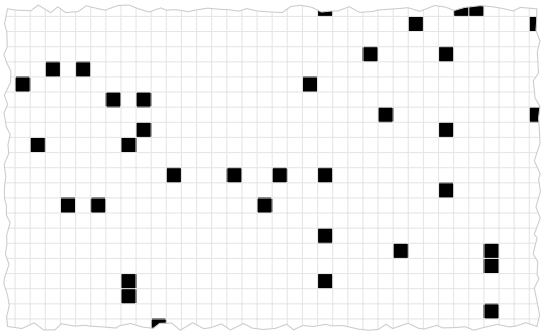


Figure 14.1: A typical pattern of the subshift $X_{1/2}$.

Theorem 14.1. *Let $d \geq 2$. For $C \in \mathbb{N}$ and $0 \leq \alpha < d-1$, consider the subshit X_α defined as:*

$$X_\alpha = \{x \in \{\square, \blacksquare\}^{\mathbb{Z}^d} : \forall n \in \mathbb{N}, \forall w \in \mathcal{A}^{[\![n]\!]^d},\ w \sqsubseteq x \implies |w|_\blacksquare \leq C \cdot n^\alpha\}.$$

If $\alpha \in \Pi_1$, then X_α is a sofic subshift; furthermore, any effective subshift $X \subseteq X_\alpha$ is also sofic.

Proof (Soficity of X_α). The main idea of this proof consists in representating patterns $w \in \{\square, \blacksquare\}^{[\![n]\!]^d}$ by the list of all their $\blacksquare$ symbols; and have the induction forbid representations of densities higher than α.

Formally, let us fix $\alpha \in \Pi_1$ such that $0 \leq \alpha < d-1$, and $r \in Q_+$ such that $\alpha < r < d-1$. For $w \in \{\square, \blacksquare\}^{[\![n]\!]^d}$, the representation function $\mathcal{R}$ will representat the pattern w with a tuple composed of:

- The *size* n;
- The *list* $L = \{i \in [\![n]\!]^d : x_i = \blacksquare\}$ of all $\blacksquare$ symbols in w;

when the number of ■ symbols in w verifies $|w|_■ \leq C \cdot n^r$. Otherwise, w has no representation and we set $\mathcal{R}(w) = \emptyset$.

For any $w \in \{□, ■\}^{[\![n]\!]^d}$ and any $m \in \mathcal{R}(w)$, the bit size of m verifies $|m| = O(n^r \cdot \log n)$. Furthermore, the induction $\mathcal{I}$ follows directly from our definition of $\mathcal{R}$: given 2^d lists of positions of ■ symbols and the size n of the domains $[\![n]\!]^d$,

- Merge these 2^d lists together into a larger list L. Since each list represents the positions inside its hypercube of fundamental domain $[\![n]\!]^d$, the coordinates of each ■ symbol first need to be recomputed to know its position in the domain $[\![2n]\!]^d$.
- If the size of the merged list L is larger than $C \cdot n^r$, reject the computation. Otherwise, compute the rational $r_\alpha \in \mathbb{Q}_+$ obtained after approximating $\alpha \in \Pi_1$ during $O(n^r)$ steps.
- For each size of square $1 \leq k \leq \log n$, check for each position of a symbol ■ in L that the square of fundamental domain $[\![k]\!]^d$ centered on this position contains less than $C \cdot k^{r_\alpha}$ symbols ■.

Using appropriate data structure (sets implemented as balanced trees, etc...), the function $\mathcal{I}$ can actually be implemented in time $O(s \cdot \text{polylog}(s))$ on inputs of size s: by Theorem 10.11, the subshift $X_{\mathcal{R},\mathcal{I}} \subseteq \{□, ■\}^{\mathbb{Z}^d}$ is sofic. Since the subshifts $X_{\mathcal{R},\mathcal{I}}$ and X_α coincide[152], X_α is in turn sofic. □

[152] Every pattern in $X_{\mathcal{R},\mathcal{I}}$ must contain less than $C \cdot n^\alpha$ symbols ■, so is valid in X_α. Reciprocally, any valid pattern in X_α of domain $[\![2^n]\!]^d$ turns out to be inductively valid by induction.

To prove that any effective subshift of X_α is sofic, we use a classical trick that comes up in most examples of the chapter: **we embed a small (but ever growing) enumeration of the forbidden patterns in the induction function** $\mathcal{I}$. However, this method only works when representations embed enough information to check them for forbidden patterns efficiently:

Proof (*general case*). Let us now consider any effective subshift $X \subseteq X_\alpha$. There exists a computably enumerable family $\mathcal{F} \subseteq \{□, ■\}^{*d}$ of forbidden patterns such that $X = X_\mathcal{F}$. We assume that there exists a RAM program $e \in \{0,1\}^*$ that enumerates $\mathcal{F}$ as a list of multidimensional arrays filled with □ and ■ symbols. *In particular, in t steps of computations, e cannot enumerate patterns of any domain whose cardinality exceeds t.*

We amend the induction function $\mathcal{I}$ from the proof above by adding the following computations:

- After computing $\log\log n$ steps of the RAM program e, collect the complete patterns that were enumerated. For each of them, compute an associatd tuple $((n_1, \dots, n_d), l)$ where $[\![n_1, \dots, n_d]\!]$ is the domain of the pattern and l the list of positions in s colored with ■ symbols.
- For every position p in the merged list L (*i.e.* every ■ position of the representationed pattern), and for every pattern w enumerated at the previous step, check whether the pattern w can occur at the position p. If it does, reject; otherwise, continue.

Since any pattern enumerated in the first step is made of less than $\log\log n$ cells, checking whether it occurs at some position can be performed in time $\text{poly}(\log\log n)$ (using suitable data structures for L). Thus, the induction $\mathcal{I}$ can still be implemented with time complexity $t(s) = O(s \cdot \text{polylog}(s))$ on inputs of size s and Theorem 10.11 still applies.

By definition, any pattern of domain $[\![2^n]\!]^d$ in X is inductively valid, since it does not contain any forbidden pattern and will never fail any check of the induction $\mathcal{I}$. Reciprocally, let $w \in \mathcal{F}$ be a forbidden pattern of X. There exists $n \in \mathbb{N}$ such that the RAM program enumerates w after n steps of computations; and let us consider any pattern w' of domain $[\![2^{n'}]\!]^d$ for $n' \geq 2^n$ such that $w' \sqsubseteq w$. Then w' is not inductively valid (it will fail its $2^{n\text{-th}}$ induction step) and does not appear in $X_{\mathcal{R},\mathcal{I}}$.

We conclude that $X = X_{\mathcal{R},\mathcal{I}}$, so that X is indeed sofic. □

14.2 Seas of squares

Another motivation for Theorem 10.11 was [Wes17], which also inspired the writing of its proof. Define the "seas of squares" subshift $X_{\blacksquare} \subseteq \{\square, \blacksquare\}^{\mathbb{Z}^2}$ as the configurations made of disjoint squares of $\blacksquare$ symbols floating over a background of $\square$ symbols[153].

[Wes17] Westrick, "Seas of squares with sizes from a Π^0_1 set".

[153] $X_{\blacksquare}$ is a $\mathbb{Z}^2$ sofic subshift: one can draw rectangles by orienting the sides and corners, and rectangles can easily be forced to be squares by also drawing diagonals of slope $(1, 1)$.

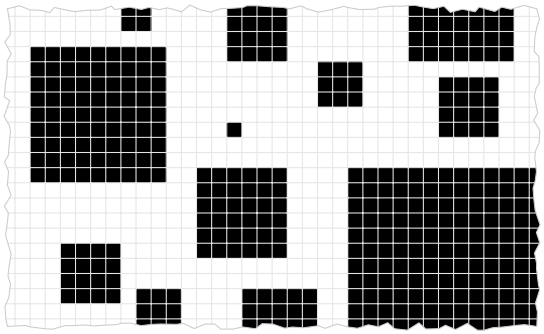


Figure 14.2: A typical configuration of the "seas of squares" subshift $X_{\blacksquare}$.

14.2.1 Classical "seas of squares" construction

For $S \subseteq \mathbb{N}$, we denote by $X_{\blacksquare:S}$ the subshift of the "seas of squares" subshift in which the side lengths of all finite squares belong in S. In [Wes17], it was proved that many "seas of squares" subshifts $X_{\blacksquare:S}$ are sofic because each square pattern of size $n \times n$ can be representationed by $\tilde{O}(n^{2/3})$ bits of information:

Theorem 14.2 ([Wes17]). *For any Π^0_1 set $S \subseteq \mathbb{N}$, the* seas of squares *subshift $X_{\blacksquare:S}$ is sofic.*

For example, the set P of prime numbers defines a sofic subshift $X_{\blacksquare:P}$, whose soficity is much less obvious than $X_{\blacksquare}$'s was.

Using Theorem 10.11, we prove Theorem 14.2 by building representations that store all the side lengths of squares that appear in the given pattern:

Proof. Let $w \in \{\square, \blacksquare\}^{[\![n]\!]^2}$ be a pattern. If w is not a valid pattern in $X_{\blacksquare}$ (*i.e.* some $\blacksquare$ symbols do not form a square, or a partial square "cut" by the border of w), then w has no representation. Otherwise, we define a single representation for w, which should contain:

- The *size* $n \in \mathbb{N}$;
- A *length list* L, which contains all the side lengths of squares appearing (even partially) in w;
- A *corner list* C containing the position $(i, j) \in [\![n]\!]^2$ and the orientation[154] of a list of square corners;

and the corners of a partial square s in w should appear in C if:

- Either s shows a single corner in w,[155] in which case C should contain this single corner;
- Or s contains exactly two corners in w *and* the distance between these corners[156] is greater than $\frac{n}{8}$.

It was noticed in [Wes17] that any pattern of domain $[\![n]\!]^2$ can only see $O(n^{2/3})$ distinct sizes of (partial) squares simultaneously: indeed, in the worst case, one of each square of side length $1, 2, \ldots, m$ appears in the domain $[\![n]\!]$, in which case $\sum_{k=1}^{m} k^2 \leq O(n^2)$. Since the *corner list* has length $O(1)$, we conclude that every representation $\mathcal{R}(w)$ is of size at most $O(n^{2/3} \cdot \log n)$ on patterns of domain $[\![n]\!]^2$.

[154] North-West, North-East, South-West or South-East.

[155] *i.e.* s straddles over one of the four corners of the domain $[\![n]\!]^2$.

[156] *i.e.* the candidate side length of the partial square s.

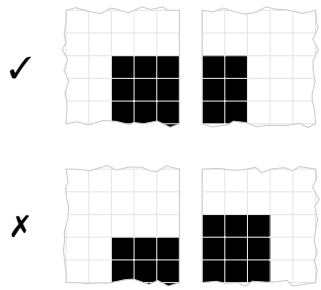


Figure 14.3: Cases of (horizontally) matching and non-matching corners.

The induction function $\mathcal{I}$ follows immediately, but is somewhat tedious to write: given a size $n \in \mathbb{N}$ and 2×2 representations of patterns of domain $[\![n]\!]^2$, we merge the list of side lengths seen in the four representations, and analyze the corners of partial squares appearing in each of the 2×2 quadrants to check whether they form valid squares;

- *List of side lengths*: merge the lists L_1, L_2, L_3, L_4 of length sides of squares into a single list L;
- *List of corners*: merge the list of all partial corners and positions C_1, C_2, C_3, C_4 from the 2×2 representations, after renormalizing the coordinates to the new domain $[\![2n]\!]$, into a single list C;
- *Merging corners*: if, in the list C of oriented positioned corners, several corners define interesecting rectangles, check that these corners actually match to form a square (reject if this isn't the case) and register the associated square length in L;
- *Deleting corners*: if four corners match together and form a complete square, or two corners match and form a square "straddling" over the border of the domain $[\![2n]\!]$ of side length less than $\frac{2n}{8}$, delete them from the representation's list of corners C;
- *Checking side lengths against S*: since S is a Π_1 set, let us compute $\log \log n$ steps of a RAM program enumerating S^c. If any enumerated size $s \in S^c$ appears in L, reject the computation.

Since $\mathcal{I}$ has time complexity $O(s \cdot \log s)$ on inputs of size s, we conclude that $X_{\mathcal{R},\mathcal{I}}$ is a sofic $\mathbb{Z}^2$ subshift. We are left with convincing ourselves that $X_{\mathcal{R},\mathcal{I}}$ is indeed $X_{\blacksquare,S}$:

- If w is a pattern of domain $[\![2^n]\!]^2$ of $X_{\blacksquare,S}$, then it is inductively valid.
- Reciprocally, let w be an inductively valid pattern of domain $[\![2^n]\!]^2$:
  - By definition of $\mathcal{I}$, the square sizes appearing in w cannot be enumerated in the $\log n$ first steps of the program enumerating S^c;
  - The $\blacksquare$ symbols in w must form valid squares. Indeed, let us assume the opposite and consider any four mismatching corners. Let us denote by $\simeq \frac{2^k}{8}$ one of the "candidate side length" of these corners. The key observation is that, even though these corners are forgotten after the k^{th} induction step, there exists 2×2 adjacent squares of the partition $(C_{k-2,i})_{i \in [\![2^{n-k}]\!]^2}$ that fully cover these mismatching corners. By definition of recurrent validity, an induction step of level $k-1$ is performed on these 2×2 squares and the mismatch is discovered.

Taking the limit over n, we conclude that $X_{\mathcal{R},\mathcal{I}} = X_{\blacksquare,S}$. □

The previous proof is an illustration of another method on representations: **to keep representations from growing too large, it is possible to discard some data once it has been checked** by the inductive validity condition[157].

[157] See condition (iii) in Definition 10.5.

14.2.2 Improved "seas of squares"

With Theorem 10.11, we can actually improve the "seas of squares" construction: since representations are of size $O(n^{2/3} \cdot \log n)$ on patterns of domain $[\![n]\!]^2$, which is much less than our $o(n)$ limit, there is plenty of room for squares to contain more data. As an example, we embed in the squares independant configurations of the Toeplitz densities lesser than some real number α (see Definition 5.7 for definitions).

More precisely, for $\alpha \in [0,1]$, let $X_{\blacksquare:\mathcal{T}(\leq\alpha)}$ denote the "seas of squares" subshift on the alphabet $\{\square, 0, 1\}^{\mathbb{Z}^2}$, whose configurations are made of squares of symbols $\{0,1\}$ over a $\square$ background; and each square contains the periodic lift of a valid pattern $w \in \mathcal{L}(\mathcal{T}(\leq \alpha))$.

