Charalampos Zinoviadis

Hierarchy and Expansiveness in Two-Dimensional Subshifts of Finite Type

Turku Centre *for* Computer Science

TUCS Dissertations
No 209, March 2016

Hierarchy and Expansiveness in Two-Dimensional Subshifts of Finite Type

Charalampos Zinoviadis

To be presented, with the permission of the Faculty of Mathematics and Natural Sciences of the University of Turku, for public criticism in Auditorium I/Tauno Nurmela-sali on March 11, 2016, at 12 noon.

University of Turku
Department of Mathematics and Statistics
FI- 20014 Turku
Finland

2016

Supervisor

Jarkko Kari
Department of Mathematics and Statistics
University of Turku
FI-20014 Turku
Finland

Reviewers

Andrei E. Romashchenko
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
University of Montpellier
Université Montpellier 2, LIRMM UMR 5506, CC477, 161 rue Ada, 34095 Montpellier Cedex 5
France

Nicolas Ollinger
Laboratoire d'Informatique Fondamentale d'Orléans
University of Orléans
LIFO 3IA, 6 rue Léonard de Vinci, BP 6759, 45067 Orléans Cedex 2
France

Opponent

Mike Hochman
Einstein Institute of Mathematics
The Hebrew University
Edmond J. Safra Campus (Givat Ram), Jerusalem 91904
Israel

Abstract

Subshifts are sets of configurations over an infinite grid defined by a set of forbidden patterns. In this thesis, we study two-dimensional subshifts of finite type (2D SFTs), where the underlying grid is $\mathbb{Z}^2$ and the set of forbidden patterns is finite. We are mainly interested in the interplay between the computational power of 2D SFTs and their geometry, examined through the concept of expansive subdynamics. 2D SFTs with expansive directions form an interesting and natural class of subshifts that lie between dimensions 1 and 2. An SFT that has only one non-expansive direction is called extremely expansive. We prove that in many aspects, extremely expansive 2D SFTs display the totality of behaviours of general 2D SFTs.

For example, we construct an aperiodic extremely expansive 2D SFT and we prove that the emptiness problem is undecidable even when restricted to the class of extremely expansive 2D SFTs. We also prove that every Medvedev class contains an extremely expansive 2D SFT and we provide a characterization of the sets of directions that can be the set of non-expansive directions of a 2D SFT. Finally, we prove that for every computable sequence of 2D SFTs with an expansive direction, there exists a universal object that simulates all of the elements of the sequence. We use the so called hierarchical, self-simulating or fixed-point method for constructing 2D SFTs which has been previously used by Gács, Durand, Romashchenko and Shen.

Acknowledgements

First of all, I would like to express my gratitude to my supervisor, Professor Jarkko Kari, for his guidance, support and patience as well as for trusting in me.

I am grateful to Dr. Andrei Romashchenko and Professor Nicolas Ollinger for agreeing to review my dissertation and for their constructive comments that helped me improve the thesis.

I would also like to thank Professor Mike Hochman for agreeing to act as my opponent. It is both a pleasure and an honour for me, since his mathematical work has largely influenced the direction of my research.

I am also heavily indebted to Dr. Pierre Guillon. Almost all of the results included in this thesis have been obtained in close, even if usually difficult, collaboration with him. We often disagree about everything, but this has helped me learn and, hopefully, improve the content and form of my thesis.

I am grateful to the Turku Center for Computer Science (TUCS), the Finnish Academy of Science and Letters, Professor Jarkko Kari and the Department of Mathematics for providing me with steady funding during the course of my PhD studies. Thanks to everybody that works in the Department of Mathematics, the working environment has always been warm and supportive, and this has helped very much my stay in Finland.

Finally, I want to thank my friends and family for their love and support.

Contents

Chapter 1

Historical overview

This thesis is about two-dimensional subshifts of finite type (2D SFTs), and more specifically, the behaviour of 2D SFTs with respect to a dynamical-geometrical notion called expansive subdynamics.

The mathematical study of 2D SFTs began with the paper of Wang [40]. A **Wang tile set** consists of a finite number of unit squares with coloured edges, which are called **tiles**. A valid tiling is a way to fill the entire plane with tiles such that the squares are edge-to-edge and such that the colors of abutting edges are the same. Wang asked the following question about Wang tile sets, which is called the **tiling problem**: Does there exist an algorithm that takes as input an arbitrary Wang tile set and decides whether it admits a valid tiling? He conjectured that the answer to this question is positive and proved that the problem is strongly correlated to the problem of the existence of an **aperiodic tile set**, that is a tile set that admits some valid tiling but no periodic valid tiling.

However, Berger [4] proved that this is not the case. In fact, he proved that the tiling problem is undecidable. In addition, his proof contained an explicit construction of an aperiodic tile set. Later, several authors have given alternative constructions of aperiodic tile sets and proofs of the undecidability of the tiling problem [37, 25, 26].

There is an alternative way of looking at and talking about the same problem. Let $\mathcal{A}$ be a finite set, called the **alphabet**. A (two-dimensional, or 2D) **configuration** is a map $c\colon \mathbb{Z}^2 \to \mathcal{A}$. The set of all configurations $\mathcal{A}^{\mathbb{Z}^2}$ is called the **full shift**. A **pattern** is a map $p\colon D \to \mathcal{A}$, where $D \subseteq \mathbb{Z}^2$ is a *finite* set. Let $\mathcal{F}$ be a set of **forbidden** patterns. The corresponding (2D) **subshift** $X_{\mathcal{F}}$ is the set of all configurations that avoid the patterns of $\mathcal{F}$: for all finite $D \subseteq \mathbb{Z}^2$ and $c \in X_{\mathcal{F}}$, $c_{|D} \notin \mathcal{F}$. If $X = X_{\mathcal{F}}$ for some *finite* set of forbidden patterns, then it is called a 2D subshift of finite type (SFT). In this thesis, we will only talk about 2D subshifts and SFTs, so that we will usually omit the dimension, except in statements of theorems.

It is not difficult to see that the set of valid tilings of a Wang tile set is an SFT. In addition, for every SFT, we can construct a Wang tile set whose set of valid tilings is, in some sense, equivalent to the given SFT. The tiling problem can thus be rephrased as the **emptiness problem** for SFTs: Given a finite set of forbidden patterns, can we algorithmically decide whether $X_{\mathcal{F}} \neq \emptyset$? The undecidability of the tiling problem then is then immediately translated to the undecidability of the emptiness problem of SFTs.

Wang tiles and forbidden patterns give a geometrical definition of SFTs, but there also exists an equivalent dynamical definition. First of all, the full shift can be endowed with the product topology of the discrete topology on $\mathcal{A}$. This gives rise to a compact, metrizable topological space. The **horizontal** and **vertical shifts**, which consist in moving a configuration one step to the left and up, respectively, are continuous with respect to this topology and obviously commute. This defines a $\mathbb{Z}^2$ action over the full shift and we can study it using the usual tools of topological dynamics.

For example, one can prove that subshifts are exactly the closed, shift-invariant subsets of the full shift, or, equivalently the subsystems of the full shift. SFTs correspond to the chain-mixing subsystems of the full shift. More importantly, for the purposes of this thesis, we can study 2D SFTs from the point of view of their **expansive subdynamics**. This notion was defined by Boyle and Lind [6] as a tool for studying multidimensional dynamical systems by looking at the (lower-dimensional) actions induced by the subgroups of the original action. Intuitively, this is the same as when we look at the lower-dimensional projections of a surface in order to understand some of its properties.

The general definition of expansive subdynamics and the main results of [6] fall out of the scope of this thesis. However, for 2D subshifts there exists an equivalent, natural geometrical definition. Let $X \subseteq \mathcal{A}^{\mathbb{Z}^2}$ be a subshift, $l \in \mathbb{P} := \mathbb{R} \sqcup \{\infty\}$ a slope and $\mathbf{l} \subset \mathbb{R}^2$ the corresponding line that passes through the origin. We say that l is an **expansive direction** of X if there exists a finite shape $V \subset \mathbb{R}^2$ such that, for all $x, y \in X$,

$$x_{|(\mathbf{l}+V)\cap\mathbb{Z}^2} = y_{|(\mathbf{l}+V)\cap\mathbb{Z}^2} \Rightarrow x = y \ .$$

In other words, there exists a fixed width $b > 0$ such that every configuration of X is uniquely defined by its restriction to the strip of slope l and width b that goes through the origin (in fact, by shift invariance, by any strip). Geometrically, this means that in X the (2D) information of the configuration is "packed" inside the one-dimensional strip of slope l. In some sense, even though X is a two-dimensional object, it is determined by a one-dimensional strip, so that subshifts with directions of expansiveness are somewhere between dimensions 1 and 2.

A direction that is not expansive is called **non-expansive**. Let $\mathcal{N}(X)$ be the set of non-expansive directions of X. Boyle and Lind proved that $\mathcal{N}(X)$ is closed in the one-point compactification topology of $\mathbb{P}$ and that $\mathcal{N}(X) \neq \emptyset$ if and only if X is infinite. Since finite subshifts are rather trivial, the most restricted non-trivial case with respect to non-expansive directions is the case when X has a unique direction of non-expansiveness. We call such a subshift **extremely expansive**. Extremely expansive SFTs form the main object of interest in this thesis. We prove that in many aspects, extremely expansive SFTs are computationally as powerful as general SFTs.

Before stating the results, we find it useful to talk about another class of SFTs with many directions of non-expansiveness, namely those that arise from deterministic tile set. A tile set is called **NW-deterministic** (the initials stand for North and West) if every tile is uniquely determined by the colors of its top and left sides[23]. Similarly, we can define SW, SE and NE deterministic tile sets (S and E stand for South and East, respectively). A tile set is called **4-way deterministic** if it is SW,NW,SE and NE deterministic [22]. One can easily see that for the SFT associated to a 4-way deterministic tile set and for every direction l that is not the vertical or the horizontal one (slopes ∞ and 0, respectively), l is an expansive direction. Guillon, Kari and Zinoviadis recently proved [13] that the vertical and the horizontal direction must indeed be non-expansive unless the associated SFT is in some sense trivial, namely vertically or horizontally periodic.

We can now start stating the results of the thesis. The first result concerns the existence of an aperiodic extremely expansive SFT. As mentioned earlier, for the unrestricted case, there exist various constructions of aperiodic SFTs. Kari and Papasoglou [22] have constructed an aperiodic 4-way deterministic tile set. According to what was said in the previous paragraph, the SFT associated to this tile set has exactly two non-expansive directions, the vertical and the horizontal one. We prove that

Theorem 1. *There exists an aperiodic extremely expansive* $2D$ *SFT.*

Of course, our construction does not use a 4-way deterministic tile set. It might seem that this result is strictly better than the one using 4-way deterministic tile sets, since we have one non-expansive direction less. However, there exists a small nuance here: 4-way deterministic tile sets give rise to SFTs with so-called **bounded radii of expansiveness**, while our construction does not have this property. In addition, in [13] it is also proved that every aperiodic SFT with bounded radii of expansiveness must have at least two non-expansive directions. Therefore, the 4-way deterministic construction is also optimal, in the class of SFTs with bounded radii of expansiveness, and it might be more precise to say that the two results are incomparable.

As mentioned already, the existence of an aperiodic tile set was originally constructed in order to prove that the tiling problem is undecidable. Kari [23] prove that the tiling problem for NW-deterministic tile sets is undecidable. In addition, Lukkarila [31] used the 4-way deterministic tile set of Kari and Papazoglou in order to prove that the tiling problem is undecidable for 4-way deterministic tile sets as well. As the reader has probably guessed already, we prove that

Theorem 2. *The emptiness problem of extremely expansive* $2D$ *SFTs is undecidable. More precisely, the emptiness problem is undecidable for* $2D$ *SFTs such that the vertical direction is the only non-expansive direction.*

One should understand the previous statement in the following sense: even if one is given an SFT X (as a finite set of forbidden patterns) and is given the additional information that X is either empty or extremely-expansive (and in this case $\mathcal{N}(X) = \{\infty\}$), even then it is not possible to decide whether $X = \emptyset$. In other words, it is not possible to algorithmically separate the sets of forbidden patterns that define empty SFTs from those that define extremely expansive non-empty SFTs.

The third result can be considered a stronger version of the undecidability of the emptiness problem. We prove that there exist extremely expansive SFT whose configurations are computationally as complicated as possible.

In order to describe this result, we need to introduce some classical notions of computation theory. For the purposes of this introduction, a **computable function** will mean a function $f\colon \mathcal{A}^{\mathbb{N}} \to \mathcal{A}^{\mathbb{N}}$ such that there exists a Turing Machine that outputs $f(c)$ when originally its reading tape contains c (*i.e.*, it outputs $f(c)$ with oracle c). Using an effective enumeration of $\mathbb{Z}^2$, it is possible to talk about computable functions with domain or range $\mathcal{A}^{\mathbb{Z}^2}$, and in general $\mathcal{A}^{\mathbb{M}}$, where $\mathbb{M}$ is any effectively enumerable set.

We say that $d \in \mathcal{A}^{\mathbb{M}}$ is **reducible** to $c \in \mathcal{A}^{\mathbb{M}'}$ if there exists a computable function f such that $f(c) = d$. This means that c is computationally at least as complicated as d, since it is possible to obtain d using c and a computable function. A subset $Y \subseteq \mathcal{A}^{\mathbb{M}}$ is called **Medvedev** reducible to $\mathcal{A}^{\mathbb{M}'}$ if every point of Y is reducible to some point of X. Intuitively, we can compute any point of Y with the help of a suitable point of X and a computable function. The relation of Medvedev reducibility is a pre-order on subsets.

Two sets are called Medvedev equivalent if they are Medvedev reducible to each other. This is an equivalence relation, whose equivalence classes are called **Medvedev degrees**. There exists a partial order on the set of Medvedev degrees given by the natural lift of the Medvedev reducibility pre-order. Computable sets are the least element of this order and, in a certain sense, the higher a set is in this hierarchy, the more difficult it is to compute a point of this set relative to the sets that lie lower in the hierarchy. The survey [16] contains a thorough study of Medvedev degrees.

A set $X \subseteq \mathcal{A}^{\mathbb{M}}$ is called **effectively closed** if its complement is semi-decidable. Effectively closed sets form the so-called Π_0^1 sets and they play a very important role in computation theory. It is easy to see that SFTs are effectively closed, even though there exist many effectively closed sets (and even effectively closed subshifts) that are not SFTs. However, Simpson [38] proved that every effective Medvedev degree (*i.e.*, the Medvedev degree of an effectively closed set) contains a 2D SFT. Therefore, in some sense, not only is the emptiness problem undecidable for 2D SFTs, but their points can be as difficult to compute as possible. We improve this result to the extremely expansive case:

Theorem 3. *Every effective Medvedev degree contains an extremely expansive 2D SFT. In other words, for every effectively closed set $Z \subseteq \mathcal{A}^{\mathbb{M}}$, there exists an extremely expansive 2D SFT Y that is Medvedev equivalent to Z.*

In fact, we prove something stronger, giving a complete characterization of the so-called **Turing degrees** of Y relative to those of Z, but it is not necessary to go into these details here.

The next result is of a dynamical flavour and it does not concern extremely expansive SFTs, but sequences of SFTs with a common rational direction of expansiveness. It also uses the notion of simulation, which is of central importance in the proofs of the previous results and, in general, for the whole thesis, even though it wasn't mentioned until now.

We say that subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^2}$ **simulates** subshift $Y \subseteq \mathcal{B}^{\mathbb{Z}^2}$ with parameters (S, T) if there exists a $\mathcal{B}$-colouring of the $S \times T$ blocks of X with the following property: Every configuration of X can be partitioned in a unique way into $S \times T$ rectangles such that when we color these rectangles with the $\mathcal{B}$-colouring we obtain a configuration of Y. Inversely, every configuration of Y can be obtained in this way.

This is weaker than the notion of simulation that we actually use, but it follows from it, is enough to describe the result and is much easier to describe. It corresponds to the definitions in [8].

It was proved in [28] that for every computable sequence of SFTs, there exists an SFT that simulates all of them. This is a surprising and really strong result. We prove a version of it in the case where all the SFTs of the sequence have a common, rational expansive direction (which without loss of generality we assume to be the horizontal one):

Theorem 4. *Let $X_0, X_1, \ldots$ be a computable sequence of 2D SFTs such that $0 \in \mathcal{N}(X_i)$, for all $i \in \mathbb{N}$. Then, there exists a 2D SFT X such that X simulates X_i for all $i \in \mathcal{N}$ and $0 \in \mathcal{N}(X)$.*

We note that there cannot exist a 2D SFT with an expansive direction that simulates all 2D SFTs with the same expansive direction, because

this would imply the decidability of the emptiness problem for extremely expansive SFTs, according to an argument of Hochman [17].

The final result of the thesis answers a natural question which arises immediately after the construction of an extremely expansive SFT. As stated already, the unique non-expansive direction of the SFT that we construct is the vertical one. Which other directions can be the unique direction of non-expansiveness for 2D SFTs? Obviously, we can achieve any rational direction with the help of an affine transformation, but can we do more? More generally, what are the sets of directions that can be the set of non-expansive directions of a 2D SFT?

Hochman [19] proved that for general 2D subshifts (not necessarily of finite type, or even effective), any closed set of directions can be the set of non-expansive directions, while any direction can be the unique direction of non-expansiveness. Recall that Boyle and Lind proved that the sets of non-expansive directions must be closed, so it turns out that in the case of general subshifts this necessary topological condition is also sufficient.

In the case of SFTs, there is an additional necessary condition, namely that the set of non-expansive directions be **effectively closed**, which is equivalent to saying that its complement is the union of an increasing, computable sequence of open intervals. It turns out that this condition is necessary and sufficient for 2D SFTs:

Theorem 5. *A set of directions $\mathcal{N}$ is the set of non-expansive directions of a 2D SFT if and only if it is effectively closed. More precisely, a direction l is the unique direction of non-expansiveness of a 2D SFT if and only if it is computable.*

This answers Question 11.2 in Boyle's Open Problems for Symbolic Dynamics [5].

Using our methods, we could easily prove Theorems 1-3 for SFTs whose unique direction of non-expansiveness is l, where l is any computable direction. This is a stronger version of the results, which we do not prove for lack of space. In any case, once one has mastered our method, it is possible to prove various new results and variants of already proved ones. Since this method is as important (if not more) as some of our results, it is probably worth saying some words about its history, too.

It is the so-called **fixed-point tile** or **self-simulating** method for constructing 2D SFTs. It was firstly described by Kurdyumov [27] in order to give a counterexample to the Positive Rates conjecture, even though only a sketch of a proof was included in this paper. It was Gács [9] who elaborated Kurdyumov's idea into a full proof of the positive rates conjecture and formalized the notion of a hierarchy of simulating SFTs (he talks about 1D cellular automata, but this does not make a big difference). Later, he significantly improved his construction and the result in a notoriously lengthy

and difficult paper [10]. Gray's reader guide to that paper [12] and the description therein of self-simulation and the problems one encounters when trying to construct a self-simulating SFT are also a very useful exposition of the ideas of Gács and Kurdymov. It was not until the work of Durand, Romashchenko and Shen [8] that the method became accessible to a broader mathematical audience. They work in the framework of 2D SFTs, which allows for a more clear, geometrical description of the basic ideas.

Gács' construction did not have any direction of expansiveness, because it was a non-reversible cellular automaton. Nonetheless, it had the horizontal direction as a direction of "semi-expansiveness". On the other hand, the construction of Durand, Romashchenko and Shen did not have neither directions of expansiveness neither directions of "semi-expansiveness". A large part of this thesis consists in making their construction expansive in the horizontal direction. We need to introduce some tricks in order to do this, but once we achieve it, then self-simulation and a previous result of Hochman immediately give an extremely expansive aperiodic SFT. Something similar was also done in [14], but the construction of that paper was significantly easier because we dealt with non-reversible cellular automata, so that we only needed a direction of "semi-expansiveness". Our current construction can be seen as an improvement of the construction of that paper, and using it we can easily retrieve its main result, which was a characterization of the numbers that can appear as the topological entropy of a (not necessarily reversible) CA.

One thing that all the constructions have in common, including ours, is that they are complicated and rather difficult to explain (for the writer) and understand (for the reader). This is unavoidable, in some degree, and the author's personal opinion is that there does not exist a "perfect" way to write them. Either the exposition is very formal, covering all details and defining every little thing, which is the road that we have chosen, or the construction is informal, in which case it is not clear what exactly the constructed SFT is, over which alphabet it is defined etc., which is the choice made by Durand, Romashchenko and Shen. Taking the middle road, as was more or less done by Gács, does not help very much, either.

Our opinion is that the best thing is to be familiar with *all* the constructions and use them accordingly. On the one hand, the constructions of Durand, Romashchenko and Shen are convincing for someone already familiar with the technique and they allow to explain a new idea concisely and efficiently, as was recently done in [7], while on the other hand our more formal presentation can be used to acquire mastery with the technique by dealing with all the unexpected little problems that arise during the construction and to convince those people who want to understand all the details.

Let us now describe the structure of the thesis:

In Chapter 2, we give the basic definition that we will need throughout the paper. In Chapter 3, we define the precise notion of simulation that we will use and give some of its properties. We believe that some of the results of this chapter are of independent interest. In Chapter 4, we describe a pseudo-programming language that will be used to describe 2D SFTs in a concise way. In Chapter 5, we construct a family of SFT (which depend on the parameters S, T) with 0 as a direction of expansiveness which are, in some sense, universal: They can simulate every SFT with 0 as a direction of expansiveness, provided that its alphabet size is small compared to S, T and it can be computed fast compared to S, T. This family of SFTs is of great importance for all subsequent constructions. This is the part of the thesis where we modify the construction of Durand, Romashchenko and Shen so as to make it reversible. In Chapter 6, we prove Theorems 1 - 4. The constructions and the proofs all follow the same pattern, but we give as many details as possible for all of them for reasons of completeness. Finally, in Chapter 7, we prove Theorem 5. This proof is a modification of the proof of the result in [19]. We try to explain what are the differences between that construction and ours and why the changes that we make are necessary.

Finally, let us mention that all of the aforementioned results have been obtain in collaboration with Pierre Guillon during various visits by him in Turku as well as of the author in Marseille. Currently, a series of joint papers is under construction that will contain even more applications of our method. Theorem 1 has also appeared in [41], even though because of lack of space, most of the details of the construction do not appear in that paper.

Chapter 2

Preliminaries

2.1 Basic definitions

We will denote by $\mathbb{Z}$, $\mathbb{N}$, $\mathbb{N}_1$, $\mathbb{Q}$ and $\mathbb{R}$ the sets of integers, non-negative integers, positive integers, rational and real numbers, respectively, by $[\![i, j[\![$ and $[\![i, j]\!]$ the integer intervals $\{i, \ldots, j-1\}$ and $\{i, \ldots, j\}$, respectively, while $[\varepsilon, \delta]$ will denote an interval of real numbers. If $f, g \colon \mathbb{N} \to \mathbb{N}$, then we will use the classical $f \in O(g)$ notation to denote that $f(n) \leq cg(n)$, for some constant c and all $n \in \mathbb{N}$.

If $f \colon X \nrightarrow Y$ is a partial function, then its **domain** $\mathcal{D}(f) \subseteq X$ is the set of elements of X whose image through f is defined. Two partial functions are equal when they have the same domain and they agree on their common domain. If $f \colon X \nrightarrow Y$ and $g \colon Y \nrightarrow Z$ are partial functions, then $g \circ f \colon X \nrightarrow Z$ is the partial function defined in the usual way (*i.e.*, $g(f(w))$ does not exist if either $w \notin \mathcal{D}(f)$ or $f(w) \notin \mathcal{D}(g)$). A **partial permutation** is a bijection over its domain onto its range, *i.e.*, an injective partial map. In the following, when defining a partial function, it will be implicit that any non-treated argument has an undefined image, and that saying that two partial functions are equal means in particular that their domains are the same. If $Z \subset X$ and $f : X \nrightarrow Y$, we may abusively consider $f_{|Z}$ as a partial map from X to Y whose domain is $Z \cap \mathcal{D}(f)$.

For $m \geq n$, $\mathbf{T} := (T_i)_{0 \leq i < m}$ and $\mathbf{t} := (t_i)_{0 \leq i < n}$ such that for all i, $0 \leq t_i < T_i$, we note $\overline{\mathbf{t}}^{\mathbf{T}} := \sum_{0 \leq i < n} t_i \prod_{0 \leq j \leq i} T_j$ the numeric value represented by the adic representation $\mathbf{t}$ in base $\mathbf{T}$. In general, T_i and t_i can belong in $\mathbb{R}$, not necessarily in $\mathbb{N}$. By convention, if $\mathbf{t}$ has length 0, then $\overline{\mathbf{t}}^{\mathbf{T}} := 0$. Similarly, for a sequence $\mathfrak{T} := (T_i)_{i \in \mathbb{N}}$, we note $\overline{\mathbf{t}}^{\mathfrak{T}} := \overline{\mathbf{t}}^{T_{[\![0,n[\![}}$. For a sequence $\mathfrak{t} := (t_i)_{i \in \mathbb{N}}$, we note $\overline{\mathfrak{t}}^{\mathfrak{T}} := \lim \overline{t_{[\![0,n[\![}}^{T_{[\![0,n[\![}}$, when this limit exists.

An **alphabet** is any finite set, whose elements are often called **symbols**. If $\mathcal{A}$ is an alphabet, $\mathcal{A}^* := \bigcup_{n \in \mathbb{N}} \mathcal{A}^n$ denotes the set of finite **words** over $\mathcal{A}$, and $\mathcal{A}^{**} := \bigcup_{m \in \mathbb{N}} (\mathcal{A}^*)^m$ the set of finite tuples of words. (Notice that

the notation $\mathcal{A}^{**}$ is a little ambiguous as it could also stand for the set $\bigcup_{m\in\mathbb{N}}(\mathcal{A}^m)^*$. Obviously, the two interpretations are isomorphic, but they are different objects.) The **empty** word is denoted by $\epsilon \in \mathcal{A}^*$.

If $w \in \mathcal{A}^n$, we write $w = w_0 \cdots w_{n-1}$, and call $|w| := n$ the **length** of w. If $\mathbf{u} \in (\mathcal{A}^*)^m$, we write $\mathbf{u} = (u_0, \ldots, u_{m-1})$, and $|\mathbf{u}| := (|u_0|, \ldots, |u_{m-1}|)$. For every $i \in \mathbb{N}$, we define the projection π_i as a partial function $\pi_i \colon \mathcal{A}^{**} \rightharpoonup \mathcal{A}^*$: $\pi_i(\mathbf{u}) = u_i$ if $\mathbf{u} \in (\mathcal{A}^*)^m$ with $m \geq i$ (and $\pi_i(\mathbf{u})$ is undefined otherwise). A **field** is a projection π_i together with a label `Field`, written in type-writer form. The notion of fields is simply a convenient way of talking about tuples of words. The names of the fields will be chosen so as to reflect the role that the field plays in the construction.

We note $\mathbb{N}^* := \bigcup_{m\in\mathbb{N}} \mathbb{N}^m$ the set of integer tuples of any dimension, where m is the **dimension** of the tuple $\mathbf{k} := (k_0, \ldots, k_{m-1}) \in \mathbb{N}^m$. Let $\mathcal{A}^{\mathbf{k}} := \mathcal{A}^{k_0} \times \ldots \times \mathcal{A}^{k_{m-1}}$; any subalphabet of $\mathcal{A}^{\mathbf{k}}$ is said to have **constant lengths**.

We will mainly use the special alphabets $\mathbb{F}_n := \{0, \ldots, n-1\}$, for $n \in \{2, 3, 4, 5\}$. Of course, instead of $\mathbb{F}_5$ we could use any alphabet with n letters. However, since some letters will have a fixed role throughout the thesis, it is better to fix the notation and get used to these roles.

The non-negative integers can be easily embedded into $\mathbb{F}_2^*$ thanks to the injection $n \mapsto \overline{n}$ which gives the shortest binary representation of $n \geq 1$. $\|n\| := |\overline{n}| = \lceil \log_2 n \rceil + 1$ is the **length of** n. By definition, $\overline{0} := \epsilon$ and $\|\epsilon\| := 0$. Inversely, if $u \in \mathbb{F}_2^*$, then $\underline{u}$ is the number represented by u in the binary representation system: for all $u \in \mathbb{F}_2^*, \underline{u}$ is the suffix of u that is obtained after removing the initial 0s. (The "lower bar" is applied before the "top bar".)

We will also need to embed some finite sets in $\mathbb{F}_2^*$. For instance, we will say that $\{-1, +1\}$ is $\mathbb{F}_2$ by identifying -1 with 0 and $+1$ with 1. Finite alphabets of bigger cardinality can be embedded into $\mathbb{F}_2^k$, for some suitable k.

Now, in the perspective of computing functions with many arguments, we are going to use symbol 2 to encode tuples into words. If $\mathbf{u} \in (\mathbb{F}_5^*)^m$ for some $m \in \mathbb{N}$, then $\chi(\mathbf{u})$ is defined as the concatenation

$$\chi(u_0)\,\chi(u_1)\,\ldots\ldots\,\chi(u_{m-1}) \in \mathbb{F}_3^*,$$

where $\chi(v) := 2\overline{v}^{\mathbb{F}_5}$, and $v \mapsto \overline{v}^{\mathbb{F}_5}$ is some monoid injection (*i.e.*, code) from $\mathbb{F}_5^*$ to $\mathbb{F}_2^*$. In this paper, we will use the code defined by $\overline{0}^{\mathbb{F}_5} := 000$, $\overline{1}^{\mathbb{F}_5} := 001$, $\overline{2}^{\mathbb{F}_5} := 010$, $\overline{3}^{\mathbb{F}_5} := 011$, $\overline{4}^{\mathbb{F}_5} := 100$. Note that the structure of the encoding of word tuples depends only on $|\mathbf{u}|$. We can also define $\chi(\mathfrak{u}) := \chi(u_0)\,\chi(u_1)\ldots \in \mathbb{F}_3^{\mathbb{N}}$ for $\mathfrak{u} \in \mathbb{F}_5^{\mathbb{N}}$.

Let us now prove a basic fact about $\chi(\cdot)$. Namely, for every $\mathbf{k} \in \mathbb{N}^*$, there exists an easily computable function that gives the positions of the 2s

in encodings of $\mathbb{F}_5^{\mathbf{k}}$ and the positions of the encodings of the components of a letter.

Fact 6. *Let $M \in \mathbb{N}$ and $\mathbf{k} \in \mathbb{N}^M$. For all $0 \leq i < M$, let us define $l_{\mathbf{k},i} := 3\sum_{j=0}^{i-1} k_j + i$. Then, for all $\mathbf{u} \in \mathbb{F}_5^{\mathbf{k}}$:*

1. $\|\chi(\mathbf{u})\| = l_{\mathbf{k},M}$,
2. $\chi(\mathbf{u})_{[\![l_{\mathbf{k},i}, l_{\mathbf{k},i+1}[\![} = 2\overline{\pi_i(\mathbf{u})}^{\mathbb{F}_5}$

These statements correspond to what Durand, Romashchenko and Shen refer to as "the TM know the place where such and such information is held in the encoding".

Symbol 3 will be used in Subsection 2.2.1 to encode the start and the end of the tape of a Turing machine.

Symbol 4 will be used in order to construct alphabets with constant lengths. In the computation, we indeed want words of various lengths to be able to represent the same objects. For this, we define $\langle u \rangle_l := 4^{l-|u|}u$, for every $l \in \mathbb{N}$ and $u \in \mathbb{F}_4^*$ with $|u| \leq l$ ($\langle u \rangle_l$ is undefined otherwise). For instance, $\langle \overline{n} \rangle_{\|n\|} = \overline{n}$ for any integer $n \in \mathbb{N}$, and the encoding $\langle \epsilon \rangle_l = 4^l$ of the empty word is a sequence of 4s. It is clear that the partial function

$$\begin{array}{rcl} \mathbb{N} \times \mathbb{F}_4^* & \twoheadrightarrow & 4^*\mathbb{F}_4^* \\ (l, u) & \mapsto & \langle u \rangle_l \end{array}$$

is injective (over its domain) and surjective; let us write $\rangle w \langle \in \mathbb{F}_4^*$ for the longest suffix in $\mathbb{F}_4^*$ of a word $w \in 4^*\mathbb{F}_4^*$, in such a way that $\rangle \langle u \rangle_l \langle = u$ for any $l \geq |u|$ and $u \in \mathbb{F}_4^*$. These two maps can be adapted to vectors in the obvious way: $\langle \mathbf{u} \rangle_{\mathbf{k}} := (\langle u_0 \rangle_{k_0}, \ldots, \langle u_{m-1} \rangle_{k_{m-1}})$ for any $\mathbf{k} \in \mathbb{N}^m$, $m \in \mathbb{N}$ and $\mathbf{u} := (u_0, \ldots, u_{m-1}) \in \mathbb{F}_4^{*m}$. Note that this is defined if and only if $\mathbf{k} \geq |\mathbf{u}|$. Similarly, $\rangle \mathbf{w} \langle := (\rangle w_0 \langle, \ldots, \rangle w_{m-1} \langle)$ for any $\mathbf{w} := (w_0, \ldots, w_{m-1}) \in (4^*\mathbb{F}_4^*)^m$.

If $\alpha : \mathbb{F}_4^{**} \twoheadrightarrow \mathbb{F}_4^{**}$ is a partial permutation that preserves the number of fields (*i.e.*, $\alpha(\mathbb{F}_4^*)^l \subseteq (\mathbb{F}_4^*)^l$ for all $l \in \mathbb{N}$), we can transform it into an equivalent permutation that also preserves the lengths:

$$\begin{array}{rccl} \langle \alpha \rangle : & (4^*\mathbb{F}_4^*)^* & \twoheadrightarrow & (4^*\mathbb{F}_4^*)^* \\ & \mathbf{w} & \mapsto & \langle \alpha(\rangle \mathbf{w} \langle) \rangle_{|\mathbf{w}|} \ . \end{array}$$

Remark 7.

- *For any $\mathbf{k} \in \mathbb{N}^*$, $\langle \alpha \rangle$ is also a partial permutation.*
- *The restriction of α to any subalphabet is implemented by that of $\langle \alpha \rangle$ to large enough words:*

$$\forall \mathbf{u} \in \mathbb{F}_4^{**}, \forall \mathbf{k} \geq \max\{|\mathbf{u}|, |\alpha(\mathbf{u})|\}, \langle \alpha \rangle (\langle \mathbf{u} \rangle_{\mathbf{k}}) = \langle \alpha(\mathbf{u}) \rangle_{\mathbf{k}} \ .$$

Proof. For the first part, assume that $\langle\alpha\rangle(\mathbf{w}) = \langle\alpha\rangle(\mathbf{w}')$. This implies that $|\mathbf{w}| = |\mathbf{w}'|$. In addition, $\alpha(\rangle\mathbf{w}\langle) = \rangle\langle\alpha\rangle(\mathbf{w})\langle = \rangle\langle\alpha\rangle(\mathbf{w}')\langle = \alpha(\rangle\mathbf{w}'\langle)$. Since α is a partial permutation, this implies that $\rangle\mathbf{w}\langle = \rangle\mathbf{w}'\langle$. Therefore, $\mathbf{w} = \langle\rangle\mathbf{w}\langle\rangle_{|\mathbf{w}|} = \langle\rangle\mathbf{w}'\langle\rangle_{|\mathbf{w}'|} = \mathbf{w}'$.