Theorem 14.3. *Let $\alpha \in \Pi_1 \cap [0,1]$. The subshift $X_{\blacksquare:\mathcal{T}(\leq\alpha)}$ is sofic.*

Sketch of proof. We mix the representation functions defined in Example 10.8 on Toeplitz subshifts, and in Theorem 14.2 above for the "seas of squares" subshifts:

- Store the size $n \in \mathbb{N}$ such that the given pattern has domain $[\![n]\!]^2$;
- For squares "straddling" over the border of the domain $[\![n]\!]$ *and whose side lengths are larger than* $\frac{n}{8}$, store a guess of the Toeplitz structure and of the sequence being Toeplitzified;[158]
- Since each square contains the Toeplitzification of the binary expansion of some real number $\beta \in [0,1]$, store the maximal binary expansion that appears in the pattern.[159]

[158] See Example 10.8 for more details.

[159] Since binary expansions are guessed, we actually store the maximal guess.

The induction $\mathcal{I}$ should build a representation from 2×2 subrepresentations (reconstructing the Toeplitz structure of patterns shared between adjacent subrepresentations, etc...) and build an upper approximation of α by computing $\log\log n$ steps of a right-approximating program. If the maximal binary expansion of the representation ever surpasses this approximation, the representation is rejected. Since representations are very non-deterministic because of the "guessing" involved when studying the structure of a Toeplitz word, some representations of valid patterns will be rejected; yet, this does not prevent valid patterns from appearing, since the "correct" guess of a valid pattern is the minimal possible binary expansion, which will not be forbidden.

Since representations are of size $O(\log n)$ and the induction has time complexity $O(s \cdot \text{polylog}(s))$, the subshift $X_{\blacksquare:\mathcal{T}(\leq\alpha)}$ is indeed sofic. □

Remark. *By adding the side lengths of the squares to the representation, we obtain representations of size $O(n^{2/3}\log n)$ that allow to even constrain the squares of symbols $\{0,1\}$ to have side lengths belonging in any Π_1 set.*

All the ideas related to the "seas of squares" Toeplitz subshifts appear in our proof of Theorem 9.28, which contains the most involved representation function of this thesis.

14.3 Lifts

14.3.1 Periodic lifts

Another famous application of the expanding tileset construction was the proof of Proposition 3.44 in [DRS10]: for any effective subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^{d-1}}$, the subshift $X^{\Uparrow} \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is sofic.

[DRS10] Durand, Romashchenko, and Shen, "Effective closed subshifts in 1D can be implemented in 2D".

A naive representation for patterns of domain $[\![n]\!]^d$ in $\mathcal{A}^{\mathbb{Z}^{d-1}\Uparrow}$ consists in storing the pattern of a $(d-1)$-dimensional facet, which results of representations of bit length $O(n^{d-1})$. This makes this example very interesting, since it appears precisely in the gap between the naive $O(n^{d-1})$ information bound and our $o(n^{d-1})$ sufficient condition (Theorem 10.11) for soficity. Yet, we prove here that we can recover Proposition 3.44 with Theorem 10.11 by distributing the $O(n^{d-1})$ bits of the representation along the direction of periodicity given by the lift.

A new proof of Proposition 3.44. (Proving Proposition 3.44 on $\mathbb{Z}^2$ makes the proof easier to understand, but the methods actually generalize to arbitrary dimensions $d \geq 2$ without issues.)

For w a pattern of $\mathcal{A}^{\mathbb{Z}\Uparrow}$ of domain $[\![n]\!]^2$, the key idea of the proof consists in "distributing" the word $w|_{\{0\}\times[\![n]\!]}$: representations should, geometrically, organize themselves in columns of n representations, each representation containing a chunk of size $O(\log n)$ of $w|_{\{0\}\times[\![n]\!]}$.

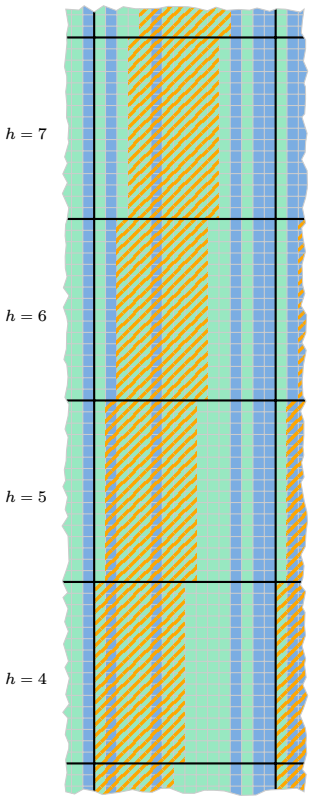


Figure 14.4: A column of square patterns ($n = 16$), an attribution of heights h, and the associated areas covered by the substrings s).

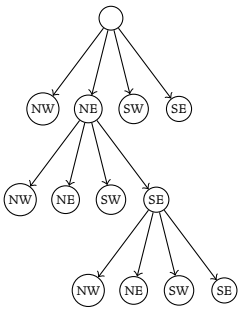


Figure 14.5: Scheme of a meta-representation of depth $d = 3$.

Distributed pattern representations More precisely, the *representation* of a pattern w in $\mathcal{A}^{\mathbb{Z}^{\Uparrow}}$ of domain $[\![n]\!]^2$ is a tuple $(n, h, s) \in \{n\} \times [\![n]\!] \times \mathcal{A}^{O(\log n)}$ (whose elements are called the *size*, the *height*, and the *substring* of the representation) such that the substring s should contain $O(\log n)$ symbols of w around the horizontal position $h \in [\![n]\!]$, *i.e.* $s = w|_{\{0\}\times(h+[-O(\log n)..O(\log n)])}$.

The induction should then enforce that representations are geometrically organized into consecutive heights (modulo n), so that n vertically consecutive representations of patterns w of size $[\![n]\!]^2$ will collectively hold the complete description of w.

Meta-representations Unfortunately, distributing the completed information into pieces makes the induction difficult to compute, since the 2×2 subrepresentations involved in a given induction step do not necessarily possess the data required to compute/check the representation of their parent.

To solve this issue, we will actually use *meta-representations*, whose role is to encode $\log n$ levels of representations at once: the meta representation of a pattern of domain $[\![n]\!]^2$ should encode a representation of said pattern, and guesses of representations for its parents, grand-parents,... for $\log n$ levels of inductions.

Remark. *Why* $\log n$*, specifically? Because after* $\log n$ *induction steps, the considered patterns have size at least* $O(n^2)$*, so that they geometrically cover at least* n *blocks of size* $[\![n]\!]^2$ *vertically, and thus covers all the "heights" of the subpatterns of size* n*.*

To encode d levels of inductions at once, we claim that the correct data structure on $\mathbb{Z}^2$ is a *4-ary comb tree* of depth d. Indeed, let us consider an intermediate representation $r^k_{\mathbf{i}}$ from Definition 10.5. It is used to compute four inductions steps: one (condition (ii)) computes the representation $r^{k+1}_{\mathbf{i}/2}$ (which will itself be used to computer higher-level induction steps: this is the branch of the comb tree), and the other three (condition (iii)) are induction steps whose results are never used again (hence, three leaves).

Thus, we define *meta representations*, which encode $\log n$ steps of induction as a *4-ary comb tree* of depth $\log n$ with $1 + 4 \log n$ nodes:

- Each non-leaf node has four children: a *North-West*, a *North-East*, a *South-West* and a *South-East* child; three of them are leafs, and one is another *4-ary comb tree*;
- Nodes at depth $k \leq \log n$ each carry the representation of a pattern of size $[\![2^k \cdot n]\!]^2$ as defined in the previous paragraph.

Intuitively, if a node has representation r, the representation of its *North-West* (resp. ...) child is the representation of the parent computed by giving r to the induction $\mathcal{I}$ as its *North-West* argument. We denote by $\mathcal{R}$ this *meta-representation* function.

Meta-induction The meta-induction $\mathcal{I}$ should then check that all the 2×2 given meta-representations agree on their guesses of higher levels, and guess the next few levels of representations that should be added to the 4-ary comb tree of the meta-representation. More precisely, given four meta-representations, *i.e.* four 4-ary comb trees T_{NW}, T_{SW}, T_{NE} and T_{SE}, the meta-induction proceeds as follows:

1. Consider the pattern representations $(n_{\text{NW}}, h_{\text{NW}}, s_{\text{NW}})$ (resp. ...) contained in the root of each tree T_{NW}, T_{SW}, T_{NE} and T_{SE}.
   a) These representations should correspond to patterns of the same size, so that n_{NW}, n_{SW}, n_{NE} and n_{SE} should be equal.
   b) These representations should have a correct geometrical arrangement: east and west heights should be equal, *i.e.* $h_{\text{NW}} = h_{\text{NE}}$ and $h_{\text{SW}} = h_{\text{SE}}$; and heights on each "column" should be vertically increasing, *i.e.* $h_{\text{NW}} = h_{\text{SW}} + 1 \bmod n$ and $h_{\text{NE}} = h_{\text{SE}} + 1 \bmod n$.

2. The four input meta-representations should agree on their guesses of higher levels: considering the North-West subtree of the representation T_{NW}, the North-East subtree of the representation T_{NE}, etc... check that they are all equal. If they are not, reject the computation. Otherwise, denote T' this unique subtree.
3. If T' is a single leaf node, accept the computation and return it as a meta-representation.
4. Otherwise, T' a 4-ary comb tree of depth $\log n - 1$ that encodes $\log n - 1$ levels of successive inductions. Since the meta-induction should return a meta-representation of a pattern of domain $[\![2n]\!]^2$, we need to non-deterministically guess 2 additional levels to T' to obtain a binary comb tree of depth $\log n + 1$.

   We non-deterministically guess these two additional levels to T' to obtain a new 4-ary comb tree T, and fill the associated nodes of level $k \in \{\log n, \log n + 1\}$ with representations as follows:

   a) The size of these nodes should be $2^k \cdot n$;
   b) The height should be some non-deterministic $h \in [\![2^k \cdot n]\!]$.
   c) The substring $s \in \mathcal{A}^{O(k+\log n)}$ should be non-deterministically filled, but s should agree with the substrings s_{NW} (resp. s_{SW}, ...) on their respective intersections.

      Denoting $I_{\text{NW}} = h_{\text{NW}} - [O(\log n) \mathinner{..} O(\log n)] \subseteq [\![n]\!]$ (resp. I_{SW}...) the intervals covered by the substring s_{NW} (resp. ...), one can compute the positions these intervals would have in the successive representations of the tree T by going down the trees T_{NW} (resp. ...) recursively. Indeed, starting from the root, an interval is left unchanged at depth k' if we go down a West subtree; or shifted by $2^{k'-1} \cdot n$ if going down an East subtree. Applying this process, denote by $I'_{\text{NW}} \subseteq [\![2^k \cdot n]\!]$ (resp. ...) the position that the interval I_{NW} occupies in the current depth-k node.

      Denote by $D = h + [-O(k + \log n) \mathinner{..} O(k + \log n)] \subseteq [\![2^k \cdot n]\!]$ the interval covered by the non-deterministically guessed substring $s \in \mathcal{A}^{O(k+\log n)}$. Considering the intersection of D and I'_{NW}, I'_{SW}, I'_{NE} and I'_{SE}, then s should agree with the corresponding bits of s_{NW} (resp. s_{SW}, ...) if their intersections are non-empty.

   Finally, return the new tree T as a valid meta-representation.

Sketch of proof We then claim that the induction defines valid representations, which in turn implies that $X_{\mathcal{R},\mathcal{I}} = \mathcal{A}^{\mathbb{Z}^{\Uparrow}}$. Indeed:

- By inductive validity, the representations indeed geometrically organize themselves as needed (step 1. of $\mathcal{I}$).
- For every pattern w, there always exists an inductively valid meta-representation (the notion of inductive validity, which computes representations inductively from subrepresentations, and only checks one step of induction from any 2×2 square, creates a recursion tree that looks like a 4-ary comb tree);
- Most importantly, by guessing $\log n$ steps of recursion in advance, all n distributed representations in a column have to guess the *same representation* simultaneously. Thus, the bits of the substrings of depth $\geq \log n$ are each checked individually by these n distributed representations, ensuring the correctness of representations of depth $\geq \log n$.

To go from $\mathcal{A}^{\mathbb{Z}^{\Uparrow}}$ to $X^{\Uparrow}$ for some $\mathbb{Z}$ effective subshift X, we modify the induction to compute some $\log \log n$ steps of an enumeration of the forbidden patterns of X, and check whether they appear in the substrings s_{NW}, s_{NE}, ... Since representations for patterns of domain $[\![n]\!]^2$ have bit length $O(\log^2 n)$ and that $\mathcal{I}$ is computable in time $t(s) = O(s)$, we conclude that $X^{\Uparrow}$ is a $\mathbb{Z}^2$ sofic subshift. □

14.3.2 Sparse lifts

A difficult question about multidimensional sofic subshifts is whether periodicity is truly needed in the previous construction: in Question 15.10, we ask whether having a sofic free lift implies that the original subshift is sofic. Or, equivalently: does there exist an effective non-sofic subshift whose free lift is sofic?

In the whole section, we consider the example of the *all-period subshift* $X_* \subseteq \{\text{␣}, *\}^{\mathbb{Z}}$ whose configurations containing at least two symbols $*$ at distance, say, $n \in \mathbb{N}$, are actually n-periodic[160].

[160] This subshift was defined in the proof of Theorem 9.2.

The patterns of this subshift admit a very simple representation: if a valid pattern w of domain $[\![n]\!]$ contains at least two symbols $*$, we associate the representation containing:

- The *domain size* $n \in \mathbb{N}$;
- A *period* $p \in [\![n]\!]$, which is the shortest distance between two $*$ symbols in w;
- A *shift* $\sigma \in [\![n]\!]$, which is the smallest index of a $*$ symbol in w;

if w contains a single symbol $*$, then we associate a representation containing the associated *domain size* and *shift*; and if w contains no symbol $*$, we associate the empty representation ε. If w is not a valid pattern in X_*, we define no representation for it. All representations are of size at most $O(\log n)$.

The associated induction checks the compatibility of two representations:

- If at least one representation contains a period, check that it is compatible with the data available on the other representation. The new shift is the left shift, and the period is left unchanged;
- If the two representations only contain a shift each, compute the associated period (as the distance between the two shifts);
- If one representation is blank, return the shift of the other representation;
- If both representations are blank, return a blank representation.

We now consider two non-periodic lifts of X_*: the *Toeplitz lift*, and the *α-sparse lift*.