For the second part, let $\mathbf{u} \in \mathbb{F}_4^{**}$ and $\mathbf{k} \geq \max\{|\mathbf{u}|, |\alpha(\mathbf{u})|\}$. Then, $\langle\mathbf{u}\rangle_{\mathbf{k}}$ and $\langle\alpha(\mathbf{u})\rangle_{\mathbf{k}}$ exist and $|\langle\mathbf{u}\rangle_{\mathbf{k}}| = |\langle\alpha(\mathbf{u}\rangle_{\mathbf{k}})| = |\mathbf{k}|$. Therefore, $\langle\alpha\rangle(\langle\mathbf{u}\rangle_{\mathbf{k}}) = \langle\alpha(\rangle\langle\mathbf{u}\rangle_{\mathbf{k}}\langle)\rangle_{\mathbf{k}} = \langle\alpha(\mathbf{u})\rangle_{\mathbf{k}}$. □

In the rest of the paper, we will often implicitly use Remark 7 both to construct partial permutations that preserve the lengths of the fields, as well as to state and prove things about them. It allows us to describe the behaviour of a partial permutation α, and then translate this result into the behaviour of $\langle\alpha\rangle$, provided that the lengths of the fields are sufficiently large, thus omitting the (confusing) $\rangle\cdot\langle$ and $\langle\cdot\rangle$ symbols.

Let $i_1, \ldots, i_l$ be a set of fields, and $w \in \mathbb{F}_5^*$. Then,

$$\mathcal{S}^w_{i_1,\ldots,i_l} := \{\, \mathbf{u} \in \mathbb{F}_5^{**} \,|\, \rangle\pi_{i_k}(u)\langle = w, \text{ for } k = 1, \ldots, l\}$$

is the set of all symbols that have fields $i_1, \ldots, i_l$ equal to w (up to the application of $\rangle\cdot\langle$). If $n \in \mathbb{N}$, let

$$\mathcal{S}^n_{i_1,\ldots,i_l} := \left\{ \mathbf{u} \in \mathbb{F}_5^{**} \,\middle|\, \underline{\rangle\pi_{i_k}(u)\langle} = n, \text{ for } k = 1, \ldots, l \right\}$$

be the set of all symbols who have the values n (in binary form) in the fields $i_1, \ldots, i_l$.

2.2 Computation

2.2.1 Turing machines

The reader is assumed to be familiar with classical concepts in computability theory. We just fix some terminology and give a variant of a definition of Turing machines, imposing some additional technical restrictions which, however, do not restrict the computational power.

A **Turing machine** (TM) is a partial ("global") map $\mathcal{M}$ from $\mathbb{F}_4^{\mathbb{Z}} \times Q \times \mathbb{Z}$ into itself, where $Q \subset \mathbb{F}_2^*$ is a finite set of **states** containing the **initial** state 0 and the **accepting** state ϵ, and depending on a partial **transition map** $\delta_{\mathcal{M}} : \mathbb{F}_4 \times Q \setminus \{\epsilon\} \dashrightarrow \mathbb{F}_4 \times Q \times \{-1, +1\}$ such that:

$$\mathcal{M}(z, q, j) = \begin{cases} (z, q, j) & \text{if } q = \epsilon \\ (z', q', j') & \text{otherwise, where } (z'_j, q', j' - j) = \delta_{\mathcal{M}}(z_j, q) \\ & \text{and } z'_i = z_i, \forall i \neq j \ , \end{cases}$$

for any $(z, q, j) \in \mathbb{F}_4^{\mathbb{Z}} \times Q \times \mathbb{Z}$, which will sometimes be called a **machine configuration**, the first component being the **tape content**, the second the (head) **internal state**, the third the **head position**.

The model of TM that we use satisfies the following assumptions, which, as can be easily seen, do not restrict the computational power of TM.

- There is only one tape, from which the TM reads the input and on which it writes the output.
- The internal states are words of $\mathbb{F}_2^*$ (this is just a semantic restriction).
- All machines have the same initial and accepting states 0 and ϵ, respectively.
- The global map is still defined after having accepted, and is then equal to the identity.
- There is no precise rejecting state (instead, we use undefined transitions over non-accepting states).
- In every accepting transition, the head (which disappears) moves to the right. This is a technical assumption which simplifies the construction of an IPPA that simulates $\mathcal{M}$ in Section 5.2.

If $\mathcal{M}^t({}^\infty 3.\chi(\mathbf{u})\, 3^\infty, 0, 0) = ({}^\infty 3.\chi(\mathbf{u}')\, 3^\infty, \epsilon, j)$, for some $t \in \mathbb{N}, j \in \mathbb{Z}$, then we say that $\mathcal{M}$ **halts** over (or **accepts**) **input** $\mathbf{u} \in \mathbb{F}_5^{**}$, and **outputs** $\mathbf{u}' \in \mathbb{F}_5^{**}$, and we define $f_\mathcal{M}(\mathbf{u}) := \mathbf{u}'$ and $t_\mathcal{M}(\mathbf{u})$ as the minimal t for which this holds (if this never holds, or if ${}^\infty 3.\chi(\mathbf{u})\, 3^\infty$ is rejected, then $t_\mathcal{M}(\mathbf{u})$ is undefined).

Notice that $f_\mathcal{M}(\mathbf{u})$ is well-defined, since when the accepting state ϵ appears, the machine configuration is no more modified.

We say that $\mathcal{M}$ **computes** the partial map $f_\mathcal{M} : \mathbb{F}_5^{**} \rightharpoonup \mathbb{F}_5^{**}$, with time **complexity**

$$\begin{array}{rccl} t_\mathcal{M} : & \mathbb{N} & \to & \mathbb{N} \\ & n & \mapsto & \max_{|\chi(\mathbf{u})|=n} t_\mathcal{M}(\mathbf{u}) \ , \end{array}$$

where, by definition, the max is taken only over *accepted* inputs. $t_\mathcal{M}$ is well-defined since there are only finitely many accepted inputs of each length.

2.2.2 Computability

A partial function $f : \mathbb{F}_5^{**} \rightharpoonup \mathbb{F}_5^{**}$ is called **computable** if there exists a TM $\mathcal{M}$ such that $f = f_\mathcal{M}$. Recall that integers (and finite sets) can be identified to words, hence allowing us to talk about computable maps between Cartesian products involving $\mathbb{N}$ and finite sets. We also say that a

set $X \subseteq \mathbb{F}_4^{**}$ is **computable** if its characteristic function $\iota_X \colon \mathbb{F}_4^{**} \to \mathbb{F}_2$ is computable, and that it is **computably enumerable** if it is the domain of a computable function. We will say that a partial function $f : X \nrightarrow \mathbb{F}_5^{**}$, with $X \subset \mathbb{F}_5^{**}$ is **computable** if both X and the extension of f to $\mathbb{F}_5^{**}$ (by not defining images outside of X) are computable.

A partial function $\Phi : \mathbb{F}_2^{\mathbb{N}} \nrightarrow \mathbb{F}_2^{\mathbb{N}}$ is called **computable** if there exists a TM $\mathcal{M}$ such that $x \in \mathcal{D}(\Phi)$ if and only if for all $n \in \mathbb{N}$, there exists $m \in \mathbb{N}$ such that $f_{\mathcal{M}}(x_{[\![0,m[\![}, n)$ is defined, in which case it is equal to $\Phi(x)_n$. Finally, by parametrizing $\mathbb{Z}$ with $\mathbb{N}$, we can talk about computable functions $\Phi : \mathbb{F}_2^{\mathbb{Z}} \nrightarrow \mathbb{F}_2^{\mathbb{Z}}$. An equivalent definition is that $\Phi : \mathbb{F}_2^{\mathbb{Z}} \nrightarrow \mathbb{F}_2^{\mathbb{Z}}$ is computable if there exists a TM $\mathcal{M}$ such that $x \in \mathcal{D}(\Phi)$ if and only if for all $n \in \mathbb{N}$, there exists $m \in \mathbb{N}'$ such that $f_{\mathcal{M}}(x_{[\![-m,m[\![}, n)$ is defined, in which case it is equal to $\Phi(x)_n$.

Since $\mathbb{R}$ can be identified with $\mathbb{F}_2^{\mathbb{N}}$, we can also talk about computable functions of real numbers. A partial function $\Psi : \mathbb{R} \nrightarrow \mathbb{R}$ is **computable** if there exists a computable function $f \colon \mathbb{R} \times \mathbb{N} \to \mathbb{Q}$ with the following property: $|\Psi(x) - f(x,n)| < 2^{-n}$. This is the classical definition of computability for real functions and it says that we can compute better and better approximations of x.

If $\mathcal{M}$ is a TM, let

$$\mathfrak{Z}_{\mathcal{M}} := \left\{ z \in \mathbb{F}_2^{\mathbb{N}} \,\middle|\, \forall t \in \mathbb{N}, \mathcal{M}^t(3^\infty.z, 0, 0) \text{ exists and is not in } \mathbb{F}_4^{\mathbb{Z}} \times \{\epsilon\} \times \mathbb{Z} \right\}$$

be the set of one-sided binary sequences over which $\mathcal{M}$ runs for an infinite amount of time. We say that a subset $X \subset \mathbb{F}_2^{\mathbb{N}}$ is **effectively closed** (or Π_1^0) if $\chi(X) = \mathfrak{Z}_{\mathcal{M}}$ for some TM $\mathcal{M}$, or equivalently if the set of words that do not prefix any sequence in it is computably enumerable. This can be extended to sets of sequences that can be encoded with words, in particular over finite alphabets: a subset $X \subset \prod_{t \in \mathbb{N}} \mathcal{A}_t$, where $\mathcal{A}_t$ is a finite subalphabet of $\mathbb{F}_5^*$, is **effectively closed** if $\chi(X) = \mathfrak{Z}_{\mathcal{M}}$ for some program $\mathcal{M}$ (we encode every finite alphabet with $\mathbb{F}_2^k$, for some suitable k which depends on $t \in \mathbb{N}$).

$\mathcal{M}$ is called **polynomial** if $t_{\mathcal{M}} \in O(P)$, for some polynomial P. A partial function f is called **polynomially computable** if $f_{\mathcal{M}} = f$ for some polynomial TM $\mathcal{M}$. It is easy to see that the class of (polynomially) computable functions with this version of TM corresponds to the classical one. Analogously, X is a polynomially computable set if its characteristic function ι_X is polynomially computable. We say that a function (or sequence) f is **polynomially checkable** if it can be computed in time $O(P(\log f))$, for some polynomial P. The terminology comes from the fact that even though f might not be polynomially computable, its graph (*i.e.*, the set of pairs element-image) is a polynomially computable set. For example $f(n) = 2^{2^n}$ is a polynomially checkable sequence even though it is not polynomially computable.

Instead of a universal TM, we use the following essentially equivalent:

Fact 8. *There exists an injection that associates to each TM $\mathcal{M}$ a* ***program*** *$p_{\mathcal{M}} \in \mathbb{F}_4^*$ such that if we denote by Q_p the state set of the TM corresponding to program p, then*

- *The language $\{p_{\mathcal{M}} | \, \mathcal{M}$ is a TM$\} \subseteq \mathbb{F}_4^*$ is polynomially decidable.*
- *The characteristic function $(p, q) \mapsto \iota_{Q_p}(q)$ that checks whether $q \in Q_p$ is polynomially computable.*
- *The "universal" transition rule*

$$\begin{array}{rccc} \delta_{\mathcal{U}} : & \mathbb{F}_4 \times \mathbb{F}_4^* \times \mathbb{F}_4^* & \nrightarrow & \mathbb{F}_4 \times \mathbb{F}_4^* \times \{-1, +1\} \\ & (a, q, p_{\mathcal{M}}) & \mapsto & \delta_{\mathcal{M}}(a, q) \end{array}$$

  is polynomially computable.

- *In addition, $|Q_p| \leq |p|$. (We can assume that p contains a list of the states of Q_p.)*

We will use the following notations: If p is the program of a TM that computes a reversible function f, then p^{-1} will denote the program of the inverse function f^{-1} (it will always be computable in our constructions). Also, t_p and $\mathfrak{Z}_p$ will be used to denote $t_{\mathcal{M}_p}$ and $\mathfrak{Z}_{\mathcal{M}_p}$, where $\mathcal{M}_p$ is the TM that corresponds to the program p.

The first examples of polynomially computable functions, which will be most useful in the sequel, are the encodings presented in Subsection 2.1. Clearly, $\langle \cdot \rangle_.$ and its (right) inverse $\rangle \cdot \langle$ are polynomially computable. Moreover, the projections $\pi_i : \mathbb{F}_5^{**} \to \mathbb{F}_5^*$, for $i \in \mathbb{N}$, are polynomially computable and so are the functions $(\mathbf{k}, i) \to l_{\mathbf{k},i}$ (as defined in Fact 6) and $\chi(\cdot)$.

2.2.3 Degrees

In the following, $\mathbb{M}$ and $\mathbb{M}'$ can stand for either $\mathbb{N}$ or $\mathbb{Z}$.

Two sets $X, Y \in \mathbb{F}_2^{\mathbb{M}}$ are **computably homeomorphic** if there exists a computable bijection between them.

We say that $d \in \mathbb{F}_2^{\mathbb{M}'}$ is **Turing-reducible** to $c \in \mathbb{F}_2^{\mathbb{M}}$ if $d = \Phi(c)$, for some computable function Φ. This yields a preorder over configurations, whose equivalence classes are called **Turing degrees**. If d is Turing-reducible to c, then in a computational sense, c is more complicated than d. A **cone** over degree d is the set of Turing degrees that are higher than d.

Moreover, we say that subset $Y \subset \mathbb{F}_2^{\mathbb{M}'}$ is **Medvedev-reducible** to subset $X \subset \mathbb{F}_2^{\mathbb{M}}$ if there is a computable partial function $\Phi : \mathbb{F}_2^{\mathbb{M}} \nrightarrow \mathbb{F}_2^{\mathbb{M}'}$ such that $\mathcal{D}(\Phi) \supseteq X$ and $\Phi(X) \subseteq Y$. This also yields a pre-order over sets, whose equivalence classes are called **Medvedev degrees**. Finally, we say

that subset $Y \subset \mathbb{F}_2^{\mathbb{M}'}$ is **Mučnik-reducible** to subset $X \subset \mathbb{F}_2^{\mathbb{M}}$ if every point of X is Turing-reducible to some point of Y (but not in a uniform way, as in Medvedev-reducibility). This again yields a pre-order over sets, whose equivalence classes are called **Mučnik degrees**.

Medvedev and Mučnik degrees of a set are an attempt to formalize the notion of how computationally difficult it is to compute a point of the set. Of course, computable homeomorphism implies having the same Turing degrees, which implies Medvedev-equivalence, which in turns implies Mučnik-equivalence.

We do not get too much into details, but the notion holds in the large setting of effective topological spaces (see for instance [11]).

2.3 Symbolic dynamics

$\mathcal{A}^{\mathbb{Z}^d}$ is the set of d-dimensional **configurations**, endowed with the product of the discrete topology, and with the **shift** dynamical system σ, defined as the action of $\mathbb{Z}^d$ by $(\sigma^{\mathbf{i}})_{\mathbf{i}\in\mathbb{Z}^d}$, where $\sigma^{\mathbf{i}}(x)_{\mathbf{k}} := x_{\mathbf{i}+\mathbf{k}}$ for any configuration $x \in \mathcal{A}^{\mathbb{Z}^d}$ and any $\mathbf{i}, \mathbf{k} \in \mathbb{Z}^d$.

A **pattern** over a (usually finite) **support** $D \subset \mathbb{Z}^d$ is a map $p \in \mathcal{A}^D$.

Two patterns $u_1 \colon D_1 \to \mathcal{A}$ and $u_2 \colon D_2 \to \mathcal{A}$ are called **disjoint** if D_1 and D_2 are disjoint shapes of $\mathbb{Z}^d$. If u_1, u_2 are disjoint, let $u_1 \vee u_2$ be the pattern over shape $D_1 \sqcup D_2$ defined by $(u_1 \vee u_2)(\mathbf{i}) = u_j(\mathbf{i})$, if $\mathbf{i} \in D_j$, $j = 1, 2$. Inductively, we can define $\bigvee_{1\leq i\leq k} u_i$, when $u_1, \dots, u_k$ are mutually disjoint pattens.

Let $E, D \subset \mathbb{Z}^2$ be two shapes, and $u \in \mathcal{A}^D$ be a 2D pattern. We denote u_E the restriction of u to $D \cap E$ (this is a pattern with support $D \cap E$).

If $I \subseteq \mathbb{Z}$ and $(c_i)_{i\in I}$ is a family of configurations of $\mathcal{A}^{\mathbb{Z}}$, $|(c_i)_{i\in I}$ denotes the (possibly infinite) pattern $u \colon \mathbb{Z} \times I \to \mathcal{A}$ such that $u_{\mathbb{Z}\times i} = c_i$, for all $i \in I$. If $I = [\![0, n[\![$, then $|(c_0, \dots, c_{n-1})$ is the horizontal strip of width n obtained by putting $c_0, \dots, c_{n-1}$ on top of each other (in this order). If $I = \mathbb{Z}$, then we obtain a configuration in $\mathcal{A}^{\mathbb{Z}^2}$.

Let $x \in \mathcal{A}^{\mathbb{Z}^d}$ and $\mathbf{S} := (S_0, \dots, S_{d-1}) \in \mathbb{N}_1^d$. The **S-bulking** (or higher-power representation) of x is the configuration $x_{[\mathbf{S}]} \in (\mathcal{A}^{S_0\times\dots S_{d-1}})^{\mathbb{Z}^d}$ such that for any $\mathbf{i} = (i_0, \dots, i_{d-1}) \in \mathbb{Z}^d$,

$$x_{[\mathbf{S}]_{\mathbf{i}}} := x_{[\![i_0 S_0,(i_0+1)S_0[\![\times\dots\times[\![i_{d-1}S_{d-1},(i_{d-1}+1)S_{d-1}[\![}.$$

A (d-dimensional) **subshift** is a closed set $X \subset \mathcal{A}^{\mathbb{Z}^d}$ such that $\sigma^{\mathbf{i}}(X) = X$ for all $\mathbf{i} \in \mathbb{Z}^d$. Equivalently, X is a subshift if and only if there exists a family of patterns $\mathcal{F} \subset \bigcup_{D\subset_{\text{finite}}\mathbb{Z}^d} \mathcal{A}^D$ such that

$$X = \left\{ x \in \mathcal{A}^{\mathbb{Z}^d} \,\middle|\, \forall \mathbf{i} \in \mathbb{Z}^d, \forall D \subset_{\text{finite}} \mathbb{Z}^d, \sigma^{\mathbf{i}}(x)_{|D} \notin \mathcal{F} \right\}.$$

If $\mathcal{F}$ can be chosen finite, we say that X is a **subshift of finite type** (SFT).

If $\mathcal{F}$ can be chosen computably enumerable, then X is called an **effective subshift**.

A map Φ from subshift X to subshift Y is a **morphism** if $\Phi\sigma = \sigma\Phi$. If it is surjective, then it is a **factor map**, and Y is a **factor** of X (this defines a preorder); if it is bijective, then it is a **conjugacy**, and X and Y are **conjugate** (this defines an equivalence relation). A subshift $Y \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is called **sofic** if it is a factor of some SFT, which is then called a **cover** for Y.

A configuration $x \in \mathcal{A}^{\mathbb{Z}^d}$ is called **periodic** with **period** $\mathbf{j} \neq \mathbf{0} \in \mathbb{Z}^d$ if $\sigma^{\mathbf{j}}(x) = x$. A subshift X is called **aperiodic** if it does not contain any periodic configurations.

Abusing notation, we use the notations $\mathcal{S}^w_{i_1,\dots,i_l}$ and $\mathcal{S}^n_{i_1,\dots,i_l}$ (where $w \in \mathbb{F}_5^{**}$ and $n \in \mathbb{N}$) also for configurations. For example, if $c \in (\mathbb{F}_5^{**})^{\mathbb{Z}}$, we will say that $c \in \mathcal{S}^w_{i_1,\dots,i_l}$ if $c_i \in \mathcal{S}^w_{i_1,\dots,i_l}$ for all $i \in \mathbb{Z}$. Finally, for $N \in \mathbb{N}$ and $n \in [\![0, N[\![$, let

$$\mathcal{P}^{n,N}_{i_1,\dots,i_l} := \{c \in (\mathbb{F}_5^{**})^{\mathbb{Z}} : \underline{\rangle \pi_{i_k}(c)_j \langle} = j + n \mod N, \text{ for all } j \in \mathbb{Z}, 1 \leq k \leq l\}$$

be the set of all configurations such that

$$\pi_{i_k}(c) = .^{\infty}(n \dots (N-1)01 \dots (n-1))^{\infty}, \text{ for } 1 \leq k \leq l.$$

2.4 Cellular automata

A (1D) **partial cellular automaton** (PCA) is a partial ("global") continuous function $F : \mathcal{A}^{\mathbb{Z}} \dashrightarrow \mathcal{A}^{\mathbb{Z}}$ whose domain is an SFT, and such that $F\sigma = \sigma F$. Equivalently by some extension of the so-called Curtis-Lyndon-Hedlund theorem, there exist a **neighbourhood** $V \subset_{\text{finite}} \mathbb{Z}$ and a partial **local rule** $f : \mathcal{A}^V \dashrightarrow \mathcal{A}$ such that for all $z \in \mathcal{A}^{\mathbb{Z}}$, $F(z)$ is defined if and only if $f(z_{|i+V})$ is defined for all $i \in \mathbb{Z}$, in which case $F(z)_i := f(z_{|i+V})$. If $V \subseteq [\![-r, r]\!]$, then r is called a **radius** of the PCA. The radius of a PCA is not uniquely determined.

A PCA is called **reversible** (RPCA) if it is injective. In this case, it is known that there exists another RPCA, denoted by F^{-1}, such that FF^{-1} and $F^{-1}F$ are restrictions of the identity, and $\mathcal{D}(F^{-1}) = F(\mathcal{A}^{\mathbb{Z}})$ (the argument for this is similar to the one in [15]). In particular, there exist so-called inverse radius and inverse local rule. If r is both a radius and an inverse radius for an RPCA F, we call it a **bi-radius** for F. In the rest of the paper, we only consider RPCA with bi-radius 1. This is not a significant restriction, since these PCA and RPCA exhibit the whole range of computational and dynamical properties of general PCA and RPCA.

For $t \in \mathbb{N}$, the t^{th}**−order range** of F is the (sofic) subshift $\Omega^t_F := F^t(\mathcal{A}^{\mathbb{Z}}) \cap F^{-t}(\mathcal{A}^{\mathbb{Z}})$ and its **limit set** is the (effective) subshift $\Omega_F := \Omega^{\infty}_F :=$

$\bigcap_{t\in\mathbb{Z}} \Omega_F^t$, containing all the configurations that are *not* **ultimately rejected** (either in the past or the future). There is a canonical way to associate a 2D SFT $\mathcal{O}_F$ to an RPCA F: it consists of the infinite space-time diagrams of the configurations that are not ultimately rejected. Formally, $\mathcal{O}_F := \{\mathcal{O}_F(x) | \, x \in \Omega_F\}$, where $\mathcal{O}_F(x) := |(F^t(x))_{t\in\mathbb{Z}} \in \mathcal{A}^{\mathbb{Z}^2}$ for any $x \in \Omega_F$. One can see that $\mathcal{O}_F$ is conjugate to the $\mathbb{Z}^2$-action of (F, σ) over Ω_F. Note nevertheless that the same SFT may correspond to distinct RPCA (if they have different transient phases).

A pattern $w \in \mathcal{A}^D$, with $D \subset \mathbb{Z}^2$, is **locally valid** for f if for any $(i, t) \in D$ such that $C := (i + [\![-1, 1]\!]) \times \{t-1\} \subset D$, we have $p_{(i,t)} = f(p_{|C})$. Note that, in general, this notion depends on the local rule and not only on the RPCA. By compactness, if there exist locally valid square patterns of arbitrarily large height and width, then $\mathcal{O}_F \neq \emptyset$, *i.e.*, there are configurations which are never rejected. If $x \in F^{-t}(\mathcal{A}^{\mathbb{Z}})$, then $|(x, F(x), \dots, F^t(x))$ is a **locally valid horizontal strip** of height $t + 1$. The notion of a locally-valid horizontal strip depends only on the RPCA and not on the local rule, *i.e.*, it is a "global" notion.

For every $m \in \mathbb{N}$, $\delta = (\delta_0, \dots, \delta_{m-1}) \in \{-1, 0, 1\}^m$, we define the shift product $\sigma^\delta = \sigma^{\delta_0} \times \dots \times \sigma^{\delta_{m-1}}$. A **partial partition** (cellular) **automaton** (PPA) is a PCA $F = \sigma^\delta \circ \alpha$ over some alphabet $\mathcal{A} = \mathcal{A}_0 \times \dots \times \mathcal{A}_{m-1}$, where α is (the parallel synchronous application of) a partial permutation of $\mathcal{A}$. $-\delta_i$ is called the **direction** of field i. The (counter-intuitive) "$-$" is due to the fact that the normal definition of σ shifts everything to the left, while we are used to thinking of the positive direction as going to the right. So, if we want to have a field with speed $+1$, then we should apply σ^{-1} to it.

Every PPA is a RPCA with bi-radius 1 and conversely every RPCA is essentially a PPA (see for instance [24, Proposition 53]). Note, however, that the inverse of a PPA is not, formally, exactly a PPA: the permutation is performed after the shifts, in the form $\alpha^{-1} \circ \sigma^{-\delta}$. Nevertheless, it is conjugate, via α, to the corresponding PPA.

In order to define families of PPA that are somehow uniform, we consider the corresponding objects acting on infinite alphabets. A **partial partition automaton with infinite alphabet** (IPPA) is a partial map $F : (\mathbb{F}_5^{*m})^{\mathbb{Z}} \dashrightarrow (\mathbb{F}_5^{*m})^{\mathbb{Z}}$, where $m \in \mathbb{N}$, $F = (\sigma^{\delta_0} \times \dots \times \sigma^{\delta_{m-1}}) \circ \alpha$, the σ^{δ_j} are shifts over infinite $\mathbb{F}_5^*$ (that is $\sigma(y)_i = y_{i+1}$ for any $y \in (\mathbb{F}_5^*)^{\mathbb{Z}}$ and $i \in \mathbb{Z}$), and $\alpha\colon \mathbb{F}_5^{*m} \to \mathbb{F}_5^{*m}$ is a partial (infinite) **permutation**. By restricting the domain and the co-domain of an IPPA to finite subsets of $\mathbb{F}_5^{*m}$, we obtain normal (finite) PPA. In our constructions, the permutation α will always be length-preserving and the restriction will be taken over an alphabet of the form $\mathbb{F}_5^{\mathbf{k}}$.

If $F\colon \mathcal{A}^{\mathbb{Z}} \to \mathcal{A}^{\mathbb{Z}}$ and $G\colon \mathcal{B}^{\mathbb{Z}} \to \mathcal{B}^{\mathbb{Z}}$ are PCA, then we say that G is a **factor** of F if there exists a continuous map $H\colon \mathcal{A}^{\mathbb{Z}} \to \mathcal{B}^{\mathbb{Z}}$ such that $GH = HF$. If F and G are RPCA and F factors onto G, then it is easy to

see that $\mathcal{O}_F$ factors onto $\mathcal{O}_G$ through the map that sends $\mathcal{O}_x$ to $\mathcal{O}_{H(x)}$, for all $x \in \mathcal{A}^{\mathbb{Z}}$. However, the notion of factoring for RPCA is stronger, since it also takes into account the transient times of the RPCA (which are not relevant in the corresponding 2D SFTs).

Let $F_0, \dots, F_{n-1}$ be RPCA such that $\mathcal{D}(F_i) \cap \mathcal{D}(F_j) = \emptyset$, for all $i \neq j$. Then, $\bigsqcup_{i \in [\![0,n[\![} F_i$ denotes the map with domain $\bigsqcup_{i \in [\![0,n[\![} \mathcal{D}(F_i)$ and that agrees with F_i on $\mathcal{D}(F_i)$, for all $i \in [\![0, n[\![$. $\bigsqcup_{i \in [\![0,n[\![} F_i$ is not always an RPCA, since there might be a configuration that is not in any $\mathcal{D}(F_i)$ but that is locally everywhere in the domains (which are SFTs). However, and this will always be the case in this paper, $\bigsqcup_{i \in [\![0,n[\![} F_i$ is also an RPCA if $\mathcal{D}(F_i)$ and $\mathcal{D}(F_j)$ are over disjoint alphabets, for $i \neq j$. In this case, $\Omega_{\bigsqcup_{i \in [\![0,n[\![} F_i} = \bigsqcup_{i \in [\![0,n[\![} \Omega_{F_i}$ and $\mathcal{O}_{\bigsqcup_{i \in [\![0,n[\![} F_i} = \bigsqcup_{i \in [\![0,n[\![} \mathcal{O}_{F_i}$.

2.4.1 Expansiveness

The projective line $\mathbb{P} := \mathbb{R} \sqcup \{\infty\}$ is seen as the set of **slopes** to the *vertical* direction. Here, quite unconventionally, the horizontal direction is represented by ∞ and the vertical one by 0. The relevance of this choice will appear later, but in any case it does not affect any set-theoretical, topological or computable property because the inversion map over $\mathbb{P}$ is a computable homeomorphism.

The projective line $\mathbb{P}$ admits a natural effective topology if seen as the quotient of the circle by central symmetry: a subset is effectively closed if the corresponding subset of the circle is effectively closed as a subset of $[0,1]^2$. This topology is equivalent to the one-point compactification of the $\mathbb{R}$ and renders $\mathbb{P}$ a compact, metric space.

Let X be a 2D subshift, $l \in \mathbb{P}$ a slope and $\mathbf{l} \subset \mathbb{R}^2$ the corresponding vectorial line. We say that direction l is **expansive** for X if there exists a bounded shape $V \subset \mathbb{R}^2$ such that, for all $x, y \in X$,

$$x_{|(\mathbf{l}+V)\cap\mathbb{Z}^2} = y_{|(\mathbf{l}+V)\cap\mathbb{Z}^2} \Rightarrow x = y \ .$$

We denote by $\mathcal{N}(X)$ the set of **non-expansive** directions (*i.e.*, the set of directions that are not expansive). The terminology comes from the fact that if $l = p/q$ is rational (or infinite), then l is expansive for X if and only if the dynamical system $(X, \sigma^{(p,q)})$ is expansive, in the classical sense of expansive dynamical systems.

Expansive directions were first introduced by Boyle and Lind [6] in a more general setting. The following fact is a particular case of [6, Theorem 3.7].

Proposition 9. *Let X be a 2D subshift. Then, $\mathcal{N}(X)$ is closed. In addition, $\mathcal{N}(X)$ is empty if and only if X is finite.*

We say that X is **extremely expansive** if $|\mathcal{N}(X)| = 1$, which is, according to Proposition 9, the most constrained non-trivial case.

In the case of SFTs (actually, of all effective subshifts), we have an additional restriction on the set of non-expansive directions that comes from computation theory, as is usually the case, see [18, 20].

A direction $l \in \mathbb{P}$ can be represented as the pair of coordinates of the intersection of the line $\mathbf{l}$ with the unit circle. This gives two (symmetric with respect to the origin) representations for each direction which are computably equivalent. Computability questions about expansive directions can then be transferred to computability questions about pairs of real numbers, which we already know how to deal with.

It can be noted that effectively closed subsets that do not contain $\{\infty\}$ are exactly the effectively closed subsets of $\mathbb{R}$. The restriction map from $\mathbb{P}$ (with the above-defined effective topology) onto $\mathbb{R}$ is actually computable, and it can be noted that the pre-image of an effectively closed set by a computable function is effectively closed.

Lemma 10. *Let X be a 2D SFT. Then, $\mathcal{N}(X)$ is effectively closed.*

In particular, if an SFT X has a unique direction of non-expansiveness, then this direction must be computable.

Proof. The statement follows from the following two facts: First, it is semi-decidable whether a direction is expansive, *i.e.*, there exists a TM that takes as input a (rational direction) and halts if the direction is expansive. This follows from [6, Lemma 3.2]. Secondly, it is semi-decidable whether two *expansive* directions belong in the same expansive component. (The expansive component of an expansive direction is the largest connected set that includes the direction and is included in the set of expansive directions. One can see that it is always an open interval.) This follows from [34], as described in [5, Appendix C].

Having these two facts in mind, it is not difficult to see that the following algorithm enumerates a sequence of intervals whose union is the complement of $\mathcal{N}(X)$: For each rational direction, check whether it is expansive. Every time you find an expansive direction, check whether it is in the same component with one of the expansive directions that you have already found. Every time this is the case, output the whole interval of directions that is between them.

□

A subshift Y is called **extremely-expansively sofic** if there exists an extremely expansive SFT that factors onto Y. Since expansive directions are not preserved through block maps, an extremely-expansively sofic subshift need not be extremely expansive itself. In fact, as we will see, there

exist extremely-expansively sofic subshifts that do not have any direction of expansiveness.

Lemma 11. *Let $X_0, X_1, \dots$ be 2D subshifts over the same alphabet $\mathcal{A}$.*

- *If $X_0 \subseteq X_1$, then $\mathcal{N}(X_0) \subseteq \mathcal{N}(X_1)$.*
- *If $\mathcal{N}(X_0) \cap \mathcal{N}(X_1) = \emptyset$, then $X_0 \cap X_1$ is a finite subshift.*
- *If $\bigsqcup_w X_w$ is a closed disjoint (possibly uncountable) union, then $\mathcal{N}(\bigsqcup_w X_w) = \bigcup_w \mathcal{N}(X_w)$.*

Proof. The first claim follows immediately from the definitions.

For the proof of the second claim, we have that $\mathcal{N}(X_0 \cap X_1) \subseteq \mathcal{N}(X_0) \cap \mathcal{N}(X_1) = \emptyset$ according to the first claim. Therefore, $\mathcal{N}(X_0 \cap X_1) = \emptyset$, and since $X_0 \cap X_1$ is a subshift, Proposition 9 gives that it is finite.

Finally, for the last claim, the inclusion $\mathcal{N}(X_w) \subseteq \mathcal{N}(\bigsqcup_w X_w)$ comes from the first point.

For the other inclusion, assume $l \in \mathcal{N}(\bigsqcup_w X_w)$ Then, there exist $x, y \in \bigsqcup_w X_w$ which coincide over an open half-plane $H_l \subseteq \mathbb{R}^2$ of slope l and disagree somewhere outside it. The orbits of x and y under the shift action have a common limit point z. Then, z is in the intersection $X_x \cap X_y$ of the subshifts that contain x and y, respectively. By disjointness, we get that $X_x = X_y = X_{w'}$, for some w', which means that $l \in \mathcal{N}(X_{w'})$.

□

If F is an RPCA, then we denote $\mathcal{N}(F) := \mathcal{N}(\mathcal{O}_F)$. It is straightforward that the horizontal direction (which according to our definition is ∞) is expansive for F. It is not much more complicated to see that, if the bi-radius is 1, $\mathcal{N}(F) \subseteq [-1, 1]$ (directions around the horizontal are expansive).