Toeplitz lift

Define the Toeplitz lift of X_* as[161]:

[161] As defined in Section 5.1, the set of configurations $X_{\mathcal{T}} \subseteq \hat{N}^{\mathbb{Z}}$ is the $\mathbb{Z}$ closure of the *ruler sequence*.

$$X_*^{\uparrow\mathcal{T}} = \{x \in \{\text{␣}, *\}^{\mathbb{Z}^2} : \exists (z_n)_{n\in\hat{N}} \in (X_*)^{\hat{N}}, \exists t \in X_{\mathcal{T}}, \forall i \in \mathbb{Z},\ x|_{\{i\}\times\mathbb{Z}} = z_{t_i}\}$$

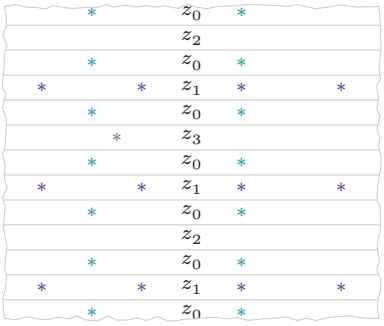


Figure 14.6: Structure of a configuration of a Toeplitz lift.

Intuitively, configurations of $X_*^{\uparrow\mathcal{T}}$ are obtained as a vertical Toeplitz of several (in fact, infinitely many) configurations of X_*. Since a pattern of domain $[\![n]\!]^2$ will only see $O(\log n)$ distinct patterns of X_*, and that X_* itself has representations of bit size $O(\log n)$, one should not be surprised that:

Example 14.4. *The $\mathbb{Z}^2$ subshift $X_*^{\uparrow\mathcal{T}}$ is sofic.*

Sketch of proof. Define the representation $\mathcal{R}$ on patterns of domain $[\![n]\!]^2$ as a tuple containing the following information:

- The domain size $n \in \mathbb{N}$;
- A guess of the vertical Toeplitz structure;
- For each of the $\log n$ known levels of vertical Toeplitz in w and the $O(1)$ lines of unknown levels, a representation of the corresponding pattern of domain $[\![n]\!]$ in X_*;

An associated induction $\mathcal{I}$ should:

- Check the compability of the vertical Toeplitz structures between the four subrepresentations;
- On each level of the Toeplitz structure, merge the two representations of $[\![n]\!]$ patterns using the induction of the X_*-representation.

A lot of details about the representation function of a Toeplitz structure has been swept under the rug: for precisions, we refer to Example 10.8. In any case, representations of $[\![n]\!]^2$ patterns are of size $O(\log^2 n)$, and $\mathcal{I}$ can be implemented with time complexity $t(s) = O(s)$: thus, $X_*^{\uparrow_{\mathcal{T}}}$ is sofic by Theorem 10.11. □

What kind of subshifts have sofic *Toeplitz lifts*? Unfortunately, the full shift $\mathcal{A}^{\mathbb{Z}}$ can be proven to have a non-sofic Toeplitz lift[162], as do all $\mathbb{Z}$ subshifts of positive entropy.

As for our construction, it can only be applied under some very heavy restrictions: since the vertical Toeplitz structure enforces equality of lines that are arbitrarily far from one another, and that our proof can only rely on the representations of the $\mathbb{Z}$ subshift to maintain this constraint, the $\mathbb{Z}$ representations need to entirely determine the $\mathbb{Z}$ patterns. We provided such representations in the case of the subshift X_*, or for the Toeplitz subshifts $X_{\mathcal{T}(U)}$; in the case of the "seas of squares" subshift, whose representations do not remember the exact positions of all the squares inside a given pattern, this requirement is not met.

[162] The proof of non-soficity actually follows the argument used on the mirror subshift: see Proposition 7.1.

α-sparse lift

For $\alpha \in [0, 1) \cap \mathbb{Q}_+$, define the α-sparse lift of X_* as:

$$X_*^{\uparrow\alpha} = \{x \in \{\text{␣}, *\}^{\mathbb{Z}^2} : \forall i \in \mathbb{Z},\ x|_{\{i\}\times\mathbb{Z}} \in X_* \text{ and } \exists I \subseteq \mathbb{Z}$$
$$I \text{ has density } O(n^\alpha) \text{ and } \forall i \in \mathbb{Z},\ x|_{\{i\}\times\mathbb{Z}} \neq \text{␣}^{\mathbb{Z}} \iff i \in I\};$$

where a set $I \subseteq \mathbb{Z}$ is said to have *density* $O(n^\alpha)$ if there exists a constant $C \in \mathbb{R}$ such that, for every interval $J \subseteq \mathbb{Z}$ of diameter $n \in \mathbb{N}$, we have $|I \cap J| \leq C \cdot n^\alpha$. In other words, $X_*^{\uparrow\alpha}$ is a free lift of X_* in which the density of non-blank lines is bounded by $O(n^\alpha)$.

Since a pattern of domain $[\![n]\!]^2$ will only see $O(n^\alpha)$ distinct patterns of X_*, and that X_* has representations of size $O(\log n)$, one should not be surprised that:

Example 14.5. *The $\mathbb{Z}^2$ α-sparse lift $X_*^{\uparrow\alpha}$ is sofic.*

Sketch of proof. Define the representation $\mathcal{R}$ on patterns of domain $[\![n]\!]^2$ as follows:

- The domain size $n \in \mathbb{N}$;
- A list of *non-trivial lines* and the X_*-representation of each of these non-trivial lines.

If a pattern of domain $[\![n]\!]^2$ has more than $O(n^\alpha)$ non-trivial lines, then $\mathcal{R}$ does not define a representation for it.

An associated induction should merge the lists of non-trivial lines, and perform the induction on the X_*-representations of the two adjacent sub-representations of each non-trivial line. Since the representations have size $O(n^\alpha \cdot \log n)$ and that the induction can be computed in time $t(s) = O(s)$, this proves that $X_*^{\uparrow\alpha}$ is sofic. □

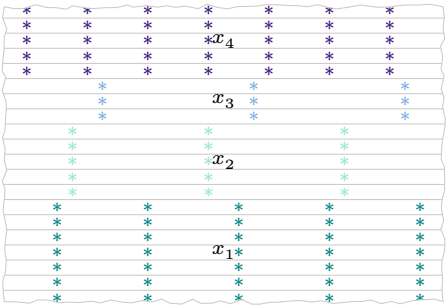


Figure 14.7: Layout of a "stripe lift" configuration.

What about other lifts? Many variations of this α-sparse lift could be defined and proved sofic using this method. For example, for a given subshift X, one could consider a subshift whose configurations are made of independant stripes from the periodic lift $X^{\Uparrow}$. If the density of stripes in a configuration is $O(n^{\alpha})$ for $\alpha < 1$, then the resulting "α-stripe lift" is sofic.

The distinction between the Toeplitz lift and such stripe lifts is that the Toeplitz lift needs to ensure equality between lines that are arbitrarily far from one another, which forces to pack a lot of information into the $\mathbb{Z}$ representations; whereas the stripe lifts forces equality between some stripes of consecutive lines, which is easy to enforce sofically.

Perspectives: soficity and communication complexity

15

We provide some perspective to Theorem 10.11 and the soficity of multidimensional subshifts in general. In particular, we argue that these problems naturally fit within the context of ressource-bounded (non-deterministic) communication complexity.

For example, Lemma 15.9 rephrases the characterization of $\mathbb{Z}$ sofic subshifts (Proposition 6.5) in terms of communication complexity. For illustrative purposes, we also build various $\mathbb{Z}^2$ subshifts based on problems with communication complexity $O(n)$ and study their soficity.

15.1 Communication complexity

15.1.1 Definitions

Let A and B be two sets, and $R \subseteq A \times B$ a relation. In traditional communication complexity, a pair of elements $(a, b) \in A \times B$ is given to Alice and Bob: Alice receives the element $a \in A$, and Bob the element $b \in B$. Intuitively, the communication complexity of the relation R is the minimal amount of information that Alice and Bob need to exchange to decide whether $(a, b) \in R$.

More precisely, *protocols* formalize the idea of exchanging messages:

Definition 15.1. A non-deterministic protocol *for R is a tuple (T, R_A, R_B) for T a finite set, and $R_A \subseteq A \times T$, $R_B \subseteq B \times T$ two relations such that:*

$$(a, b) \in R \quad \iff \quad \exists t \in T,\ (a, t) \in R_A \text{ and } (b, t) \in R_B.$$

In this definition, T is the set of possible *transcripts*, *i.e.* (intuitively) the possible communications between Alice and Bob while they verify whether a given pair (a, b) verifies $(a, b) \in R$. In the following examples, we will very often interpret protocols as communication protocols, which Alice and Bob follow to exchange messages[163] (*i.e.* bits) and determine the validity of their input pair. An element $t \in T$ is then considered as the summary of the non-deterministic messages they exchanged.

[163] We will, for example, say in Example 15.4 that "Alices sends her input to Bob".

Computationally, the relations R_A and R_B in a protocol can be arbitrary; in other words, Alice and Bob are given unlimited computational power. Thus, up to numbering the elements of T, the amount of information exchanged by Alice and Bob in a session of the protocol is $\log |T|$:

Definition 15.2. *The* size *of a protocol (T, R_A, R_B) is $\log |T|$.*

Going back to relations $R \subseteq A \times B$, we can then define the complexity of the communication problem R:

Definition 15.3. *Given a relation $R \subseteq A \times B$, its* non-deterministic communication complexity *$\mathcal{N}(R)$ is the minimal size of protocols for R.*

We refer the reader to [KN97] for a more exhaustive overview of communication complexity and its applications.

[KN97] Kushilevitz and Nisan, *Communication complexity*.

15.1.2 Examples

Example 15.4. *Consider the problem* EQ_n *of string equality: Alice and Bob are respectively given binary words* $a \in \{0,1\}^n$ *and* $b \in \{0,1\}^n$*, and want to check whether their inputs are equal. Formally,*

$$\mathrm{EQ}_n = \{(a,b) \in (\{0,1\}^n)^2 : a = b\}.$$

Then $\mathcal{N}(\mathrm{EQ}_n) = n$.

Proof. On the one hand, we have $\mathcal{N}(\mathrm{EQ}_n) \leq n$. Indeed, we make Alice send its whole input $a \in \{0,1\}^n$ to Bob, and Bob accepts if the message received matches its own string $b \in \{0,1\}^n$.

On the other hand, assume by contradiction that $\mathcal{N}(\mathrm{EQ}_n) \leq n-1$. Then EQ_n admits a protocol with less than 2^{n-1} possible transcripts, so that two distinct valid pairs $(a,b) \in \mathrm{EQ}_n$ and $(a',b') \in \mathrm{EQ}_n$ must share a common transcript[164]. This is a contradiction: indeed, (a,b') and (b',a) should also be valid for EQ_n, despite having $a \neq b'$ and $a' \neq b$. □

[164] This argument is often called the *fooling pair* argument.

As is often the case with non-deterministic computations, the complexity of a problem and its negation do not always match:

Example 15.5. *Consider the problem* NE_n *of string difference: Alice and Bob are respectively given binary words* $a \in \{0,1\}^n$ *and* $b \in \{0,1\}^n$*, and want to check whether their inputs differ. Formally:*

$$\mathrm{NE}_n = \{(a,b) \in (\{0,1\}^n)^2 : a \neq b\}.$$

Then $\mathcal{N}(\mathrm{NE}_n) = \log n + O(1)$.

Proof. On the one hand, we have $\mathcal{N}(\mathrm{NE}_n) \leq \log n + 1$. Indeed, Alice non-deterministically guesses an index $i \in [\![n]\!]$, and sends i to Bob along with the i^{th} bit a_i of its input; and Bob accepts only if its own bit b_i differs from a_i. This defines a valid protocol for NE_n: indeed, for every pair $(a,b) \in (\{0,1\}^n)^2$, $a \neq b$ if and only if there exists $i \in [\![n]\!]$ such that $a_i \neq b_i$.

On the other hand, let $P = (T, R_A, R_B)$ be an arbitrary protocol for NE_n, and let us consider the following protocol $P' = (T', R'_A, R'_B)$:

- $T' = 2^T$;
- The pair (a,t') is accepted by Alice if $t' \subseteq \{t \in T : (a,t) \notin R_A\}$;
- The pair (b,t') is accepted by Bob if $t'^c \subseteq \{t \in T : (b,t) \notin R_B\}$.

Then P' is a protocol for EQ_n, since $(a,b) \notin \mathrm{NE}_n$ if and only if there exists a partition of $T = T_A \sqcup T_B$ such that R_A rejects all (a,t) for $t \in T_A$ and R_B rejects all (b,t) for $t \in T_B$.[165] By the previous example, we have $|T'| \geq 2^n$; but since $|T'| = 2^{|T|}$, we deduce that $|T| \geq n$. Thus, $\mathcal{N}(\mathrm{NE}_n) \geq \log n$. □

[165] P' is a determinization of the protocol P.

The method in the proof above generalizes to arbitrary communication problems, and shows that the communication complexities of a problem and its negation can differ by at most an exponential factor.

15.1.3 Direct sums

Direct sums are parallel instances of two communication problem R, R': Alice and Bob are respectively given two words $(a,a') \in A^2$ and $(b,b') \in B^2$, and want to verify whether $(a,b) \in R$ and $(a',b') \in R'$ hold simultaneously.

Definition 15.6. *Let* $R \subseteq A \times B$ *and* $R' \subseteq A \times B$ *be two communication problems. Their* direct sum $R \wedge R'$ *is the logical conjuction of* R *and* R'*:*

$$R \wedge R' = \{((a,a'),(b,b')) \in A^2 \times B^2 : (a,b) \in R \text{ and } (a',b') \in R'\}.$$

Direct sums generalize to arbitrary sums $\wedge_{i=1}^{k} R_k$ of k communication problems $R_1, \dots, R_k$. They come up with the very natural question: is there any advantage in solving k parallel instances of a problem simultaneously rather than sequentially? This is answered in the following proposition:

[KN97] Kushilevitz and Nisan, *Communication complexity*.