Conversely, it can be shown that, up to a recoding, every 2D SFT for which the horizontal direction is expansive is equal to $\mathcal{O}_F$, for some RPCA F.

Chapter 3

Simulation

3.1 Simulation

If $S, T \in \mathbb{N}_1$ and $Q \in \mathbb{Z}$, we say that RPCA $F : \mathcal{A}^{\mathbb{Z}} \nrightarrow \mathcal{A}^{\mathbb{Z}}$ (S, T, Q)-**simulates** RPCA $G : \mathcal{B}^{\mathbb{Z}} \nrightarrow \mathcal{B}^{\mathbb{Z}}$ if there is a partial continuous **decoding** surjection $\Phi : \mathcal{A}^{\mathbb{Z}} \nrightarrow \mathcal{B}^{\mathbb{Z}}$ such that $\sigma\Phi = \Phi\sigma^S$, $G\Phi = \Phi\sigma^Q F^T$, $G^{-1}\Phi = \Phi\sigma^{-Q}F^{-T}$ and the **simulating** subshift $\tilde{\mathcal{D}}(\Phi) := \bigsqcup_{\substack{0 \leq t < T \\ 0 \leq s < S}} \sigma^s F^t(\mathcal{D}(\Phi))$ is a disjoint union.In other words, 1 step of G is encoded into T steps of F, up to some shift by Q, and the intermediary steps used are not valid encodings. We note $F \underset{S,T,Q,\Phi}{\succeq} G$, or when some parameters are clear from the context or not so important, $F \underset{S,T,\Phi}{\succeq} G$, $F \underset{S,T,Q}{\succeq}$, $F \underset{S,T}{\succeq} G$, or $F \succeq G$ (each time this symbol will be used, F and G are meant to be RPCA).

We remind the reader that according to our notations, $\sigma\Phi = \Phi\sigma^S$ and $G\Phi = \Phi\sigma^Q F^T$ and $G^{-1}\Phi = \Phi\sigma^{-Q}F^{-T}$ imply that the domains of the two partial functions are identical. This is in fact crucial for understanding the notion of simulation and it will be used extensively in the proofs and constructions to come. For example, this means that the equality $G\Phi = \Phi\sigma^Q F^T$ does not immediately imply $G^{-1}\Phi = \Phi\sigma^{-Q}F^{-T}$, because the domains of $G^{-1}\Phi$ and $\Phi\sigma^{-Q}F^{-T}$ might be different (if we only had the equality $G\Phi = \Phi\sigma^Q F^T$, it could happen that $x \in \mathcal{D}(G^{-1}\Phi)$ but $G^{-1}\Phi(x) \notin \Phi(\mathcal{A}^{\mathbb{Z}})$).

In fact, one can see that the couple of conditions $G\Phi = \Phi\sigma^Q F^T$ and $G^{-1}\Phi = \Phi\sigma^{-Q}F^{-T}$ is equivalent to the triple of conditions $G\Phi = \Phi\sigma^Q F^T$, $\mathcal{D}(G\Phi) = \mathcal{D}(\Phi\sigma^{-Q}F^{-T})$ and $\mathcal{D}(G^{-1}\Phi) = \mathcal{D}(\Phi F^T\sigma^Q)$.

F **exactly simulates** G if Φ is actually bijective. In other words, there exists a well-defined **encoding** function $\Phi^{-1} : \mathcal{B}^{\mathbb{Z}} \to \mathcal{D}(\Phi)$. F **completely simulates** G if, besides, $\Omega_F \subset \tilde{\mathcal{D}}(\Phi)$. In other words, every bi-infinite orbit of F will eventually encode some orbit of G. Actually, in our constructions we will even have the stronger $\mathcal{D}(F^{t'}) \subset \tilde{\mathcal{D}}(\Phi)$, for some $t' \in \mathbb{Z}$.

Remark 12.

1. $\mathcal{D}(\Phi) = \sigma^S(\mathcal{D}(\Phi))$.
2. $F \underset{S,T,DS}{\succeq} G$ *if and only if* $F \underset{S,T,0}{\succeq} \sigma^D G$.
3. *For any* $s \in [\![0, S[\![\, , t \in [\![0, T[\![$, $\sigma^s F^t(\mathcal{D}(\Phi))_{[S]}$ *is an SFT.*
4. *Since the union* $\tilde{\mathcal{D}}(\Phi)$ *is disjoint, there exists a shape* $U \subset_{finite} \mathbb{Z}$ *such that for any* $x \in \tilde{\mathcal{D}}(\Phi)$ *,* $x_{|U}$ *determines the (unique)* $s \in [\![0, S[\![$ *and* $t \in [\![0, T[\![$ *such that* $x \in \sigma^s F^t(\mathcal{D}(\Phi))$.

Proof. The first two claims follow immediately from the definitions.

For the third claim, notice that since Φ is continuous and $\sigma\Phi = \Phi\sigma^S$, this means that $\mathcal{D}(F)$ is the domain of a PCA over $\mathcal{A}^{\mathbb{Z}}_{[S]}$, so it is an SFT. Since σ and F are invertible maps and the property of being an SFT is preserved under invertible maps, we have that $\sigma^s F^t(\mathcal{D}(\Phi))_{[S]}$ is an SFT for all $s \in [\![0, S[\![$ and $t \in [\![0, t[\![$.

The last claim follows easily from the disjointness using a classical compactness argument. □

We can prove an analogue of Curtis-Lyndon-Hedlund theorem for decoding and encoding functions.

Remark 13. *The decoding function* Φ *admits a neighbourhood* $V \subset_{finite} \mathbb{Z}$ *and a partial* ***bulked local rule*** $\phi : \mathcal{A}^V \nrightarrow \mathcal{B}$ *such that for all* $x \in \mathcal{A}^{\mathbb{Z}}$, $\Phi(x)$ *is defined if and only if* $\phi(x_{|iS+V})$ *is defined for any* $i \in \mathbb{Z}$*, in which case the latter is equal to* $\Phi(x)_i$.
If the simulation is exact, the encoding function Φ^{-1} *admits a neighbourhood* $V \subset_{finite} \mathbb{Z}$ *and a partial* ***unbulked local rule****, abusively noted* $\phi^{-1} : \mathcal{B}^V \nrightarrow \mathcal{A}^S$ *such that for all* $y \in \mathcal{B}^{\mathbb{Z}}$, $\Phi^{-1}(y)$ *is defined if and only if* $\phi^{-1}(y_{|i+V})$ *is defined for any* $i \in \mathbb{Z}$*, in which case the latter is equal to* $\Phi^{-1}(y)_{[\![iS,(i+1)S[\![}$.

Exact complete vertical (*i.e.*, $Q = 0$) simulation is stronger than most notions found in the literature. In particular:

- $\mathcal{O}_F$ simulates $\mathcal{O}_G$ in the sense of [8].
- The $\mathbb{Z}^2$-action (F, σ) over the limit set Ω_F (or the 2D SFT $\mathcal{O}_F$) is conjugate to a suspension of Ω_G in the sense of a homeomorphism

$$\begin{array}{rccl} \Psi : & \Omega_F & \to & \Omega_G \times [\![0, S[\![\times [\![0, T[\![\\ & x & \mapsto & (\Phi F^{-t}\sigma^{-s}(x), s, t), \text{ where } F^{-t}\sigma^{-s}(x) \in \mathcal{D}(\Phi) \end{array}$$

- The $\mathbb{Z}^2$-action (G, σ) over the limit set Ω_G (or the 2D SFT $\mathcal{O}_G$) is conjugate to the $\mathbb{Z}^2$-action (F^T, σ^S) restricted to $\mathcal{D}(\Phi) \cap \Omega_F$ (see [10]);

- G is a sub-automaton of a rescaling of F, so that F simulates G according to the definition of simulation given in [35]. While it is not necessary to formally define this notion of simulation, we can intuitively say that rescaling corresponds to the role of parameters S and T in our definition, while the sub-automaton condition corresponds to the decoding function Φ. We notice, however, that Ollinger's definition is more general than ours, since it does not require $\tilde{\mathcal{D}}(\Phi)$ to be a disjoint union, while the simulated can also be rescaled.

But the definition above also involves the transient part: every locally valid horizontal strip of height $t+1$ for G gives a locally valid horizontal strip of height $Tt+1$ for F.

Each kind of simulation is a conjugacy invariant. If F simulates G (resp. exactly), then it simulates (resp. exactly) any of its subsystems (but clearly, completeness is not preserved). Complete simulation clearly includes factor; for instance $F \times G \underset{1,1,0}{\succeq} F$ completely if G does not have empty domain. Also $F \times G \underset{1,1,0}{\succeq} F$ exactly if G includes a singleton subsystem. The simulation is simultaneously exact and complete if G is a singleton system. The surjectivity of Φ implies that only systems with empty domain can be simulated by systems with empty domain. We will mainly focus on **non-trivial** simulations: $S, T > 1$ and G does not have empty domain.

Remark 14. *If $F \underset{S,T,Q,\Phi}{\succeq} G$ non-trivially, then for all $j \in [\![0, T]\!]$:*

$$F^j(\mathcal{D}(G\Phi)) = \sigma^{-Q} F^{-(T-j)}(\mathcal{D}(G^{-1}\Phi) \neq \emptyset$$

More specifically, a configuration "in the middle" of the work period, *i.e.*, when $j = \lfloor T/2 \rfloor$ has at least $\lfloor T/2 \rfloor$ forward and backward images, or, in other words, it belongs to $\Omega_F^{\lfloor T/2 \rfloor}$.

The following lemma states that the limit sets correspond, in the case of complete simulation. It is a more mathematical and detailed version of the comment that we made earlier, that a valid strip horizontal of height $t+1$ in G gives a valid horizontal strip of height $Tt+1$ in F (provided that the strip is simulated).

Lemma 15. *Assume $F \underset{S,T,Q,\Phi}{\succeq} G$.*

1. *If $j \in \mathbb{N} \sqcup \{\infty\}$, then*

$$\tilde{\mathcal{D}}^j(\Phi) := \bigsqcup_{\substack{0 \leq t < T \\ 0 \leq s < S}} \sigma^s F^t \Phi^{-1}(\Omega_G^j)$$

is a disjoint union and a subshift, included in $\Omega_F^{(j-1)T+1}$.
In addition, $\tilde{\mathcal{D}}^j(\Phi) \supset \tilde{\mathcal{D}}^{j+1}(\Phi)$ and $\tilde{\mathcal{D}}^\infty(\Phi) = \bigcap_{j \in \mathbb{N}} \tilde{\mathcal{D}}^j(\Phi)$.

2. $\Omega_F \supset \tilde{\mathcal{D}}^\infty(\Phi)$.

3. If the simulation is complete, then $\Omega_F = \tilde{\mathcal{D}}^\infty(\Phi)$.

Proof.

1. It is clear that $\tilde{\mathcal{D}}^j(\Phi)$ is a disjoint union and a subshift, each subset in the union being (syntactically) included in one in the expression of $\tilde{\mathcal{D}}(\Phi)$. Assume that $F \underset{S,T,Q,\Phi}{\succeq} G$ for some $Q \in \mathbb{Z}$. Now,

$$\begin{aligned}\Phi^{-1}(\Omega_G^j) &= \mathcal{D}(G^j\Phi) \cap \mathcal{D}(G^{-j}\Phi) \\ &= \mathcal{D}(\Phi\sigma^{jQ}F^{jT}) \cap \mathcal{D}(\Phi\sigma^{-jQ}F^{-jT}) \\ &\subset \mathcal{D}(F^{jT}) \cap \mathcal{D}(F^{-jT}) = \Omega_F^{jT}\ .\end{aligned}$$

   Hence, for any $s \in [\![0, S[\![$ and any $t \in [\![0, T[\![$, $\sigma^s F^t \Phi^{-1}(\Omega_G^j) \subset \Omega_F^{jT-T+1}$. The other claims follow from the definitions.

2. It is obvious from the previous point that $\bigcap_{j\in\mathbb{N}} \Omega_F^{jT} \supset \bigcap_{j\in\mathbb{N}} \tilde{\mathcal{D}}^j(\Phi) = \tilde{\mathcal{D}}^\infty(\Phi)$.

3. Conversely, assume $x \in \Omega_F$, so that clearly $\forall k \in \mathbb{Z}, F^k(x) \in \Omega_F$. By completeness, there exist $y \in \mathcal{D}(\Phi)$ and $s \in [\![0, S[\![, t \in [\![0, T[\![$ such that $\sigma^s F^t(y) = x$. Disjointness and a direct induction give that for all $k \in \mathbb{Z}$, $F^k(y) \in F^{k \bmod T}\sigma^{Q\lfloor k/T \rfloor}(\mathcal{D}(\Phi))$. In particular, for all $j \in \mathbb{Z}$, $G^j\Phi(y) = \Phi F^{jT}\sigma^{jQ}(y)$ is defined. This gives that $\Phi(y) \in \Omega_G$, so $x \in \sigma^{-s}F^{-t}\Phi^{-1}(\Omega_G) = \sigma^{S-s}F^{T-t}\Phi^{-1}(\Omega_G)$.

□

The following remark links the periodic points of the simulating and simulated systems. It is essential for proving aperiodicity of the subshifts that we construct. The same result appears in [8, 36], even though the argument essentially goes back to the kite-and-dart tile set of Penrose. We give a slightly more general version of the usual result also takes into consideration the shift by Q.

Remark 16. *If* $F \underset{S,T,Q}{\succeq} G$ *completely, then* $\mathcal{O}_F$ *admits a configuration with period* $(s - lQ, t)$ *if and only if* $\mathcal{O}_G$ *admits a configuration with period* (k, l)*, where* $s = kS$ *and* $t = lT$.

We will only use the case $Q = 0$, for which it is intuitively clear to see that it holds true. When $q \neq 0$, one has to have in mind that for every T time steps of a configuration of F, the simulated configuration is shifted Q steps to the left.

3.2 Nested simulations

In the sequel, we will be most interested in infinite sequences of simulations of the form: $F_0 \succeq F_1 \succeq F_2 \succeq \ldots$. This looks like a formidable task, since every RPCA of the sequence must contain the information about an infinite number of configurations and update this information within a determined time, but, as the results of this section will imply, an infinite sequence of simulations gives RPCA with very useful properties. The construction of these sequences forms the basic part of our constructions and will be done in the following chapters.

If $\mathbf{S} = (S_i)_{0 \le i \le n-1}$ is a sequence of numbers, then $\mathbf{1S}$ is the sequence whose first element is equal to 1 with the elements of $\mathbf{S}$ shifted by one after it. If $\mathbf{S} = (S_i)_{0 \le i \le n-1}$ and $\mathbf{T} = (T_i)_{0 \le i \le n-1}$ are finite sequences of non-zero numbers, then $\mathbf{1S}/\mathbf{T}$ is the sequence $(S_{i-1}/T_i)_{0 \le i \le n-1}$, where $S_{-1} := 1$. A short calculation shows that

$$\overline{\mathbf{Q}}^{\mathbf{1S/T}} \prod T_i = \sum_{0 \le i \le n-1} \left(Q_i \prod_{0 < j < i} S_j \prod_{i < j \le n-1} T_j \right).$$

Lemma 17. *Simulation (resp. exact, complete, exact and complete) is a preorder.*
More precisely, if $F_0 \underset{S_0,T_0,Q_0,\Phi_0}{\succeq} F_1 \underset{S_1,T_1,Q_1,\Phi_1}{\succeq} \cdots \underset{S_{n-1},T_{n-1},Q_{n-1},\Phi_{n-1}}{\succeq} F_n$ *(resp. exactly, completely) for some* $n \in \mathbb{N}$, *then* $F_0 \underset{S,T,Q,\Phi}{\succeq} F_n$ *(resp. exactly, completely), where* $(S, T, Q, \Phi) = (\prod S_i, \prod T_i, \overline{\mathbf{Q}}^{\mathbf{1S/T}} \prod T_i, \Phi_{n-1} \cdots \Phi_0)$.

The products range from 0 to $n-1$. If there were no shifts in the simulation (*i.e.*, if $Q_i = 0$ for all i) the above statement would be more or less trivial. Even in the presence of shifts, the proof is essentially a simple verification.

Proof.

- Clearly $F \underset{1,1,0,\mathrm{id}}{\succeq} F$.
- Now suppose $F \underset{S,T,Q,\Phi}{\succeq} G \underset{S',T',Q',\Phi'}{\succeq} H$. Then it is clear that $\sigma\Phi'\Phi = \Phi'\sigma^{S'}\Phi = \Phi'\Phi\sigma^{S'S}$ and $H\Phi'\Phi = \Phi'\sigma^{Q'}G^{T'}\Phi = \Phi'\Phi\sigma^{QT'+SQ'}F^{T'T}$.

Moreover:

$$\begin{aligned}
\bigsqcup_{\substack{0\le t<T\\0\le s<S}} \sigma^s F^t(\mathcal{D}(\Phi)) &\supset \bigsqcup_{\substack{0\le t<T\\0\le s<S}} \sigma^s F^t \Phi^{-1}\Big(\bigsqcup_{\substack{0\le t'<T'\\0\le s'<S'}} \sigma^{s'} G^{t'}(\mathcal{D}(\Phi'))\Big)\\
&= \bigsqcup_{\substack{0\le t<T\\0\le s<S}} \bigsqcup_{\substack{0\le t'<T'\\0\le s'<S'}} \sigma^s F^t \Phi^{-1}(\sigma^{s'} G^{t'}(\mathcal{D}(\Phi')))\\
&= \bigsqcup_{\substack{0\le t<T\\0\le s<S}} \bigsqcup_{\substack{0\le t'<T'\\0\le s'<S'}} F^{t+t'T} \sigma^{s+s'S+t'Q}(\mathcal{D}(\Phi'\Phi))\\
&= \bigsqcup_{\substack{0\le t<TT'\\0\le s<SS'}} \sigma^s F^t(\mathcal{D}(\Phi'\Phi)) =: \tilde{\mathcal{D}}(\Phi'\Phi)\ .
\end{aligned}$$

This proves that $F \underset{SS',TT',QT'+SQ',\Phi'\Phi}{\succeq} H$.

- If Φ and Φ' are bijections, then $\Phi'\Phi$ is also a bijection.
- If both simulations are complete, then by Point 3 of Lemma 15,

$$\begin{aligned}
\Omega_F &= \bigsqcup_{\substack{0\le t<T\\0\le s<S}} \sigma^s F^t \Phi^{-1}(\Omega_G)\\
&\subset \bigsqcup_{\substack{0\le t<T\\0\le s<S}} \sigma^s F^t \Phi^{-1}(\tilde{\mathcal{D}}(\Phi'))\\
&= \tilde{\mathcal{D}}(\Phi'\Phi).
\end{aligned}$$

- A direct induction gives the expected results.

□

Similarly to simulations, which involve a decomposition of the system in terms of how much is shifted the grid on which to read the encoding, a sequence of simulations involves a nested decomposition, which gives a full skeleton, inside each configuration, as expressed by the following lemma. Here, and in the following, we use gothic letters to denote sequences, but the corresponding normal letters to denote the elements of the sequences. Also, if $\mathfrak{S}$ is an infinite sequence and $n \in \mathbb{N}$, then $\mathfrak{S}_{[\![0,n[\![}$ is the finite prefix of length n of $\mathfrak{S}$. Finally, if $(\Phi_i)_{i\in\mathbb{N}}$ is a sequence of decoding functions, then $\Phi_{[\![0,n[\![}$ will be the decoding function $\Phi_{n-1}\cdots\Phi_0$.

Lemma 18.

1. *If* $F_0 \underset{S_0,\overline{T_0},\Phi_0}{\succeq} F_1 \underset{S_1,\overline{T_1},\Phi_1}{\succeq} \cdots \underset{S_{n-1},\overline{T_{n-1}},\Phi_{n-1}}{\succeq} F_n \underset{S_n,\overline{T_n},\Phi_n}{\succeq} \cdots$ *and* $j \in \mathbb{N} \sqcup \{\infty\}$, *then*

$$\begin{aligned}\tilde{\mathcal{D}}^j(\Phi) &:= \bigcap_{n\in\mathbb{N}} \tilde{\mathcal{D}}^j(\Phi_{[\![0,n[\![}) \\ &= \bigsqcup_{\substack{\mathfrak{t}\in\prod_{i\in\mathbb{N}}[\![0,T_i[\![\\ \mathfrak{s}\in\prod_{i\in\mathbb{N}}[\![0,S_i[\![}} \bigcap_{n\in\mathbb{N}} \sigma^{\overline{\mathfrak{s}_{[\![0,n[\![}}^{\mathfrak{S}}} F_0^{\overline{\mathfrak{t}_{[\![0,n[\![}}^{\mathfrak{T}}} \Phi_0^{-1}\cdots\Phi_{n-1}^{-1}(\Omega^j_{F_n})\end{aligned}$$

is a disjoint union and a subshift.
In addition, $\tilde{\mathcal{D}}^j(\Phi) \supset \tilde{\mathcal{D}}^{j+1}(\Phi)$ *and* $\tilde{\mathcal{D}}^\infty(\Phi) = \bigcap_{j\in\mathbb{N}} \tilde{\mathcal{D}}^j(\Phi)$.

2. *If, besides, all simulations are nontrivial, then* $\tilde{\mathcal{D}}^2(\Phi) = \tilde{\mathcal{D}}^\infty(\Phi) \subset \Omega_{F_0}$ *is uncountable.*

3. *If the simulations (in the hypothesis of Point 1) are complete, then* $\tilde{\mathcal{D}}^\infty(\Phi) = \Omega_{F_0}$.

4. *If the sequence* $(\Phi_n)_{n\in\mathbb{N}}$ *is computable, then the map* $x \in \tilde{\mathcal{D}}^\infty(\Phi) \to (s_i,t_i)_{i\in\mathbb{N}}$, *where* $x \in \bigcap_n \sigma^{\overline{\mathfrak{s}_{[\![0,n[\![}}^{\mathfrak{S}}} F_0^{\overline{\mathfrak{t}_{[\![0,n[\![}}^{\mathfrak{T}}} \mathcal{D}(\Phi_{[\![0,n[\![})$ *is computable.*

Point 2 implies nonemptiness of Ω_{F_0} and $\mathcal{O}_{F_0}$, and of any Ω_{F_n}, since all those statements can be applied to the sequence starting from n. Point 4 states that we can always recover the skeleton from a valid configuration. In particular the skeleton map is continuous.

Proof.

1. By Lemma 17 and compactness, it is clear that $\tilde{\mathcal{D}}^j(\Phi)$ is a subshift. The equality is rather easily checkable.
   We can see that the union is disjoint: if $(\mathfrak{s},\mathfrak{t}) \neq (\mathfrak{s}',\mathfrak{t}')$, say $(s_m, t_m \neq s'_m, t'_m)$, then $\bigcap_{n\in\mathbb{N}} \sigma^{\overline{\mathfrak{s}_{[\![0,n[\![}}^{\mathfrak{S}}} F_0^{\overline{\mathfrak{t}_{[\![0,n[\![}}^{\mathfrak{T}}} \Phi_0^{-1}\cdots\Phi_{n-1}^{-1}(\Omega^j_{F_n})$ is included in $\sigma^{\overline{\mathfrak{s}_{[\![0,m[\![}}^{\mathfrak{S}}} F_0^{\overline{\mathfrak{t}_{[\![0,m[\![}}^{\mathfrak{T}}} \Phi_0^{-1}\cdots\Phi_{m-1}^{-1}(\Omega^j_{F_m})$, which is, according to Lemma 17, disjoint from $\sigma^{\overline{\mathfrak{s}'_{[\![0,m[\![}}^{\mathfrak{S}}} F_0^{\overline{\mathfrak{t}'_{[\![0,m[\![}}^{\mathfrak{T}}} \Phi_0^{-1}\cdots\Phi_{m-1}^{-1}(\Omega^j_{F_m})$ which includes $\bigcap_{n\in\mathbb{N}} \sigma^{\overline{\mathfrak{s}'_{[\![0,n[\![}}^{\mathfrak{S}}} F_0^{\overline{\mathfrak{t}'_{[\![0,n[\![}}^{\mathfrak{T}}} \Phi_0^{-1}\cdots\Phi_{n-1}^{-1}(\Omega^j_{F_n})$.

2. Since for any $n \in \mathbb{N}$ and $m \geq n$, $F_n \underset{S_n\cdots S_{m-1},\overline{T_n\cdots T_{m-1}},\Phi_{m-1}\cdots\Phi_n}{\succeq} F_m$, then Point 1 of Lemma 15 says that

$$\Omega_{F_n^{T_n\cdots T_{m-1}+1}} \supset \tilde{\mathcal{D}}^2(\Phi_{[\![n,m[\![}) \ .$$

If the simulations are nontrivial, then $T_n \cdots T_{m-1} \to \infty$ when n is fixed and $m \to \infty$, and

$$\Omega_{F_n} \supset \bigcap_{m\in\mathbb{N}} \Omega_{F_n^{T_0\cdots T_{m-1}+1}} \supset \bigcap_{m\in\mathbb{N}} \tilde{\mathcal{D}}^2(\Phi_{[\![n,m[\![}) = \tilde{\mathcal{D}}^2(\Phi_{[\![n,\infty[\![}) \ .$$

Injecting this inclusion in the definition of $\tilde{\mathcal{D}}^\infty(\Phi_{[\![0,n[\![})$ gives that

$$\tilde{\mathcal{D}}^\infty(\Phi) \supset \bigcap_{m\geq 0} \tilde{\mathcal{D}}^2(\Phi_{[\![0,m[\![}) \supset \tilde{\mathcal{D}}^2(\Phi).$$

The converse is trivially true, and Point 3 of Lemma 15 already tells us that $\tilde{\mathcal{D}}^\infty(\Phi) \subset \Omega_{F_0}$.
Moreover, since $F_n \underset{S_nS_{n+1},T_nT_{n+1}}{\succeq} F_{n+2}$ with $T_nT_{n+1} \geq 4$, then Remark 14 gives that $\Omega^2_{F_n}$ is non-empty. Therefore, each of the uncountably many subsets in the disjoint union expressing $\tilde{\mathcal{D}}^2(\Phi)$ is a closed non-empty intersection.

3. If $n \in \mathbb{N}$ is such that $F_0 \underset{S_0\cdots S_{n-1},T_0\cdots T_{n-1},\Phi_{n-1}\cdots\Phi_0}{\succeq} F_n$ completely, then by Point 3 of Lemma 15, $\Omega_{F_0} = \tilde{\mathcal{D}}^\infty(\Phi_{[\![0,n[\![})$.

4. This follows from repeated application of Remark 5.4 and the fact that $\Phi_{[\![0,n[\![}$ is a decoding function for all $n \in \mathbb{N}$.

□

The following extends Lemma 18 (which can be recovered by $\mathcal{B}_i$ being singletons). In this case, every RPCA simulates a disjoint union of RPCA, each one of which simulates a disjoint union of RPCA and so on. In this way, we obtain an "infinite tree" of simulations. Along any branch of this tree, Lemma 18 is true, but, more importantly, something similar is true even when we take all the (possibly uncountable) branches of this tree together.

Lemma 19.

1. *Let $(\mathcal{B}_n)_{n\in\mathbb{N}}$ be a sequence of finite alphabets, such that for any word $u \in \prod_{i<n} \mathcal{B}_i$ of length $n \in \mathbb{N}$, there exist $S_u, T_u, Q_u \in \mathbb{N}$, a decoding function Φ_u and a RPCA F_u such that $F_u \underset{S_u,T_u,\Phi_u}{\succeq} \bigsqcup_{b\in\mathcal{B}_n} F_{ub}$.*

   Let $\tilde{\mathcal{D}}^j_z(\Phi) := \bigcap_{n\in\mathbb{N}} \tilde{\mathcal{D}}^j(\Phi^z_{[\![0,n[\![}) $ for all $j \in \mathbb{N} \sqcup \{\infty\}, z \in \prod_{i\in\mathbb{N}} \mathcal{B}_i$.

   Then, for any $j \in \mathbb{N} \sqcup \{\infty\}$ and any closed $Y \subset \prod_{i\in\mathbb{N}} \mathcal{B}_i$,

$$\tilde{\mathcal{D}}^j_Y(\Phi) := \bigsqcup_{z\in Y} \tilde{\mathcal{D}}^j_z(\Phi)$$

   is a disjoint union and a subshift, and $\tilde{\mathcal{D}}^2_Y(\Phi) = \tilde{\mathcal{D}}^\infty_Y(\Phi) \subset \Omega_{F_\epsilon}$.

2. *Besides, the set* $Z := \left\{ z \in \prod_{i\in\mathbb{N}} \mathcal{B}_i \,\middle|\, \tilde{\mathcal{D}}^{\infty}_z(\Phi) \neq \emptyset \right\}$ *corresponding to nested nontrivial, non-empty simulations is closed. If the simulations are complete, then* $\tilde{\mathcal{D}}^{2}_Z(\Phi) = \tilde{\mathcal{D}}^{\infty}_Z(\Phi) = \Omega_{F_\epsilon}$.

In the above statement, the notation $\Phi^z_{[\![0,n[\![}$ stands for the composition $\Phi_{z_{[\![0,n[\![}} \cdots \Phi_{z_0}\Phi_\epsilon$, which is the decoding function from $F_{z_{[\![0,n[\![}}$ onto F_ϵ.

Proof.

1. Point 2 of Lemma 18 gives that $\tilde{\mathcal{D}}^{\infty}_z(\Phi) \neq \emptyset$ if $F_{z_{[\![0,n[\![}} \succeq F_{z_{[\![0,n+1]\!]}}$ non trivially for any $n \in \mathbb{N}$, *i.e.*, all these RPCA have non-empty domain. The converse is obvious.
   By the same distributivity of decreasing intersections over unions as for Point 1 of Lemma 18, it can be easily seen that

$$\tilde{\mathcal{D}}^{j}_Y(\Phi) = \bigcap_{n\in\mathbb{N}} \bigsqcup_{u\in\mathcal{L}_n(Y)} \bigsqcup_{\substack{0\le t<\prod_{i<n} T_{u_{[\![0,i[\![}} \\ 0\le s<\prod_{i<n} S_{u_{[\![0,i[\![}}} \sigma^s F^t_\epsilon \Phi_\epsilon^{-1}\Phi_{u_0}^{-1}\cdots\Phi_u^{-1}(\Omega^j_{F_u}) \ ,$$

   which is a decreasing intersection of finite unions of subshifts, and we have $\tilde{\mathcal{D}}^{2}_z(\Phi) = \tilde{\mathcal{D}}^{\infty}_z(\Phi) \subset \Omega_{F_\epsilon}$ for all $z \in Y$.

2. If $F_u \succeq \bigsqcup_{a\in\mathcal{B}_n} F_{ua}$ completely, then Point 3 of Lemma 15 gives

$$\Omega_{F_u} = \bigsqcup_{\substack{0\le t<T_u \\ 0\le s<S_u}} F^t_u \sigma^s \Phi_u^{-1}\Big(\bigsqcup_{a\in\mathcal{B}_n} \Omega_{F_{ua}}\Big) \ .$$

   An immediate induction gives for any $n \in \mathbb{N}$,

$$\Omega_{F_\epsilon} = \bigsqcup_{u\in\mathcal{L}_{n+1}(Z)} \bigsqcup_{\substack{0\le t<\prod_{i<n} T_{u_{[\![0,i[\![}} \\ 0\le s<\prod_{i<n} S_{u_{[\![0,i[\![}}} \sigma^s F^t_\epsilon \Phi_\epsilon^{-1}\Phi_{u_0}^{-1}\cdots\Phi^{-1}_{u_{[\![0,n[\![}}(\Omega_{F_u}) \ .$$

   Being true for any n, this gives the result.

□

Lemmas 18 and 19 can be seen as extensions of Lemma 15 in the case of an infinite nested simulation. The following lemma can be seen as such an extension of Remark 16.

Lemma 20. *If* $F_0 \underset{S_0,T_0}{\succeq} F_1 \underset{S_1,T_1}{\succeq} \cdots \underset{S_{n-1},T_{n-1}}{\succeq} F_n \underset{S_n,T_n}{\succeq} \cdots$ *completely, with* $S_n, T_n > 1$ *for any* $n \in \mathbb{N}$*, then* $\mathcal{O}_{F_0}$ *is aperiodic.*

In particular, either $\Omega_{F_n} = \emptyset$ $(= \mathcal{O}_{F_n})$ for all $n \in \mathbb{N}$ or, Ω_{F_n} (and $\mathcal{O}_{F_n}$) is aperiodic uncountable, for all $n \in \mathbb{N}$.

Proof. From Lemma 17, $F_0 \underset{S_0\cdots S_{n-1},T_0\cdots T_{n-1}}{\succeq} F_n$ completely. By Remark 16, $\mathcal{O}_{F_0}$ cannot have any nontrival period less than $S_0 \cdots S_{n-1}$ horizontally and less than $T_0 \cdots T_{n-1}$ vertically. If these two products go to infinity, we get that there cannot exist any periodic points. □

In fact, it follows from the proof that it is enough that one of the products $\prod_{i\in\mathbb{N}} S_i$ and $\prod_{i\in\mathbb{N}} T_i$ is infinite. It is well known that a non-empty, aperiodic 2D SFT is uncountable. Lemma 18 gives some additional information about how uncountability occurs in the case of an infinite nested simulation.

3.3 Expansiveness and simulation

The following lemmas highlight the relation between the notions of simulation and expansive directions. This subsection extends slightly Section 5 in [19]. The following lemmas correspond to Lemma 5.1 and Lemma 5.3 in [19], which examine how the so-called "shape of prediction" evolves. It also motivates the choice of considering the horizontal direction as ∞, which will make many future expressions clearer.

Lemma 21. *Suppose* $F \underset{S,T,Q}{\succeq} G$ *exactly. Then* $\mathcal{N}(F) \supseteq \frac{1}{T}(Q + S\mathcal{N}(G))$. *Moreover, if the simulation is complete, then* $\mathcal{N}(F) = \frac{1}{T}(Q + S\mathcal{N}(G))$.

In particular, $\mathcal{N}(\sigma^{-Q}G) = \mathcal{N}(G) + Q$ and $\mathcal{N}(G^T) = \frac{1}{T}\mathcal{N}(G)$.

Proof. Let us consider the matrix $M := \begin{bmatrix} S & Q \\ 0 & T \end{bmatrix}$ as acting over $\mathbb{R}^2$. Consider a slope $l \in \mathbb{P}$, $\mathbf{l} \subset \mathbb{R}^2$ the corresponding vectorial line, $\mathbf{l}' := M\mathbf{l}$ the vectorial line corresponding to slope $\frac{S}{T}l + \frac{Q}{T}$. Roughly, $\mathbf{l}'$ for F corresponds to $\mathbf{l}$ for G.

- Consider a finite shape $W' \subset \mathbb{R}^2$, U and f the neighbourhood and local rule of F, V and ϕ^{-1} those of Φ^{-1}, as defined in Remark 13. Without loss of generality, we can assume that $U = [\![-uS, uS]\!]$, for some $u \in \mathbb{N}$.