Proposition 15.7 ([KN97, Corollary 4.9])**.** *For any $k \in \mathbb{N}$ and any relation $R \subseteq A \times B$, the communication complexity of k parallel instances of R verifies:*

$$\mathcal{N}(\wedge_{i=1}^{k} R) \geq k \cdot (\mathcal{N}(R) - \log n - O(1)).$$

In other words, gains are negligeable on problems of large complexity (*i.e.* $\Omega(\log n)$). However, the logarithmic substractive factor in the previous proposition is important, as illustrated by the following example:

Example 15.8. *Consider the problem $\wedge_{i=1}^{n} \mathrm{NE}_n$ of n parallel instances of the string difference problem: Alice and Bob each hold n binary strings $(a^{(k)})_{1\leq k\leq n} \in (\{0,1\}^n)^n$ and $(b^{(k)})_{1\leq k\leq n} \in (\{0,1\}^n)^n$, and want to check whether each pair of words $a^{(k)}$ and $b^{(k)}$ are distinct for $1 \leq k \leq n$. Formally,*

$$\wedge_{i=1}^{n} \mathrm{NE}_n = \{((a^{(k)}, b^{(k)}))_{1\leq k\leq n} \in (((\{0,1\}^n)^2)^n : \forall 1 \leq k \leq n,\ a^{(k)} \neq b^{(k)}\}.$$

Then $\mathcal{N}(\wedge_{i=1}^{n} \mathrm{NE}_n) = \Theta(n)$.

Sketch of proof. By the previous proposition, we have $\mathcal{N}(\wedge_{i=1}^{n} \mathrm{NE}_n) = \Omega(n)$.

The other direction is often proved by exhibiting a protocol of size $O(n)$ for $\wedge_{i=1}^{n} \mathrm{NE}_n$ in a public coin representation of randomized communication complexity, and then use derandomization. Introducing these methods is beyond the scope of this thesis; however, we prove an equivalent result on the similar problem $\mathrm{N1}_n$ in Proposition 15.12. □

15.2 Communication complexity in $\mathbb{Z}$ subshifts

Going back to languages of finite words $L \subseteq \mathcal{A}^*$, it is well-known that regular languages can be characterized in terms of communication complexity. Indeed, let $\mathrm{CC}_{n,L}$ denote the communication problem in which Alice and Bob are respectively given finite words $u \in \mathcal{A}^*$ and $v \in \mathcal{A}^*$ such that $|u| + |v| = n$, and should determine whether the concatenation uv is a valid word of the language L. Formally:

$$\mathrm{CC}_{n,L} = \{(u, v) \in \bigcup_{k=0}^{n} \mathcal{A}^k \times \mathcal{A}^{n-k} : uv \in L\}.$$

By Myhill-Nerode's theorem, a language $L \subseteq \mathcal{A}^*$ verifies $\mathrm{CC}_{n,L} = O(1)$ if and only if L is regular.[166]

[166] Indeed, if L is regular, Alice just sends to Bob the state it ended up in after reading a in a Deterministic Finite Automaton. Reciprocally, if $\mathrm{CC}_{n,L} = O(1)$, then L admits finitely many residuals, and is thus regular.

As mentioned in Proposition 6.5, Myhill-Nerode's theorem generalizes to $\mathbb{Z}$ subshifts: and a given $\mathbb{Z}$ subshift X is sofic if and only if $\{F_X(w) : w \in \mathcal{A}^*\}$ is finite[167]. We prove here that this result can also be rephrased in terms of communication complexity.

[167] Recall the definition of *follower sets* $F_X(w) = \{y \in \mathcal{A}^{\mathbb{N}} : \exists x \in \mathcal{A}^{-\mathbb{N}},\ xwy \in X\}$.

For X a subshift and $n \in \mathbb{N}$, consider the problem $\mathrm{CC}_{n,X}$ in which Alice and Bob are respectively given finite words $u \in \mathcal{A}^*$ and $v \in \mathcal{A}^*$ such that $|u| + |v| = n$, and should determine whether uv forms a valid pattern in X. Formally,

$$\mathrm{CC}_{n,X} = \{(u, v) \in \bigcup_{k=0}^{n} \mathcal{A}^k \times \mathcal{A}^{n-k} : \exists x \in X,\ uv \sqsubseteq x\}.$$

Lemma 15.9. *A $\mathbb{Z}$ subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is sofic if and only if $\mathcal{N}(\mathrm{CC}_{n,X}) = O(1)$.*

Proof.

$\Longrightarrow$ If X is sofic, then fix a local cover $X' \subseteq \mathcal{B}^{\mathbb{Z}}$ of X a projection $\pi\colon \mathcal{B} \to \mathcal{A}$ such that $\pi(X') = X$. Then Alice can pick an arbitrary preimage of u under π, Bob an arbitrary preimage of v under π, and communicate their respective rightmost and leftmost symbols to ensure that the concatenation of these preimages is valid in X'.

$\Longleftarrow$ Fix $n \in \mathbb{N}$, and assume that $\mathcal{N}(\mathrm{CC}_{n+N,X}) \leq \log K$ for some constant $K \in \mathbb{R}_+$ and for every $N \in \mathbb{N}$. By definition, for every $N \in \mathbb{N}$, there exists a protocol (T, R_A, R_B) for $C_{n+N,X}$ such that $|T| \leq K$.

Notice that for any $u \in \mathcal{A}^n$:

$$F_X(u)|_{[\![N]\!]} = \bigcup_{t \in T\colon (u,t) \in R_A} \{v \in \mathcal{A}^N\colon (v,t) \in R_B\}.$$

In particular, $F_X(u)|_{[\![N]\!]}$ is entirely determined by $\{t \in T\colon (u,t) \in R_A\}$. Since T is finite, this implies that, for every $n \in \mathbb{N}$ and every $N \in \mathbb{N}$:

$$|\{F_X(u)|_{[\![N]\!]}\colon u \in \mathcal{A}^n\}| \leq 2^{|T|} \leq 2^K.$$

Now, for any two words $u, u' \in \mathcal{A}^n$ such that $F_X(u) \neq F_X(u')$, there exists $N \in \mathbb{N}$ such that $F_X(u)|_{[\![N]\!]} \neq F_X(u')|_{[\![N]\!]}$ (compactness). Thus, since the previous bound was valid for every $N \in \mathbb{N}$, we obtain:

$$|\{F_X(u)\colon u \in \mathcal{A}^n\}| \leq 2^K.$$

By Proposition 6.5, we conclude that X is sofic. □

15.3 Communication complexity of multidimensional subshifts

A generalization of communication complexity to an infinite setting was introduced quite recently [GJ15] with the explicit goal to study multidimensional sofic subshifts: to the best of our knowledge, this was the first use of communication complexity within symbolic dynamics. In fact, as opposed to Kolmogorov complexity – which is already mentioned in [DLS08], [DRS12, Section 7], [Sim15] or [DR22] – communication complexity seems to have gained little traction in the subshift community; although it may be more popular within the twin community of *picture languages*[168] [Ter19].

Yet, communication complexity could be a great tool for the study of sofic multidimensional subshifts, in particular in a ressource-bounded context. For example, the statement of Theorem 10.11 could certainly be understood as a quantification of the data communicated between 2^d adjacent (hyper)cubic patterns with limited computational power to decide whether they are, in a sense, "locally admissible". While it does not provide a full characterization of multidimensional soficity, communication complexity seems like a fitting framework for formalizations of the information-bounding intuitions.

Many questions arise from the intersection of communication complexity and multidimensional sofic subshifts. For example, [GJ15] explicitely aimed at studying the soficity of free lifts of one-dimensional subshifts.

Question 15.10. *Let $X \subseteq \mathcal{A}^{\mathbb{Z}}$ be a subshift. If $X^{\rightleftharpoons} \subseteq \mathcal{A}^{\mathbb{Z}^2}$ is a sofic subshift, must the subshift X be sofic?*

On $\mathbb{Z}^2$, the free lift $X^{\rightleftharpoons}$ of a $\mathbb{Z}$ sofic subshift X must be sofic; and while Proposition 3.44 (which shows that effective $\mathbb{Z}$ subshifts have sofic periodic lifts on $\mathbb{Z}^2$) cannot be generalized to free lifts[169], we cannot exclude at the

[GJ15] Guillon and Jeandel, *Infinite Communication Complexity*.

[DLS08] Durand, Levin, and Shen, "Complex tilings".

[DRS12] Durand, Romashchenko, and Shen, "Fixed-point tile sets and their applications".

[Sim15] Simpson, "Symbolic dynamics: entropy = dimension = complexity".

[DR22] Destombes and Romashchenko, "Resource-bounded Kolmogorov complexity provides an obstacle to soficness of multidimensional shifts".

[168] Essentially, a picture language is a language of finite rectangular patterns; and a picture language is *recognizable* if it is the projection of a local picture language. See Section 15.5.1 for more details.

[Ter19] Terrier, "Communication complexity tools on recognizable picture languages".

[169] For example, consider the mirror subshift on $\mathbb{Z}$: its free lift is not sofic, as the $\mathbb{Z}^2$ mirror subshift would be sofic otherwise.

moment the existence of $\mathbb{Z}$ non-sofic subshifts admitting sofic free lifts. Indeed, as in Proposition 15.7 and Example 15.8, it might be possible to find a non-sofic subshift of low communication complexity (for example, $\sim \log n$) such that the communication complexity of n parallel configurations reduces to $O(n)$ (instead of the expected $\sim n \log n$).

The perceptive reader might notice that such examples fall within the case of *linear complexity*, *i.e.* when the amount of information crossing the domain of a pattern $[\![n]\!]^d$ equals the size of its border $O(n^{d-1})$. It is also the case missed by our Theorem 10.11.

At the moment, I believe that we lack the tools to properly study the difficult cases of multidimensional soficity. Our most used tool for non-soficity is a counting argument from the $\mathbb{Z}^2$ mirror subshift (Proposition 7.1), which goes wrong when exchanging patterns in configurations does not necessarily introduce forbidden patterns[170]. Parallels could be drawn with communication complexity: for example, the computation of $\mathcal{N}(\text{EQ}_n)$ in Example 15.4 is an instance in which the counting argument applies, because of the existence of *fooling pairs*[171]. Even more interesting, the complexity of dual communication problem NE_n can be explicitely computed (Example 15.5); even though it admits no fooling pairs, and thus no counting argument.

Our hope is that communication complexity could provide the necessary tools to answer the equivalent questions in symbolic dynamics. To illustrate this perspective with an example, the rest of this chapter begins the study of various subshifts based on the communication problem $\wedge_{i=1}^n \text{N1}_n$. This endeavor leaves many questions unanswered, and should be considered as an opening to later studies.

[170] We explicitely consider where said counting argument can go wrong in Question 15.26.

[171] Fooling pairs are a list of pairs $(a^{(i)}, b^{(i)})_{i\in I}$ such that the pair $(a^{(i)}, b^{(j)})$ is valid if and only if $i = j$.

15.4 Example: the problem N1

15.4.1 The problem N1

Fix $n \in \mathbb{N}$, and let us denote $W_{n,1} = \{w \in \{0,1\}^n : |w|_1 = 1\}$ the set of words of length n containing exactly one symbol 1. We define the communication problem N1_n in which Alice and Bob are respectively given two strings $u, v \in W_{n,1}$, and want to check that *u and v are not mirrors of each other*.

Formally, denote $v^{(i)} \in W_{n,1}$ the binary string of length n whose single symbol 1 is at position $i \in [\![n]\!]$; and $u^{(i)} \in W_{n,1}$ the mirror of $v^{(i)}$, *i.e.* the string whose single symbol 1 is at position $n-1-i$. Then:

$$\text{N1}_n = \{(u^{(p)}, v^{(q)}) \in W_{n,1}^2 : p \neq q\}.$$

✓ $(\dots 0100, 1000 \dots)$
✗ $(\dots 0100, 0010 \dots)$

Figure 15.1: Examples of valid and non-valid entries of N1_n.

Proposition 15.11. $\mathcal{N}(\text{N1}_n) = \theta(\log\log n)$.

Proof. To prove that $\mathcal{N}(\text{N1}_n) = O(\log\log n)$, take $(u^{(p)}, v^{(q)}) \in \text{N1}_n$. Alice and Bob have to ensure that $p \neq q$: to proceed, Alice non-deterministically picks an index $i \in [\![\log n]\!]$ and sends i to Bob, along with the i^{th} bit of the binary expansion of p; Bob compares it with the i^{th} bit of the binary expansion of q, and accepts only if these two bits differ.

The converse $\mathcal{N}(\text{N1}_n) \geq \log\log n$ is similar to Example 15.5. □

The complexity of N1_n is not additive under direct sums, which makes $\mathbb{Z}^2$ subshifts based on N1_n interesting candidates for our purposes:

Proposition 15.12. *Consider the problem $\wedge_{i=1}^n \text{N1}_n$ in which Alice and Bob are respectively given matrices $U \in \{0,1\}^{n\times n}$ and $V \in \{0,1\}^{n\times n}$ containing exactly one symbol 1 per line, and want to check that each pair of lines $(U_i, V_i) \in W_{n,1}$ verifies $(U_i, V_i) \in \text{N1}_n$, i.e. U_i and V_i are not symmetric of each other. Formally,*

$$\wedge_{i=1}^n \text{N1}_n = \{(U, V) \in (W_{n,1}^n)^2 : \forall i, (U_i, V_i) \in \text{N1}_n\}.$$

Then $\mathcal{N}(\wedge_{i=1}^{n} \mathbf{N1}_n) = O(n)$.

While this result is well-known, as a direct special case of Example 15.8, this section is actually dedicated to another proof of this result. We aim at keeping the methods involved as elementary and simple as possible, so that we can implement them as a picture language in Section 15.5.

15.4.2 Sample spaces and Linear Feedback Shift Registers (LFSRs)

Before we begin the proof of Proposition 15.12, we make a quick digression by the lands of sample spaces and linear feedback shift registers:

Definition 15.13 (Sample space)**.** *For* $n \in \mathbb{N}$*, a* sample space *is a subset* $S \subseteq \{0,1\}^n$*. It is* (ε, k)-independent *if for every set of positions* $i_1 < \cdots < i_k$ *and every binary string* $t \in \{0,1\}^k$*, we have:*

$$|P_{r \in S}(r_{i_1} \dots r_{i_k} = t) - 2^{-k}| \leq \varepsilon,$$

where $r \in S$ *is chosen uniformly at random in* S*.*

In other words, a sample space is a subset of the whole set $\{0,1\}^n$; it is (ε, k)-independent if is (up to ε) indistinguishable from a true uniform random sampling on $\{0,1\}^n$ for subwords of length k.

Definition 15.14 (LFSR)**.** A Linear Feedback Shift Register (LFSR) *is a pair of finite sequences* $f \in \{0,1\}^m$ *and* $s \in \{0,1\}^m$*. It generates a* shift register sequence $(r_i)_{i \in \mathbb{N}}$ *defined as follows:*

$$r_i = \begin{cases} s_i & \text{if } i < m \\ \sum_{j=0}^{m-1} f_i r_{i-m+j} & \text{otherwise.} \end{cases}$$

We are interested in LFSRs for two reasons: they are easy to implement into tilings (see Section 15.5); and they generate (ε, k)-independent samples spaces.

Lemma 15.15 ([Alo+90])**.** *Let* $S_{m,n}$ *be the set of all shift register sequences generated by* non-degenerate LFSRs[172]*. Then* $S_{m,n}$ *is a* $(\frac{2n}{2^m}, k)$*-independent sample space for every* $k \in [\![n]\!]$.

For the sake of completeness, we mention the standard proof:

Proof. A sample space $S \subseteq \{0,1\}^n$ is said to be ε*-biased* (with respect to linear tests) if, for X chosen uniformly at random and for every $\alpha \in \{0,1\}^n \setminus \{0\}^n$, we have $|P(\langle X, \alpha \rangle = 1) - P(\langle X, \alpha \rangle = 0)| \leq \varepsilon$.

The proof is divided in two claims:

Claim ([Alo+90, Proposition 1])**.** *For every* $m, n \in \mathbb{N}$*, the sample space* $S_{m,n}$ *is* $\frac{n-1}{2^m} \cdot (1 + O(2^{m/2}))$*-biased.*

For fixed $f \in \{0,1\}^m$ and $\alpha \in \{0,1\}^n$, we define the following two polynomials $f(t) = t^m + \sum_{j=0}^{m-1} f_j t^j$ and $g(t) = \sum_{j=0}^{n-1} \alpha_j t^j$.

We then consider the inner products $\langle r, \alpha \rangle$, for $r \in \{0,1\}^n$ ranging among the shift register sequences generated by all seeds $s \in \{0,1\}^m$. Notice that the bits $\{r_j\}_{0 \leq j < n}$ are, by induction, linear combinations of s_0, s_1, ..., s_{m-1}; and that the coefficients of these linear combinations turn out to be exactly the coefficients obtained by reducing any t^j modulo $f(t)$. Since $\langle r, \alpha \rangle$ is a linear combination of $\{r_j\}_{0 \leq j < n}$ (whose coefficients are the reduction of g modulo f), two cases occur:

[Alo+90] Alon et al., "Simple constructions of almost k-wise independent random variables".

[172] *i.e* The feedback rule $f \in \{0,1\}^m$ is non-degenerate if the associated polynomial $t^m + \sum_{j=0}^{m-1} f_j \cdot t^j$ is irreducible.

- Either $\langle r, \alpha\rangle$ is identically 0, in which case $g(t)$ is divided by $f(t)$.
- Or $\langle r, \alpha\rangle$ is not identically 0, in which case it is a non-constant linear combination of s_0, ..., s_{m-1} and is thus uniformly distributed when $s \in \{0,1\}^m$ is chosen uniformly.

Hence, the bias of $\langle r, \alpha\rangle$ (for r chosen uniformly from $S_{m,n}$) is exactly the probability that $f(t)$ divides $g(t)$, which is bounded by the number of irreducible monic polynomials of degree m dividing a given polynomial of degree $n-1$. Since a polynomial of degree $n-1$ can be divided by at most $\frac{n-1}{m}$ such irreducible monic polynomials, dividing by the number of said irreducible monic polynomials $\frac{1}{m}\cdot(2^m + O(2^{m/2}))$ concludes the proof[173].

[173] Formally, this number is

$$\frac{1}{m}\sum_{d|m}\mu(\frac{m}{d})2^d,$$

where μ is the usual Möbius function.

Claim ([Alo+90, Lemma 1][174]). *Let $S \subseteq \{0,1\}^n$ be an ε-biased sample space (with respect to linear tests). Then S is (ε, k)-independent for every $k \in \mathbb{N}$.*

[174] This result is attributed to Vazirani. Usually proved using Fourier analysis, the claim is considerably simplified on finite groups.

Let X be a uniform random variable in S such that, for any $\beta \neq 0$, we have $|P(\langle X, \beta\rangle = 1) - P(\langle X, \beta\rangle = 0)| \leq \varepsilon$. And pick $0 \leq i_1 < \cdots < i_k < n$ some indices. Denoting $\chi_\beta(\alpha) = (-1)^{\langle \alpha, \beta\rangle}$ for $\alpha, \beta \in \{0,1\}^k$, we have:

- For any $\alpha, \beta \in \{0,1\}^k$, $\sum_\beta \chi_\beta(\alpha) = \begin{cases} 2^k & \text{if } \alpha = 0^k \\ 0 & \text{otherwise.} \end{cases}$
- For any $\alpha, \alpha', \beta \in \{0,1\}^k$, $\chi_\beta(\alpha)\cdot\chi_\beta(\alpha') = \chi_\beta(\alpha+\alpha')$.

For $\beta \in \{0,1\}^k$, we introduce[175] $c_\beta = \sum_\alpha P(X_{i_1,\ldots,i_k} = \alpha)\cdot\chi_\beta(\alpha)$. Notice that $c_\beta = 1$ if $\beta = 0^k$, and that if $\beta \neq 0^k$ we have:

[175] Look, a Fourier coefficient in the wild!

$$\begin{aligned} c_\beta &= \sum_{\alpha|\langle\alpha,\beta\rangle=0} P(X_{i_1,\ldots,i_k} = \alpha) - \sum_{\alpha|\langle\alpha,\beta\rangle=1} P(X_{i_1,\ldots,i_k} = \alpha) \\ &= P(\langle X_{i_1,\ldots,i_k}, \beta\rangle = 0) - P(\langle X_{i_1,\ldots,i_k}, \beta\rangle = 1), \end{aligned}$$

so that $|c_\beta| \leq \varepsilon$ (by hypothesis). Then, for $\alpha \in \{0,1\}^k$, we have:

$$\begin{aligned} \sum_\beta \chi_\beta(\alpha)\cdot c_\beta &= \sum_\beta\sum_{\alpha'} \chi_\beta(\alpha)\cdot\chi_\beta(\alpha')\cdot P(X_{i_1,\ldots,i_k} = \alpha') \\ &= \sum_{\alpha'} P(X_{i_1,\ldots,i_k} = \alpha')\sum_\beta \chi_\beta(\alpha+\alpha') \\ &= \sum_{\alpha'} P(X_{i_1,\ldots,i_k} = \alpha')\cdot 2^k\delta_{\alpha=\alpha'} \\ &= 2^k\cdot P(X_{i_1,\ldots,i_k} = \alpha). \end{aligned}$$

Thus:

$$P(X_{i_1,\ldots,i_k} = \alpha) \leq \frac{1}{2^k}\Big|\sum_{\beta\neq 0}\chi_\beta(\alpha)\cdot c_\beta\Big| \leq \frac{1}{2^k}\sum_{\beta\neq 0}|\chi_\beta(\alpha)|\cdot|c_\beta| \leq \frac{2^{k-1}}{2^k}\,\varepsilon. \qquad \square$$

15.4.3 A construction for $\wedge_{i=1}^n \mathbf{N1}_n$

We now go back to $\wedge_{i=1}^n \mathrm{N1}_n$ and, using the sample space $S_{m,n}$ for well-chosen values of m, provide an explicit non-deterministic protocol with communication $O(n)$.

Proof. Let $U, V \in \{0,1\}^{n\times n}$ be the matrices given respectively to Alice and Bob. We denote their lines by $U_i \in W_{n,1}$ and $V_i \in W_{n,1}$, and denote by p_i (resp. q_i) the integers such that $U_i = u^{(p_i)}$ and (resp. $V_i = v^{(q_i)}$).

Inner products Let us assume that Alice and Bob can generate, non-determnistically, *the same* binary matrix $C \in \{0,1\}^{n\times n}$. Then Alice and Bob can solve the problem $\wedge_{i=1}^n \mathrm{N1}_n$ by exchanging $O(n)$ bits:

- Alice and Bob non-deterministically generate a common binary matrix $C \in \{0,1\}^{n\times n}$ such that $C_{i,p_i} \neq C_{i,q_i}$ (which is only possible if $p_i \neq q_i$).

- Alice and Bob each compute the inner products $\langle U_i, \bar{C_i}\rangle$ and $\langle V_i, C_i\rangle$ (where $\bar{C_i}$ is the mirror of C_i) as elements of $\mathbb{F}_2$.
- Alice sends the result of her inner products as a vector $R \in \mathbb{F}_2^n$, and Bob accepts if and only if $R_i \neq \langle V_i, C_i\rangle$ for every i.

Transmitting binary matrices efficiently We thus reduced the problem $\wedge_{i=1}^n \mathrm{N1}_n$ to exchanging a binary matrix with only $O(n)$ bits, which might not seem much easier (in fact, there are 2^{n^2} distinct binary matrices, so it is completely hopeless).

However, Alice and Bob do not actually need to generate *the whole set* $\{0,1\}^{n\times n}$ of binary matrices: they can just generate a subset $S \subseteq \{0,1\}^{n\times n}$ that is actually large enough so that, for all sets of distinct integers $p_i, q_i \in [\![n]\!]$ ($i \in [\![n]\!]$), there exists $C \in S$ such that $C_{i,p_i} \neq C_{i,q_i}$ for every $i \in [\![n]\!]$.

Let us fix $m = 5\log n$. By Lemma 15.15, the space $S_{m,n\log n}$ of all shift register sequences generated by non-degenerate LFSRs is[176] $(\frac{1}{n^3}, k)$-independent for every $k \in [\![n]\!]$.

[176] It is $(\frac{2n\log n}{2^{5\log n}}, k)$-independent, thus *a fortiori* $(\frac{1}{n^3}, k)$ independent.

- Let us divide $[\![n]\!] \times [\![n]\!]$ into block of $\log n$ rows and n columns.
- Send $\frac{n}{\log n}$ tuples $(f_N, s_N) \in (\{0,1\}^m)^2$ to associate one LFSR per such block, and use these LFSRs to generate the blocks of the matrix C.

Let us denote by $S \subseteq \{0,1\}^{n\times n}$ the set of matrices generated by this process. Each element $C \in S$ is determined by the pairs (f,s) generating the LFSRs, which amounts to $\frac{n}{\log n} \cdot (5\log n) = O(n)$ bits. We are left with checking that the set of matrices generated by this process is able to distinguish all pairs of positions, *i.e.* that for any $(p_i, q_i)_{0\leq i<n}$, there exists $C \in S$ such that $C_{i,p_i} \neq C_{i,q_i}$.

Since all blocks of $\log n$ rows are defined by independent LFSRs, let us focus on the first lines and prove that there exists an LFSR that generates a block $C' \in \{0,1\}^{\log n\times n}$ such that $C'_{i,p_i} \neq C'_{i,q_i}$ for $0 \leq i < \log n$. Notice that each these blocks needs to distinguish between $2\log n$ pairs of positions. Once again, let $X \in S_{m,n\log n}$ be taken uniformly at random. By $(\frac{1}{n^3}, k)$-independence,

$$\left| P(\wedge_{i=0}^{\log n-1}(X_{i,p_i} = 1 \wedge X_{i,q_i} = 0)) - \frac{1}{2^{2\log n}} \right| \leq \frac{1}{n^3},$$

so that $P(\wedge_{i=0}^{\log n-1} X_{i,p_i} \neq X_{i,q_i}) > 0$: there exists a non-degenerate LFSR (and, thus, an LFSR) that distinguishes all needed pairs of positions in the block. Joining all blocks, there must exists $C \in S$ that distinguishes between the pairs $(p_i, q_i)_{0\leq i<n}$. □

15.5 N1 as picture languages

We proved that n parallel instances of the problem $\mathrm{N1}_n$ can actually be solved by communicating only $O(n)$ bits. We now consider the implementation of $\wedge_{i=1}^n \mathrm{N1}_n$, and of the relevant protocol, into a picture language.

15.5.1 Picture languages

Let $\mathcal{A}$ be a finite alphabet, and let us fix a symbol $\# \notin \mathcal{A}$. Let us denote $\mathcal{R} = \bigcup_{m,n\in\mathbb{N}} [\![m]\!] \times [\![n]\!]$ the set of all finite rectangles in $\mathbb{Z}^2$, and $\mathcal{A}^R$ the set of all colorings of $\mathcal{R}$ by the symbols of $\mathcal{A}$.

Recall that $\mathcal{I}(D)$ denotes the interior of a domain $D \subseteq \mathbb{Z}^d$, and $\partial(R)$ its border.

Definition 15.16. *For $R' \in \mathcal{R}$ and $R = \mathcal{I}(R')$, a* picture *is a coloring $w \in \mathcal{A}^R$. Given a picture $w \in \mathcal{A}^R$, the associated* bordered picture *$\hat{w}$ is the coloring of $(\mathcal{A} \cup \{\#\})^R$ such that $\hat{w}|_{\mathcal{I}(R')} = w \in \mathcal{A}^R$ and $\hat{w}_i = \#$ for $i \in \partial(R')$.*

A picture language *is a subset* $L \subseteq \mathcal{A}^{\mathcal{R}}$. *The associated* bordered language *is the set of associated bordered pictures* $\hat{L} = \{\hat{w} : w \in L\}$.

As in traditional formal languages, picture languages can be classified depending on their complexity. For our needs, we focus on the classes of *local* and *recognizable* picture languages.

Definition 15.17. A *picture language* $L \subseteq \mathcal{A}^{\mathcal{R}}$ *is* local *if there exists a set of local* 2×1 *and* 1×2 *forbidden patterns* $\mathcal{F}$ *such that* $\hat{L}$ *is the set of bordered pictures in which none of the patterns of* $\mathcal{F}$ *appear.*

Definition 15.18. A *picture language* $L \subseteq \mathcal{A}^{\mathcal{R}}$ *is* recognizable *if there exists an alphabet* $\mathcal{B}$*, a local picture language* $L' \subseteq \mathcal{B}^{\mathcal{R}}$ *and a projection* $\pi : \mathcal{B} \to \mathcal{A}$ *such that* $L = \pi(L')$.