  Let $W := M^{-1}W' + (T\,[\![-u, u]\!] + V + [\![-Q, 0[\![) \times \{0\} + [-1, 2[\times[0, 1[$. If $l \in \mathcal{N}(G)$, then there exist configurations $x \neq y \in \Omega_G$ such that $\mathcal{O}_G(x)_{|\mathbf{l}+W} = \mathcal{O}_G(y)_{|\mathbf{l}+W}$.

  Then, $\mathcal{O}_F(\Phi^{-1}(x)) \neq \mathcal{O}_F(\Phi^{-1}(y))$, but we claim that

  $$\mathcal{O}_F(\Phi^{-1}(x))_{|\mathbf{l}'+W'} = \mathcal{O}_F(\Phi^{-1}(y))_{|\mathbf{l}'+W'}.$$

  Since W' was an arbitrary finite shape, this implies that $\frac{S}{T}l + \frac{Q}{T} \in \mathcal{N}(F)$, which proves that $\mathcal{N}(F) \supseteq \frac{1}{T}(Q + S\mathcal{N}(G))$.

Let us proceed with the proof of the claim. Let $(p_1, p_2) \in \mathbf{l}' + W'$ and write $p_1 =: mS + r$, $p_2 =: nT + q$ and $n := m'S + r'$, where $m, r, n, q, m', r' \in \mathbb{Z}$ and $0 \leq r, r' < S$ and $0 \leq q < T$. Intuitively, we can think that (p_1, p_2) belongs to the encoding of the m'th letter of $\sigma^{-m'Q}G^n(x)$ and $G^n(y)$.

More precisely, a straightforward computation shows that

$$M^{-1}(p_1, p_2) = (m - Qm' + r/S - r'/S + q/T, n + q/T),$$

so that $(m - Qm', n) \in M^{-1}(p_1, p_2) + [-1, 2[\times[0, 1[$. This, in turn, implies that $(m-Qm', n)+(T[\![-u, u]\!]+[\![-Q, 0[\![+V)\times\{0\}$ is included in $\mathbf{l} + W$, so that

$$\mathcal{O}_G(x)_{|(m-Qm',n)+(T[\![-u,u]\!]+[\![-Q,0[\![+V)\times\{0\}} = \\ = \mathcal{O}_G(y)_{|(m-Qm',n)+(T[\![-u,u]\!]+[\![-Q,0[\![+V)\times\{0\}}.$$

Using the facts that V is the neighbourhood of ϕ^{-1} and that Φ^{-1} "blows-up" letters into blocks of size $S \times T$ with an additional shift of Q for every vertical time step, we deduce that

$$\mathcal{O}_F(\Phi^{-1}(x))_{|([\![(m-Qm')S,(m-Qm'+1)S[\![+T[\![-uS,uS]\!]+[\![-QS,0[\![+nQ)\times\{nT\}} = \\ = \mathcal{O}_F(\Phi^{-1}(y))_{|([\![(m-Qm')S,(m-Qm'+1)S[\![+T[\![-uS,uS]\!]+[\![-QS,0[\![+nQ)\times\{nT\}}.$$

Notice that $nQ - Qm'S = r'Q$. Now, using the fact that $T[\![-uS, uS]\!]$ is a neighbourhood for f^q and $[\![-QS, 0[\![$ for $\sigma^{-r'Q}$, we obtain that

$$\sigma^{-r'Q}F^q\left(\mathcal{O}_F(\Phi^{-1}(x))\right)_{|[\![mS+r'Q,(m+1)S+r'Q[\![\times\{nT\}} = \\ = \sigma^{-r'Q}F^q\left(\mathcal{O}_F(\Phi^{-1}(y))\right)_{|[\![mS+r'Q,(m+1)S+r'Q[\![\times\{nT\}}.$$

The last equality implies that $\mathcal{O}_F(\Phi^{-1}(x))_{|(p_1,p_2)} = \mathcal{O}_F(\Phi^{-1}(y))_{|(p_1,p_2)}$, because

$$\begin{aligned} & \sigma^{-r'Q}F^q\left(\mathcal{O}_F(\Phi^{-1}(x))\right)_{|[\![mS+r'Q,(m+1)S+r'Q[\![\times\{nT\}} \\ = \; & \mathcal{O}_F(\Phi^{-1}(x))_{|[\![mS,(m+1)S[\![\times\{nT+q\}} \\ = \; & \mathcal{O}_F(\Phi^{-1}(x))_{|[\![mS,(m+1)S[\![\times\{p_2\}} \end{aligned}$$

and $p_1 \in [\![mS, (m + 1)S[\![$.

- Consider a finite shape $W \subset \mathbb{R}^2$, U the synchronizing shape as defined in Remark 12, V and ϕ the neighbourhood and local rule of Φ as defined in Remark 13, and $W' := MW + (V \cup U) \times \{0\} - [\![0, S[\![\times [\![0, T[\![$.

  If $\frac{S}{T}l + \frac{Q}{T} \in \mathcal{N}(F)$, then there exist configurations $x \neq y \in \Omega_F$ such that $\mathcal{O}_F(x)_{|\mathbf{l}'+W'} = \mathcal{O}_G(y)_{|\mathbf{l}'+W'}$.

  By Remark 12 and completeness of the simulation, there exist common $s \in [\![0, S[\![$ and $t \in [\![0, T[\![$ such that $x' := \sigma^{-s}F^{-t}(x)$ and $y' := \sigma^{-s}F^{-t}(y)$ are in $\mathcal{D}(\Phi)$. It follows easily from the definitions that $\mathcal{O}_F(x')_{|\mathbf{l}'+MW+V\times\{0\}} = \mathcal{O}_F(y')_{|\mathbf{l}'+MW+V\times\{0\}}$.

  By injectivity of Φ, $\mathcal{O}_G(\Phi(x'))$ and $\mathcal{O}_G(\Phi(y'))$ are also distinct, but we claim that they coincide in $\mathbf{l} + W$. Since W is an arbitrary finite shape, this implies that $l \in \mathcal{N}(G)$, which proves that $\mathcal{N}(F) \subseteq \frac{1}{T}(Q + S\mathcal{N}(G))$.

  Let $(p_1, p_2) \in \mathbf{l}+W$. Then, $M(p_1, p_2)+V\times\{0\} \subset \mathbf{l}'+MW+V\times\{0\}$; it follows from this that

  $$\mathcal{O}_F(x')_{|(p_1S+p_2Q+V,p_2T)} = \mathcal{O}_F(y')_{|(p_1S+p_2Q+V,p_2T)}.$$

  In addition, we have that

  $$\begin{aligned}
  \mathcal{O}_G(\Phi(x'))_{(p_1,p_2)} &= G^{p_2}\Phi(x')_{p_1} \\
  &= \Phi\sigma^{p_2Q}F^{p_2T}(x')_{p_1} \\
  &= \phi(\sigma^{p_2Q}F^{p_2T}(x')_{|p_2S+V}) \\
  &= \phi(\mathcal{O}_F(x')_{|(p_1S+V+p_2Q,p_2T)})
  \end{aligned}$$

  The same holds for y', and since, as we have noticed earlier, the final expression is the same for x' and y', we get that $\mathcal{O}_G(\Phi(x'))_{|(p_1,p_2)} = \mathcal{O}_G(\Phi(y'))_{|(p_1,p_2)}$, as claimed.

□

Lemmas 17 and 21 can be combined to obtain expansive directions in nested simulations, which will be used extensively in Section 7.

Lemma 22. *If* $F_0 \underset{S_0,T_0,D_0S_0}{\succeq} F_1 \underset{S_1,T_1,D_1S_1}{\succeq} \dots \underset{S_{n-1},T_{n-1},D_{n-1}S_{n-1}}{\succeq} F_n$ *completely exactly, and all these RPCA have bi-radius* 1*, then*

$$\mathcal{N}(F_0) \subseteq \overline{\mathbf{SD}}^{\mathbf{1S/T}} + \left(\prod_{i<n}\frac{S_i}{T_i}\right)[-1,1] = \overline{\mathbf{D}}^{\mathbf{S/T}} + \left(\prod_{i<n}\frac{S_i}{T_i}\right)[-1,1].$$

Proof. We already noted that the radius of a RPCA F_n with bi-radius 1 has $\mathcal{N}(F_n) \subseteq [-1,1]$. From Lemma 17, we know that $F_0 \underset{S,T,Q}{\succeq} F_n$ exactly completely, where $(S,T,Q) = (\prod S_i, \prod T_i, \overline{\mathbf{SD}}^{\mathbf{1S/T}} \prod T_i)$ and from Lemma 21, we deduce that:

$$\mathcal{N}(F_0) = \overline{\mathbf{SD}}^{\mathbf{1S/T}} + \left(\prod_{i<n} \frac{S_i}{T_i}\right) \mathcal{N}(F_n) \subseteq \overline{\mathbf{SD}}^{\mathbf{1S/T}} + \left(\prod_{i<n} \frac{S_i}{T_i}\right) [-1,1] \ .$$

Also, by definition we have that $\overline{\mathbf{SD}}^{\mathbf{1S/T}} = \overline{\mathbf{D}}^{\mathbf{S/T}}$. □

In the limit case of an infinite nested simulation, we obtain the following proposition, which slightly extends Theorem 5.4 in [19].

Proposition 23. *If $F_i \underset{S_i,T_i,D_iS_i}{\succeq} F_{i+1}$ completely exactly, for all $i \in \mathbb{N}$, then*

$$\mathcal{N}(F_0) \subseteq \overline{\mathfrak{D}}^{\mathfrak{S}/\mathfrak{T}} + \left(\inf_{n\in\mathbb{N}} \prod_{i<n} \frac{S_i}{T_i}\right) [-1,1] \ .$$

In particular, if the simulations are non-trivial and $\prod_{i<n} S_i/T_i$ converges to 0, then $\mathcal{N}(F_0) = \{\overline{\mathfrak{D}}^{\mathfrak{S}/\mathfrak{T}}\}$.

Proof. From Lemma 22, we know that

$$\mathcal{N}(F_0) \subseteq \bigcap_{n\in\mathbb{N}} \left(\prod_{i<n} \frac{S_i}{T_i}\right) [-1,1] + \overline{\mathfrak{D}_{[\![0,n[\![}}^{\mathfrak{S}_{[\![0,n[\![}/\mathfrak{T}_{[\![0,n[\![}},$$

which gives the wanted inclusion, when n goes to ∞.

For the second claim, if all the simulations are non-trivial, then from Lemma 18 we know that $\mathcal{O}_{F_0}$ is uncountable, hence by Proposition 9, it has at least one non-expansive direction. In addition, by the first claim and the assumption $\prod_{i<n} S_i/T_i \to 0$, we know that $\mathcal{N}(F_0) \subseteq \{\overline{\mathfrak{D}}^{\mathfrak{S}/\mathfrak{T}}\}$, and we must actually have equality. □

3.4 Explicit simulation

In the previous sections of this chapters, we defined a notion of simulation and then proved some facts about this notion, which suggest that it is a good choice. However, we have not given any non-trivial example of simulation until now, nor have we explained how this could happen. For example, the decoding function Φ could be anything.

The simulation that we construct all have the same basic "form". We call these simulation **explcit**, because the simulated configuration is explicitly

written letter by letter in the simulating configuration. In order to make this more precise, we need to give some more definitions and notations.

Let us fix a some fields $\texttt{Addr}$, $\texttt{Addr}_{+1}$, $\texttt{Clock}$ and $\texttt{Clock}_{+1}$ (In fact, these are just distinct numbers that we use to project letters on). These are sometimes called *coordinate* fields. Let $\Sigma^{s,t,S,T} := \mathcal{P}^{s,S}_{\texttt{Addr},\texttt{Addr}_{+1}} \cap \mathcal{S}^{t}_{\texttt{Clock},\texttt{Clock}_{+1}}$. In $\Sigma^{s,t,S,T}$ the values of $\texttt{Addr}$ grow by 1 modulo S from left to right and the value of $\texttt{Clock}$ is constant and equal to t, while the origin has $\texttt{Addr}$ s. This is the usual way to break up a configuration into blocks, with one small difference. Normally, we only need the fields $\texttt{Addr}$ and $\texttt{Clock}$ to do this. However, since we are using PPA, we need to have some right- (or left-) moving copies of these fields in order to check the compatibility of these fields. Having this in mind, we define $\Sigma^{s,t,S,T}$ in the above way, since it will make notation a little lighter later on. The union $\bigsqcup_{\substack{0\leq t<T\\0\leq s<S}} \Sigma^{s,t,S,T}$ is disjoint.

In addition, let $\Sigma^{s,S} := \mathcal{P}^{s,S}_{\texttt{Addr},\texttt{Addr}_{+1}}$. In $\Sigma^{s,S}$, we do not care about the value of $\texttt{Clock}$ (or if it is even constant). Clearly, $\Sigma^{s,t,S,T} \subseteq \Sigma^{s,S}$. For $c \in \Sigma^{s,S}$ and $i \in \mathbb{Z}$, the pattern

$$B_i^c = c_{[\![-s+iS,-s+(i+1)S[\![}$$

is called a **colony** of c. Clearly, $(B_i^c)_{i\in\mathbb{Z}} = \sigma^{-s}(c)_{[S]}$ and in B_i^c, the value of $\texttt{Addr}$ (and $\texttt{Addr}_{+1}$) grows from 0 to $S-1$ from left to right.

This is the natural way to break a configuration into colonies of size S. Now, we are going to use every colony to encode one letter of the simulated configuration. For this, we have to define the appropriate decoding function.

Let $\tilde{\phi}\colon (\mathbb{F}_5^*)^* \dashrightarrow \mathbb{F}_5^{**}$ be the following function, which is the basis of all the decoding functions that we will use: Let $w \in (\mathbb{F}_5^*)^*$ be a *word* over the infinite alphabet $\mathbb{F}_5^*$ (we look at w as a finite part of some 1D configuration over $\mathbb{F}_5^*$). If $\rangle w \langle = \chi(\mathbf{u})\, 3^{|w|-|\chi(\mathbf{u})|}$, where $\mathbf{u} \in \mathbb{F}_5^{**}$ (we look at $\mathbf{u}$ as a tuple of elements of $\mathbb{F}_5^*$), then we define $\tilde{\phi}(w) = \mathbf{u}$.

Notice that $\chi(\mathbf{u}) \in \mathbb{F}_3^*$. In other words, w is equal to $\chi(\mathbf{u})$ up to appending some 3s at the end of $\chi(\mathbf{u})$ (this gives a word in $\mathbb{F}_4^*$) and then adding some 4s in front of every *letter* of $\chi(\mathbf{u})\, 3^{|w|-|\chi(\mathbf{u})|}$ (which gives a word in $(\mathbb{F}_5^*)^*$). Unless w has this very specific form, $\tilde{\phi}(w)$ is not defined.

$\tilde{\phi}$ is well-defined because $\chi(\cdot)$ is an injection and because 3 does not appear as a letter of $\chi(\mathbf{u}) \in \mathbb{F}_3^*$. A necessary condition so that $\tilde{\phi}(w) = \mathbf{u}$ is that $|w| \geq |\chi(\mathbf{u})|$.

Let $\texttt{Field}$ be a new field and $\tilde{\phi}_{\texttt{Field}}\colon (\mathbb{F}_5^{**})^* \dashrightarrow \mathbb{F}_5^{**}$ be defined as $\tilde{\phi}\pi_{\texttt{Field}}$. $\tilde{\phi}_{\texttt{Field}}$ can read words over letters with many fields by ignoring the other fields and using $\tilde{\phi}$ on $\texttt{Field}$.

We can extend $\tilde{\phi}$ in a natural way to a map $\tilde{\Phi}\colon (\mathbb{F}_5^*)^{\mathbb{Z}} \dashrightarrow (\mathbb{F}_5^{**})^{\mathbb{Z}}$ as follows: for all $c \in (\mathbb{F}_5^*)^{\mathbb{Z}}$ and $i \in \mathbb{Z}$, $\tilde{\Phi}(c)_i = \tilde{\phi}(c_{|[\![iS,(i+1)S[\![}})$. Similarly, $\tilde{\phi}_{\texttt{Field}}$ can be naturally extended to a map $\tilde{\Phi}_{\texttt{Field}}\colon (\mathbb{F}_5^{**})^{\mathbb{Z}} \dashrightarrow (\mathbb{F}_5^{**})^{\mathbb{Z}}$.

The idea is that every configuration will be divided into colonies using the coordinate fields and then $\tilde{\phi}_{\texttt{Field}}$ will be used on every colonies so as to obtain a letter. Putting these letters together, we obtain the simulated configuration.

Formally, a decoding function Φ will be equal to $\tilde{\Phi}_{\texttt{Field}|\Sigma}$, where $\Sigma \subseteq \Sigma^{0,S}$, for some S that is large enough. If $b_i = \tilde{\phi}_{\texttt{Field}}(B_i^c)$, then $\Phi(c) = \tilde{\Phi}_{\texttt{Field}}(c) = (b_i)_{i\in\mathbb{Z}}$. We call b_i the **simulated letter** of the i'th colony and the letters of c are the **simulating letters**.

The decoding functions that we will use in our constructions will *always* be of the form $\tilde{\Phi}_{\texttt{Field}|\Sigma}$, where $\Sigma \subseteq \Sigma^{0,S}$. For such functions, we immediately obtain two of the conditions of a decoding function of a simulation:

Remark 24. *Let us fix a field list* $\mathcal{C} = [\mathtt{Addr}, \mathtt{Addr}_{+1}, \mathtt{Clock}, \mathtt{Clock}_{+1}, \mathtt{Tape}]$, $S \in \mathbb{N}_1$ *and vectors* $\mathbf{k}, \mathbf{k}' \in \mathbb{N}^*$ *such that the following inequalities hold:*

$$\begin{cases} k_{\mathtt{Addr}} \geq \|S\| \\ k_{\mathtt{Tape}} \geq 1 \\ S \geq \left|\chi\left(\mathbb{F}_5^{\mathbf{k}'}\right)\right| \end{cases},$$

Let $\Sigma := (\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}} \cap \Sigma^{0,S} \cap \tilde{\Phi}^{-1}_{\mathtt{Tape}}((\mathbb{F}_5^{\mathbf{k}'})^{\mathbb{Z}})$. *Then* $\Phi := \tilde{\Phi}_{\mathtt{Tape}|\Sigma} : (\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}} \to (\mathbb{F}_5^{\mathbf{k}'})^{\mathbb{Z}}$ *is surjective and* $\Phi\sigma^S = \sigma\Phi$.

In addition, for every $b \in (\mathbb{F}_5^{**})^{\mathbb{Z}}$, *we are free to chose the values of the anonymous fields in any way we like in a pre-image.*

In the above remark, Σ contains those configurations over $\mathbb{F}_5^{\mathbf{k}}$ that are well-structured (*i.e.*, divided into colonies with the origin having address 0) and such that in the i'th colony we have the encoding of a letter of $\mathbb{F}_5^{\mathbf{k}'}$, for all $i \in \mathbb{Z}$.

Chapter 4

The programming language

4.1 Definitions and basic permutations

In our constructions, we want to use permutations that are computed fast. It is not possible to formally state what fast means, but polynomially computable and, more generally, polynomially checkable permutations is fast enough. This is a common feature of all self-similar and hierarchical constructions and the reasons why it is needed are explained very thoroughly in [12]. For our purposes, it is enough to describe a pseudo-programming language, with which we will write "programs" that are interpreted as permutations $\alpha\colon \mathbb{F}_5^{**} \nrightarrow \mathbb{F}_5^{**}$.

Let us start describing this programming language: It has four types, **terms** (that are denoted $t, t' \ldots$), **valuations** (that are denoted $v, v', \ldots$), **conditions** (that are denoted $c, c', \ldots$) and **permutations** (that are denoted $\alpha, \alpha', \ldots$). Each type is semantically interpreted as a different kind of mathematical object. Terms are interpreted as maps $t\colon \mathbb{F}_5^{**} \nrightarrow \mathbb{F}_5^{*}$. They represent some word information that can be extracted from a tuple. Valuations are interpreted as functions $v\colon \mathbb{F}_5^{**} \nrightarrow \mathbb{N}$. Valuations represent numerical information that can be extracted from tuples. Conditions are predicates over $\mathbb{F}_5^{**}$, or equivalently maps $q\colon \mathbb{F}_5^{**} \nrightarrow \{0,1\}$. Finally, permutations are, rather predictably, interpreted as (partial) permutations $\mathbb{F}_5^{**} \nrightarrow \mathbb{F}_5^{**}$ which will be used to define IPPA.

Let us describe each type with more details. We are not going to try to give a formal definition of the programming language, since it is would be unnecessarily complicated. It would involve a global induction on the various types, starting from some basic objects and taking a closure under some inductive operations. Instead, we will simply list the objects that we are actually going to use in the rest of the thesis. The proofs that they are polynomially computable are often trivial and will be omitted in most cases.

Terms

- Every word $w \in \mathbb{F}_5^*$ is a term (understood as the constant function);
- for all $i \in \mathbb{N}$, the projection π_i of the i'th field is a term;
- if t is a term, then $\chi(t)$ is also a term ($\chi(t)(\mathbf{u}) = \chi(t(\mathbf{u}))$, for all $\mathbf{u}$ in $\mathbb{F}_5^{**}$);
- if v is a valuation and t is a term, then $t_{|v}$ is also a term, where $t_{|v}(\mathbf{u}) := t(\mathbf{u})_{|v(\mathbf{u})}$. In other words, $t_{|v}$ uses v as a pointer for t and it gives the letter at the $v(\mathbf{u})$'th position of $t(\mathbf{u})$.

Valuations

- Every natural $n \in \mathbb{N}$ is a valuation, understood as a constant function;
- if t is a term, then $|t|$ is a valuation;
- For all vectors $\mathbf{k} \in \mathbb{N}^*$ and $i \in \mathbb{N}$, the function $l_{\mathbf{k},i}$ defined in Fact 6 is a valuation.
- If $\mathfrak{S}\colon \mathbb{N} \to \mathbb{N}$ is a sequence of numbers and v a valuation, then S_v (where $S_v(\mathbf{u}) := S_{v(\mathbf{u})}$) is also a valuation. (In general, the complexity of this valuation depends on the complexity of $\mathfrak{S}$ and it is not polynomially computable if $\mathfrak{S}$ is not.)
- Basic arithmetical operations (addition, subtraction, multiplication etc) of valuations are still valuations.

In fact, we will need the following, more general version of the third bullet:

- For all valuations v, vector sequences $\mathbf{k}\colon \mathbb{N} \to \mathbb{N}^M$ and $i \in \mathbb{N}$, $l_{\mathbf{k}_v,i}$ (where $l_{\mathbf{k}_v,i}(\mathbf{u}) := l_{\mathbf{k}_{v(\mathbf{u})},i}(\mathbf{u})$) is also a valuation. In this version, the vector whose structure $l_{\mathbf{k}_v,i}$ gives depends on the input letter. Of course, if $\mathbf{k}$ is not a polynomially computable sequence, then neither is $l_{\mathbf{k}_v,i}$.

A **vector valuation** is a collection $\mathbf{v} = (v_i)_{0 \leq i \leq M-1}$ of valuations, for some $M \in \mathbb{N}$. Vector valuations are used to obtain lengths of alphabets in a polynomially computable way.

Conditions

- If v_1, v_2 are valuations, then $v_1 \geq v_2$ is a condition whose interpretation is clear;
- if t_1, t_2 are terms, then $t_1 = t_2$ is a condition;
- if t, t_1 are terms and $(Q_w)_{w \in \mathbb{F}_5^*}$ is a sequence of subsets of $\mathbb{F}_5^*$, then $t_1 \in Q_t$ is a condition. ($\mathbf{u}$ satisfies $t_1 \in Q_t$ if $t_1(\mathbf{u}) \in Q_{t(\mathbf{u})}$.)
- if t is a term and $i_1, \ldots, i_n$ are fields, then $\mathcal{S}^t_{i_1,\ldots,i_n}$ is a condition (that is true for $\mathbf{u}$ if and only if $\mathbf{u} \in \mathcal{S}^{t(\mathbf{u})}_{i_1,\ldots,i_n}$);
- $\mathcal{H}^v_p(t)$ is a condition, where $\mathbf{u}$ satisfies $\mathcal{H}^v_p(t)$ if and only if the TM defined by program p does not stop within $v(\mathbf{u})$ steps over term $t(\mathbf{u})$;
- boolean operations of conditions are also conditions.

Permutations

- For every condition q, $Check[q]$ is a permutation. $Check[q](\mathbf{u})$ is equal to $\mathbf{u}$ if and only if $\mathbf{u}$ satisfies q (and is undefined otherwise). This is an involution.
- For every valuation v and field $i \in \mathbb{N}$, $\mathbf{incr}[v, i]$ is a permutation defined in the following way: Let $\mathbf{u} \in \mathbb{F}_5^{**}$ and define $\mathbf{u}'$ in the following way: $u'_j := u_j$ for all $j \neq i$, and $u'_i := \langle \gamma \rangle_{|u_i|}(u_i)$, where $\gamma(w) := \overline{\underline{w} + 1 \bmod v(\mathbf{u})}$ when $\underline{w} < v(\mathbf{u})$ (undefined otherwise); then $\mathbf{incr}[v; i](\mathbf{u}) := \mathbf{u}'$ if $v(\mathbf{u}) = v(\mathbf{u}')$ (undefined otherwise).

  Essentially $\mathbf{incr}[v; i]$ adds 1 modulo $v(\mathbf{u})$ to the i'th field of $\mathbf{u}$. The additional complications are due to the fact that we want this rule to always be reversible (which would not necessarily be true if $v(\mathbf{u}')$ is not equal to $v(\mathbf{u})$) and length preserving (which is the reason that we use the strange γ function).
- $\alpha_{\mathcal{U}}[t; \texttt{Tape}, \texttt{Head}_{-1}, \texttt{Head}_{+1}]$ is a permutation for every term t and fields $\texttt{Tape}$, $\texttt{Head}_{-1}$, $\texttt{Head}_{+1}$. We direct the reader to Section 5.2 for the definition of this permutation, since it uses a permutation that is defined and examined therein.
- Let t be a term and i be a field such that t **does not depend** on i. In other words, if $\mathbf{u}, \mathbf{u}' \in \mathbb{F}_5^{**}$ and $\pi_j(\mathbf{u}) = \pi_j(\mathbf{u}')$ for all $j \neq i$, then $t(\mathbf{u}) = t(\mathbf{u}')$.

  Then, $Write[t; i]$ is a permutation defined as follows: Let $\mathbf{u} \in \mathbb{F}_5^{**}$. $Write[t; i](\mathbf{u})$ is defined if and only if $\rangle u_i \langle = \epsilon$. In this case, all fields remain the same except for i which becomes equal to $\langle t(\mathbf{u}) \rangle_{|u_i|}$.

Essentially, we check that the field i is empty and then write $t(\mathbf{u})$ on it, while preserving the lengths. The condition that t does not depend on i is essential to ensure reversibility.

$Write[t;i]^{-1}$ first checks that the i'th field is equal to $t(\mathbf{u})$ and then empties it, while preserving the lengths. This is a way to reversibly erase some information from a letter, namely compare it with some other place of the letter where the same information is held.

- For all fields i, i', $Swap[i,i']$ is a permutation defined as follows: Let $\mathbf{u} \in \mathbb{F}_5^{**}$. $Swap[i,i'](\mathbf{u})$ is defined if and only if $|u_i| = |u_{i'}|$. In this case, all fields are unchanged except for i and i' whose values are exchanged.

  This is a length-preserving involution.

- For every condition q and permutation α, **if** q **then** α is a permutation. On input $\mathbf{u} \in \mathbb{F}_5^{**}$, it applies α if condition $q(\mathbf{u})$ is satisfied and $q(\mathbf{u}) = q(\alpha(\mathbf{u}))$. If $q(\mathbf{u})$ is satisfied and $q(\mathbf{u}) \neq q(\alpha(\mathbf{u}))$, then it is not defined on $\mathbf{u}$ (this ensures reversibility). Finally, if $q(\mathbf{u})$ is *not* satisfied, it is equal to the identity.

- The composition of permutations is also a permutation. In constructions, we will denote the composition $\alpha_2 \circ \alpha_1$ by writing α_2 below α_1.

In the definition, we check that the values of the valuations, terms and conditions that are given as parameters do not change. This is a technical point that ensures that they are interpreted as reversible functions. In all our constructions, these conditions will easily be satisfied because the valuations, terms and conditions will either be constant or depend on fields that are not modified by the rule at hand.

If we were giving a complete, formal description of a language, then this would be the point where by a large, tedious induction we would prove that, given some natural conditions on the parameters, every permutation of the language is polynomially computable, or, more precisely, polynomially computable in its parameters (this means that its complexity is a polynomial of the complexity of its parameters) and that short programs exist for the permutations. Namely, the size of the program is $O(p_{t,v,\dots})$, where t, v etc. are the parameters of the permutation.

We can also prove that the size of a program of a permutation is approximately the same as the size of the program of its inverse.

4.2 Conventions about defining IPPA

In the first part of this chapter, we gave a short exposition of the programming language that will be used in the rest of the thesis in order to define

permutations of $\mathbb{F}_5^{**}$. However, in order to define a PPA, the number of fields and the directions of the fields also have to be fixed.

Recall that we want to define PPA, *i.e.*, RPCA of the form $F = \sigma^\delta \circ \alpha$, where $\delta \in \{-1, 0, +1\}^M$ is the shift vector and α is a partial permutation of $\mathcal{A} = \mathcal{A}_0 \times \ldots \mathcal{A}_{M-1}$, for some $M \in \mathbb{N}$. In our case, F will always be the restriction of an IPPA, *i.e.*, $\mathcal{A}$ will be equal to $\mathbb{F}_5^{\mathbf{k}}$, for some $\mathbf{k} \in \mathbb{N}^M$ and $\alpha := \beta_{|\mathbb{F}_5^{\mathbf{k}}}$ will be the restriction of some (infinite) permutation β defined in the programming language.

We will use the following conventions when constructing such PPA:

- We first give a list of so-called **explicit** field labels. Such a list will often be noted in the form $\mathcal{C} := [\texttt{Field}_e, \ldots, \texttt{Field}'_{e'}]$, where $e, e' \in \{-1, 0, +1\}$. The subscripts $e, \ldots, e'$ correspond to the *directions* of the fields (if the direction is equal to 0, then it will be omitted). The field list is a tuple of pairwise different natural projections, that are used by the permutation, together with their directions, that will be used by the shift. (The labels of the fields will make the permutations more understandable than the corresponding indices $i, i', \ldots$). The field list is not fixed, so in fact for every field list, we give a different permutation, even though they only differ in the enumeration of the fields.

  The permutation is assumed to reject any element of $\mathbb{F}_5^{**}$ that does not involve all field numbers in the list, but note that it does not reject tuples that have more fields; the so-called **anonymous** fields, that are not in the list, are not modified by the permutation (but they might be used by some other PPA with which we compose). This allows us to define some simple PPA with few fields and then use them as "building blocks" in order to build more complicated ones in the following sense: the complicated PPA has more fields than the simple one, but, if it does not "touch" any of its fields, its behaviour on those fields is described by the corresponding behaviour of the building block.

  If $\mathcal{C}$ and $\mathcal{C}'$ are two lists of field labels, then $\mathcal{C} \cup \mathcal{C}'$ is the list that contains the fields of $\mathcal{C}$ and $\mathcal{C}'$. Usually, the lists will be disjoint, so that we will use the notation $\mathcal{C} \sqcup \mathcal{C}'$.

- After giving the field list, we describe an (infinite) permutation using the programming language defined in the first part of this chapter.

- Then, we need to fix $M \in \mathbb{N}$ and $\mathbf{k} \in \mathbb{N}^M$. If we do not care about the existence of anonymous fields, then we always assume that M is some number greater than or equal to the largest natural appearing in the field list $\mathcal{C}$. In this way, we ensure that the configurations will not be

rejected simply because the program tries to access a field that is not there.

When we do not want anonymous fields to exist (for example, when we want to achieve exactness of a simulation), then we assume that the field list $\mathcal{C}$ is equal to $[0, \ldots, M-1]$ and we choose this M for the number of fields.

In any case, after choosing M, we fix some vector $\mathbf{k} \in \mathbb{N}^M$ satisfying some appropriate conditions (which are case-specific).

- Finally, we need to define the directions of the fields. However this has already been done in the definition of the field list with the use of the subscripts e, e' etc. The directions of the anonymous fields can be anything. In fact, our statements will be true for *all* directions of the anonymous fields, since we will not refer to them.

Chapter 5

The universal simulator

In this chapter, our aim is to construct an RPCA (a family of RPCA in fact, depending on some parameters) that can simulate every other RPCA that satisfies some conditions. This is done in Lemma 33. This RPCA is extremely helpful and it will be part of all our subsequent constructions. Since it is difficult to overstress the importance of this RPCA, we will give a step-by-step description of its construction with as many details as possible.

In Section 5.1, we will embed a periodic rectangular grid in every configuration. This is a standard procedure in hierarchical constructions and it will allow us to partition every configuration into colonies and use the decoding function $\tilde{\Phi}$. In Section 5.2, we will make a slight digression and show how we can simulate any TM with an RPCA in real-time. This is needed in order to preserve the expansiveness of the horizontal direction. Then, in Section 5.3, we construct an RPCA to simulate an RPCA whose direction vectors are null (all its fields are still). There are some tricks involved in this phase, mainly having to do with deleting the previous simulated letter and synchronizing the computations. Then, in Section 5.4, we construct an RPCA that can simulate any RPCA whose permutation is the identity *i.e.*, any shift. Finally, in Section 5.5, we construct the universal IPPA *Simulate* that can simulate any RPCA, when it is restricted to the appropriate alphabet.

5.1 Imposing a periodic structure

Let $\mathcal{C}_{Grid} = [\texttt{Addr}, \texttt{Addr}_{+1}, \texttt{Clock}, \texttt{Clock}_{+1}]$.

- `Clock` and `Addr` are meant to localize the cell in its macrocell, and they correspond to the projections involved in the definition of explicit simulation in Section 3.4.

- $\texttt{Clock}_{+1}$ and $\texttt{Addr}_{+1}$ are used to communicate with the neighbour cells, so that consistency between the `Clock` and `Addr` fields is achieved.

$Grid[v_{\texttt{MAddr}}, v_{\texttt{MClock}}]$

1: $Check[\underline{\pi_{\texttt{Addr}_{+1}}} = \underline{\pi_{\texttt{Addr}}}$ **and** $\underline{\pi_{\texttt{Clock}_{+1}}} = \underline{\pi_{\texttt{Clock}}}]$ {Check left-neighbour information coherence.}
2: **incr**$[v_{\texttt{MAddr}}; \texttt{Addr}_{+1}]$ {Increment $\texttt{Addr}_{+1}$ so that the right neighbour can check coherence.}
3: **incr**$[v_{\texttt{MClock}}; \texttt{Clock}]$ {Update `Clock`.}
4: **incr**$[v_{\texttt{MClock}}; \texttt{Clock}_{+1}]$ {Update $\texttt{Clock}_{+1}$.}

By the discussion of Chapter 4, we know that $Grid[v_{\texttt{MAddr}}, v_{\texttt{MClock}}; \mathcal{C}_{Grid}]$ is polynomially computable with respect to its parameters $v_{\texttt{MAddr}}$ and $v_{\texttt{MClock}}$.