As the definitions may suggest, picture languages are *very similar* to sofic subshifts. In particular, when given a picture language L, we can define the subshift $X[L]$ made of non-overlapping pictures of L floating over a background of symbols #:

$$X[L] = \Big\{ x \in (\mathcal{A} \cup \{\#\})^{\mathbb{Z}^2} : \exists J, \exists (R_j)_{j\in J} \in \mathcal{R}^J, \mathcal{I}(R_{j_1}) \cap \mathcal{I}(R_{j_2}) = \emptyset \text{ if } j_1 \neq j_2, \\ \bigsqcup_{j\in J} R_j = \{i \in \mathbb{Z}^2 : x_i \neq \#\} \text{ and } \forall j \in J, x|_{R_j} \in \hat{L} \Big\}$$

Proposition 15.19. *If* L *is a recognizable picture language, then* $X[L]$ *is a sofic subshift.*

Sketch of proof. Let L' be a local language projecting onto L, and $\hat{L}'$ be the associated bordered language. The forbidden patterns of $\hat{L}'$ define an SFT $X \subseteq (\mathcal{A}' \cup \{\#\})^{\mathbb{Z}^2}$. By additionally enforcing that patterns over the alphabet $\mathcal{A}'$ are actually organized into rectangles, we obtain a sofic subshift whose projection to $\mathcal{A}$ is the subshift $X[L]$. □

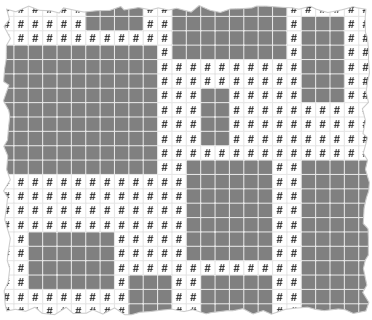


Figure 15.2: Sketch of a configuration of $X[L]$: gray rectangles represent valid pictures in L.

15.5.2 Implementing N1 as a picture language

We now associate a two-dimensional picture language to the communication problem N1. We proceed naively, by drawing the matrices U and V of Alice and Bob and separating them by a column of symbols @. Formally, we define the picture language $L_{\text{N1}} \subseteq \{0, 1, @\}^{\mathcal{R}}$ associated with N1 as:

$$L_{\text{N1}} = \bigcup_{n\in\mathbb{N}} \Big\{ w \in \{0,1,@\}^{[\![2n+1]\!]\times[\![n]\!]} : \forall i \in [\![n]\!], \exists p, q \in [\![n]\!], \\ p \neq q \text{ and } w|_{[\![2n+1]\!]\times\{i\}} = u^{(p)} @ v^{(q)} \Big\}.$$

0	0	0	0	0	0	1	0	0	0	@	0	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	0	0	@	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	@	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0	@	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	@	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	@	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0	0	@	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	@	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0	@	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	@	0	0	0	0	0	0	1	0	0

Figure 15.3: A valid picture of L_{N1}: the central column induces no symmetry on any line.

Unfortunately, we do not know whether L_{N1} is a recognizable picture language. We can, however, prove its recognizability if we allow ourselves to sparsify the columns. More precisely, for $w \in L_{\text{N1}}$ of size $(2n+1) \times n$, we define $w^{(s)}$ of size $(2n \log n + 1) \times n$ on the alphabet $\{0, 1, \sqcup, @\}$ as:

$$w^{(s)}_{i,j} = \begin{cases} @ & \text{if } i = n \log n \\ w_{j,(\frac{j}{\log n})} & \text{if } i < n \log n \text{ and } \log n \mid i \\ w_{j,(\frac{j-1}{\log n}+1)} & \text{if } i > n \log n \text{ and } \log n \mid i-1 \\ \sqcup & \text{otherwise.} \end{cases}$$

In other words, the columns in the left and right side of $w^{(s)}$ have been shifted by $\log n$ from one another, and the blanks have been filled with the symbol $\sqcup$. We then define the *sparse picture language* associated with N1 as $L^{(s)}_{\text{N1}} = \{w^{(s)} : w \in L_{\text{N1}}\} \subseteq \{0, 1, \sqcup, @\}^{\mathcal{R}}$.

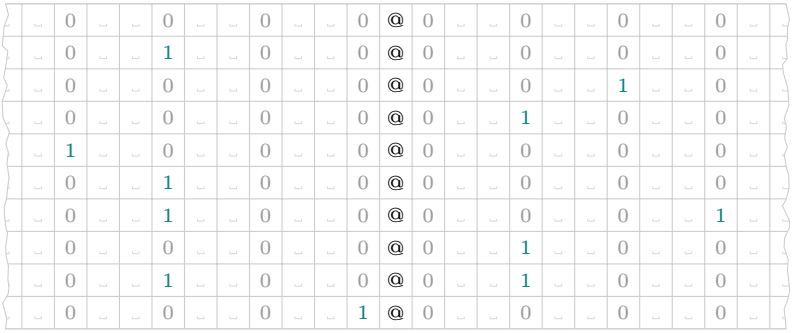


Figure 15.4: Piece of a valid sparse picture of $L^{(s)}_{\text{N1}}$.

We can then state the main result of this chapter:

Lemma 15.20. *The picture language $L^{(s)}_{\text{N1}}$ is recognizable.*

To prove that $L^{(s)}_{\text{N1}}$ is recognizable, we implement the protocol from thre previous Section 15.4.3 into a picture language.

Proof. We begin by superimposing a grid of size $\log n \times \log n$ on each halve of the pictures, where n is the height of the picture: $\log n$ can be measured, for example, by running a binary counter vertically (after n steps, the binary counter has width $\log n$). Using this grid, we may assume that, as in Section 15.4.3, the pictures are divided in blocks of $\log n$ rows, and draws a special column every $\log n$ columns.

To ensure the right size of the pictures in our local language, send a signal from the bottom-left and bottom-right corner horizontally towards the center of the picture, which goes up each time it crosses a vertical line of the grid, and should meet on the top row. This ensures that all pictures of our local language are of size $n \times (2n \log n + 1)$.

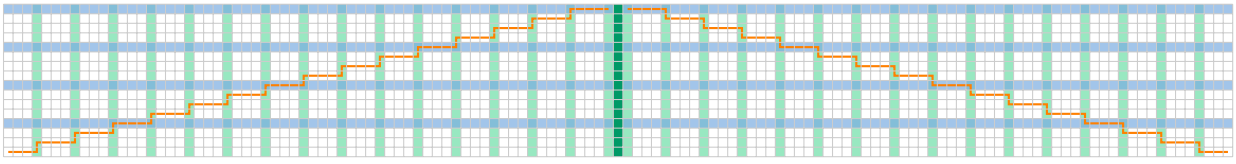


Figure 15.5: Division of a picture ($n = 4$) into blocks of $\log n$ rows with marked ■-columns. Each ■-line marks the beginning of a new block. The two ■ signals ensure the correct size of the picture and positionning of the central column.

For the rest of this proof, we reason on each block of $\log n$ rows independently.

Computing an LFSR inside a block We want to make each column in a block of $\log n$ rows contain:

- On a *rule layer*, a word $f \in \{0,1\}^{5\log n}$ written vertically (the same word is written on all columns));
- On a *random layer*, a word $w \in \{0,1\}^{5\log n}$ written vertically (this word may differ from column to column);
- On a *computation layer*, an auxiliary word $a \in \{0,1\}^{5\log n}$ (this word may differ from column to column).

Since $5\log n > \log n$, we compress the words by writing five letters per cell.

We then implement the computations of an LFSR run from the central column (which, we recall, will have its cells projected to the symbol @) towards the exterior. Since the left and right halves of the picture should implement mirror computations of said LFSR, let us focus on the right half of the block.

Let $f \in \{0,1\}^{5\log n}$ and $s \in \{0,1\}^{5\log n}$ be the words respectively written on the rule layer and the random layer of the 0^{th} (*i.e.* the central) column. For $0 \leq i \leq \log n$, let us denote $w^{(i)}$ the word written on the random layer of the i^{th} column[177], and $a^{(i)}$ the word written on its computation layer. Recall that by definition, all columns have the word f written on their rule layer; and let $(r_n)_{n\in\mathbb{N}}$ be the sequence written by the LFSR (f,s).

- In the i^{th} column, we compute the inner product $\langle f, w^{(i)}\rangle$ from bottom to top by using the letters of the auxiliary word $a^{(i)}$ as memory (in other words, $a^{(i)}_j = \langle f|_{[\![j]\!]}, w^{(i)}|_{[\![j]\!]}\rangle$). The last bit of $a^{(i)}$ is then $\langle f, w^{(i)}\rangle$.
- Going from the i^{th} to the $(i+1)^{\text{th}}$ column, we shift $w^{(i)}$ one bit down (*i.e.* $w^{(i+1)}|_{[\![5\log n-1]\!]} = w^{(i)}|_{1+[\![5\log n-1]\!]}$), and copy the last bit of $a^{(i)}$ as the new value for $w^{(i+1)}_{5\log n-1}$.

[177] So that $w^{(0)} = s$.

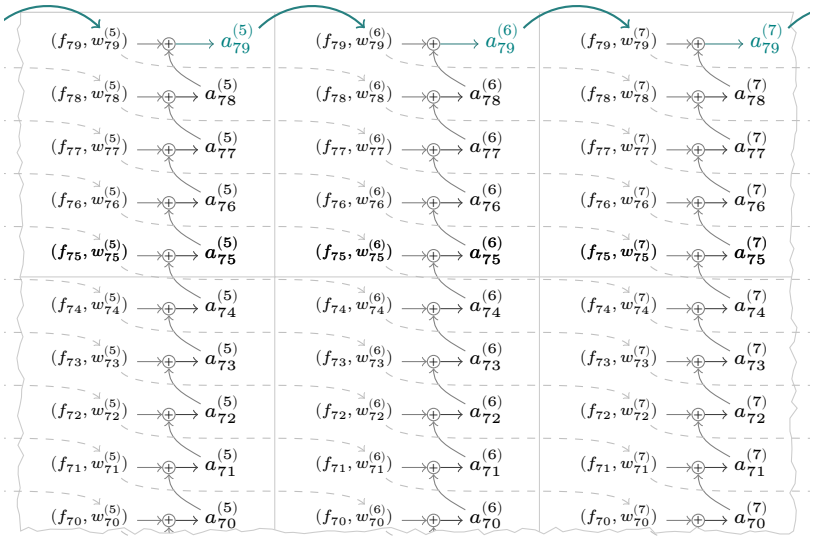


Figure 15.6: A few steps of computation of the LFSR in the local tiling.

In a block of height $n = 16$, the LFSR has length $80 = 16 \cdot 5$. The value of the $(w^{(i)}_j)_{j\in[\![80]\!]}$ in the column i are entirely determined by the column of index $i-1$: if $j < 79$, then $w^{(i)}_j = w^{(i-1)}_{j+1}$; and if $j = 79$, then $w^{(i)}_j = a^{(i-1)}_{79}$.
By construction, $\langle f, w^{(i)}\rangle = a^{(i)}_{79}$, so that the LFSR sequence $(r_n)_{n\in\mathbb{N}}$ generated by $(f, w^{(0)})$ verifies $r_{80+i} = a^{(i)}_{79}$.

At this point of the proof, every block of $\log n$ rows contains, in its top row (more precisely, we consider the last bit in every column's computation layer), the shift register sequence of length $n\log n$ generated by the pair $(f,s) \in (\{0,1\}^{5\log n})^2$.

Sending the generated bits to the next column Using wires, we move the bits from this top row to the next special columns, so that cells of each marked column contain distinct bits from the shift register sequence we just built:

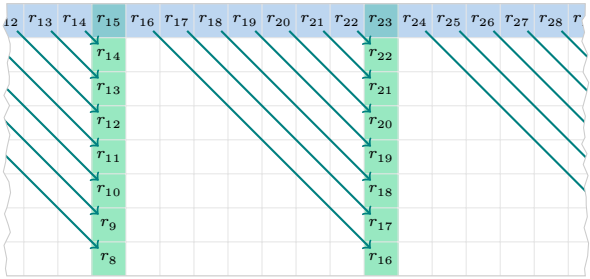


Figure 15.7: Moving the shift register sequence from the top row of a block to the marked columns.

We say that the bits of the shift register sequence appearing on the marked columns form the *random layer* of the picture.

Final adjustements We can finally add the input bits that Alice and Bob (respectively on the left and right half of each picture) will be comparing. These input bits appear on all cells that belong to a marked column. Since the language associated with the expression $0^* \cdot 1 \cdot 0^*$ is regular, we can ensure that at most a single symbol 1 appears on each halve of each line.

Then, using horizontal wires, we make all marked cells containing a symbol 1 send the bit of its *random layer* towards the central column of the picture, and compare the two bits sent respectively by the left and right half of the picture. On every line, we finally forbid these two bits to be equal.

Since we implemented the protocol introduced in Section 15.4.3, the resulting picture language is indeed $L_{\mathrm{N1}}^{(s)}$. □

From Lemma 15.20, we immediately obtain that the associated subshift is sofic:

Corollary 15.21. *The* sparse subshift $X[L_{\mathrm{N1}}^{(s)}] \subseteq \{\#, 0, 1, \_, @\}^{\mathbb{Z}^2}$ *is sofic.*

15.5.3 Perspectives

Remark 15.22. *We chose sparsification* $\log n$ *in* $L_{\mathrm{N1}}^{(s)}$ *because it is the size of the LFSRs. Yet, by superimposing several layers of the same construction, we can actually strengthen this result and prove that* $L_{\mathrm{N1}}^{(s)}$ *is recognizable for any sparsification that is an iterated logarithmic function* $\log \log \log \dots$.

This immediately leads to the following question:

Question 15.23. *Can we remove the sparsification between columns? In other words, is* L_{N1} *a recognizable picture language?*

Trying to solve this, we came up with this seemingly unrelated question[178] on cellular automata:

Question 15.24. *Does there exist a cellular automaton* $\varphi : \mathcal{A}^{\mathbb{Z}} \to \mathcal{A}^{\mathbb{Z}}$ *such that, for every positions* $i_1, j_1, i_2, j_2 \in \mathbb{Z}$ *and for and every configuration* $y \in \mathcal{A}^{\mathbb{Z}}$ *that verifies* $y_{i_2} \neq y_{j_2}$*, there exists another configuration* $x \in \mathcal{A}^{\mathbb{Z}}$ *such that* $x_{i_1} \neq x_{j_1}$ *and* $\varphi(x) = y$*?*

Essentially, φ would allow, in its space-time diagrams, to have arbitrary pairs of unrelated constraints at each time step.

[178] The core idea follows the same principle: how do we create a set of $2^{O(n)}$ binary matrices C that can distinguish between any pairs of positions in every line while being easily implemented as a picture language?

15.6 N1 as subshifts

15.6.1 The subshift X_{N1}

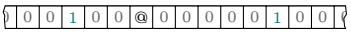


Figure 15.8: A configuration of X_{N1}.