In practice, the two valuation parameters $v_{\texttt{MAddr}}$ and $v_{\texttt{MClock}}$ will be constant over the alphabet of the PPA, in which case the behaviour will be described by the following:

Lemma 25. *Let us fix a field list $\mathcal{C}_{Grid} \in \mathbb{N}^4$ and integers $S, T \in \mathbb{N}_1$.*
Let F be the IPPA defined by the permutation $Grid[S, T; \mathcal{C}_{Grid}]$ and directions ν_{Grid} given by the label indices, and let $\mathbf{k} \in \mathbb{N}^*$ be a vector satisfying:

$$\begin{cases} k_{Addr}, k_{Addr_{+1}} \geq \|S\| \\ k_{Clock}, k_{Clock_{+1}} \geq \|T\| \end{cases}.$$

Let $c \in (\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}}$. Then, $c \in F^{-2}((\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}})$ if and only if there exist $s \in [\![0, S[\![$ and $t \in [\![0, T[\![$ such that $c \in \Sigma^{s,t,S,T}$. In this case, $F(c) \in \Sigma^{s,t+1 \bmod T,S,T}$.

In the previous statement, S and T should be understood as the **width** and **height** of the macrocells. Notice, also, that the statement holds for all vectors $\mathbf{k} \in \mathbb{N}^*$ that satisfy the inequalities, which means that there can be other fields in the alphabet. This means that if we use *Grid* together with other rules that do not change the values of the fields in $\mathcal{C}_{Grid}$, the statement of the lemma will still be true.

The restrictions about the lengths of $\mathbf{k}$ ensure that fields are large enough that we can write the binary representation of S and T on them.

Proof. We prove the stronger claim that if $F^2(c)$ exists, then there exist $0 \leq s < S$ and $0 \leq t < T$ such that for all $n \in \mathbb{Z}$, $\underline{\pi_{\texttt{Addr}}(c_n)} = \underline{\pi_{\texttt{Addr}_{+1}}(c_n)} = s + n \bmod S$ and $\underline{\pi_{\texttt{Clock}}(c_n)} = \underline{\pi_{\texttt{Clock}_{+1}}(c_n)} = t$.

Suppose that $\underline{\pi_{\texttt{Addr}}(c_n)} \neq \underline{\pi_{\texttt{Addr}_{+1}}(c_n)}$ or $\underline{\pi_{\texttt{Clock}}(c_n)} \neq \underline{\pi_{\texttt{Clock}_{+1}}(c_n)}$, for some $n \in \mathbb{Z}$. Then, line 1 would not be defined at cell n, $\underline{F(c)}$ would not exist, which is a contradiction.

Suppose, then, that there exists $n \in \mathbb{Z}$ with $\underline{\pi_{\texttt{Addr}}(c_{n+1})} \neq \underline{\pi_{\texttt{Addr}}(c_n)} + 1 \bmod S$. Line 2 and the fact that $\texttt{Addr}_{+1}$ is a right-going field imply that $\underline{\pi_{\texttt{Addr}_{+1}} F(c)_{n+1}} = \underline{\pi_{\texttt{Addr}}(c_n)} + 1 \mod S$. Then, line 1 is not defined at cell $n+1$ of $F(c)$ since $\underline{\pi_{\texttt{Addr}_{+1}} F(c)_{n+1}} = \underline{\pi_{\texttt{Addr}}(c_n)} + 1 \mod S \neq \underline{\pi_{\texttt{Addr}}(c_{n+1})}$. Therefore $F^2(c)$ does not exist, which contradicts the hypothesis. Similarly, we can prove that $\underline{c_n.\texttt{Clock}} = \underline{c_{n+1}.\texttt{Clock}}$, for all $n \in \mathbb{Z}$. Thus, the stronger claim we made at the beginning of the proof is true.

If $\underline{\pi_{\texttt{Addr}}(c_0)} = s$ and $\underline{\pi_{\texttt{Clock}}(c_0)} = t$, then the previous claim implies that for all $n \in \mathbb{Z}$, $\underline{\pi_{\texttt{Addr}}(c_n)} = s+n \bmod S$ and $\underline{\pi_{\texttt{Clock}}(c_n)} = t$. Furthermore, since the value of `Addr` is not changed by F and the value of `Clock` is increased by $1 \mod T$ every time step by line 3, we have that $\underline{\pi_{\texttt{Addr}} F(c)_n} = s + n \bmod S$ and $\underline{\pi_{\texttt{Clock}} F(c)_n} = t + 1 \bmod T$, for all $n \in \mathbb{Z}$.

□

In general, when using IPPA, we have to use a similar rule every time we want to impose some horizontal restriction on the configuration. Namely, we have to use an additional right-moving (or left-moving, it does not make a difference) field, and then we need 2 steps in order to verify that the field is constant.

All of the rules we construct will factor onto $Grid[S, T; \mathcal{C}_{Grid}]$, for some $S, T \in \mathbb{N}_1$. The following remark will give the disjointness condition in the definition of simulation.

Remark 26. *Assume that $F\colon \mathcal{A}^{\mathbb{Z}} \nrightarrow \mathcal{A}^{\mathbb{Z}}$ factors onto $Grid[S, T; \mathcal{C}_{Grid}]$ through the factor map H and let $\Sigma_F^{s,t} := H^{-1}(\Sigma^{s,t,S,T})$. Then, the union $\bigsqcup_{\substack{0 \leq t < T \\ 0 \leq s < S}} \Sigma_F^{s,t}$ is disjoint and $F(\Sigma_F^{s,t}) \subseteq \Sigma_F^{s,t+1 \bmod T}$.*

Therefore, if $\Phi\colon \mathcal{A}^{\mathbb{Z}} \nrightarrow \mathcal{B}^{\mathbb{Z}}$ satisfies that $\mathcal{D}(\Phi) \subseteq \Sigma_F^{0,0}$, for some s, t, then the union $\bigsqcup_{\substack{0 \leq t < T \\ 0 \leq s < S}} F^t \sigma^s(\mathcal{D}(\Phi))$ is disjoint.

$\Sigma_F^{s,t}$ implicitly depends on the factor map H. However, in applications, H will be equal to $\pi_{\mathcal{C}_{Grid}}$ so that no ambiguity arises by omitting it.

5.2 Simulating TM with IPPA

The IPPA *Grid* allows us to divide every configuration into colonies with a periodical clock. We want to use this space-time structure in order to do computations within the **work-periods** (the "time" between two subsequent steps where the clock is 0). We are going to introduce the elements needed for this one by one, since, hopefully, it will make some of the ideas more clear. First, let us show how to simulate TMs in real time with PPA.

For all programs $p = p_{\mathcal{M}} \in \mathbb{F}_4^*$, we construct an IPPA that simulates $\mathcal{M}$ in real-time. This subsection is inspired by [32].

Let $\mathcal{C}_{\mathcal{U}} := [\texttt{Tape}, \texttt{Head}_{-1}, \texttt{Head}_{+1}]$.

The key item to maintaining reversibility, is to keep track of the history of the computation. Some kind of *archive* of each past step is shifted in the direction opposite to the head, in order for the head to always have space to write the new history. Recall the definition of the function $\mathcal{U}$ from Subsection 2.2.1. Let $\gamma_{\mathcal{U}}[p] \colon (\mathbb{F}_4^*)^3 \nrightarrow (\mathbb{F}_4^*)^3$ be defined by the following transitions: (a, h_{-1}, h_{+1}) is mapped to

- $(a', \chi(a, q, \delta), \chi(a, q, \delta))$, if $(h_\delta, h_{-\delta}) = (q, \epsilon)$ ($\texttt{Head}_\delta$ contains the TM head) and $\mathcal{U}(a, q, p) = (a', \epsilon, +1)$. If $\texttt{Head}_\delta$ contains a head and the transition is an accepting one, then we write an encoding of the last transition on the `Head` fields, modify `Tape` and the TM heads disappear. Here, the assumption that the TM head (which has disappeared) moves to the right is convenient to ensure injectivity.

- (a', h'_{-1}, h'_{+1}), where $(h'_{\delta'}, h'_{-\delta'}) = (q', \chi(a, q, \delta))$ if $(h_\delta, h_{-\delta}) = (q, \epsilon)$ and $\mathcal{U}(a, q, p) = (a', \epsilon, \delta')$. If the transition is not an accepting one, then `Tape` is modified, the TM head is written on the appropriate `Head` field and on the other `Head` field we write an encoding of the transition and of the position of the head before the transition.

- (a, h_{-1}, h_{+1}) if $h_{-1}, h_{+1} \notin Q_p \setminus \{\epsilon\}$. If none of the `Head` fields contains a TM head, then do nothing.

It is not difficult (by a tedious case enumeration) to see that $\gamma_{\mathcal{U}}[p]$ is a partial permutation and *polynomially computable*, thanks in particular to the disjointness of $Q_p \subset \mathbb{F}_2^*$ and $\chi(\mathbb{F}_4 \times Q_p \times \{-1, 1\}) \subset 2\mathbb{F}_3^*$. Basically, $\gamma_{\mathcal{U}}[p]$ identifies the accepting state ϵ with the absence of state (for which it just performs identity). In other cases, it prevents from having two (non-accepting) head states at the same cell; then it applies the transition rule and sends the new state to the correct direction (depending on δ), while sending an archive of the last performed operation in the opposite direction. At the moment that the accepting state appears, it just sends two (identical) archives in opposite directions (there is no new state to send).

We say that $c \in (\mathbb{F}_4^{**})^{\mathbb{Z}}$ **represents** $(z, q, j) \in \mathbb{F}_4^{\mathbb{Z}} \times Q \times \mathbb{Z}$ of the machine $\mathcal{M}$ corresponding to program p if:

- For all $i \in \mathbb{Z}$, $\pi_{\texttt{Tape}}(c_i) = z_i$;

- $(\pi_{\texttt{Head}_{-1}}(c_j), \pi_{\texttt{Head}_{+1}}(c_j)) \in \{q\} \times \{\epsilon\} \cup \{\epsilon\} \times \{q\}$;

- For all $i \neq j$ and $\delta \in \{-1, +1\}$, $\pi_{\texttt{Head}_\delta}(c_i) \notin Q_p \setminus \{\epsilon\}$

- For all $i \neq j$ and $\delta \in \{-1, +1\}$, if $\pi_{\texttt{Head}_\delta}(c_i) \neq \epsilon$ then δ has the sign of $j - i$.

Intuitively, this means that in c there is at most one (non-accepting) head at position j, no head elsewhere, and nothing (represented by ϵ, like the accepting state) in $\texttt{Head}_{-1}$ on its right nor in $\texttt{Head}_{+1}$ on its left. The possible archives go away from the head position. We can thus see that the head will never move into a cell where there is an archive, so that one of the transitions of $\gamma_{\mathcal{U}}[p]$ will always be applicable.

Formally, we have the following lemma about the behaviour of $\gamma_{\mathcal{U}}[p]$.

Lemma 27. *Let us fix a field list $\mathcal{C}_{\mathcal{U}} \in \mathbb{N}^3$ and a program $p = p_{\mathcal{M}} \in \mathbb{F}_4^*$. Consider the IPPA F defined by permutation $\langle \gamma_{\mathcal{U}}[p] \rangle$ and directions $\nu_{\mathcal{U}}$ given by the label indices.*
Let $\mathbf{k} \in \mathbb{N}^*$ be a vector satisfying:

$$\begin{cases} k_{\mathit{Head}_{-1}}, k_{\mathit{Head}_{+1}} \geq \|\chi\left(\mathbb{F}_4 \times Q_p \times \{-1, +1\}\right)\| \\ k_{\mathit{Tape}} \geq 1 \ . \end{cases}$$

Let $c \in (\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}}$ and suppose that $\rangle c \langle$ represents configuration $(z, q, j) \in \mathbb{F}_4^{\mathbb{Z}} \times Q \times \mathbb{Z}$ of $\mathcal{M}$.
Then, for all $t \in \mathbb{N}$, $\big\rangle F^t(c) \big\langle$ represents $\mathcal{M}^t(z, q, j)$.

As in Lemma 25, the inequalities about the lengths of $\mathbf{k}$ simply state that the fields are long enough. Using Lemma 7, we will omit the $\langle\cdot\rangle$ and $\rangle\cdot\langle$ from $\langle \gamma_{\mathcal{U}}[p] \rangle$ and $\rangle c \langle$ in the following proof, since they are only used to make F have constant lengths.

Proof. We will prove the claim for $t = 1$; the general claim then follows by induction.

Suppose, first, that $\mathcal{M}(z, q, j)$ does not exist. This means that $\delta_{\mathcal{M}}(z_j, q)$ does not exist, or equivalently, that $\mathcal{U}(z_j, q, p) = \mathcal{U}(c_j.\texttt{Tape}, q, p)$ does not exist. From the definition of $\gamma_{\mathcal{U}}[p]$ and the fact that c represents (z, q, j), we have that $\gamma_{\mathcal{U}}[p](c_j)$ does not exist, which implies that $F(c)$ does not exist.

Suppose, then, that $\mathcal{M}(z, q, j) = (z', q', j')$ exists. This means that $\mathcal{U}(z_j, q, p) = (z'_j, q', j' - j)$, and $z'_i = z_i$ for any $i \neq j$. By assumption, for any $i \in \mathbb{Z}$, $(\pi_{\texttt{Tape}}(c_i), \pi_{\texttt{Head}_{-1}}(c_i), \pi_{\texttt{Head}_{+1}}(c_i)) = (z_i, h_{-1,i}, h_{+1,i})$ for some $h_{-1,i}, h_{+1,i} \in Q_p \cup \chi\left(\mathbb{F}_4 \times Q_p \times \{-1, +1\}\right) \cup \{\epsilon\}$.

Moreover, for any $i \neq j$, and $\delta \in \{-1, +1\}$, $h_{\delta,i} \notin Q_p$, so the identity rule is applied. After applying the shifts, it gives that for any $i < j - 1$,

$$\begin{aligned}(\pi_{\texttt{Tape}} F(c_i), \pi_{\texttt{Head}_{-1}} F(c_i), \pi_{\texttt{Head}_{+1}} F(c_i)) &= (z_i, h_{-1,i+1}, h_{+1,i-1}) \\ &= (z'_i, h_{-1,i+1}, \epsilon)\end{aligned}$$

with $h_{-1,i+1} \notin Q_p$, and for any $i > j + 1$,

$$\begin{aligned}(\pi_{\texttt{Tape}} F(c_i), \pi_{\texttt{Head}_{-1}} F(c_i), \pi_{\texttt{Head}_{+1}} F(c_i)) &= (z_i, h_{-1,i+1}, h_{+1,i-1}) \\ &= (z'_i, \epsilon, h_{+1,i-1})\end{aligned}$$

with $h_{+1,i-1} \notin Q$.

Now, assume $(h_{-1,i}, h_{+1,i}) = (q, \epsilon)$ and that $\mathcal{U}(z_j, q, p) = (z'_j, q', -1)$ (the other cases can be dealt with in a similar way). Then the transition $(z_j, q, \epsilon) \to (z'_j, q', \chi(z_j, q, -1))$ is applied by $\gamma_{\mathcal{U}}[p]$. After the application of $\gamma_{\mathcal{U}}[p]$ and the shifts, we obtain

$$\begin{aligned}
(\pi_{\mathtt{Tape}} F(c_j), \pi_{\mathtt{Head}_{-1}} F(c_j), \pi_{\mathtt{Head}_{+1}} F_{\mathcal{U}}[p](c_j)) &= (z'_j, \epsilon, \epsilon), \\
(\pi_{\mathtt{Tape}} F(c_{j-1}), \pi_{\mathtt{Head}_{-1}} F(c_{j-1}), \pi_{\mathtt{Head}_{+1}} F(c_{j-1})) &= (z'_{j-1}, q', \epsilon) \text{ and,} \\
(\pi_{\mathtt{Tape}} F(c_{j+1}), \pi_{\mathtt{Head}_{-1}} F(c_{j+1}), \pi_{\mathtt{Head}_{+1}} F(c_{j+1})) &= (z'_{j+1}, \epsilon, \chi(z_j, q, -1)).
\end{aligned}$$

All conditions are hence satisfied for $F(c)$ to represent $\mathcal{M}(z, q, j)$. □

Note that due to the parallel nature of IPPA, some configurations may involve several machine heads, and valid simulations may take place in parallel, provided that there is enough space between them so that the archive and the heads do not collide. For this reason, we need to give a "finite version" of the previous lemma.

Lemma 28. *Let us fix a field list $\mathcal{C}_{\mathcal{U}} \in \mathbb{N}^3$ and a program $p = p_{\mathcal{M}} \in \mathbb{F}_4^*$. Consider the IPPA F defined by permutation $\langle \gamma_{\mathcal{U}}[p] \rangle$ and directions $\nu_{\mathcal{U}}$ given by the label indices. Let $\mathbf{k} \in \mathbb{N}^*$ be a vector satisfying:*

$$\begin{aligned}
&k_{\mathit{Head}_{-1}}, k_{\mathit{Head}_{+1}} \geq \|\chi(\mathbb{F}_4 \times Q_p \times \{-1, +1\})\| \\
&k_{\mathit{Tape}} \geq 1.
\end{aligned}$$

Let $c \in (\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}}$, $c' = \rangle c \langle$ and assume that there exists $n \in \mathbb{N}$ such that the set

$$J := \left\{ j \in \mathbb{Z} \middle| \, (\pi_{\mathit{Head}_{-1}}(c'_j), \pi_{\mathit{Head}_{+1}}(c'_j)) \neq (\epsilon, \epsilon) \right\}$$

satisfies that for any $j \neq j' \in J$, we have $|j' - j| > 2n$, and that for all $j \in J$, $(c'_j.\mathit{Head}_{-1}, c'_j.\mathit{Head}_{+1}) \in Q_p \times \{\epsilon\} \cup \{\epsilon\} \times Q_p$. For $j \in J$, let $q_j := c'_j.\mathit{Head}_{-1}$ if $(c'_j.\mathit{Head}_{-1}, c'_j.\mathit{Head}_{+1}) \in Q_p \times \{\epsilon\}$ and $q_j := c'_j.\mathit{Head}_{+1}$ if $(c'_j.\mathit{Head}_{-1}, c'_j.\mathit{Head}_{+1}) \in \{\epsilon\} \times Q_p$.

Then, $F^n(c)$ exists if and only if $(z^j, q'_j, j') := \mathcal{M}^n(c'.\mathit{Tape}, q_j, j)$ exists, for all $j \in J$. In addition:

- *$\pi_{\mathit{Tape}} F^n(c)_{[\![j-n, j+n]\!]} = z^j_{[\![j-n, j+n]\!]}$, for all $j \in J$;*
- *$\pi_{\mathit{Tape}} F^n(c)_i = c_i$ if $i \notin J + [\![-n, n]\!]$;*
- *$(\pi_{\mathit{Head}_{-1}} F^n(c)_{j'}, \pi_{\mathit{Head}_{+1}} F^n(c)_{j'}) \in \{(q'_j, \epsilon)\} \cup \{(\epsilon, q'_j\}$, for all $j \in J$;*
- *$(\pi_{\mathit{Head}_{-1}} F^n(c)_i, \pi_{\mathit{Head}_{+1}} F^n(c)_i) \notin Q_p \times \{\epsilon\} \cup \{\epsilon\} \times Q_p$, if $i \notin \{j' | \, j \in J\}$*

Proof. First note that the identity is always applied when the head is absent; in particular it is applied outside $J + [\![-t, t]\!]$ at time $t \in \mathbb{N}$ (because F has radius 1) and initially the heads are only in the positions in J.

According to the assumptions, for all $j \in J$, the configuration c'^j obtained by turning all $(c_i.\mathtt{Head}_{-1}, c_i.\mathtt{Head}_{+1})$ to (ϵ, ϵ) except at position j represents $(c'.\mathtt{Tape}, q_j, j)$. Thanks to Lemma 27, for all $0 \leq t \leq n$, $F^t(c'^j)$ exists if and only if $\mathcal{M}^n(c'.\mathtt{Tape}, q_j, j)$ exists.

In that case, since c'^j coincides with c' over interval $[\![j - 2n, j + 2n]\!]$ and since the radius is 1, a simple induction can show that $F^t(c'^j)$ coincides with $F^t(c')$ over interval $[\![j - 2n + t, j + 2n - t]\!]$. Lemma 27 hence gives the main claim.

Conversely, suppose that $F^t(c'^j)$ is undefined for some $j \in J$ with $t \leq n$ minimal. Then, $F^{t-1}(c'^j)$ exists, and by Lemma 27 involves a unique (non-accepting) head, in some cell $j' \in [\![j - t, j + t]\!]$. Therefore, $\gamma_{\mathcal{U}}[p](F^{t-1}(c'^j)_i$ is defined for any $i \neq j'$. This means that $\gamma_{\mathcal{U}}[p](F^{t-1}(c'^j)_{j'})$ is undefined; we have already noted that this is equal to $\gamma_{\mathcal{U}}[p](F^{t-1}(c')_{j'})$, which proves that $F^t(c')$ is undefined. □

Lemma 28 will be used in the following way: Every configuration will be divided into colonies by *Grid*. Initially (when the clock is equal to 0), inside every colony there will be exactly one TM head at the leftmost cell of the colony. These TM will perform some computation for a small amount of time compared to the the width of the colonies (the S of Lemma 25) so that the heads will not meet. Lemma 28 will immediately imply that at the end of the computation, in every colony, `Tape` contains the output of the computation. Finally, the output of the computation will be copied onto some new field and then the computation will be run backwards (remember that $\gamma_{\mathcal{U}}[p]$ is a permutation).

We are now ready to give the details of the definition of the permutation $\alpha_{\mathcal{U}}[t; \mathtt{Tape}, \mathtt{Head}_{-1}, \mathtt{Head}_{+1}]$: Let $\mathbf{u} \in \mathbb{F}_5^{**}$ and define $\mathbf{u}'$ in the following way:

$$(u'.\mathtt{Tape}, u'.\mathtt{Head}_{-1}, u'.\mathtt{Head}_{+1}) = \langle \gamma_{\mathcal{U}}[t(\mathbf{u})] \rangle (u.\mathtt{Tape}, u.\mathtt{Head}_{-1}, u.\mathtt{Head}_{+1}),$$

and $u'_j := u_j$ for all $j \notin \{\mathtt{Tape}, \mathtt{Head}_{-1}, \mathtt{Head}_{+1}\}$. Then,

$$\alpha_{\mathcal{U}}[t; \mathtt{Tape}, \mathtt{Head}_{-1}, \mathtt{Head}_{+1}](\mathbf{u}) := \mathbf{u}',$$

if $t(\mathbf{u}) = t(\mathbf{u}')$ (and it is undefined otherwise.).

Again, the definition gets a little more complicated due to the need to preserve the lengths, to have arbitrarily many fields and to ensure reversibility. When $t = \pi_{\mathtt{Prog}}$, where `Prog` is a new field, then the condition $t(\mathbf{u}) = t(\mathbf{u}')$ is always satisfied.

5.3 Computing the simulated permutation

Let $\mathcal{C}_{Compute} = \mathcal{C}_{\mathcal{U}} \sqcup [\texttt{NTape}]$.

- $\texttt{Head}_{-1}, \texttt{Head}_{+1}$ are used by to simulate a TM with the rule of Subsection 5.2.
- The output of this computation is written on NTape and then the computation is reversed (the Bennett trick, see [3]).

$Compute[v_{\texttt{Addr}}, v_{\texttt{Clock}}, v_{\texttt{Alarm}}, t_{\texttt{Prog}}, t_{\texttt{RProg}}]$

1: **if** $v_{\texttt{Clock}} = 0$ **and** $v_{\texttt{Addr}} = 0$ **then**
2: $Write[0; \texttt{Head}_{-1}]$ {Write the machine initial state in the left head field.}
3: **end if**
4: **if** $0 \leq v_{\texttt{Clock}} < v_{\texttt{Alarm}}$ **then**
5: $\langle \gamma_{\mathcal{U}}[t_{\texttt{Prog}}; \texttt{Tape}, \texttt{Head}_{-1}, \texttt{Head}_{+1}] \rangle$ {Run the machine in order to compute the permutation.}
6: **else if** $v_{\texttt{Clock}} = v_{\texttt{Alarm}}$ **then**
7: $Check[\pi_{\texttt{Head}_{-1}}, \pi_{\texttt{Head}_{+1}} \notin Q_{t_{\texttt{Prog}}} \setminus \{\epsilon\}]$ {Check that the computation halted.}
8: $Write[\texttt{Tape}; \texttt{NTape}]$ {Copy the output onto a different tape.}
9: $Swap[\texttt{Head}_{-1}, \texttt{Head}_{+1}]$ {The directions of the fields of $F_{\mathcal{U}}[p]^{-1}$ are opposite to those of $F_{\mathcal{U}}[p]$.}
10: **else if** $v_{\texttt{Alarm}} < v_{\texttt{Clock}} \leq 2v_{\texttt{Alarm}}$ **then**
11: $\langle \gamma_{\mathcal{U}}[t_{\texttt{Prog}}; \texttt{Tape}, \texttt{Head}_{+1}, \texttt{Head}_{-1}]^{-1} \rangle$ {Unwind the computation in order to delete the archive.}
12: **end if**
13: **if** $2v_{\texttt{Alarm}} \leq v_{\texttt{Clock}} < 3v_{\texttt{Alarm}}$ **then**
14: $\langle \gamma_{\mathcal{U}}[t_{\texttt{RProg}}; \texttt{NTape}, \texttt{Head}_{-1}, \texttt{Head}_{+1}] \rangle$ {Compute the inverse of the permutation, in order to recover Tape.}
15: **else if** $v_{\texttt{Clock}} = 3v_{\texttt{Alarm}}$ **then**
16: $Check[\pi_{\texttt{Head}_{-1}}, \pi_{\texttt{Head}_{+1}} \notin Q_{t_{\texttt{RProg}}} \setminus \{\epsilon\}]$ {Check that the computation halted.}
17: $Write[\texttt{Tape}; \texttt{NTape}]^{-1}$ {Empty NTape.}
18: $Swap[\texttt{Head}_{-1}, \texttt{Head}_{+1}]$ {Reverse the directions again.}
19: **else if** $3v_{\texttt{Alarm}} < v_{\texttt{Clock}} \leq 4v_{\texttt{Alarm}}$ **then**
20: $\langle \gamma_{\mathcal{U}}[t_{\texttt{RProg}}; \texttt{Tape}, \texttt{Head}_{+1}, \texttt{Head}_{-1}]^{-1} \rangle$ {Unwind the second computation, too.}
21: **end if**
22: **if** $v_{\texttt{Clock}} = 4v_{\texttt{Alarm}}$ **and** $v_{\texttt{Addr}} = 0$ **then**
23: $Write[0; \texttt{Head}_{-1}]^{-1}$ {Erase the machine initial state.}

24: **end if**

$Compute[v_{\texttt{Addr}}, v_{\texttt{Clock}}, v_{\texttt{Alarm}}, v_{\texttt{Prog}}, v_{\texttt{RProg}}; \mathcal{C}_{Compute}]$ is polynomially computable with respect to its parameters.

Note that, depending on the value of $v_{\texttt{Addr}}$, only a small amount of these permutation are applied.

In applications, the three parameters $v_{\texttt{Alarm}}, t_{\texttt{Prog}}, t_{\texttt{RProg}}$ will be constant. $v_{\texttt{Alarm}}$ contains a natural number that controls how long the computation lasts. $t_{\texttt{Prog}}$ and $t_{\texttt{RProg}}$ are interpreted as the program and the reverse program (*i.e.*, the program of the inverse IPPA) of the IPPA that we want to simulate.

We are now able to simulate uniformly RPCA with "radius 0" (null direction vector).

Lemma 29. *Let us fix a field list $\mathcal{C}_{Grid} \sqcup \mathcal{C}_{Compute} \in \mathbb{N}^8$, vectors $\mathbf{k}, \mathbf{k}' \in \mathbb{N}^*$, integers $S, T \in \mathbb{N}_1, t_0, U \in \mathbb{N}$ and programs $p, p^{-1} \in \mathbb{F}_4^*$ of a partial permutation $\alpha : \mathbb{F}_5^{**} \nrightarrow \mathbb{F}_5^{**}$ and its inverse α^{-1}, respectively and let G the IPPA corresponding to permutation α and null direction vector.*

Consider the IPPA F with directions $\nu_{Grid \cup Compute}$, and permutation

$$Grid[S,T] \circ Compute[\underline{\pi_{\texttt{Addr}}}, \underline{\pi_{\texttt{Clock}}} - t_0, U, p, p^{-1}] \ ,$$

and assume that the following inequalities hold:

$$\begin{cases} U \geq \max\{t_p(\mathbb{F}_5^{\mathbf{k}'}), t_{p^{-1}}(\mathbb{F}_5^{\mathbf{k}'})\} \\ S \geq \max\{2t_0, \left\|\chi\left(\mathbb{F}_5^{\mathbf{k}'}\right)\right\|\} \\ T \geq 4U + t_0 \\ k_{\texttt{Addr}}, k_{\texttt{Addr}_{+1}} \geq \|S\| \\ k_{\texttt{Clock}}, k_{\texttt{Clock}_{+1}} \geq \|T\| \\ k_{\texttt{Head}_{-1}}, k_{\texttt{Head}_{+1}} \geq \left|\chi\left(\mathbb{F}_4 \times (Q_p \cup Q_{p^{-1}}) \times \{-1, +1\}\right)\right| \\ k_{\texttt{Tape}}, k_{\texttt{NTape}} \geq 1. \end{cases}$$

Then, $F_{|(\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}}} \underset{S,T,0,\Phi}{\succeq} G_{|(\mathbb{F}_5^{\mathbf{k}'})^{\mathbb{Z}}}$, where $\Phi := \tilde{\Phi}_{\texttt{Tape}|\Sigma}$ and

$$\Sigma := (\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}} \cap \Sigma^{0,0,S,T} \cap \mathcal{S}^{\epsilon}_{\texttt{NTape},\texttt{Head}_{-1},\texttt{Head}_{+1}} \cap \tilde{\Phi}^{-1}_{\texttt{Tape}}((\mathbb{F}_5^{\mathbf{k}'})^{\mathbb{Z}}).$$

The number t_0 should be understood as a delay before which to apply this rule, and U as the maximal time that we allow to the (forward and backward) computation.

Proof. It follows from Remarks 24 and 26 that Φ is surjective, $\Phi\sigma^S = \sigma\Phi$ and that $\mathcal{D}(\Phi) := \bigsqcup_{\substack{0 \leq t < T \\ 0 \leq s < S}} F^t\sigma^s(\mathcal{D}(\Phi))$ is a disjoint union.

Therefore, in order to prove the simulation we only have to prove that $G\Phi = \Phi F^T$. This is equivalent (since we are talking about partial functions) to the facts that if $\Phi(c) = b$, then $F^T(c)$ exists if and only if $G(b)$ exists, and in that case $\Phi F^T(c) = G(b)$.

We are actually going to prove the following stronger:

Fact 30. *If* $c \in (\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}} \cap \Sigma^{0,t_0,S,T} \cap \mathcal{S}^{\epsilon}_{NTape,Head_{-1},Head_{+1}} \cap \tilde{\Phi}^{-1}_{Tape}(b)$*, then* $F^{4U}(c)$ *exists if and only if* $G(b)$ *exists, and in that case* $F^{4U}(c) \in \Sigma^{0,t_0+4U,S,T} \cap \mathcal{S}^{\epsilon}_{NTape,Head_{-1},Head_{+1}} \cap \tilde{\Phi}^{-1}_{Tape}(G(b))$.

Since the only rule applied outside $t_0 \leq \texttt{Clock} \leq t_0 + 4U$ is $Grid$, Fact 30 implies that if $\Phi(c) = b$, then $\Phi F^T(c) = G(b)$, which concludes the proof of the lemma.

For the rest of the proof, let $c \in (\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}} \cap \Sigma^{0,t_0,S,T} \cap \mathcal{S}^{\epsilon}_{\texttt{NTape},\texttt{Head}_{-1},\texttt{Head}_{+1}} \cap \tilde{\Phi}^{-1}_{\texttt{Tape}}(b)$.

Suppose, first, that $F^{4U}(c)$ exists.

- $\texttt{Clock} = t_0$: Initially, $c \in \mathcal{S}^{\epsilon}_{\texttt{NTape},\texttt{Head}_{-1},\texttt{Head}_{+1}}$. Line 2 writes the initial head of the TM on $\texttt{Head}_{-1}$ of the leftmost cell of every colony.

- $t_0 \leq \texttt{Clock} < t_0 + U$: Only the permutation of line 5 is applied. Together with the directions of $\nu_{Compute}$, this implies that we apply U steps of the IPPA $F_{\mathcal{U}}[p]$ on configuration $(\pi_{\texttt{Tape}}(c), \pi_{\texttt{Head}_{-1}}(c), \pi_{\texttt{Head}_{+1}}(c))$. This configuration has a starting TM head at the leftmost cell of every colony, and since $c \in \Phi^{-1}(b)$, we have that in the i'th colony the input of the TM is $\chi(b_i)$.

- $t_0 = t_0 + U$: Line 7 checks that in no place of the tape does there appear a head of the TM defined by p. This means that all the TM have accepted within the first U steps and since p is the program of α, we take that $\alpha(b_i)$ exists, for all $i \in \mathbb{Z}$, or equivalently that $G(b)$ exists.

Therefore, if $F^{4U}(c)$ exists, then $G(b)$ exists.

For the other direction, suppose that $G(b)$ exists, or, equivalently, that $\alpha(b_i)$ exists, for all $i \in \mathbb{N}$.

- $\texttt{Clock} = t_0$: By assumption, $c \in \mathcal{S}^{\epsilon}_{\texttt{NTape},\texttt{Head}_{-1},\texttt{Head}_{+1}}$.Line 2 writes the initial state of the TM on the $\texttt{Head}_{-1}$ of the leftmost cell of every colony.

- $t_0 \leq \texttt{Clock} < t_0 + U$: Only the permutation of line 5 is applied. Together with the directions of $\nu_{Compute}$, this implies that at each step, we apply the IPPA $F_{\mathcal{U}}[p]$ on the configuration $(\pi_{\texttt{Tape}}(c), \pi_{\texttt{Head}_{-1}}(c), \pi_{\texttt{Head}_{+1}}(c))$. There is a TM head at the leftmost

position of every colony and the input in the i'th colony is equal to $\chi(b_i)$. In other words, $\tilde{\phi}_{\texttt{Tape}}(B_i^c) = b_i$, for all $i \in \mathbb{Z}$.

- $\texttt{Clock} = t_0 + U$ Since $\alpha(b_i)$ is defined for all $i \in \mathbb{Z}$, $U \geq t_p(\mathbb{F}_5^{\mathbf{k}'})$ and $S \geq 2U$, we can see that the conditions of Corollary 28 are satisfied with $n = U$. This means that the computation of the TM in every colony has accepted and that the output of the computation is written on the `Tape` of every colony. In other words, $\tilde{\phi}_{\texttt{Tape}}(B_i^{F^U(c)}) = \alpha(b_i)$, for all $i \in \mathbb{Z}$.

  The check of line 7 is true, since by assumption all of the TM have accepted before $\texttt{Clock} = U$, and when a TM halts its head disappears. Line 8 copies the contents of `Tape` on `NTape`. Therefore, after the application of line 8, we have that $\tilde{\phi}_{\texttt{NTape}}(B_i^{F^U(c)}) = \alpha(b_i)$, for all $i \in \mathbb{Z}$.

  Finally, line 9 swaps the fields $\texttt{Head}_{-1}$ and $\texttt{Head}_{+1}$. This can be thought of as "reversing" the directions of these fields. We do this because we want to reverse the computation done by $F_{\mathcal{U}}[p]$ and in order to achieve this, it is not enough to apply $\langle \gamma_{\mathcal{U}}[p]^{-1} \rangle$, but we also need to use directions $-\nu_{Compute}$.

- $t_0 + U + 1 \leq \texttt{Clock} \leq t_0 + 2U$: Only the permutation of line 11 is applied. Together with the fact that the shift directions have been reversed and that the fields $\texttt{Head}_{-1}$, $\texttt{Head}_{+1}$ have also been exchanged in the rules of lines 5 and 11, this implies that the IPPA $(F_{\mathcal{U}}[p])^{-1}$ is applied for U time steps on the configuration
  $(F_{\mathcal{U}}[p])^U(\pi_{\texttt{Tape}}(c), \pi_{\texttt{Head}_{-1}}(c), \pi_{\texttt{Head}_{+1}}(c))$.

  Therefore,

$$(\pi_{\texttt{Tape}} F^{2U}(c), \pi_{\texttt{Head}_{-1}} F^{2U}(c), \pi_{\texttt{Head}_{+1}} F^{2R}(c)) = \\ = (\pi_{\texttt{Tape}}(c), \pi_{\texttt{Head}_{-1}}(c), \pi_{\texttt{Head}_{+1}}(c)).$$

  In other words, the computation has been "run backwards" until the beginning, but the output of the computation is on `NTape`. This is the trick used by Bennet in [3] to simulate arbitrary TM with reversible ones.

  At this point, $F^{2U}(c) \in \mathcal{S}^{\epsilon}_{\texttt{Head}_{+1}}$, $\texttt{Head}_{-1}$ is empty except at the leftmost cell of every colony, where it contains the initial state 0, and finally, $\phi_{\texttt{Tape}}(B_i^{F^{2U}(c)}) = b_i$ and $\phi_{\texttt{NTape}}(B_i^{F^{2U}(c)}) = \alpha(b_i)$, for all $i \in \mathbb{Z}$.

- $t_0 + 2U \leq \texttt{Clock} < t_0 + 3U$: Only the permutation of line 14 is applied. Together with the directions of $\nu_{Compute}$, this implies that at each step, we apply the IPPA $F_{\mathcal{U}}[p^{-1}]$ on the configuration

$(\pi_{\texttt{NTape}}(c), \pi_{\texttt{Head}_{-1}}(c), \pi_{\texttt{Head}_{+1}}(c))$. Notice that we use NTape as the TM tape and we use the program $p^{-1} = t_{\texttt{RProg}}$.

- $\texttt{Clock} = t_0 + 3U$: Since $\alpha^{-1}(\alpha(b_i))$ is defined for all $i \in \mathbb{Z}$, $U > t_{p^{-1}}(\mathbb{F}_5^{\mathbf{k}'})$ and $S \geq 2U$, the conditions of Corollary 28 are satisfied with $n = U$. This implies that $\phi_{\texttt{NTape}}(B_i^{F^{3U}(c)}) = b_i$, for all $i \in \mathbb{Z}$.

  The check of line 16 is true, since all of the TM have accepted before $\texttt{Clock} = 3U$. Line 17 copies the contents of Tape on NTape. Since, at this point these fields are equal in every cell, this is equivalent to emptying fields NTape (in a reversible way, though). Therefore, after applying this permutation, $F^{3U}(c) \in \mathcal{S}^{\epsilon}_{\texttt{NTape}}$.

  We still have to empty the Head fields, too. For this, we have to run the computation backwards. Line 18 swaps the fields $\texttt{Head}_{-1}$ and $\texttt{Head}_{+1}$, "reversing" the directions of these fields.

- $t_0 + 3U + 1 \leq \texttt{Clock} \leq t_0 + 4U$: Only the permutation of line 20 is applied. Together with the fact that the shift directions have been reversed and that the head fields inside the rules are also exchanged, this implies that the IPPA $F_{\mathcal{U}}[p^{-1}]^{-1}$ is applied for U time steps on the configuration $F_{\mathcal{U}}[p^{-1}]^{3U}(\pi_{\texttt{Tape}}(c), \pi_{\texttt{Head}_{-1}}(c), \pi_{\texttt{Head}_{+1}}(c))$.

  Therefore,

$$(\pi_{\texttt{Tape}}F^{4U}(c), \pi_{\texttt{Head}_{-1}}F^{4U}(c), \pi_{\texttt{Head}_{+1}}F^{4U}(c)) = \\ = (\pi_{\texttt{Tape}}F^{2U}(c), \pi_{\texttt{Head}_{-1}}F^{2U}(c), \pi_{\texttt{Head}_{+1}}F^{2U}(c)).$$

  Notice that now we are using Tape as the tape of the TM, while during the forward computation we used NTape. This is not a problem, though, because the two fields were equal at the end of the forward computation at step $3U$.

  At this point, we have that $\phi_{\texttt{Tape}}(B_i^{F^{4U}(c)}) = \alpha(b_i)$, for all $i \in \mathbb{Z}$. Also, in the Head fields, there exists the initial state 0 of the TM on $\texttt{Head}_{-1}$ of the leftmost cell of every colony, while the rest of them are empty.

- Finally, line 23 deletes these initial states from all the colonies, and we get that $F^{4U}(c) \in \mathcal{S}^{\epsilon}_{\texttt{NTape},\texttt{Head}_{-1},\texttt{Head}_{+1}}$.

Therefore, we have proved that if $G(b)$ exists, then $F^{4U}(c)$ exists and $F^{4U}(c) \in (\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}} \cap \Sigma^{0,t_0+4U,S,T} \cap \mathcal{S}^{\epsilon}_{\texttt{Head}_{-1},\texttt{Head}_{+1},\texttt{NTape}} \cap \tilde{\Phi}^{-1}_{\texttt{Tape}}(\alpha(b))$, which finishes the proof of the lemma. □

In a nutshell, this is how the construction works. First, use the program p to compute α. At the end of this phase, Tape contains $\alpha(b)$ (in the

colonies). Copy $\alpha(b)$ onto NTape and in the second phase, run the computation backwards so as to erase all auxiliary information written by the TM during the computation. At the end of the second phase, Tape contains b and NTape contains $\alpha(b)$. In the third and fourth phases of the construction, perform the reverse of what was done in the first two phases, while exchanging the roles of NTape and Tape. First, use p^{-1} with tape field NTape so as to compute $\alpha^{-1}(\alpha(b)) = b$, then copy Tape onto NTape (thus emptying NTape) and then perform the computation backwards. At the end, NTape is again empty and Tape contains $\alpha(b)$ and everything was done in a reversible way.

Notice for all $b \in (\mathbb{F}_5^{\mathbf{k}'})$ and all $c \in \Phi^{-1}(b)$, the values of the fields in $\mathcal{C}_{Grid} \sqcup \mathcal{C}_{Compute}$ of c are uniquely determined. This implies that if there are no anonymous fields, or if the values of the anonymous fields were determined by the fields of $\mathcal{C}_{Grid} \sqcup \mathcal{C}_{Compute}$, then the simulation is also exact.

5.4 Shifting

Let $\mathcal{C}_{MacroShift} := [\texttt{Tape}, \texttt{Tape}_{-1}, \texttt{Tape}_{+1}]$.

$\texttt{Tape}_{+1}$ and $\texttt{Tape}_{-1}$ are used to exchange the information of Tape between colonies.

In the following algorithm, $M \in \mathbb{N}_1$ has to be thought of as the number of fields in the simulated alphabet, $\nu \in \{-1, 0, +1\}^M$ as the vector of directions of the simulated IPPA, and $\mathbf{k}' \colon \mathbb{F}_5^{**} \nrightarrow \mathbb{N}^M$ is a vector valuation that gives the lengths of the alphabet of the simulated IPPA. $\mathbf{k}'$ represents the field lengths of the simulated letters, whose information is then "known" to all the letters of the simulating PPCA.

$MacroShift[M, \nu, \mathbf{k}', v_{\texttt{Addr}}, v_{\texttt{Clock}}, v_{\texttt{MAddr}}]$

1: **if** $v_{\texttt{Clock}} = 0$ **or** $v_{\texttt{Clock}} = v_{\texttt{MAddr}}$ **then**
2: **for** $0 \leq i < M$ **do**
3: **if** $l_{\mathbf{k}',i} \leq v_{\texttt{Addr}} < l_{\mathbf{k}',i+1}$ **then**
4: $Swap[\texttt{Tape}, \texttt{Tape}_{\nu_i}]$ {Letters are moved to the corresponding moving fields, and back after $v_{\texttt{MAddr}}$ steps.}
5: **end if**
6: **end for**
7: **end if**

This is polynomially computable in the parameters.

Lemma 31. *Let us fix a field list $\mathcal{C}_{Grid} \sqcup \mathcal{C}_{MacroShift} \in \mathbb{N}^7$, an integer $M \in \mathbb{N}_1$, a direction vector $\nu \in \{-1, 0, +1\}^M$, a vector $\mathbf{k}' \in \mathbb{N}^M$, a vector $\mathbf{k} \in \mathbb{N}^*$ and integers $S, T \in \mathbb{N}_1$, $t_0 \in \mathbb{N}$.*

Consider the IPPA F defined by directions $\nu_{Grid \sqcup Compute}$, given by the label indices, and permutation

$$MacroShift[M, \nu_{Grid \sqcup Compute}, \mathbf{k}', \underline{\pi_{\mathit{Addr}}}, \underline{\pi_{\mathit{Clock}}} - t_0, S]$$
$$Grid[S, T],$$

and assume that the following inequalities hold:

$$\begin{cases} S \geq \left\| \chi\left(\mathbb{F}_5^{\mathbf{k}'}\right) \right\| \\ T \geq t_0 + S \\ k_{\mathit{Addr}}, k_{\mathit{Addr}_{+1}} \geq \|S\| \\ k_{\mathit{Clock}}, k_{\mathit{Clock}_{+1}} \geq \|T\| \\ k_{\mathit{Tape}}, k_{\mathit{Tape}_{-1}}, k_{\mathit{Tape}_{+1}} \geq 1 \ . \end{cases}$$

Then $F_{|(\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}}} \underset{S,T,0,\Phi}{\succeq} \sigma^{-\nu}_{|(\mathbb{F}_5^{\mathbf{k}'})^{\mathbb{Z}}}$, where $\Phi := \tilde{\Phi}_{|\Sigma}$ and $\Sigma := (\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}} \cap \Sigma^{0,0,S,T} \cap \mathcal{S}^{\epsilon}_{\mathit{Tape}_{-1}, \mathit{Tape}_{+1}} \cap \tilde{\Phi}^{-1}((\mathbb{F}_5^{\mathbf{k}'})^{\mathbb{Z}})$.

Recall that, by convention, the directions of the fields are the opposite of the shift that is actually applied.

Proof. Again, by the definition of Φ and Remarks 24 and 26, we know that Φ is surjective, $\Phi\sigma^S = \sigma\Phi$ and that $\mathcal{D}(\Phi) := \bigsqcup_{\substack{0 \leq t < T \\ 0 \leq s < S}} F^t\sigma^s(\mathcal{D}(\Phi))$ is a disjoint union.

Therefore, we only have to show that $\sigma^{-\nu}\Phi = \Phi F^T$, which is equivalent to showing, since $\sigma^{-\nu}(b)$ is defined for all $b \in (\mathbb{F}_5^{\mathbf{k}'})^{\mathbb{Z}}$, that if $\Phi(c) = b$, then $\Phi F^T(c) = \sigma^{-\nu}(b)$.

As in the proof of Lemma 29, we are going to prove the following stronger

Fact 32. *If $c \in (\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}} \cap \Sigma^{0,t_0,S,T} \cap \mathcal{S}^{\epsilon}_{\mathit{Tape}_{-1}, \mathit{Tape}_{+1}} \cap \tilde{\Phi}^{-1}_{\mathit{Tape}}(b)$, then $F^S(c) \in \Sigma^{0,t_0+S,S,T} \cap \mathcal{S}^{\epsilon}_{\mathit{Tape}_{-1} = \mathit{Tape}_{+1}} \cap \tilde{\Phi}^{-1}(\sigma^{-\nu}(b))$.*

- $\mathtt{Clock} = t_0$: By assumption, $c \in \mathcal{S}^{\epsilon}_{\mathtt{Tape}_{-1}, \mathtt{Tape}_{+1}}$.

  Lines 3 and 4 copy the encodings of the fields of b on the correct `Tape`, in the following sense:

  Since $c \in \Phi^{-1}(b)$, $\pi_{\mathtt{Tape}}(B_i^c)_{|[\![l_{\mathbf{k}',j}, l_{\mathbf{k}',j+1}[\![} = \chi(\pi_j(b_i))$, for all $i \in \mathbb{Z}$ and $0 \leq j < M$. Line 4 moves the letter in `Tape` to $\mathtt{Tape}_{\nu_j}$ when $l_{\mathbf{v},j} \leq \mathtt{Addr} < l_{\mathbf{v},j+1}$, or in other words, moves the encoding of the j'th field of b_j onto $\mathtt{Tape}_{+1}$ if j is a right-moving field and to $\mathtt{Tape}_{-1}$ if j is a left-moving field.

  This means that after the application of line 4, we have that $\pi_{\mathtt{Tape}_{\nu_j}}(B_i^c)_{|[\![l_{\mathbf{k}',j}, l_{\mathbf{k}',j+1}[\![} = \chi(\pi_j(b_i))$, while $\pi_{\mathtt{Tape}_{-\nu_j}}(B_i^c)_{|[\![l_{\mathbf{k}',j}, l_{\mathbf{k}',j+1}[\![} = \pi_{\mathtt{Tape}}(B_i^c)_{|[\![l_{\mathbf{k}',j}, l_{\mathbf{k}',j+1}[\![} = \epsilon$, for all $i \in \mathbb{Z}$ and $0 \leq j < M$.

- $t_0 \leq \texttt{Clock} < t_0 + S$: No permutation is applied during these time steps. Only the Tape fields are shifted to the corresponding direction.
- $t_0 = t_0 + S$: Every symbol that was part of the encoding of the j'th field of b has travelled S steps to the direction indicated by ν_j. This means that before the application of line 4, we have that $\pi_{\texttt{Tape}_{\nu_i}}(B_i^{F^S(c)})_{|[\![l_{\mathbf{k}',j}, l_{\mathbf{k}',j+1}[\![} = \chi\left(\pi_i(b_{i-\nu_i})\right)$, while $\pi_{\texttt{Tape}_{-\nu_j}}(B_n^{F^S(c)})_{|[\![l_{\mathbf{k}',j}, l_{\mathbf{k}',j+1}[\![} = \pi_{\texttt{Tape}}(B_i^{F^S(c)})_{|[\![l_{\mathbf{k}',j}, l_{\mathbf{k}',j+1}[\![} = \epsilon$, for all $i \in \mathbb{Z}$ and $0 \leq j < M$.

  Line 4 moves the letter from the Tape fields back to Tape. Therefore, after the application of line 4, we have that $F^S(c) \in \mathcal{S}^{\epsilon}_{\texttt{Tape}_{-1},\texttt{Tape}_{+1}}$ and $\Phi(F^S(c)) = \sigma^{-\nu}(b)$, which concludes the proof of the Lemma.

□

In this proof, it is of great importance that all the letters of b have the same lengths, because this implies that the j'th field of every letter of b is encoded at the same positions inside every colony. In fact, the reason that we deal only with alphabets of constant lengths is that this shifting procedure can work so easily.

5.5 Simulating any fixed rule

In this subsection, we will use Lemma 29 and 31 to construct an IPPA that can simulate non-trivially any PCA, when restricted to an appropriate finite subalphabet.

Let $\mathcal{C}_{Simulate} := \mathcal{C}_{Compute} \cup \mathcal{C}_{MacroShift} \in \mathbb{N}^6$ ($\mathcal{C}_{Compute}$ and $\mathcal{C}_{MacroShift}$ share the field Tape).

$Simulate[M, \nu, \mathbf{k}', v_{\texttt{Addr}}, v_{\texttt{Clock}}, v_{\texttt{MAddr}}, v_{\texttt{MClock}}, v_{\texttt{Alarm}}, t_{\texttt{Prog}}, t_{\texttt{RProg}}]$

1: $Compute[v_{\texttt{Addr}}, v_{\texttt{Clock}}, v_{\texttt{Alarm}}, t_{\texttt{Prog}}, t_{\texttt{RProg}}]$
2: $MacroShift[M, \nu, v_{\texttt{Addr}}, v_{\texttt{Clock}} - 4v_{\texttt{Alarm}}, v_{\texttt{MAddr}}, \mathbf{k}']$

This is easily seen to be polynomially computable in the parameters.

Notice that $MacroShift$ and $Compute$ are used at "different time steps", *i.e.*, at different values of $v_{\texttt{Clock}}$. $Compute$ starts being used when $v_{\texttt{Clock}} = 0$ and, by definition of $v_{Compute}$, is equal to the identity when $v_{\texttt{Clock}} \geq 4v_{\texttt{Alarm}}$, while $MacroShift$ starts being used when $v_{\texttt{Clock}} = 4v_{\texttt{Alarm}}$ (it has a delay of $4v_{\texttt{Alarm}}$). Formally, this means that for every value of $v_{\texttt{Clock}}$, at most one of the rules $Compute$ and $MacroShift$ is not equal to the identity.

The following lemma is the fruit of all our efforts until now. It provides an IPPA that can simulate any PCA when restricted to a sufficiently large alphabet.

Lemma 33. *Let us fix a field list $\mathcal{C}_{Simulate} \sqcup \mathcal{C}_{Grid} \in \mathbb{N}^{10}$, an integer $M \in \mathbb{N}_1$, programs $p, p^{-1} \in \mathbb{F}_2^*$ of a partial permutation $\alpha : \mathbb{F}_5^{**} \nrightarrow \mathbb{F}_5^{**}$ and its inverse α^{-1}, respectively, a direction vector $\nu \in \{-1, 0, +1\}^M$, a vector $\mathbf{k}' \in \mathbb{N}^M$, a vector $\mathbf{k} \in \mathbb{N}^*$ and integers $S, T, t_0, U \in \mathbb{N}$.*

Let $G = \sigma^{-\nu}\alpha$ and F be the IPPA defined by directions $\nu_{Grid \sqcup Simulate}$, given by the label indices, and permutation

$$\begin{cases} Simulate[M, \nu_{Simulate}, \mathbf{k}', \underline{\pi_{\texttt{Addr}}}, \underline{\pi_{\texttt{Clock}}} - t_0, S, T, U, p, p^{-1}] \\ Grid[S, T] \ , \end{cases}$$

and assume that the following inequalities hold:

$$\begin{cases} U \geq \max\{t_p(\mathbb{F}_5^{\mathbf{k}'}), t_{p^{-1}}(\mathbb{F}_5^{\mathbf{k}'})\} \\ S \geq \max\{2U, \left\|\chi\left(\mathbb{F}_5^{\mathbf{k}'}\right)\right\|\} \\ T \geq 4U + S + t_0 \\ k_{\texttt{Addr}}, k_{\texttt{Addr}_{+1}} \geq \|S\| \\ k_{\texttt{Clock}}, k_{\texttt{Clock}_{+1}} \geq \|T\| \\ k_{\texttt{Head}_{-1}}, k_{\texttt{Head}_{+1}} \geq \left|\chi\left(\mathbb{F}_4 \times (Q_p \cup Q_{p^{-1}}) \times \{-1, +1\}\right)\right| \\ k_{\texttt{Tape}}, k_{\texttt{NTape}}, k_{\texttt{Tape}_{-1}}, k_{\texttt{Tape}_{+1}} \geq 1. \end{cases}$$

Then, $F_{|(\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}}} \underset{S,T,0,\Phi}{\succeq} G_{|(\mathbb{F}_5^{\mathbf{k}'})^{\mathbb{Z}}}$ completely, where $\Phi := \tilde{\Phi}_{\texttt{Tape}|\Sigma}$ and

$$\Sigma := (\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}} \cap \Sigma^{0,t_0,S,T} \cap \mathcal{S}^{\epsilon}_{\texttt{Head}_{-1},\texttt{Head}_{+1},\texttt{Tape}_{-1},\texttt{Tape}_{+1},\texttt{NTape}} \cap \tilde{\Phi}^{-1}_{\texttt{Tape}}((\mathbb{F}_5^{\mathbf{k}'})^{\mathbb{Z}}).$$

Proof. Notice that line 1 together with *Grid* make up the permutation whose behavior is described in Lemma 29, while line 2 and *Grid* make up the permutation of Lemma 31. Also, as already noted, for any value of `Clock`, at most one permutation of lines 1 and 2 is not equal to the identity.

Like in the proofs of Lemmas 29 and 31, we can easily see that we only have to show that if $\Phi(c) = b$, then $\Phi F^T(c) = G(b)$, and that this follows from the following stronger fact:

Fact 34. *If $c \in (\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}} \cap \Sigma^{0,t_0,S,T} \cap \mathcal{S}^{\epsilon}_{\texttt{NTape},\texttt{Head}_{-1},\texttt{Head}_{+1},\texttt{Tape}_{-1},\texttt{Tape}_{+1}} \cap \tilde{\Phi}^{-1}_{\texttt{Tape}}(b)^{\mathbb{Z}}$, then $F^{4U+S}(c)$ exists if and only if $G(b)$ exists, and in that case $F^{4U+S}(c) \in \Sigma^{0,t_0+4U+S,S,T} \cap \mathcal{S}^{\epsilon}_{\texttt{NTape},\texttt{Head}_{-1},\texttt{Head}_{+1}} \cap \tilde{\Phi}^{-1}_{\texttt{Tape}}(G(b))$.*

Indeed, according to Fact 30, we have that $F^{4U}(c)$ exists if and only if $\alpha(b)$ exits, or, equivalently, if and only if $G(b)$ exists, and in this case

$$F^{4U}(c) \in \Sigma^{0,t_0+4U,S,T} \cap \mathcal{S}^{\epsilon}_{\texttt{Head}_{-1},\texttt{Head}_{+1},\texttt{Tape}_{-1},\texttt{Tape}_{+1},\texttt{NTape}} \cap \tilde{\Phi}^{-1}_{\texttt{Tape}}(\alpha(b)).$$

Similarly, Fact 32 implies that

$$F^{4U+S}(c) \in \Sigma^{0,t_0+4U+S,S,T} \cap \mathcal{S}^{\epsilon}_{\mathtt{Head}_{-1},\mathtt{Head}_{+1},\mathtt{Tape}_{-1},\mathtt{Tape}_{+1},\mathtt{NTape}} \cap \\ \cap \tilde{\Phi}^{-1}_{\mathtt{Tape}}(\sigma^{-\nu}\alpha(b)),$$

which concludes the proof of the lemma, since $G = \sigma^{-\nu}\alpha(b)$. □

Simulate is a rule (family of rules in fact, since they depend on parameters) that can simulate any PPA. Every IPPA F that we will construct later will factor onto *Simulate*. They might have some additional fields for which we apply a different rule, and this rule might even take into consideration the fields of $\mathcal{C}_{Simulate}$, but none of these other rules is going to *change* the fields of $\mathcal{C}_{Simulate}$. Therefore, by projecting onto $\mathcal{C}_{Simulate}$, we will immediately obtain that F simulates G, even though the simulation might not be exact.

However, if $\mathbf{k}$ does not have any anonymous fields, then the simulation is exact, since all the fields of $\mathcal{C}_{Grid} \sqcup \mathcal{C}_{Simulate}$ are uniquely determined by Φ.

5.5.1 Satisfying the inequalities

Lemma 33 is true only under the assumption that the following set of inequalities are satisfied:

$$\begin{cases} U \geq \max\{t_p(\mathbb{F}_5^{\mathbf{k}'}), t_{p^{-1}}(\mathbb{F}_5^{\mathbf{k}'})\} \\ S \geq \max\{2U, \left\|\chi\left(\mathbb{F}_5^{\mathbf{k}'}\right)\right\|\} \\ T \geq 4U + S + t_0 \\ k_{\mathtt{Addr}}, k_{\mathtt{Addr}_{+1}} \geq \|S\| \\ k_{\mathtt{Clock}}, k_{\mathtt{Clock}_{+1}} \geq \|T\| \\ k_{\mathtt{Head}_{-1}}, k_{\mathtt{Head}_{+1}} \geq \left|\chi\left(\mathbb{F}_4 \times (Q_p \cup Q_{p^{-1}}) \times \{-1, +1\}\right)\right| \\ k_{\mathtt{Tape}}, k_{\mathtt{NTape}}, k_{\mathtt{Tape}_{-1}}, k_{\mathtt{Tape}_{+1}} \geq 1\ . \end{cases}$$

We will denote this set of inequalities by $\mathcal{I}(\mathbf{k}, \mathbf{k}', S, T, U, t_0, p, p^{-1})$. When $t_0 = 0$, *i.e.*, when the computation starts at $\mathtt{Clock} = 0$, we will omit it and write $\mathcal{I}(\mathbf{k}, \mathbf{k}', S, T, U, p)$, instead. Let us explain again intuitively why each inequality is needed:

- $k_{\mathtt{Addr}}, k_{\mathtt{Addr}_{+1}} \geq \|S\|$: The fields $k_{\mathtt{Addr}}$ and $k_{\mathtt{Addr}+1}$ have to be large enough so that we can write the binary representation of S in them.
- $k_{\mathtt{Clock}}, k_{\mathtt{Clock}_{+1}} \geq \|T\|$: The fields $k_{\mathtt{Clock}}$ and $k_{\mathtt{Clock}+1}$ have to be large enough so that we can write the binary representation of T in them.
- $U \geq \max\{t_p(\mathbb{F}_5^{\mathbf{k}'}), t_{p^{-1}}(\mathbb{F}_5^{\mathbf{k}'})\}$: We have to run the TM long enough so that the computation of p onto the encoded letters halts.

- $S \geq \max\{2U, \left\|\chi\left(\mathbb{F}_5^{\mathbf{k}'}\right)\right\|\}$: The colonies have to be wide enough so that we can encode the letters of $\mathbb{F}_5^{\mathbf{k}'}$ in them. In addition, they have to be wide enough so that the heads of the computation do not "collide".

- $T \geq 4U + S + t_0$: T has to be large enough so that the computation and the shifting are done before the next working period starts.

- $k_{\mathtt{Head}_{-1}}, k_{\mathtt{Head}_{+1}} \geq \left|\chi\left(\mathbb{F}_4 \times (Q_p \cup Q_{p^{-1}}) \times \{-1, +1\}\right)\right|$: The head fields have to be large enough so that states of $\gamma_{\mathcal{U}}[p]$ and $\gamma_{\mathcal{U}}[p^{-1}]$ can be written on them.

- $k_{\mathtt{Tape}}, k_{\mathtt{NTape}}, k_{\mathtt{Tape}_{-1}}, k_{\mathtt{Tape}_{+1}} \geq 1$. Empty fields are of no use, in general.

Remark 35. *If $p, p^{-1} \in \mathbb{F}_2^*, \mathbf{k}' \in \mathbb{N}^M$ and $t_0 \in \mathbb{N}$ are fixed, then we can choose $\mathbf{k}, U, S$ and T such that the inequalities of Lemma 33 is satisfied.*

Proof. Since $\mathbf{k}', p$ and t_0 are fixed, given U, S, T we can choose $\mathbf{k} := \mathbf{k}_{U,S,T}$ such that all of the inequalities except for the three first are satisfied as equalities. Then, given S, U we can choose $T := T_{S,U}$ such that the third inequality is satisfied as an equality. Similarly, given U we can choose $S := S_U$ such that the second inequality is satisfied. Finally, U can be chosen independently from the rest of the parameters, since it only depends on p and $\mathbf{k}'$, which are fixed. □

In later constructions, the choice of U will not be so straightforward, as $\mathbf{k}'$ will depend on $\mathbf{k}$. In this case, we first fix G and then look for the suitable values of the parameters. The situation becomes trickier when the simulated RPCA depends on the choice of the parameters, as will be the case in the following chapters. Then, we have to be careful not to fall into a circular argument.

Chapter 6

Infinite hierarchies

For every PPA G and sufficiently large S, T, Lemma 33 shows that is is possible to construct a PPA F that $(S, T, 0)$-simulates G. In addition, the simulation can be made exact. We also want to make it complete. The most direct way is to restrict F to $\tilde{\Phi}^{-1}_{\mathtt{Tape}}(\mathcal{D}G)$, which, by definition makes the simulation complete. However, this is not good because it is a radius-S SFT condition and, if we wanted to have an infinite nested simulation, we would have to impose an infinite number of such restrictions, so that the subshift we would obtain would not be an SFT.

The idea, which is the basic idea behind all hierarchical constructions, is that if the simulated alphabet is determined in an easy way by the simulating alphabet, then it is possible to design a simple IPPA that ensures that the simulating configuration is in $\tilde{\Phi}^{-1}_{\mathtt{Tape}}(\mathcal{D}G)$.

6.1 Son-father checks

The first thing is to check that the simulated letter, which is written in an encoded form bit by bit in `Tape`, has the correct structure, *i.e.*, it is the encoding of a letter with the correct number of fields and lengths.

$\mathbf{k}' \in \mathbb{F}_5^{**} \dashrightarrow \mathbb{N}^M$ is a vector valuation, that gives the lengths of the simulated alphabet as a function of the lengths of the simulating alphabet. In applications, it will be easily computable from every letter of the simulating alphabet, or, in other words, the information about the structure of the simulated letter will be known to all of the letters of the simulating IPPA.

$CkAlph[M, v_{\mathtt{Addr}}, t_{\mathtt{Tape}}, \mathbf{k}']$

1: **if** $v_{\mathtt{Addr}} \geq l_{\mathbf{k}',M}$ **then**
2: $\quad Check(t_{\mathtt{Tape}} = 3)$ {On the right side of the encoding, `Tape` is 3.}
3: **end if**

4: **for** $0 \leq i < M$ **do**
5: **if** $v_{\texttt{Addr}} = l_{\mathbf{k}',i}$ **then**
6: $Check[t_{\texttt{Tape}} = 2]$ {Field separators are at the expected positions.}
7: **else if** $l_{\mathbf{k}',i} < v_{\texttt{Addr}} < l_{\mathbf{k}',i+1}$ **then**
8: $Check[t_{\texttt{Tape}} \in \mathbb{F}_2]$ {Proper field encodings are binary.}
9: **end if**
10: **end for**

This permutation is polynomially computable in its parameters. (In every case that we use it, it will be easily checkable that the parameters are polynomially computable.)

The following lemma follows simply by inspection of the definition of $\chi(\cdot)$ and $CkAlph[M, v_{\texttt{Addr}}, t_{\texttt{Tape}}, \mathbf{k}']$:

Lemma 36. *Let us fix a field list* $[\texttt{Addr}, \texttt{Tape}] \in \mathbb{N}^2$*, an integer* $M \in \mathbb{N}_1$*, a vector* $\mathbf{k}' \in \mathbb{N}^M$*,* $S \in \mathbb{N}_1$ *and a vector* $\mathbf{k} \in \mathbb{N}^*$*.*

Let F *be the IPPA defined by a null direction vector and permutation* $CkAlph[M, \underline{\pi_{\texttt{Addr}}}, \pi_{\texttt{Tape}}, \mathbf{k}']$*, and assume that the following inequalities hold:*

$$\begin{cases} S \geq \left\| \chi\left(\mathbb{F}_5^{\mathbf{k}'}\right) \right\| \\ k_{\texttt{Addr}} \geq \|S\| \\ k_{\texttt{Tape}} \geq 1\ . \end{cases}$$

Let $c \in (\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}} \cap \Sigma^{s,S} \cap \tilde{\Phi}^{-1}_{\texttt{Tape}}(b)$*, where* $s \in [\![0, S[\![$ *and* $b \in (\mathbb{F}_5^{**})^{\mathbb{Z}}$*. Then,* $F(c)$ *exists if and only if* $b \in (\mathbb{F}_5^{\mathbf{k}'})^{\mathbb{Z}}$ *and in this case* $F(c) = c$*.*

In other words, if a configuration is split into colonies using the `Addr` field and every colony has the encoding of some letter on its `Tape` tape, then $CkAlph[M, \underline{\pi_{\texttt{Addr}}}, \pi_{\texttt{Tape}}, \mathbf{k}']$ ensures that this encoded letter belongs in $\mathbb{F}_5^{\mathbf{k}'}$.

We can also check that some field i in the simulated letter has a prescribed prefix (given by a term t).

$HCheck[M, v_{\texttt{Addr}}, t_{\texttt{Tape}}, \mathbf{k}', i, t]$

1: **if** $l_{\mathbf{k}',i} < v_{\texttt{Addr}} \leq l_{\mathbf{k}',i} + |\chi(t)|$ **then**
2: $Check[t_{\texttt{Tape}} = \chi(t)_{|v_{\texttt{Addr}} - l_{\mathbf{k}',i}}]$
3: **end if**

Lemma 37. *Let us fix a field list* $[\texttt{Addr}, \texttt{Tape}] \in \mathbb{N}^2$*, an integer* $M \in \mathbb{N}_1$*, a field* $i \in [\![0, M[\![$*, a covector* $\mathbf{k}' \in \mathbb{N}^M$*,* $S \in \mathbb{N}$*, a term* $t \colon \mathbb{F}_5^{**} \to \mathbb{F}_5^*$ *and a vector* $\mathbf{k} \in \mathbb{N}^*$*.*

Let F be the IPPA defined by a null direction vector and permutation $HCheck[M, \underline{\pi_{Addr}}, \mathtt{Tape}, \mathbf{k}', i, t]$, and assume that the following inequalities hold:

$$\begin{cases} S \geq \left\| \chi\left(\mathbb{F}_5^{\mathbf{k}'}\right) \right\| \\ k_{Addr} \geq \|S\| \\ k_{Tape} \geq 1\ . \end{cases}$$

Let $c \in (\mathbb{F}_5^{\mathbf{k}})^{\mathbb{Z}} \cap \Sigma^{s,S} \cap \phi^{-1}(b)$, where $0 \leq s < S$ and $b \in (\mathbb{F}_5^{\mathbf{k}'})^{\mathbb{Z}}$ and assume that $t(c_n) = t(c_{n'}) := t_c$ for all $n, n' \in \mathbb{Z}$.
Then, $F(c)$ exists if and only if $\pi_i(b_j)_{[\![0,\|t_c\|[\![} = t_c$, for all $j \in \mathbb{Z}$ and in this case $F(c) = c$.

We implicitly assume that if $l > \|w\|$, where $w \in \mathcal{A}^*$, then $w_l = \epsilon$.

In other words, if all letters of c have the "same idea" about what $\pi_i(b_j)$ should be, then, they can check in one step that this indeed happens. In practice, t will usually be equal to $\pi_{\mathtt{Field}}$, where `Field` is a horizontally constant field, so that the condition $t(c_n) = t(c_{n'})$ will be true. In this case, we just check that $\pi_{\mathtt{Field}}(c_n)$ is a prefix of $\pi_{\mathtt{Field}}(b_j)$, for all $j, n \in \mathbb{Z}$.