Instead of going through picture languages, one can directly define a subshift $X_{\text{N1}} \subseteq \{0,1,@\}^{\mathbb{Z}}$ as:

$$X_{\text{N1}} = \overline{\{0^{-\mathbb{N}}10^p\,@\,0^q10^{\mathbb{N}} : p \neq q\}};$$

in other words, if a configuration contains a symbol @, then the symbols 1's (if they exist) are not mirrors of each other.

Of course, this subshift is not sofic:

Proposition 15.25. *$X_{\text{N1}} \subseteq \{0,1,@\}^{\mathbb{Z}}$ is not sofic.*

Proof. The words $w_p = 10^p@$ all have distinct extender sets in X_{N1}, since they can be followed by the word 0^q1 if and only if $q \neq p$. By Proposition 6.5, X_{N1} is not sofic. □

15.6.2 Free lift and soficity

Let us consider the free lift $X_{\text{N1}}^{\rightleftarrows} \subseteq \{0,1,@\}^{\mathbb{Z}^2}$ of X_{N1}. Its soficity is still an open problem:

Question 15.26. *Is $X_{\text{N1}}^{\rightleftarrows} \subseteq \{0,1,@\}^{\mathbb{Z}^2}$ a sofic subshift?*

I believe this to be an interesting case to study Question 15.10: the subshift $X_{\text{N1}}^{\rightleftarrows}$ is not obviously sofic; but disproving its soficness would probably require novel methods and arguments. Indeed, it is interesting to consider where typical proofs of non-soficity go wrong in the case of $X_{\text{N1}}^{\rightleftarrows}$:

Argument sketch. Let us consider $S \subseteq \{0,1,@\}^{[\![n,n]\!]}$, the square patterns of size $n \times n$ whose right border is a column of symbols @. Since there can be at most a single symbol 1 per line, there are n possibilities per line for these patterns, thus $|S| = n^n$.

However, assuming that $X_{\text{N1}}^{\rightleftarrows}$ is a sofic subshift, there could only be $2^{O(n)}$ preimages for the border $\partial([\![n,n]\!])$. Thus, in valid configurations of $X_{\text{N1}}^{\rightleftarrows}$, there exists several patterns of S whose occurences can be exchanged.

Where does this proof go wrong? As opposed to the mirror subshift case, exchanging patterns in a configuration of $X_{\text{N1}}^{\rightleftarrows}$ does not necessarily introduce a forbidden pattern. In fact, the pattern $w_p = \dots 10^p@$ can be completed by all partial configurations *except one*. Thus, the usual combinatorial argument for non-soficity (comparing the number of valid patterns with the size of the border) does not apply here, in the same way the "fooling pair" argument from communication complexity does not apply on N1. □

15.6.3 Perspectives

One main obstacle to proving the soficness of X_{N1} is that lines of a configuration need to be completely independent. However, we would not be surprised if $X_{\text{N1}}^{|} \subseteq \{0,1,@\}^{\mathbb{Z}^2}$ (the restriction of $X_{\text{N1}}^{\rightleftarrows}$ in which symbols @ are vertically aligned) turned out to be a sofic subshift.

Thus, we see two directions of study for the soficity of X_{N1}:

1. Study the soficity of $X_{\text{N1}}^{|} \subseteq \{0,1,@\}^{\mathbb{Z}^2}$, in which symbols @ are vertically aligned. A first step would be to remove the sparsification introduced in the proof of Lemma 15.20, see Question 15.23.

2. Study the picture language obtained from a variant of N1 in which the symbols @ are no longer vertically aligned (in other words, Alice and Bob's binary strings are no longer necessarily of the same length).

Bibliography

Personal bibliography

[CH20] Antonin Callard and Mathieu Hoyrup. "Descriptive complexity on non-Polish spaces". In: *STACS 2020*. Volume 154. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020, 8:1–8:16.
DOI: 10.4230/LIPIcs.STACS.2020.8 .

[CV21] Antonin Callard and Pascal Vanier. "Computational characterization of surface entropies for $\mathbb{Z}^2$ subshifts of finite type". In: *ICALP 2021*. Volume 198. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 122:1–122:20.
DOI: 10.4230/LIPIcs.ICALP.2021.122 .

[CH22] Antonin Callard and Benjamin Hellouin de Menibus. "The aperiodic Domino problem in higher dimension". In: *STACS 2022*. Volume 219. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 19:1–19:15.
DOI: 10.4230/LIPIcs.STACS.2022.19 .

[CS24] Antonin Callard and Ville Salo. "Distortion element in the automorphism group of a full shift". In: *Ergodic Theory and Dynamical Systems* 44.7 (2024), pages 1757–1817.
DOI: 10.1017/etds.2023.67 .

[CPV25] Antonin Callard, Léo Paviet Salomon, and Pascal Vanier. "Computability of extender sets in multidimensional subshifts". In: *STACS 2025*. Volume 327. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025, 21:1–21:19.
DOI: 10.4230/LIPIcs.STACS.2025.21 .

General bibliography

[Akl85] Selim G. Akl. *Parallel sorting algorithms*. Volume 12. Notes and Reports in Computer Science and Applied Mathematics. Academic Press, 1985, pages xiii+229.
DOI: 10.1016/C2013-0-10281-4 .

[Alo+90] Noga Alon et al. "Simple constructions of almost k-wise independent random variables". In: *31st Annual Symposium on Foundations of Computer Science*. Volume II. 1990, pages 544–553.
DOI: 10.1109/FSCS.1990.89575 .

[AS13] Nathalie Aubrun and Mathieu Sablik. "Simulation of effective subshifts by two-dimensional subshifts of finite type". In: *Acta Applicandae Mathematicae* 126 (2013), pages 35–63.
DOI: 10.1007/s10440-013-9808-5 .

[AS14] Nathalie Aubrun and Mathieu Sablik. "Multidimensional effective S-adic subshifts are sofic". In: *Uniform Distribution Theory* 9.2 (2014), pages 7–29.
URL: https://pcwww.liv.ac.uk/~karpenk/JournalUDT/vol09/no2/02AubrunSablick.pdf .

[AV79] Dana Angluin and Leslie G. Valiant. "Fast probabilistic algorithms for Hamiltonian circuits and matchings". In: *Journal of Computer and System Sciences* 18.2 (1979), pages 155–193.
DOI: 10.1016/0022-0000(79)90045-X .

[Bat68] Kenneth E. Batcher. "Sorting networks and their applications". In: *AFIPS '68 (American Federation of Information Processing Societies)*. 1968 Spring Joint Computer Conference. Volume 32. AFIPS Conference Proceedings. 1968, pages 307–314.
DOI: 10.1145/1468075.1468121 .

[Ber64] Robert Berger. "The undecidability of the Domino problem". PhD thesis. Harvard University, 1964
Published as "The undecidability of the Domino problem". In: *Memoirs of the American Mathematical Society* 66 (1966), pages 1–72.

[Cap08] Silvio Capobianco. "Multidimensional cellular automata and generalization of Fekete's lemma". In: *Discrete Mathematics & Theoretical Computer Science* (DMTCS) 10.3 (2008), pages 95–104.
DOI: 10.46298/dmtcs.442 .

[Cas10] Julien Cassaigne. *Odd shift*. Unpublished results (see these slides for pointers). 2010 .

[CR73] Stephen A. Cook and Robert A. Reckhow. "Time bounded random access machines". In: *Journal of Computer and System Sciences* 7.4 (1973), pages 354–375.
DOI: 10.1016/S0022-0000(73)80029-7 .

[CS92] Peter F. Corbett and Isaac D. Scherson. "Sorting in Mesh Connected Multiprocessors". In: *IEEE Transactions on Parallel & Distributed Systems* 3.5 (1992), pages 626–632.
DOI: 10.1109/71.159046 .

[Des06] Angela Desai. "Subsystem entropy for $\mathbb{Z}^d$ sofic shifts". In: *Indagationes Mathematicae* 17.3 (2006), pages 353–359.
DOI: 10.1016/S0019-3577(06)80037-6 .

[Des21] Juline Destombes. "Algorithmic complexity and soficness of shifts in dimension two". (In French). PhD thesis. Université de Montpellier, 2021.
arXiv: 2309.12241 [cs.IT].
URL: https://theses.fr/2021MONTS129 .

[DLS08] Bruno Durand, Leonid A. Levin (Леони́д Анато́лійович Ле́він), and Alexander K. Shen (Александр Ханиевич Шень). "Complex tilings". In: *The Journal of Symbolic Logic* 73.2 (2008), pages 593–613.
DOI: 10.2178/jsl/1208359062 .

[DR22] Julien Destombes and Andrei E. Romashchenko (Андрей Евгеньевич Ромащенко). "Resource-bounded Kolmogorov complexity provides an obstacle to soficness of multidimensional shifts". In: *Journal of Computer and System Sciences* 128 (2022), pages 107–134.
DOI: 10.1016/j.jcss.2022.04.002 .

[DRS08] Bruno Durand, Andrei E. Romashchenko (Андрей Евгеньевич Ромащенко), and Alexander K. Shen (Александр Ханиевич Шень). “Fixed point and aperiodic tilings”. In: *DLT 2008 (Developments in Language Theory)*. Volume 5257. Lecture Notes in Computer Science. 2008, pages 276–288.
DOI: 10.1007/978-3-540-85780-8_22 .

[DRS10] Bruno Durand, Andrei E. Romashchenko (Андрей Евгеньевич Ромащенко), and Alexander K. Shen (Александр Ханиевич Шень). “Effective closed subshifts in 1D can be implemented in 2D”. In: *Fields of logic and computation*. Volume 6300. Lecture Notes in Computer Science. Springer, 2010, pages 208–226.
DOI: 10.1007/978-3-642-15025-8_12 .

[DRS12] Bruno Durand, Andrei E. Romashchenko (Андрей Евгеньевич Ромащенко), and Alexander K. Shen (Александр Ханиевич Шень). “Fixed-point tile sets and their applications”. In: *Journal of Computer and System Sciences* 78.3 (2012), pages 731–764.
DOI: 10.1016/j.jcss.2011.11.001 .

[ER64] Calvin C. Elgot and Abraham Robinson. “Random-access stored-program machines, an approach to programming languages”. In: *Journal of the Association for Computing Machinery* 11.4 (1964), pages 365–399.
DOI: 10.1145/321239.321240 .

[FP19] Thomas French and Ronnie Pavlov. “Follower, predecessor, and extender entropies”. In: *Monatshefte für Mathematik* 188.3 (2019), pages 495–510.
DOI: 10.1007/s00605-018-1224-5 .

[Fre16a] Thomas K. French. “Characterizing follower and extender set sequences”. In: *Dynamical Systems* 31.3 (2016), pages 293–310.
DOI: 10.1080/14689367.2015.1111865 .

[Fre16b] Thomas K. French. “Follower and extender sets in symbolic dynamics”. PhD thesis. University of Denver, 2016.
URL: https://digitalcommons.du.edu/etd/1152 .

[Gác01] Péter Gács. “Reliable cellular automata with self-organization”. In: *Journal of Statistical Physics* 103.1-2 (2001), pages 45–267.
DOI: 10.1023/A:1004823720305 .

[Gác86] Péter Gács. “Reliable computation with cellular automata”. In: *Journal of Computer and System Sciences* 32.1 (1986), pages 15–78.
DOI: 10.1016/0022-0000(86)90002-4 .

[GJ15] Pierre Guillon and Emmanuel Jeandel. *Infinite Communication Complexity*. 2015.
arXiv: 1501.05814 [cs.CC] .

[GS23] Léo Gayral and Mathieu Sablik. “Arithmetical hierarchy of the Besicovitch-stability of noisy tilings”. In: *Theory of Computing Systems* 67.6 (2023), pages 1209–1240.
DOI: 10.1007/s00224-023-10142-y .

[Har71] Juris Hartmanis. “Computational complexity of random access stored program machines”. In: *Mathematical Systems Theory* 5 (1971), pages 232–245.
DOI: 10.1007/BF01694180 .

[Har86] David Harel. “Effective transformations on infinite trees, with applications to high undecidability, dominoes, and fairness”. In: *Journal of the Association for Computing Machinery* 33.1 (1986), pages 224–248.
DOI: 10.1145/4904.4993 .

[Hed69] Gustav A. Hedlund. “Endomorphisms and automorphisms of the shift dynamical system”. In: *Mathematical Systems Theory* 3 (1969), pages 320–375.
DOI: 10.1007/BF01691062 .

[HM10] Michael Hochman and Tom Meyerovitch. “A characterization of the entropies of multidimensional shifts of finite type”. In: *Annals of Mathematics* 171.3 (2010), pages 2011–2038.
DOI: 10.4007/annals.2010.171.2011 .

[Hoc09] Michael Hochman. “On the dynamics and recursive properties of multidimensional symbolic systems”. In: *Inventiones Mathematicae* 176.1 (2009), pages 131–167.
DOI: 10.1007/s00222-008-0161-7 .

[Hoc10] Michael Hochman. “On the automorphism groups of multidimensional shifts of finite type”. In: *Ergodic Theory and Dynamical Systems* 30.3 (2010), pages 809–840.
DOI: 10.1017/S0143385709000248 .

[HS74] Juris Hartmanis and Janos Simon. “On the power of multiplication in random access machines”. In: SWAT 1974. Cornell University, 1974, pages 13–23.
DOI: 10.1109/SWAT.1974.20 .

[JK12] Timo Jolivet and Jarkko Kari. “Consistency of multidimensional combinatorial substitutions”. In: *Theoretical Computer Science* 454 (2012), pages 178–188.
DOI: 10.1016/j.tcs.2012.03.050 .

[JV15] Emmanuel Jeandel and Pascal Vanier. “Characterizations of periods of multi-dimensional shifts”. In: *Ergodic Theory and Dynamical Systems* 35.2 (2015), pages 431–460.
DOI: 10.1017/etds.2013.60 .

[Kle38] Stephen Cole Kleene. “On notation for ordinal numbers”. In: *Journal of Symbolic Logic* 3.4 (1938), pages 150–155.
DOI: 10.2307/2267778 .

[KM13] Steve Kass and Kathleen Madden. “A sufficient condition for non-soficness of higher-dimensional subshifts”. In: *Proceedings of the American Mathematical Society* 141.11 (2013), pages 3803–3816.
DOI: 10.1090/S0002-9939-2013-11646-1 .

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997, pages xiv+189.
DOI: 10.1017/CBO9780511574948 .

[KR84] David Kirkpatrick and Stefan Reisch. “Upper bounds for sorting integers on random access machines”. In: *Theoretical Computer Science* 28.3 (1984), pages 263–276.
DOI: 10.1016/0304-3975(83)90023-3 .

[Kůr03] Petr Kůrka. *Topological and symbolic dynamics*. Volume 11. Cours spécialisés. Société Mathématique de France, 2003, pages xii+315. ISBN: 2-85629-143-0 .

[Lei92] F. Thomson Leighton. *Introduction to parallel algorithms and architectures. Arrays, trees, hypercubes*. Morgan Kaufmann Publishers, 1992, pages xx+831.
DOI: 10.1016/C2013-0-08299-0 .

[LM95] Douglas A. Lind and Brian Marcus. *An introduction to symbolic dynamics and coding*. Cambridge University Press, 1995.
DOI: 10.1017/CBO9780511626302 .

[Mel61] Zdzislaw A. Melzak. “An informal arithmetical approach to computability and computation”. In: *Canadian Mathematical Bulletin* 4.3 (1961), pages 279–293.
DOI: 10.4153/CMB-1961-031-9 .

[Mey11] Tom Meyerovitch. “Growth-type invariants for $\mathbb{Z}^d$ subshifts of finite type and arithmetical classes of real numbers”. In: *Inventiones Mathematicae* 184.3 (2011), pages 567–589.
DOI: 10.1007/s00222-010-0296-1 .

[MH38] Marston H. C. Morse and Gustav A. Hedlund. “Symbolic dynamics”. In: *American Journal of Mathematics* 60.4 (1938), pages 815–866.
DOI: 10.2307/2371264 .

[Min61] Marvin L. Minsky. “Recursive unsolvability of Post’s problem of “tag” and other topics in theory of Turing machines”. In: *Annals of Mathematics* 74.3 (1961), pages 437–455.
DOI: 10.2307/1970290 .

[Mor21] Marston Morse. “Recurrent geodesics on a surface of negative curvature”. In: *Transactions of the American Mathematical Society* 22.1 (1921), pages 84–100.
DOI: 10.2307/1988844 .

[Moz89] Shahar Mozes. “Tilings, substitution systems and dynamical systems generated by them”. In: *Journal d’Analyse Mathématique* 53 (1989), pages 139–186.
DOI: 10.1007/BF02793412 .

[MP22] Benoît Monin and Ludovic Patey. *Calculabilité. Degrés Turing, théorie algorithmique de aléatoire, mathématiques à rebours et hypercalculabilité*. Tableau noir. Calvage & Mounet, 2022 .

[NS79] David Nassimi and Sartaj Sahni. “Bitonic sort on a Mesh-Connected Parallel Computer”. In: IEEE *Transactions on Computers* 28.1 (1979), pages 2–7.
DOI: 10.1109/TC.1979.1675216 .

[OP16] Nic Ormes and Ronnie Pavlov. “Extender sets and multidimensional subshifts”. In: *Ergodic Theory and Dynamical Systems* 36.3 (2016), pages 908–923.
DOI: 10.1017/etds.2014.71 .

[Pav13] Ronnie Pavlov. “A class of nonsofic multidimensional shift spaces”. In: *Proceedings of the American Mathematical Society* 141.3 (2013), pages 987–996.
DOI: 10.1090/S0002-9939-2012-11382-6 .

[PV23] Léo Paviet Salomon and Pascal Vanier. “Realizing finitely presented groups as projective fundamental groups of SFTs”. In: *MFCS 2023*. Volume 272. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 75:1–75:15.
DOI: 10.4230/LIPIcs.MFCS.2023.75 .

[Pyt02] N. Pythéas Fogg. *Substitutions in dynamics, arithmetics and combinatorics*. Volume 1794. Lecture Notes in Mathematics. Springer, 2002, pages xviii+402.
DOI: 10.1007/b13861 .

[QT00] Anthony N. Quas and Paul B. Trow. “Subshifts of multi-dimensional shifts of finite type”. In: *Ergodic Theory and Dynamical Systems* 20.3 (2000), pages 859–874.
DOI: 10.1017/S0143385700000468 .

[Ric72] Daniel Richardson. “Tessellations with local transformations”. In: *Journal of Computer and System Sciences* 6 (1972), pages 373–388.
DOI: 10.1016/S0022-0000(72)80009-6 .

[Rob71] Raphael M. Robinson. “Undecidability and nonperiodicity for tilings of the plane”. In: *Inventiones Mathematicae* 12 (1971), pages 177–209.
DOI: 10.1007/BF01418780 .

[RS59] Michael O. Rabin and Dana S. Scott. “Finite automata and their decision problems”. In: IBM *Journal of Research and Development* 3.2 (1959), pages 114–125.
DOI: 10.1147/rd.32.0114 .

[Sch95] Klaus Schmidt. “The cohomology of higher-dimensional shifts of finite type”. In: *Pacific Journal of Mathematics* 170.1 (1995), pages 237–269.
DOI: 10.2140/pjm.1995.170.237 .

[Sim15] Stephen G. Simpson. “Symbolic dynamics: entropy = dimension = complexity”. In: *Theory of Computing Systems* 56.3 (2015), pages 527–543.
DOI: 10.1007/s00224-014-9546-8 .

[Sli78] Anatol O. Slissenko (Анатоль Олесьевич Слисенко). “Методы вычислений, основанные на адресной организации памяти” [“Models of Computations Based on Address Organization of Storage”]. In: *Всесоюзн. симпозиум « Искусств, интеллект и автоматизация исследова ний в матем. » Тезисы докладов и сообщений* [*Proc. Soviet Symp. on AI and Automation of Research in Mathematics*]. Інститут кібернетики, Киев [Institute of Cybernetics, Kiev], 1978, pages 94–96 .

[Sli79] Anatol O. Slissenko (Анатоль Олесьевич Слисенко). “Сложностные задачи теории вычислений”. In: *Научн. совет по компл. проблеме «Кибернетика»*. (Preprint), 1979.
DOI: 10.1070/RM1981v036n06ABEH003102
Translated as “Complexity problems in computational theory”. In: *Russian Mathematical Surveys* 36.6 (1981), pages 23–125.

[Sli81] Anatol O. Slissenko (Анатоль Олесьевич Слисенко). “Поиск периодичностей и идентификация полслов в реальное время”. In: *Теоретические применения методов математической логики. III*. Volume 105. Записки научных семинаров ЛОМИ. Записки научных семинаров, 1981, pages 62–173.
DOI: 10.1007/BF01084395
Translated as “Detection of periodicities and string-matching in real time”. In: *Journal of Soviet Mathematics* 22 (1983), pages 1316–1387.

[SSS86] Sandeep Sen, Isaac D. Scherson, and Adi Shamir. “Shear Sort: A True Two-Dimensional Sorting Techniques for VLSI Networks”. In: ICPP’86 (*International Conference on Parallel Processing*). 1986, pages 903–908 .

[Ter19] Véronique Terrier. “Communication complexity tools on recognizable picture languages”. In: *Theoretical Computer Science* 795 (2019), pages 194–203.
DOI: 10.1016/j.tcs.2019.05.040 .

[TK77] Clark D. Thompson and Kung Hsiang-Tsung (孔祥重). “Sorting on a Mesh-Connected Parallel Computer”. In: *Communications of the ACM (Association for Computing Machinery)* 20.4 (1977), pages 263–271.
DOI: 10.1145/359461.359481 .

[Tör21] Ilkka Törmä. “Fixed point constructions in tilings and cellular automata”. In: AUTOMATA 2021 (*International Workshop on Cellular Automata and Discrete Complex Systems*). Volume 90. Open Access Series in Informatics (OASIcs). Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2021, 4:1–4:13.
DOI: 10.4230/oasics.automata.2021.4 .

[Wan57] Wang Hao (王浩). “A variant to Turing’s theory of computing machines”. In: *Journal of the Association for Computing Machinery* 4 (1957), pages 61–92.
DOI: 10.1145/320856.320867 .

[Wan61] Wang Hao (王浩). “Proving theorems by pattern recognition – II”. In: *Bell System Technical Journal* 40.1 (1961), pages 1–41.
DOI: 10.1002/j.1538-7305.1961.tb03975.x .

[Wei73] Benjamin Weiss. “Subshifts of finite type and sofic systems”. In: *Monatshefte für Mathematik* 77 (1973), pages 462–474.
DOI: 10.1007/BF01295322 .

[Wes17] Linda B. Westrick. “Seas of squares with sizes from a Π^0_1 set”. In: *Israel Journal of Mathematics* 222.1 (2017), pages 431–462.
DOI: 10.1007/s11856-017-1596-6 .

[Wie83] Jiří Wiedermann. “Deterministic and Nondeterministic Simulation of the RAM by the Turing Machine”. In: *IFIP 9th World Computer Congress*. 1983, pages 163–168 .

[Zin15] Charalampos Zinoviadis. “Hierarchy and expansiveness in 2D subshifts of finite type”. In: LATA 2015. Volume 8977. Lecture Notes in Computer Science. Springer, 2015, pages 365–377.
DOI: 10.1007/978-3-319-15579-1_28 .

[ZW01] Zheng Xizhong (郑锡忠) and Klaus Weihrauch. “The arithmetical hierarchy of real numbers”. In: *Mathematical Logic Quarterly* 47.1 (2001), pages 51–65.
DOI: 10.1002/1521-3870(200101)47:1<51::AID-MALQ51>3.0.CO;2-W .

Soficity of multidimensional subshifts

Keywords: tilings, sofic subshifts, symbolic dynamics, computability

Résumé (FR) En dynamique symbolique, un sous-shift multidimensionnel est un language formel $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ de coloriages infinis de l'espace discret $\mathbb{Z}^d$ défini en termes de motifs interdits. Comme les langages de mots finis, pour lesquels ont été définies des classes de complexité (qui incluent classiquement les langages locaux, rationnels, algébriques ou calculablement énumérables…) en fonction de l'expressivité des différentes machines qui les reconnaissent (respectivement : les automates locaux, les automates finis, les automates à piles et les machines de Turing), les sous-shifts ont été classifiés en sous-shifts de types finis (définis par des familles finies de motifs interdits), sous-shifts effectifs (définis par des familles calculablement énumérables) et sous-shifts sofiques : ces derniers forment une classe intermédiaire entre les deux précédentes, et sont définis comme les images morphiques des sous-shifts de types finis par des automates cellulaires.

Nous nous intéressons à la question suivante : quand un sous-shift donné $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ est-il sofique ? Autrement dit, comment prouve-t-on (ou réfute-t-on) la soficité d'un sous-shift ? Si cette question est résolue en dimension 1 (les sous-shifts sofiques sur $\mathbb{Z}$ étant similaires aux langages rationnels, le théorème de Myhill-Nerode caractérise la soficité en dimension 1 par comptage du nombre de « contextes »), décrire la frontière entre les sous-shifts multidimensionnels sofiques et effectifs reste un problème ouvert en dynamique symbolique.

Cette thèse se divise en deux parties indépendantes, précédées de chapitres préliminaires d'introduction et de définitions des notions étudiées (de dynamique symbolique, calculabilité…). Dans la première partie, nous étudions les ensembles d'extensions des sous-shifts sur $\mathbb{Z}^d$ (qui, informellement, comptent les classes de motifs qui peuvent être librement échangés dans les configurations d'un sous-shift) selon leur (in)calculabilité : en particulier, nous prouvons que les *entropies d'extensions* des sous-shifts (*i.e.* le taux de croissance du nombre d'ensemble d'extensions) peuvent être entièrement caractérisées calculablement dans la hiérarchie arithmétique des nombres réels, le niveau précis dépendant de la complexité et des propriétés dynamiques vérifiées par le sous-shift considéré. Dans la seconde partie, nous prouvons une condition suffisante pour la soficité des sous-shifts multidimensionnels s'appuyant sur une quantification de « l'information utile » contenue dans les motifs : plus précisément, nous introduisons une notion de *représentation inductive* (qui, informellement, décrit l'information échangée par des motifs adjacents d'une taille donnée pour vérifier la validité locale d'une configuration), et nous prouvons qu'admettre des représentations calculables de petites complexité est une condition suffisante pour la soficité d'un sous-shift. Enfin, nous présentons ces résultats comme une complexité de communication sur des coloriages infinis, et argumentons que la complexité de communication non-déterministe forme un cadre riche pour l'étude de la soficité des sous-shifts multidimensionnels.

Abstract (EN) In symbolic dynamics, a multidimensional subshift is a formal language $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ of infinite colorings of the discrete space $\mathbb{Z}^d$ defined in terms of forbidden patterns. As languages of finite words have been classified into several complexity classes (which, classically, include the local regular, context-free, and computably enumerable languages…) depending on the expressiveness of the various devices used for their descriptions (respectively: local automata, finite automata, pushdown automata, Turing machines…), subshifts have been classified into subshifts of finite type (definied by finite families of forbidden patterns), effective subshifts (defined by computably enumerable families of forbidden patterns) and the *sofic subshifts*: the latter form an intermediary class, and are defined as the morphic images of subshifts of finite type by cellular automata.

We are interested in the following question: when is a given subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ actually sofic? In other words, how does one prove or disprove the soficity of a subshift? While this question is entirely solved in the one-dimensional setting (as $\mathbb{Z}$ sofic subshifts are very similar to regular languages of finite words, the Myhill-Nerode theorem characterizes one-dimensional soficity by counting the number of possible "contexts"), describing the frontier between sofic and effective multidimensional subshifts is still an open problem in symbolic dynamics.

This thesis is divided in two independent parts, with preliminary chapters of introduction and definitions of the relevant notions being considered (from symbolic dynamics, computability theory…). In the first part, we study the extender sets of $\mathbb{Z}^d$ subshifts (which, informally, count the classes of patterns that can be freely exchanged in the configurations of a subshift) using computability theory: in particular, we prove that *extender entropies* of $\mathbb{Z}^d$ subshifts (*i.e.* the growth rate of the number of extender sets) can be fully characterized computationally in the arithmetical hierarchy of real numbers, the precise level depending on the complexity and the dynamical properties verified by the considered subshifts. In the second part, we prove a sufficient condition for multidimensional soficity based on a quantification of the "useful information" contained in patterns: more precisely, we introduce a notion of *inductive representations* (which, informally, describe the information exchanged between adjacent patterns of a given size to check the local validity of a configuration), and prove that admitting computable representations of small complexity is a sufficient condition for soficity. Finally, we describe these results as a variant of communication complexity on infinite colorings, and argue that non-deterministic communication complexity is a fruitful context of the study of multidimensional soficity.