Lemmas 36 and 37 correspond to what in [8] is achieved by mentioning that "the TM knows at which place the information of every field is held". For many people, this argument is one of the most confusing things in that construction. This is the reason why we have tried to explain this point as clearly as possible and show exactly how the cells of the simulating IPPA can collectively check that some constant information of the simulating alphabet is the same in the simulated alphabet. In fact, we use a general term t in Lemma 37, which essentially allows us to impose any (polynomially computable) condition on the simulated alphabet.

6.2 Self-simulation

We are now ready to construct a self-simulating RPCA. This is the simplest and first example of nested simulation. We just check that the simulated letter has the same lengths as the simulating ones and that some "hierarchical" fields (which contain the values p, p^{-1}, U, S, T that are fixed in Lemma 33) have the same value in the simulated letter as in the simulating ones (where their values is already fixed).

Let $\mathcal{C}_{\mathtt{Self}} := \mathcal{C}_{Simulate} \sqcup [\mathtt{MAddr}, \mathtt{MClock}, \mathtt{Alarm}, \mathtt{Prog}, \mathtt{RProg}] \in \mathbb{N}^{15}$.

`MAddr` and `MClock` are used to obtain the values of $v_{\mathtt{MAddr}}$ and $v_{\mathtt{MClock}}$. Similarly, `Prog`, `RProg` and `Alarm` are used to obtain the values $t_{\mathtt{Prog}}, t_{\mathtt{RProg}}$ and $v_{\mathtt{Alarm}}$. All of these fields will be horizontally constant.

$\texttt{Self}[M, \nu]$

1: **if** $\underline{\pi_{\texttt{Clock}}} = 0$ **then**
2: $\quad Check[\mathcal{S}^{\epsilon}_{\texttt{Head}_{-1},\texttt{Head}_{+1},\texttt{Tape}_{-1},\texttt{Tape}_{+1},\texttt{NTape}}]$
3: $\quad CkAlph[M, \underline{\pi_{\texttt{Addr}}}, \pi_{\texttt{Tape}}, (|\pi_j|)_{j<M}]$ {Check that the lengths of the simulated letter are the same}
4: $\quad$ **for** $i \in \{\texttt{MAddr}, \texttt{MClock}, \texttt{Alarm}, \texttt{Prog}, \texttt{RProg}\}$ **do**
5: $\quad\quad HCheck[M, \underline{\pi_{\texttt{Addr}}}, \texttt{Tape}, (|\pi_j|)_{j<M}, i, \pi_i]$ {Check that the hierarchical fields of the simulated letter are the same}
6: $\quad$ **end for**
7: **end if**
8: $Simulate[M,\nu,(|\pi_j|)_{j<M},\underline{\pi_{\texttt{Addr}}},\underline{\pi_{\texttt{Clock}}}, \underline{\pi_{\texttt{MAddr}}}, \underline{\pi_{\texttt{MClock}}}, \underline{\pi_{\texttt{Alarm}}}, \pi_{\texttt{Prog}}, \pi_{\texttt{RProg}}]$ {The alphabet is as expected; we can simulate.}
9: $Grid[\underline{\pi_{\texttt{MAddr}}}, \underline{\pi_{\texttt{MClock}}}]$

In the next lemma, we do not want to have any anonymous fields, but only those fields that are used in `Self`. There are 15 fields in $\mathcal{C}_{\texttt{Self}}$, so we take the field list $[0, \dots, 14]$, which means that we assign a number of $[\![0, 15[\![$ to every field in $\mathcal{C}_{\texttt{Self}}$ in some random (but fixed) way. Once we have done this, the corresponding vector of directions is also well-defined.

Lemma 38. *Let us fix the field list $\mathcal{C}_{\mathtt{Self}} := [0, \dots, 14]$, the corresponding direction vector $\nu_{\mathtt{Self}}$, integers $S, T, U \in \mathbb{N}_1$ and vector $\mathbf{k} \in \mathbb{N}^{15}$. Let F be the IPPA with directions $\nu_{\mathtt{Self}}$ and permutation $\mathtt{Self}[15, \nu_{\mathtt{Self}}]$ and p, p^{-1} be the programs for this permutation and its inverse, respectively.*

Let $F_{\mathbf{k},S,T,U}$ be the restriction of F to the subalphabet

$$\mathcal{A}_{\mathbf{k},S,T,U} := \mathbb{F}_5^{\mathbf{k}} \cap \mathcal{S}^{S}_{\mathtt{MAddr}} \cap \mathcal{S}^{T}_{\mathtt{MClock}} \cap \mathcal{S}^{U}_{\mathtt{Alarm}} \cap \mathcal{S}^{p}_{\mathtt{Prog}} \cap \mathcal{S}^{p^{-1}}_{\mathtt{RProg}},$$

and assume that the following inequalities are satisfied:

$$\begin{cases} \mathcal{I}(\mathbf{k}, \mathbf{k}, S, T, U, p, p^{-1}) \\ k_{\mathtt{Prog}} \geq \|p\| \\ k_{\mathtt{RProg}} \geq \left\|p^{-1}\right\| \\ k_{\mathtt{MAddr}} \geq \|S\| \\ k_{\mathtt{MClock}} \geq \|T\| \\ k_{\mathtt{Alarm}} \geq \|U\| \end{cases}.$$

Then, $F_{\mathbf{k},S,T,U} \underset{S,T,0,\Phi}{\succeq} F_{\mathbf{k},S,T,U}$ completely exactly, where $\Phi := \tilde{\Phi}_{\mathtt{Tape}|\Sigma}$ and

$$\Sigma := \mathcal{A}^{\mathbb{Z}}_{\mathbf{k},S,T,U} \cap \Sigma^{0,0,S,T} \cap \mathcal{S}^{\epsilon}_{\mathtt{Head}_{-1},\mathtt{Head}_{+1},\mathtt{Tape}_{-1},\mathtt{Tape}_{+1},\mathtt{NTape}} \cap \tilde{\Phi}^{-1}_{\mathtt{Tape}}(\mathcal{A}^{\mathbb{Z}}_{\mathbf{k},U,S,T,p}).$$

It is important to notice that F is a fixed rule that does not depend on $\mathbf{k}, S, T, U$. Therefore, its program p is a *fixed* word which we can "feed" to itself by restricting the alphabet to $\mathcal{S}^p_{\texttt{Prog}}$. This is the basic idea of self-simulation. Notice also that the fields for which we do a hierarchical check are exactly those that are fixed in the definition of $\mathcal{A}_{\mathbf{k},S,T,U}$. We need to ensure that these fields have the correct value in the simulated letter. The correct way to do this is to check that the value of the simulated letter is in a good relation with the values in the letters of the simulating IPPA. Here, the relation is simply equality. Later it will be something more complicated.

We will try to give as many details as possible in the following proof because it will serve as a prototype for the rest of the hierarchical simulations.

Proof. We have to show three things: First of all, that $F_{\mathbf{k},S,T,U}$ $(S,T,0)$-simulates $F_{\mathbf{k},S,T,U}$ with decoding function Φ (simulation), second, that Φ is injective (exactness) and, finally, that $\Omega_{F_{\mathbf{k},S,T,U,p}} \subseteq \tilde{\mathcal{D}}(\Phi)$ (completeness).

For the simulation part, let $b \in \mathcal{A}^{\mathbb{Z}}_{\mathbf{k},S,T,U}$ and $c \in \mathcal{A}^{\mathbb{Z}}_{\mathbf{k},S,T,U} \cap \Sigma^{0,0,S,T} \cap \mathcal{S}^{\epsilon}_{\texttt{Head}_{-1},\texttt{Head}_{+1},\texttt{Tape}_{-1},\texttt{Tape}_{+1},\texttt{NTape}} \cap \tilde{\Phi}^{-1}(b)$. By definition, c is not rejected by the checks of lines 2,3 and 5.

Indeed, line 2 checks that the fields $\texttt{Head}_{-1}$, $\texttt{Head}_{+1}$, $\texttt{Tape}_{-1}$, $\texttt{Tape}_{+1}$, $\texttt{NTape}$ are empty, which is true since

$$c \in \Sigma^{0,0,S,T} \cap \mathcal{S}^{\epsilon}_{\texttt{Head}_{-1},\texttt{Head}_{+1},\texttt{Tape}_{-1},\texttt{Tape}_{+1},\texttt{NTape}}.$$

Line 3 checks that the lengths of the fields of b and c are the same, while the checks of line 5 check that b and c have the same values in the fields $\texttt{MAddr}, \texttt{MClock}, \texttt{Prog}$ and $\texttt{RProg}$, which are true by definition.

Since c is not rejected by these checks and $F_{\mathbf{k},S,T,U}$ is a subrule of

$$Grid[S,T] \circ Simulate_{15,\nu_{\texttt{Self}}}[(|\pi_j|)_{j<M}, \underline{\pi_{\texttt{Addr}}}, \underline{\pi_{\texttt{Clock}}}, S, T, U, p, p^{-1}]$$

and, by assumption, the inequalities of Lemma 33 are satisfied, and p is the program of $\texttt{Self}[15, \nu_{\texttt{Self}}]$, Lemma 33 gives that $F_{\mathbf{k},S,T,U}$ $(S,T,0)$-simulates $F_{\mathbf{k},S,T,U}$ with decoding function Φ.

For the exactness part, we have already noted various times that the values of the fields in $\mathcal{C}_{Simulate}$ are uniquely determined for all $c \in \Phi^{-1}(b)$. For the hierarchical fields (*i.e.*, $\texttt{MAddr}, \texttt{MClock}, \texttt{Alarm}, \texttt{Prog}, \texttt{RProg}$) the values are fixed for all $c \in \mathcal{A}^{\mathbb{Z}}_{\mathbf{k},S,T,U}$. In addition, there do not exist any anonymous fields (since we chose $M = 15$). Therefore, Φ is injective and the simulation is exact.

For the completeness part, we will first show that if $c \in \Sigma^{0,0,S,T} \cap F^{-T}_{\mathbf{k},S,T,U}(\mathcal{A}^{\mathbb{Z}}_{\mathbf{k},S,T,U})$, then $c \in \Phi^{-1}(\mathcal{A}^{\mathbb{Z}}_{\mathbf{k},S,T,U})$.

Indeed, line 2 checks that $c \in \Sigma^{0,0,S,T} \cap \mathcal{S}^{\epsilon}_{\texttt{Head}_{-1},\texttt{Head}_{+1},\texttt{Tape}_{-1},\texttt{Tape}_{+1},\texttt{NTape}}$ (in the sense that if this is not true, then $F_{\mathbf{k},S,T,U,p}$ would not be defined, so there would be a contradiction). According to Lemma 36, line 3 checks that

for every colony B_i^c, $\pi_{\texttt{Tape}}(B_i^c)$ has the structure of the encoding of a letter in $\mathbb{F}_5^{\mathbf{k}}$. (We cannot immediately say that $\pi_{\texttt{Tape}}(B_i^c)$ is the encoding of a letter in $\mathbb{F}_5^{\mathbf{k}}$ because there are some triplets that are not used by $\chi(\cdot)$. So for example, if the first three letters in the `Tape` tape are 111, then $\pi_{\texttt{Tape}}(B_i^c)$ is not the encoding of a letter in $\mathbb{F}_5^{\mathbf{k}}$, even though the 2's and 3's are in the correct positions.) In addition, since $F^T_{\mathbf{k},S,T,U}$ exists and the inequalities $\mathcal{I}(\mathbf{k},\mathbf{k},S,T,U,p,p^{-1})$ are satisfied, this means that the computation of p on input $\pi_{\texttt{Tape}}(B_i^c)$ halts, therefore for all $i \in \mathbb{Z}$, $\tilde{\phi}_{\texttt{Tape}}(B_i^c) = b_i \in \mathbb{F}_5^{**}$. Lemma 36 now implies that $\tilde{\phi}_{\texttt{Tape}}(B_i^c) = b_i \in \mathbb{F}_5^{\mathbf{k}}$. Finally, line 5 checks that $\tilde{\phi}_{\texttt{Tape}}(b_i) \in \mathcal{S}^S_{\texttt{MAddr}} \cap \mathcal{S}^T_{\texttt{MClock}} \cap \mathcal{S}^U_{\texttt{Alarm}} \cap \mathcal{S}^p_{\texttt{Prog}} \cap \mathcal{S}^{p^{-1}}_{\texttt{RProg}}$ by checking that the hierarchical fields of b_i have the same values as the corresponding fields of the letters of c (notice that the hierarchical fields are constant for the letters of c, so that Lemma 37 applies). Summarizing, we have that $b \in \mathcal{A}^{\mathbb{Z}}_{\mathbf{k},S,T,U}$, so that $c \in \mathcal{A}^{\mathbb{Z}}_{\mathbf{k},S,T,U} \cap \Sigma^{0,0,S,T} \cap \mathcal{S}^{\epsilon}_{\texttt{Head}_{-1},\texttt{Head}_{+1},\texttt{Tape}_{-1},\texttt{Tape}_{+1},\texttt{NTape}} \cap \tilde{\Phi}^{-1}_{\texttt{Tape}}(\mathcal{A}^{\mathbb{Z}}_{\mathbf{k},S,T,U})$.

Finally, if $c \in F^{-2T}_{\mathbf{k},S,T,U}(\mathcal{A}^{\mathbb{Z}}_{\mathbf{k},S,T,U})$ then $F^t\sigma^s(c) \in \Sigma^{0,0,S,T}$, for some $s \in [\![0,S[\![$ and $t \in [\![0,T[\![$. Therefore, $F^t_{\mathbf{k},S,T,U}\sigma^s(c) \in \Sigma^{0,0,S,T} \cap F^{-T}_{\mathbf{k},S,T,U}(\mathcal{A}^{\mathbb{Z}}_{\mathbf{k},S,T,U})$ so that $F^t_{\mathbf{k},S,T,U}\sigma^s(c) \in \Phi^{-1}(\mathcal{A}^{\mathbb{Z}}_{\mathbf{k},S,T,U})$. This implies that $\Omega_{F_{\mathbf{k},S,T,U}} \subseteq F^{-2T}_{\mathbf{k},S,T,U}(\mathcal{A}^{\mathbb{Z}}_{\mathbf{k},S,T,U}) \subseteq \bigsqcup_{\substack{0\le t<T\\ 0\le s<S}} F^t\sigma^s(\mathcal{D}(\Phi))$, which means that the simulation is also complete. □

6.2.1 Satisfying the inequalities

It is not as straightforward to see that the inequalities $\mathcal{I}(\mathbf{k},\mathbf{k},S,T,U,p,p^{-1})$ can be satisfied as it was for Lemma 33, because in this case $\mathbf{k}'$ is equal to $\mathbf{k}$, which means that we cannot fix $\mathbf{k}'$ and then choose $\mathbf{k},S,T,U$ sufficiently big.

Remark 39. *We can find $\mathbf{k},S,T,U$ such that the inequalities of Lemma 38 are satisfied. In addition, for all $\epsilon > 0$, S/T can be made larger than $1-\epsilon$. (Intuitively, the macro-tiles can be made as close to a square as we want.)*

Proof. We have to satisfy the following inequalities:

$$\begin{cases} U \geq \max\{t_p(\mathbb{F}_5^{\mathbf{k}}), t_{p^{-1}}(\mathbb{F}_5^{\mathbf{k}})\} \\ S \geq \max\{2U, \left\|\chi\left(\mathbb{F}_5^{\mathbf{k}}\right)\right\|\} \\ T \geq 4U + S \\ k_{\texttt{Addr}}, k_{\texttt{Addr}_{+1}} \geq \|S\| \\ k_{\texttt{Clock}}, k_{\texttt{Clock}_{+1}} \geq \|T\| \\ k_{\texttt{Head}_{-1}}, k_{\texttt{Head}_{+1}} \geq \left|\chi\left(\mathbb{F}_4 \times (Q_p \cup Q_{p^{-1}}) \times \{-1, +1\}\right)\right| \\ k_{\texttt{Tape}}, k_{\texttt{NTape}}, k_{\texttt{Tape}_{-1}}, k_{\texttt{Tape}_{+1}} \geq 1 \\ k_{\texttt{Prog}} = \|p\| \\ k_{\texttt{RProg}} = \left\|p^{-1}\right\| \\ k_{\texttt{MAddr}} = \|S\| \\ k_{\texttt{MClock}} = \|T\| \\ k_{\texttt{Alarm}} = \|U\| \ . \end{cases}$$

For all S, T, U, let us choose $\mathbf{k} := \mathbf{k}_{S,T,U}$ such that all of the inequalities except the first three are equalities. Then, $\left\|\chi\left(\mathbb{F}_5^{\mathbf{k}}\right)\right\| \leq P_1(\log S, \log T, \log U)$ and $\max\{t_p(\mathbb{F}_5^{\mathbf{k}}), t_{p^{-1}}(\mathbb{F}_5^{\mathbf{k}})\} \leq P_2(\log S, \log T, \log U)$, for some polynomials P_1, P_2. These follow by definition of $\chi(\cdot)$ and $\mathbf{k}_{S,T,U}$ and the fact that the program p is fixed and has polynomial complexity.

Therefore, it is enough to find S, T, U that satisfy the following inequalities:

$$\begin{cases} U \geq P_2(\log S, \log T, \log U) \\ S \geq \max\{2U, P_1(\log S, \log T, \log U)\} \\ T \geq 4U + S \ . \end{cases}$$

For all S, U, let us choose $T := T_{S,U} = S + 4U$. Also, for all S, S_0, r, let us choose $U := U_{S,S_0,r} = \log^r(S + S_0)$. Then, the third inequality is satisfied and the other two are written as follows:

$$\begin{cases} \log^r(S + S_0) \geq & P_2(\log S, \log(S + 4\log^r(S + S_0)), \log(\log^r(S + S_0)) \\ S \geq & \max\{2\log^r(S + S_0), \\ & P_1(\log S, \log(S + 4\log^r(S + S_0)), \log(\log^r(S + S_0)))\}. \end{cases}$$

There exist *fixed* $r, S_0 \in \mathbb{N}$ such that the first inequality is satisfied for all S, because P_2 is a fixed polynomial (hence its degree is a fixed number). Let us choose such r, S_0. Then, if S is sufficiently large we also have that the second inequality is also satisfied (since r, S_0 are now fixed), because the right hand side grows only polylogarithmically in S, which finishes the proof of the claim.

For the second claim, $S/T = \frac{S}{S + \log^r(S + S_0)}$, which can be made larger than $1 - \epsilon$ by choosing S sufficiently large. □

Corollary 40. *There exists an RPCA G such that $\mathcal{O}_G$ is non-empty, aperiodic and $\mathcal{N}(G) = \{0\}$.*

Proof. Let $G := F_{\mathbf{k},S,T,U} \colon \mathcal{A}^{\mathbb{Z}}_{\mathbf{k},U,S,T} \to \mathcal{A}^{\mathbb{Z}}_{\mathbf{k},U,S,T}$, for some parameters that satisfy $\mathcal{I}(\mathbf{k}, \mathbf{k}, S, T, U, p, p^{-1})$. This is possible, according to Remark 39. By definition, we have that $S < T$.

It is not difficult to see that $G^{-1}(\mathcal{A}^{\mathbb{Z}}_{\mathbf{k},U,S,T})$ is nonempty. Then, Lemmas 38, 20, 18 and Proposition 23 imply that $\mathcal{O}_G$ is non-empty, uncountable, aperiodic and $\mathcal{N}(G) = \{0\}$. □

This finishes the construction of an extremely-expansive, aperiodic 2D SFT. Once we achieved self-simulation, then extreme expansiveness follows immediately from Proposition 23.

6.3 Hierarchical simulation

We now want to construct more general nested hierarchical simulations, where the parameters of the simulation might vary in every simulation level. This structure is more flexible than a simple self-simulating RPCA, and it will be more useful in the various applications.

Let us fix the field list $\mathcal{C}_{HSimul} := \mathcal{C}_{Simulate} \sqcup [\texttt{Level}, \texttt{Prog}, \texttt{RProg}]$ and let ν_{HSimul} be the corresponding vector of directions.

- `Prog` and `RProg` are used as in the previous section.
- `Level` is used to obtain the values of $v_{\texttt{MAddr}}, v_{\texttt{MClock}}$ and $v_{\texttt{Alarm}}$, not through a direct projection, as in the previous case, but in a polynomially computable way.

In the following, let $\mathfrak{S}, \mathfrak{T}, \mathfrak{U} \in \mathbb{N}^{\mathbb{N}}$ be sequences of integers and and $(\mathbf{k} \colon \mathbb{N} \to \mathbb{N}^M)_{n \in \mathbb{N}}$ is a *sequence* of vectors depending on n (It can give rise to a vector valuation by using $\underline{\pi_{\texttt{Field}}}$ as the index of the sequence).

$HSimul[M, \nu, \mathbf{k}, \mathfrak{S}, \mathfrak{T}, \mathfrak{U}]$

1: **if** $\underline{\pi_{\texttt{Clock}}} = 0$ **then**
2: $\quad Check[\mathcal{S}^{\epsilon}_{\texttt{Head}_{-1},\texttt{Head}_{+1},\texttt{Tape}_{-1},\texttt{Tape}_{+1},\texttt{NTape}}]$
3: $\quad CkAlph[M, \underline{\pi_{\texttt{Addr}}}, \pi_{\texttt{Tape}}, \mathbf{k}_{\underline{\pi_{\texttt{Level}}}+1}]$ {Check that the lengths of the simulated letter are correct}
4: $\quad HCheck[M, \underline{\pi_{\texttt{Addr}}}, \pi_{\texttt{Tape}}, \mathbf{k}_{\underline{\pi_{\texttt{Level}}}+1}, \texttt{Prog}, \pi_{\texttt{Prog}}]$ {`Prog` of the simulated letter is the same}
5: $\quad HCheck[M, \underline{\pi_{\texttt{Addr}}}, \pi_{\texttt{Tape}}, \mathbf{k}_{\underline{\pi_{\texttt{Level}}}+1}, \texttt{RProg}, \pi_{\texttt{RProg}}]$ {`RProg` is also the same}
6: $\quad HCheck[M, \underline{\pi_{\texttt{Addr}}}, \pi_{\texttt{Tape}}, \mathbf{k}_{\underline{\pi_{\texttt{Level}}}+1}, \texttt{Level}, \overline{\underline{\pi_{\texttt{Level}}} + 1}]$ {`Level` of the simulated letter increases by 1}
7: **end if**

8: $Simulate[M,\ \nu,\ \mathbf{k}_{\underline{\pi_{\texttt{Level}}}},\ \underline{\pi_{\texttt{Addr}}}\ ,\underline{\pi_{\texttt{Clock}}},\ \mathfrak{S}_{\underline{\pi_{\texttt{Level}}}},\ \mathfrak{T}_{\underline{\pi_{\texttt{Level}}}},\ \mathfrak{U}_{\underline{\pi_{\texttt{Level}}}},\ \pi_{\texttt{Prog}},\ \pi_{\texttt{RProg}}]$ {Simulate}
9: $Grid[\mathfrak{S}_{\underline{\pi_{\texttt{Level}}}}, \mathfrak{T}_{\underline{\pi_{\texttt{Level}}}}]$

We will now construct a nested simulation of RPCA where the simulation parameters are different at every level.

Lemma 41. *Let $\mathfrak{U}, \mathfrak{S}, \mathfrak{T}$ be polynomially checkable sequences of integers. Let us fix the field list $\mathcal{C}_{HSimul} := [0, \dots, 12]$, the corresponding* fixed *direction vector ν_{HSimul} and a polynomially checkable sequence of* 13*-uples $\mathbf{k} \in (\mathbb{N}^{13})^{\mathbb{N}}$. Let F be the IPPA with directions ν_{HSimul} and permutation $HSimul[13, \nu_{HSimul}, \mathbf{k}, \mathfrak{S}, \mathfrak{T}, \mathfrak{U}; \mathcal{C}_{HSimul}]$ and p, p^{-1} be the programs for this permutation and its inverse, respectively.*

For all $n \in \mathbb{N}$, let F_n be the restriction of F to the subalphabet

$$\mathcal{A}_n := \mathbb{F}_5^{\mathbf{k}_n} \cap \mathcal{S}^n_{\mathit{Level}} \cap \mathcal{S}^p_{\mathit{Prog}} \cap \mathcal{S}^{p^{-1}}_{\mathit{RProg}},$$

and assume that the following inequalities hold for all $n \in \mathbb{N}$:

$$\begin{cases} \mathcal{I}(\mathbf{k}_n, \mathbf{k}_{n+1}, S_n, T_n, U_n, p, p^{-1}) \\ k_{n,\mathit{Prog}} \geq \|p\| \\ k_{n,\mathit{RProg}} \geq \left\|p^{-1}\right\| \\ k_{n,\mathit{Level}} \geq \|n\| \ . \end{cases}$$

Then, $F_n \underset{S_n,T_n,0,\Phi_n}{\succeq} F_{n+1}$ completely exactly, where $\Phi_n := \tilde{\Phi}_{\mathit{Tape}|\Sigma_n}$ and $\Sigma_n := \mathcal{A}_n^{\mathbb{Z}} \cap \Sigma^{0,0,S_n,T_n} \cap \mathcal{S}^{\epsilon}_{\mathit{Head}_{-1},\mathit{Head}_{+1},\mathit{Tape}_{-1},\mathit{Tape}_{+1},\mathit{NTape}} \cap \tilde{\Phi}^{-1}(\mathcal{A}_{n+1}^{\mathbb{Z}})$

The proof is very similar to the proof of Lemma 38. There exists some differences, though. For example, we do not have the fields `MAddr`, `MClock` and `Alarm`. These fields are computed with the aid of field `Level`, so we perform a hierarchical check for `Level`. Apart from that, the proof follows the same pattern.

Another difference, which will be important when we prove that the inequalities can be satisfied is that the program is not fixed once we fix M and ν, as in the self-similar case, but depends on $\mathbf{k}, \mathfrak{S}, \mathfrak{T}$ and $\mathfrak{U}$. Therefore, its complexity also depends on these parameters. More precisely, $t_p(\mathcal{A}_n) = P(|\mathcal{A}_n| + t_{\mathbf{k}}(n) + t_{\mathfrak{S}}(n) + t_{\mathfrak{T}}(n) + t_{\mathfrak{U}}(n))$, for some polynomial P that does not depend on the parameters. This is due to the fact that the program consists in a bounded number (independent of n) of polynomially computable functions and a bounded number of calls to the parameters. Similarly, $|p| = O(|p_{\mathbf{k}}| + |p_{\mathfrak{S}}| + |p_{\mathfrak{T}}| + |p_{\mathfrak{U}}|)$. (The same things hold for p^{-1}.)

Proof. Let us fix $n \in \mathbb{N}$. We have to show three things: that F_n $(S_n, T_n, 0)$-simulates F_{n+1} with decoding function Φ_n (simulation), that Φ_n is injective (exactness) and that $\Omega_{F_n} \subseteq \mathcal{D}(\Phi_n)$ (completeness).

For the simulation part, let $b \in \mathcal{A}_{n+1}^{\mathbb{Z}}$ and $c \in \Phi^{-1}(b) \in \mathcal{A}_n^{\mathbb{Z}} \cap \Sigma^{0,0,S_n,T_n} \cap \mathcal{S}^{\epsilon}_{\mathtt{Head}_{-1},\mathtt{Head}_{+1},\mathtt{Tape}_{-1},\mathtt{Tape}_{+1},\mathtt{NTape}}$. By definition, c is not rejected by the checks of lines 2,3,4, 5 and 6, .

Since c is not rejected by these checks and F_n factors onto

$$Simulate[13, \nu_{Simulate}, (|\pi_j|)_{j<M}, \underline{\pi_{\mathtt{Addr}}}, \underline{\pi_{\mathtt{Clock}}}, S_n, T_n, U_n, p, p^{-1}]$$
$$Grid[S_n, T_n]$$

and, by assumption, the inequalities of Lemma 33 are satisfied by $\mathbf{k}_n$ and $\mathbf{k}_{n+1}$, and p is the program of $HSimul[13, \nu_{HSimul}, \mathbf{k}, \mathfrak{S}, \mathfrak{T}, \mathfrak{U}]$, Lemma 33 gives that F_n $(S_n, T_n, 0)$-simulates F_{n+1} with decoding function Φ_n.

For the exactness part, we have already noted various times that the values of the fields in $\mathcal{C}_{Simulate}$ are uniquely determined for all $c \in \Phi^{-1}(b)$. For the hierarchical fields (*i.e.*, $\mathtt{Level}, \mathtt{Prog}, \mathtt{RProg}$) the values are fixed for all $c \in \mathcal{A}_n^{\mathbb{Z}}$. In addition, there do not exist any anonymous fields (since we chose $M = 13$). Therefore, Φ_n is injective and the simulation is exact.

For the completeness part, we will only show that if $c \in \Sigma^{0,0,S_n,T_n} \cap F_n^{-T}(\mathcal{A}_n^{\mathbb{Z}})$, then $c \in \Phi^{-1}(\mathcal{A}_{n+1}^{\mathbb{Z}})$. Having shown this, it is easy to conclude that the simulation is complete using the same argument as in the proof of Lemma 38

Indeed, line 2 checks that $c \in \Sigma^{0,0,S_n,T_n} \cap \mathcal{S}^{\epsilon}_{\mathtt{Head}_{-1},\mathtt{Head}_{+1},\mathtt{Tape}_{-1},\mathtt{Tape}_{+1},\mathtt{NTape}}$. According to Lemma 36, line 3 checks that for every colony B_i^c, $\pi_{\mathtt{Tape}}(B_i^c)$ has the structure of the encoding of a letter in $\mathbb{F}_5^{\mathbf{k}_{n+1}}$. In addition, since $F_n^{T_n}(c)$ exists and the equations $\mathcal{I}(\mathbf{k}_n, \mathbf{k}_{n+1}, S_n, T_n, U_n, p, p^{-1})$ are satisfied, this means that the computation of p on input $\pi_{\mathtt{Tape}}(B_i^c)$ halts, therefore for all $i \in \mathbb{Z}$, $\tilde{\phi}_{\mathtt{Tape}}(B_i^c) = b_i \in \mathbb{F}_5^{**}$. Lemma 36 now implies that $\tilde{\phi}_{\mathtt{Tape}}(B_i^c) = b_i \in \mathbb{F}_5^{\mathbf{k}_{n+1}}$. Finally, lines 6,4 and 5 check that $\tilde{\phi}_{\mathtt{Tape}}(b_i) \in \mathcal{S}^{n+1}_{\mathtt{Level}} \cap \mathcal{S}^{p}_{\mathtt{Prog}} \cap \mathcal{S}^{p^{-1}}_{\mathtt{RProg}}$.

Summarizing, we have that $b \in \mathcal{A}_{n+1}^{\mathbb{Z}}$, so that $c \in \mathcal{A}_n^{\mathbb{Z}} \cap \Sigma^{0,0,S_n,T_n} \cap \mathcal{S}^{\epsilon}_{\mathtt{Head}_{-1},\mathtt{Head}_{+1},\mathtt{Tape}_{-1},\mathtt{Tape}_{+1},\mathtt{NTape}} \cap \tilde{\Phi}^{-1}_{\mathtt{Tape}}(\mathcal{A}_{n+1}^{\mathbb{Z}}) = \Sigma_n$, or equivalently that $c \in \Phi^{-1}(\mathcal{A}_{n+1}^{\mathbb{Z}})$. □

6.3.1 Satisfying the inequalities

Let us now show that the inequalities of Lemma 41 can be satisfied:

Remark 42. *We can find* $\mathbf{k} \in (\mathbb{N}^{13})^{\mathbb{N}}$ *and* $\mathfrak{S}, \mathfrak{T}, \mathfrak{U} \in \mathbb{N}_1^{\mathbb{N}}$ *such that the inequalities of Lemma 41 are satisfied. In addition,* $\prod_{i<n} S_i/T_i$ *can be made both* 0 *and* $\neq 0$*.*

We have to deal with two problems, which were not present in the previous cases: First, there is an infinite set of inequalities, since there is also an infinite set of RPCA, and they must all be satisfied simultaneously. Second, the size of the program and the complexity of the permutations depends on the choice of the parameters $\mathfrak{S}$ and $\mathfrak{T}$.

Proof. We have to satisfy the following inequalities, for all $n \in \mathbb{N}$

$$\begin{cases} U_n \geq \max\{t_p(\mathbb{F}_5^{\mathbf{k}_{n+1}}), t_{p^{-1}}(\mathbb{F}_5^{\mathbf{k}_{n+1}})\} \\ S_n \geq \max\{2U_n, \left\|\chi\left(\mathbb{F}_5^{\mathbf{k}_{n+1}}\right)\right\|\} \\ T_n \geq 4U_n + S_n \\ k_{n,\texttt{Prog}} \geq \|p\| \\ k_{n,\texttt{RProg}} \geq \left\|p^{-1}\right\| \\ k_{n,\texttt{Head}_{-1}}, k_{n,\texttt{Head}_{+1}} \geq \left|\chi\left(\mathbb{F}_4 \times (Q_p \cup Q_{p^{-1}}) \times \{-1,+1\}\right)\right| \\ k_{n,\texttt{Addr}}, k_{n,\texttt{Addr}_{+1}} \geq \|S_n\| \\ k_{n,\texttt{Clock}}, k_{n,\texttt{Clock}_{+1}} \geq \|T_n\| \\ k_{n,\texttt{Tape}}, k_{n,\texttt{NTape}}, k_{n,\texttt{Tape}_{-1}}, k_{n,\texttt{Tape}_{+1}} \geq 1 \\ k_{n,\texttt{Level}} = \|n\| \ . \end{cases}$$

For all $n \in \mathbb{N}$ and $\mathfrak{S}, \mathfrak{T}$ and $\mathfrak{U}$, let us choose $\mathbf{k}_n := \mathbf{k}_{n,\mathfrak{S},\mathfrak{T},\mathfrak{U}}$ such that the last four inequalities are satisfied as equalities. Then, we can see that

$$\left\|\chi\left(\mathbb{F}_5^{\mathbf{k}_n}\right)\right\| \leq P_1(\log S_n, \log T_n, \log n, k_{n,\texttt{Head}_{-1}}, k_{n,\texttt{Head}_{+1}}, k_{n,\texttt{RProg}}, k_{n,\texttt{Prog}}),$$

for some polynomial P_1.

We claim that $|p| \leq c(|p_{\mathbf{k}}| + |p_{\mathfrak{S}}| + |p_{\mathfrak{T}}| + |p_{\mathfrak{U}}|)$, for some constant c. (The same holds for p^{-1} and we can assume that the constant c is the same.) This is because, as we have already noticed, the program of p uses a fixed number of polynomial operations and a bounded number of calls to the parameters $|p_{\mathbf{k}}|$, $|p_{\mathfrak{S}}|$, $|p_{\mathfrak{T}}|$, $|p_{\mathfrak{U}}|$.

For the same reason, we have that

$$\begin{aligned} \max\{t_p(\mathbb{F}_5^{\mathbf{k}}), t_{p^{-1}}(\mathbb{F}_5^{\mathbf{k}})\} \leq P_2(&\log S_n, \log T_n, \log n, k_{n,\texttt{Head}_{-1}}, \\ &k_{n,\texttt{Head}_{+1}}, k_{n,\texttt{RProg}}, k_{n,\texttt{Prog}}, t_{\mathbf{k}}(n), t_{\mathfrak{S}}(n), , t_{\mathfrak{T}}(n), t_{\mathfrak{U}}(n)), \end{aligned}$$

for some *fixed* polynomial P_2 that does not depend on the parameter sequences.

Therefore, it is enough to find sequences $\mathbf{k}, \mathfrak{S}, \mathfrak{T}, \mathfrak{U}$ that satisfy the fol-

lowing inequalities, for all $n \in \mathbb{N}$:

$$\begin{cases} k_{n,\texttt{Prog}}, k_{n,\texttt{RProg}} \geq & c(|p_{\mathbf{k}}| + |p_{\mathfrak{S}}| + |p_{\mathfrak{T}}| + |p_{\mathfrak{U}}|) \\ k_{n,\texttt{Head}_{-1}}, k_{n,\texttt{Head}_{+1}} \geq & \left|\chi\left(\mathbb{F}_4 \times (Q_p \cup Q_{p^{-1}}) \times \{-1,+1\}\right)\right| \\ U_n \geq & P_2(\log S_{n+1}, \log T_{n+1}, \log(n+1), \\ & k_{n+1,\texttt{Head}_{-1}}, k_{n+1,\texttt{Head}_{+1}}, k_{n+1,\texttt{RProg}}, k_{n+1,\texttt{Prog}}, \\ & t_{\mathbf{k}}(n+1), t_{\mathfrak{S}}(n+1), t_{\mathfrak{T}}(n+1), t_{\mathfrak{U}}(n+1)) \\ S_n \geq & \max\{2U_n, P_1(\log S_{n+1}, \log T_{n+1}, \log(n+1), \\ & |p_{\mathbf{k}}|, |p_{\mathfrak{S}}|, |p_{\mathfrak{T}}|, |p_{\mathfrak{U}}|)\} \\ T_n \geq & 4U_n + S_n \ . \end{cases}$$

Recall that in the above inequalities, Q_p and $Q_{p^{-1}}$ depend on the choice of parameter sequences.

We will show two ways to do this. The first one does not give an extremely-expansive SFT, because $\prod_{i<n} S_i/T_i$ does not converge to 0, while the second one does.

1. For all sequences $\mathfrak{S}$ and $\mathfrak{U}$, let us choose $T_n := T_{n,\mathfrak{S},\mathfrak{U}} = S_n + 4U_n$. Also, for all n_0, r and $Q \geq 2$, let us choose $U_{n,n_0,r} := U_n = (n+n_0)^r$, $S_{n,n_0} := S_n = Q^{n+n_0}$, $k_{n,\texttt{Prog}} = k_{n,\texttt{RProg}} = n_0 Q r$ and $k_{n,\texttt{Head}_{-1}} = k_{n,\texttt{Head}_{+1}} = n_0$ for all $n \in \mathbb{N}$.

   Then, the last inequality is satisfied by definition. In addition, for all n_0, r, Q, we have that $|p_{\mathfrak{S}}| \leq \|c_1 n_0 Q\|$, $|p_{\mathfrak{U}}| \leq \|c_2 r n_0\|$ and $|p_{\mathfrak{T}}|$, $|p_{\mathbf{k}}| \leq \|c_3 r n_0 Q\|$, for some *constants* c_1, c_2, c_3. This is true because the sequence $(n+n_0)^r$ is uniformly (polynomially) computable in n, n_0, r, which means that there exists an algorithm that takes as input (n, n_0, r) and outputs $(n+n_0)^r$, for *all* values of n, n_0 and r. If we use the program for this algorithm together with a description of n_0 and r, then we obtain a program of length bounded by $\|c_2 r n_0\|$ for the sequence $((n+n_0)^r)_{n\in\mathbb{N}}$. A similar argument holds for the sequence Q^{n+n_0}.

   Since this algorithm works for *all* choices of n_0, Q, r, it means that Q_p and $Q_{p^{-1}}$ are actually *fixed*.

   In addition, all of the algorithms are polynomially computable, which means that

   $$t_{\mathfrak{S}}(n), t_{\mathfrak{T}}(n), t_{\mathbf{k}}(n) \leq P_3(\log Q^{n+n_0}), t_{\mathfrak{U}}(n) \leq P_4(\log(n+n_0)^r),$$

   for some *fixed* polynomials P_3, P_4.

   Therefore, substituting these in the inequalities above and doing some regrouping of the terms in parentheses (that is omitted), the inequalities that need to be satisfied are written as follows:

$$\begin{cases} n_0 Q r \geq & c' \log(n_0 Q r) \\ n_0 \geq & c' \\ (n+n_0)^r \geq & P_5(\log Q^{n+n_0+1}, \log(n+n_0+1)^r, \log(n+1)) \\ Q^{n+n_0} \geq & \max\{2(n+n_0+1)^r, P_6(\log Q^{n+n_0+1}, \log(n+1))\} \ , \end{cases}$$

for some polynomials P_5, P_6 and constant c' that do not depend on r, n_0 or Q.

Since c' is fixed, the first two inequalities are true for all but a finite number of triples n_0, Q, r. Without loss of generality, we assume that it is always true. We can choose n_Q and r such that the second inequality is true for all $n \in \mathbb{N}$ and all $n_0 \geq n_Q$, because the right hand of the inequality is bounded by a fixed polynomial of $(n+n_0)$ and r, while the left-hand side grows like n^r. With fixed r, we can also find n'_Q such that the third inequality is satisfied for all $n \in \mathbb{N}$ and all $n_0 \geq n'_Q$, because the left-hand side grows exponentially in $(n+n_0)$ and the right hand only polynomially (since r is fixed). By choosing $n_0 = \max\{n_Q, n'_Q\}$ we can satisfy both inequalities for all n at the same time.

Note that $\prod_{i\in\mathbb{N}} S_i/T_i = \prod_{i\in\mathbb{N}}(1+(n+n_0)^r/Q^{n+n_0}) \neq 0$. Therefore, if we choose the sequences like this, we do not obtain a unique direction of non-expansiveness, but rather a cone of non-expansive directions.

2. For all $n_0 \in \mathbb{N}$ and $Q \geq 2$, let us choose $S_{n,n_0} := S_n = Q^{n+n_0}$, $T_{n,n_0} := T_n = 2S_n$ and $U_{n,n_0} := U_n = \frac{S_n}{2Q}$, $k_{n,\texttt{Prog}} = k_{n,\texttt{RProg}} = n_0 Q$ and $k_{n,\texttt{Head}_{-1}} = k_{n,\texttt{Head}_{+1}} = n_0$ for all $n \in \mathbb{N}$. We can use a similar argumentation as in previous case to show that it is enough to satisfy the following inequalities:

$$\begin{cases} n_0 Q \geq \|Q n_0\| \\ \dfrac{Q^{n+n_0}}{4} \geq P_3(\log Q^{n+n_0+1}, \log(n+1)) \\ Q^{n+n_0} \geq \max\{\dfrac{Q^{n+n_0+1}}{2Q}, P_4(\log Q^{n+n_0+1}, \log(n+1))\} \ , \end{cases}$$

for some polynomials P_3, P_4 and constant c that do not depend on n_0 and Q.

Obviously, for all Q these inequalities are satisfied when n_0 is sufficiently large.

In this case, $\prod_{i\in\mathbb{N}} S_i/T_i = \prod_{i\in\mathbb{N}} S_i/2S_i = 0$, therefore the corresponding SFT is extremely expansive.

□

For both cases, we have a lot of freedom in choosing the sequences. In the previous proof, we just described two of the possible ways which are enough for the results we want to obtain and help in presenting the basic ideas of the proof that is needed in any possible case.

6.4 Universality

Let $\mathcal{C}_{\texttt{Other}} = [\texttt{OTape}_{-1}, \texttt{OTape}, \texttt{OTape}_{+1}]$ and let us fix the field list $\mathcal{C}_{Univ} = \mathcal{C}_{HSimul} \sqcup \mathcal{C}_{\texttt{Other}}$ and the corresponding direction vector ν_{Univ}.

For any n, consider an RPCA G_n with permutation $\alpha_n \colon (\mathbb{F}_5^{l_n})^3 \to (\mathbb{F}_5^{l_n})^3$ over $\mathcal{C}_{\texttt{Other}}$. This is not a strict restriction in itself: all RPCA can be represented in this way, up to a simple alphabet renaming and use of Remark 7

If the sequence of permutations $(\alpha_n)_{n\in\mathbb{N}}$ is polynomially computable, we can build a PPA that simulates G_n, for all $n \in \mathbb{N}$.

$Univ[M, \nu, \mathbf{k}, \mathfrak{S}, \mathfrak{T}, \mathfrak{U}, \alpha]$

1: $\alpha_{\underline{\texttt{Level}}}[\mathcal{C}_{\texttt{Other}}]$ {G_n on the $\mathcal{C}_{\texttt{Other}}$ fields.}
2: **if** $\pi_{\underline{\texttt{Clock}}} = 0$ **then**
3: $\quad Check[\mathcal{S}^{\epsilon}_{\texttt{Head}_{-1},\texttt{Head}_{+1},\texttt{Tape}_{-1},\texttt{Tape}_{+1},\texttt{NTape}}]$
4: $\quad CkAlph[M, \pi_{\underline{\texttt{Addr}}}, \pi_{\texttt{Tape}}, \mathbf{k}_{\pi_{\underline{\texttt{Level}}}+1}]$
5: $\quad HCheck[M, \pi_{\underline{\texttt{Addr}}}, \pi_{\texttt{Tape}}, \mathbf{k}_{\pi_{\underline{\texttt{Level}}}+1}, \texttt{Prog}, \pi_{\texttt{Prog}}]$
6: $\quad HCheck[M, \pi_{\underline{\texttt{Addr}}}, \pi_{\texttt{Tape}}, \mathbf{k}_{\pi_{\underline{\texttt{Level}}}+1}, \texttt{RProg}, \pi_{\texttt{RProg}}]$
7: $\quad HCheck[M, \pi_{\underline{\texttt{Addr}}}, \pi_{\texttt{Tape}}, \mathbf{k}_{\pi_{\underline{\texttt{Level}}}+1}, \texttt{Level}, \overline{\pi_{\underline{\texttt{Level}}} + 1}]$
8: **end if**
9: $Simulate[M,\ \nu,\ \mathbf{k}_{\pi_{\underline{\texttt{Level}}}},\ \pi_{\underline{\texttt{Addr}}},\ \pi_{\underline{\texttt{Clock}}},\ \mathfrak{S}_{\pi_{\underline{\texttt{Level}}}},\ \mathfrak{T}_{\pi_{\underline{\texttt{Level}}}},\ \mathfrak{U}_{\pi_{\underline{\texttt{Level}}}},\ \pi_{\texttt{Prog}},\ \pi_{\texttt{RProg}}]$ {Simulate}
10: $Grid[\mathfrak{S}_{\pi_{\underline{\texttt{Level}}}}, \mathfrak{T}_{\pi_{\underline{\texttt{Level}}}}; \mathcal{C}_{Grid}]$

The only difference of this rule with $HSimul$ is that it has 3 additional fields (which implies that $\mathbf{k}$ will be chosen in $(\mathbb{N}^{16})^{\mathbb{N}}$) and that we apply $\alpha_{\underline{\texttt{Level}}}$ onto the field list $\mathcal{C}_{\texttt{Other}}$ *independently* from what we do on $\mathcal{C}_{HSimul}$.

Lemma 43. *Let $\mathfrak{U}, \mathfrak{S}, \mathfrak{T}$ be polynomially checkable sequences of integers and α a polynomially computable sequence of permutations. Let us fix the field list $\mathcal{C}_{Univ} := [0, \dots, 15]$, the corresponding* fixed *direction vector ν_{Univ} and a polynomially checkable sequence of M-uples $\mathbf{k} \in (\mathbb{N}^{15})^{\mathbb{N}}$. Let F be the IPPA with directions ν_{Univ} and permutation $Univ[15, \nu_{Univ}, \mathbf{k}, \mathfrak{S}, \mathfrak{T}, \mathfrak{U}; \mathcal{C}_{Univ}]$ and p, p^{-1} be the programs for this permutation and its inverse, respectively.*

For all $n \in \mathbb{N}$, let F_n be the restriction of F to the subalphabet

$$\mathcal{A}_n := \mathbb{F}_5^{\mathbf{k}_n} \cap \mathcal{S}^n_{Level} \cap \mathcal{S}^p_{Prog} \cap \mathcal{S}^{p^{-1}}_{RProg},$$

and assume that the following inequalities hold:

$$\begin{cases} \mathcal{I}(\mathbf{k}_n, \mathbf{k}_{n+1}, S_n, T_n, U_n, p) \\ k_{n,\mathit{Prog}} \geq |p| \\ k_{n,\mathit{RProg}} \geq \left|p^{-1}\right| \\ k_{n,\mathit{Level}} \geq \|n\| \\ k_{n,\mathit{OTape}} = k_{n,\mathit{OTape}_{-1}} = k_{n,\mathit{OTape}_{+1}} \geq l_n \ . \end{cases}$$

If $\Omega_{G_n} \neq \emptyset$, then F_n completely $(S_n, T_n, 0)$-simulates F_{n+1} with decoding function $\Phi_n = \tilde{\Phi}_{\mathit{Tape}|\Sigma_n}$, where

$$\Sigma_n := \mathcal{A}_n^{\mathbb{Z}} \cap \Sigma^{0,0,S,T} \cap \mathcal{S}^{\epsilon}_{\mathit{Head}_{-1},\mathit{Head}_{+1},\mathit{Tape}_{-1},\mathit{Tape}_{+1},\mathit{NTape}} \cap \Phi^{-1}(\mathcal{A}_{n+1}^{\mathbb{Z}}).$$

The simulation is exact if and only if Ω_{G_n} is a singleton.
In addition, if $c \in F_n^{-1}(\mathcal{A}_n^{\mathbb{Z}})$, then $G_n \pi_{\mathcal{C}_{\mathit{Other}}}(c) = \pi_{\mathcal{C}_{\mathit{Other}}} F_n(c)$.

As mentioned before, the proof is very similar to the proof of Lemma 41. Therefore, we are going to omit most of the details and only stress those points where there is a difference.

Proof. Let us fix $n \in \mathbb{N}$. For the first claim, we have to show that F_n $(S_n, T_n, 0)$-simulates F_{n+1} with decoding function Φ_n (simulation), that Φ_n is an injection if and only if Ω_{G_n} is a singleton and that $\Omega_{F_n} \subseteq \mathcal{D}(\Phi_n)$ (completeness).

For the simulation part, let $b \in \mathcal{A}_{n+1,p}^{\mathbb{Z}}$. We can find $c \in \mathcal{A}_n^{\mathbb{Z}}$ that simulates b: we choose $c \in \Phi^{-1}(b) \in \mathcal{A}_n^{\mathbb{Z}} \cap \Sigma^{0,0,S,T} \cap \mathcal{S}^{\epsilon}_{\mathtt{Head}_{-1},\mathtt{Head}_{+1},\mathtt{Tape}_{-1},\mathtt{Tape}_{+1},\mathtt{NTape}}$ such that $\pi_{\mathcal{C}_{\mathtt{Other}}}(c) \in \Omega_{G_n}$ (this is possible by the assumption that $\Omega_{G_n} \neq \emptyset$). Then, it is easy to see that c simulates b, because $\mathcal{C}_{\mathtt{Other}}$ is only "touched" by $\alpha_{\underline{\mathtt{Level}}} := \alpha_n$, p is the program of $Univ[17, \nu_{Univ}, \mathbf{k}, \mathfrak{S}, \mathfrak{T}, \mathfrak{U}; \mathcal{C}_{Univ}]$ and $G_n^T(\pi_{\mathcal{C}_{\mathtt{Other}}}(c))$ exists.

For the exactness part, as usual $\pi_{\mathcal{C}_{HSimul}}(c)$ is uniquely determined by b. However, $\pi_{\mathcal{C}_{\mathtt{Other}}}(c)$ can be chosen independently from b to be any element of Ω_{G_n}, so that the simulation is exact if and only if Ω_{G_n} is a singleton.

Finally, for the completeness part, an argument almost identical to the argument in the proof of Lemma 41 shows that if $c \in \Sigma^{0,0,S,T} \cap F_n^{-T}(\mathcal{A}_n^{\mathbb{Z}})$, then $c \in \Phi^{-1}(\mathcal{A}_{n+1}^{\mathbb{Z}})$. As we know, this is enough to show that the simulation is complete.

The second claim, that if $c \in F_n^{-1}(\mathcal{A}_n^{\mathbb{Z}})$, then $G_n \pi_{\mathcal{C}_{\mathtt{Other}}}(c) = \pi_{\mathcal{C}_{\mathtt{Other}}} F_n(c)$ is straightforward from the definition of F_n, since the only rule that "touches" the fields $\mathcal{C}_{\mathtt{Other}}$ is G_n. □

Remark 44. *1. $\Omega_{F_0} \neq \emptyset$ if and only if $\Omega_{G_n} \neq \emptyset$, for all $n \in \mathbb{N}$.*

2. If $\Omega_{F_0} \neq \emptyset$, then F_0 completely simulates G_n for all $n \in \mathbb{N}$.

Proof. 1. If $\Omega_{G_n} = \emptyset$ for some $n \in \mathbb{N}$, then $\Omega_{F_n} = \emptyset$, so that since F_0 simulates F_n (by transitivity of simulation), we obtain that $\Omega_{F_0} = \emptyset$ by Lemma 20.

If, on the other hand, $\Omega_{G_n} \neq \emptyset$ for all $n \in \mathbb{N}$, then Lemma 18 gives that $\Omega_{F_0} \neq \emptyset$.

2. If $\Omega_{F_0} \neq \emptyset$, then the second claim of Lemma 43 implies that F_n factors onto G_n. Since F_0 simulates F_n, for all $n \in \mathbb{N}$, we obtain that F_0 simulates G_n, for all $n \in \mathbb{N}$.

□

Remark 45. *Even if $\prod_{i\in\mathbb{N}} S_i/T_i = 0$, F_0 is not necessarily extremely expansive, since we might have non-expansive directions coming from the G_n part. However, in the special case that Ω_{G_n} is a singleton for all $n \in \mathbb{N}$, then all the simulations are exact and it is straightforward to see that $\mathcal{N}(F_0) = \{0\}$, because Proposition 23 applies.*

6.4.1 Satisfying the inequalities

Remark 46. *We can find $\mathbf{k} \in (\mathbb{N}^{16})^{\mathbb{N}}$ and $\mathfrak{S}, \mathfrak{T}, \mathfrak{U} \in \mathbb{N}_1^{\mathbb{N}}$ such that the inequalities of Lemma 43 are satisfied and $\prod_{i\in\mathbb{N}} S_i/T_i = 0$.*

We only state the case $\prod_{i\in\mathbb{N}} S_i/T_i = 0$ (even though we can make it $\neq 0$) too, because it is what we will need and use in the applications.

Proof. Let us write explicitly the inequalities that we need to satisfy:

$$\begin{cases} U_n \geq \max\{t_p(\mathbb{F}_5^{\mathbf{k}_{n+1}}), t_{p^{-1}}(\mathbb{F}_5^{\mathbf{k}_{n+1}})\} \\ S_n \geq \max\{2U_n, \left|\chi\left(\mathbb{F}_5^{\mathbf{k}_{n+1}}\right)\right|\} \\ T_n \geq 4U_n + S_n \\ k_{n,\texttt{Prog}} \geq |p| \\ k_{n,\texttt{RProg}} \geq \left|p^{-1}\right| \\ k_{n,\texttt{Head}_{-1}}, k_{n,\texttt{Head}_{+1}} \geq \left|\chi\left(\mathbb{F}_4 \times (Q_p \cup Q_{p^{-1}}) \times \{-1,+1\}\right)\right| \\ k_{n,\texttt{Addr}}, k_{n,\texttt{Addr}_{+1}} \geq \|S_n\| \\ k_{n,\texttt{Clock}}, k_{n,\texttt{Clock}_{+1}} \geq \|T_n\| \\ k_{n,\texttt{Tape}}, k_{n,\texttt{NTape}}, k_{n,\texttt{Tape}_{-1}}, k_{n,\texttt{Tape}_{+1}} \geq 1 \\ k_{n,\texttt{Level}} = \|n\| \\ k_{n,\texttt{OTape}} = k_{n,\texttt{OTape}_{-1}} = k_{n,\texttt{OTape}_{+1}} = l_n \ . \end{cases}$$

For all $\mathfrak{S}, \mathfrak{T}$ and $\mathfrak{U}$, let us choose $\mathbf{k}_n := \mathbf{k}_{n,\mathfrak{S},\mathfrak{T},\mathfrak{U}}$ such that the last five inequalities are satisfied as equalities. Then, we have the crucial inequality

$$\left\|\chi\left(\mathbb{F}_5^{\mathbf{k}_n}\right)\right\| \leq P_1(\log S_n, \log T_n, \log n, l_n, \\ k_{n,\texttt{Head}_{-1}}, k_{n,\texttt{Head}_{+1}}, k_{n,\texttt{RProg}}, k_{n,\texttt{Prog}}),$$

where l_n is the size of the alphabet of α_n.

The other crucial inequality of the proof of Lemma 42 also holds without any essential changes:

$$\max\{t_p(\mathbb{F}_5^{\mathbf{k}}), t_{p^{-1}}(\mathbb{F}_5^{\mathbf{k}})\} \leq P_2(\log S_n, \log T_n, \log n, l_n, k_{n,\texttt{Head}_{-1}}, \\ k_{n,\texttt{Head}_{+1}}, k_{n,\texttt{RProg}}, k_{n,\texttt{Prog}}, t_{\mathbf{k}}(n), t_{\mathfrak{S}}(n), , t_{\mathfrak{T}}(n), t_{\mathfrak{U}}(n))$$

for some polynomial P_2 that does not depend on the parameters.

This holds because the permutation applied consists in a number of polynomial operations (recall that α is polynomially computable and fixed for this specific construction) and a bounded number of calls to $\mathfrak{S}$, $\mathfrak{T}$,$\mathfrak{U}$ and $\mathbf{k}$.

Also, since α is polynomially computable, l_n (which is part of the output of α_n) is also bounded by a polynomial of n so that we can "remove" l_n from the right-hand side of the previous inequalities and "incorporate" it in the polynomials P_1, P_2. From this point on, the proof is identical to the proof of Remark 42. (We are free to chose whether $\prod_{i\in\mathbb{N}} S_i/T_i$ is equal to 0 or not.) □

6.4.2 Domino problem

Theorem 47. *It is undecidable whether an extremely expansive SFT is empty.*

Proof. Let $\mathcal{M}$ be an arbitrary TM with program p'. For all $n \in \mathbb{N}$, we define α_n as follows:

$l_n = 1$. $\alpha_n(0,0,0) = (0,0,0)$ if $\mathcal{H}^n_{p'}(0^n)$ is true (*i.e.*, if $\mathcal{M}$ does not halt within n steps). α_n is undefined in all other cases.

$(\alpha_n)_{n\in\mathbb{N}}$ is a polynomially computable sequence of permutations. Ω_{G_n} is a singleton, equal to $\{^\infty 0^\infty\}$, if and only if $\mathcal{M}$ does not halt within n steps. Otherwise Ω_{G_n} is empty.

Let us construct the sequence of RPCA $(F_n)_{n\in\mathbb{N}}$ as in Lemma 43 corresponding to the α and $\mathbf{k}, \mathfrak{S}, \mathfrak{T}, \mathfrak{U}$ that satisfy the inequalities and for which $\prod_{i\in\mathbb{N}} S_i/T_i = 0$. Then, Remark 44 implies that Ω_{F_0} (equivalently, $\mathcal{O}_{F_0}$) is non-empty if and only if Ω_{G_n} is non-empty for all n, which is equivalent to that $\mathcal{M}$ does not halt over input 0^∞.

In addition, Remark 45 implies that if Ω_{F_0} is non-empty, then $\mathcal{N}(F_0) = \{0\}$.

Therefore, for every TM $\mathcal{M}$, we have constructed a 2D SFT $\mathcal{O}_{F_0}$ that is non-empty if and only if $\mathcal{M}$ does not halt over input 0^∞ and if $\mathcal{O}_{F_0} \neq \emptyset$, then $\mathcal{O}_{F_0}$ is extremely expansive. This concludes the proof of the undecidability. □

It follows from the previous proof that we have actually proved the following: Let A be the family of forbidden patterns that define empty SFT, and let B be the family that defines non-empty extremely expansive SFT (with unique direction of non- expansiveness ∞). There does not exists a recursively enumerable set X that contains B and is disjoint from A. In other words, if an algorithm correctly recognizes all non- empty extremely-expansive SFT, then it must also (falsely) recognize an empty SFT.

6.4.3 Intrinsic universality

The second application concerns the universality properties of RPCA.

Theorem 48. *For any computably enumerable set of non-empty PPA, there exists a PPA that completely simulates all of them.*

Proof. First of all, we can assume that all the PPA are over the field list $\mathcal{C}_{\texttt{Other}}$ with the corresponding directions. This is true because we can encode, in polynomial time, all the left-moving fields into a unique left-moving field, and similarly for the other types of fields. Then, saying that a set of PPA is computably enumerable is equivalent to saying that the corresponding set of permutations that define these PPA is computable enumerable.

In addition, for every computably enumerable set of PPA X (over the field list $\mathcal{C}_{\texttt{Other}}$), there exists a *polynomially computable sequence* $(G_n)_{n\in\mathbb{N}}$ of PPA that contains exactly the elements of X. Equivalently, there exists a *polynomially computable sequence* of permutations $(\alpha_n)_{n\in\mathbb{N}}$ that contains exactly the permutations of the PPA in X.

(Let g be a fixed element of X. The polynomial algorithm of $(\alpha_n)_{n\in\mathbb{N}}$ takes as input n, runs the algorithm that enumerates X for n steps and sets α_n equal to the last permutation that was output. If no permutation has yet been output, then α_n is set equal to g.)

If we use this sequence α and sequences $\mathbf{k}, \mathfrak{S}, \mathfrak{T}, \mathfrak{U}$ that satisfy the inequalities to define the sequence $(F_n)_{n\in\mathbb{N}}$ as in Lemma 43, then, since by assumption $\Omega_{G_n} \neq \emptyset$ for all $n \in \mathbb{N}$, Remark 44 implies that F_0 completely simulates G_n, for all $n \in \mathbb{N}$. □

Theorem 48 applies, up to a conjugacy, to computably enumerable sets of nonempty RPCA. In some sense, it gives a determinstc version of the result in [29]. The same result is not true for the non-computably-enumerable set of all nonempty RPCA, thanks to an argument by Hochman [17] and Ballier [2]. Nevertheless, the corollary applies to the family of all reversible (complete) cellular automata, since the family of RCA is computably enumerable. Unfortunately, it gives an RPCA (partial CA) that simulates all RCA (full CA) instead of an RCA. The existence of an RCA that simulates all RCA seems to be a much more difficult question and is still open, (see for instance [39].

6.5 Synchronizing computation

We now introduce one more trick in our construction: the encoding of an infinite sequence inside an infinite nested simulation by encoding increasing finite prefixes of the infinite sequence inside the alphabets of the RPCA of the nested simulation.

Let $\mathcal{C}_{SyncComput} := \mathcal{C}_{Simulate} \sqcup [\texttt{MHist}, \texttt{MHist}_{+1}, \texttt{Prog}, \texttt{RProg}]$. In this simulation, we do not use a field `Level` in order to store the parameter n. Instead, it will be obtained as the length of field `MHist`. p' is the program of a TM. It is used to reject some nested simulation sequences, depending on the infinite sequence that is stored (through its increasing finite prefixes) in the alphabets of the RPCA.

$SyncComput[M, \nu, \mathbf{k}, \mathfrak{S}, \mathfrak{T}, \mathfrak{U}, p']$

1: $Check[\pi_{\texttt{MHist}} = \pi_{\texttt{MHist}_{+1}}]$
2: **if** $\underline{\pi_{\texttt{Clock}}} = 0$ **then**
3: $\quad Check[\mathcal{H}_{p'}^{|\texttt{MHist}|}(\texttt{MHist})]$
4: $\quad Check[\mathcal{S}^{\epsilon}_{\texttt{Head}_{-1},\texttt{Head}_{+1},\texttt{Tape}_{-1},\texttt{Tape}_{+1},\texttt{NTape}}]$
5: $\quad CkAlph[M, \underline{\pi_{\texttt{Addr}}}, \pi_{\texttt{Tape}}, \mathbf{k}_{|\texttt{MHist}|+1}]$ {Check that the lengths of the simulated letter are correct}
6: $\quad HCheck[M, \underline{\pi_{\texttt{Addr}}}, \pi_{\texttt{Tape}}, \mathbf{k}_{|\texttt{MHist}|+1}, \texttt{Prog}, \pi_{\texttt{Prog}}]$ {`Prog` of the simulated letter is the same}
7: $\quad HCheck[M, \underline{\pi_{\texttt{Addr}}}, \pi_{\texttt{Tape}}, \mathbf{k}_{|\texttt{MHist}|+1}, \texttt{RProg}, \pi_{\texttt{RProg}}]$ {`RProg` is also the same}
8: $\quad HCheck[M, \underline{\pi_{\texttt{Addr}}}, \pi_{\texttt{Tape}}, \mathbf{k}_{|\texttt{MHist}|+1}, \texttt{MHist}, \pi_{\texttt{MHist}}]$ {`MHist` of the simulating letters is a prefix of `MHist` of the simulated}
9: **end if**
10: $Simulate[M, \nu, \mathbf{k}_{|\texttt{MHist}|}, \underline{\pi_{\texttt{Addr}}}, \underline{\pi_{\texttt{Clock}}}, \mathfrak{S}_{|\texttt{MHist}|}, \mathfrak{T}_{|\texttt{MHist}|}, \mathfrak{U}_{|\texttt{MHist}|}, \pi_{\texttt{Prog}}, \pi_{\texttt{RProg}}]$
11: $Grid[\mathfrak{S}_{|\texttt{MHist}|}, \mathfrak{T}_{|\texttt{MHist}|}]$

Lemma 49. *Let $\mathfrak{S}, \mathfrak{T}, \mathfrak{U}$ be polynomially checkable sequences of integers and p' be the program of a TM. Let us fix the field list $\mathcal{C}_{SyncComput} := [0, \ldots, 13]$, the corresponding* fixed *direction vector $\nu_{SyncComput}$ and a polynomially checkable sequence of 14-uples $\mathbf{k} \in (\mathbb{N}^{14})^{\mathbb{N}}$. Let F be the IPPA with directions $\nu_{SyncComput}$ and permutation*

$$SyncComput[14, \nu_{SyncComput}, \mathbf{k}, \mathfrak{S}, \mathfrak{T}, \mathfrak{U}, \pi_{\texttt{MHist}}, p']$$

and p, p^{-1} be the programs for this permutation and its inverse, respectively.

For all $w \in \mathbb{F}_2^*$, let $S_w := S_{|w|}$, $T_w := T_{|w|}$ and F_w be the restriction of F to the subalphabet

$$\mathcal{A}_w := \mathbb{F}_5^{\mathbf{k}_{|w|}} \cap \mathcal{S}^w_{\mathit{MHist},\mathit{MHist}_{+1}} \cap \mathcal{S}^p_{\mathit{Prog}} \cap \mathcal{S}^{p^{-1}}_{\mathit{RProg}},$$

and assume that the following inequalities hold for all $n \in \mathbb{N}$*:*

$$\begin{cases} \mathcal{I}(\mathbf{k}_n, \mathbf{k}_{n+1}, S_n, T_n, U_n, p, p^{-1}) \\ k_{n,\mathit{Prog}} \geq |p| \\ k_{n,\mathit{RProg}} \geq \left|p^{-1}\right| \\ k_{n,\mathit{Level}} \geq \|n\| \ . \end{cases}$$

Then, $F_w \underset{S_w,T_w,0,\Phi_w}{\succeq} \bigsqcup_{a\in\mathbb{F}_2} F_{wa}$ *completely exactly, where*

$$\Sigma_w := \mathcal{A}_w^{\mathbb{Z}} \cap \Sigma^{0,0,S_w,T_w} \cap \mathcal{S}^{\epsilon}_{\mathit{Head}_{-1},\mathit{Head}_{+1},\mathit{Tape}_{-1},\mathit{Tape}_{+1},\mathit{NTape}} \cap \tilde{\Phi}^{-1}(\bigsqcup_{a\in\mathbb{F}_2} \mathcal{A}_{wa}^{\mathbb{Z}}),$$

and $\Phi_w := \tilde{\Phi}_{\mathit{Tape}|\Sigma_w}$ *.*

Proof. Let $w \in \mathbb{F}_2^*$ and $|w| = n$. By definition, $S_w := S_n$, $T_w := T_n$ and $U_w := U_n$. If p' halts on input w within n steps, then the check of line 3 will reject every configuration, which means that $\mathcal{D}(F_w) = \emptyset$. But, in this case, wa will also be rejected by p' within $n+1$ steps, for all $a \in \mathbb{F}_2$, so that $\mathcal{D}(\bigsqcup_{a\in\mathbb{F}_2} F_{wa}) = \emptyset$, too. By definition, the empty PCA strongly, completely simulates itself for all possible choices of the simulating parameters, so that the claim is true in this case.

Suppose, then, that p' does not halt on input w within n steps. Then, the check of line 3 is always true, so that we can ignore it in the rest of the proof. As in the previous proofs, we have to show three things: that F_w $(S_n, T_n, 0)$-simulates $\bigsqcup_{a\in\mathbb{F}_2} F_{wa}$ with decoding function Φ_w (simulation), that Φ_w is injective (exactness) and that $\Omega_{F_w} \subseteq \mathcal{D}(\Phi_w)$ (completeness).

For the simulation, it is easy to see that if $b \in \mathcal{A}_{wa}^{\mathbb{Z}}$, where $a \in \mathbb{F}_2$ and $c \in \Phi_w^{-1}(b)$, then c is not rejected by the checks of lines 1,4,5,6, 7 and 8. Then, simulation follows easily from the choice of the program p and Lemma 33.

Exactness is also direct. The values of all the fields of c are uniquely determined by b and the form Φ_w.

Completeness also follows the general pattern of the previous proofs, but there is a small difference: we can show that if $c \in \Sigma^{0,0,S_n,T_n} \cap F_w^{-2T}(\mathcal{A}_w^{\mathbb{Z}})$ (the difference is that we have $2T$ instead of T in the exponent), then $c \in \Phi_w^{-1}(\bigsqcup_{a\in\mathbb{F}_2} \mathcal{A}_{wa}^{\mathbb{Z}})$. This is enough to ensure completeness of the simulation.

Indeed, if

$$c \in \Sigma^{0,0,S_n,T_n} \cap F_w^{-T}(\mathcal{A}_w^{\mathbb{Z}}),$$

then lines 4,6,7 and 8 ensure that

$$c \in \mathcal{A}_w^{\mathbb{Z}} \cap \Sigma^{0,0,S_n,T_n} \cap \mathcal{S}^{\epsilon}_{\mathtt{Head}_{-1},\mathtt{Head}_{+1},\mathtt{Tape}_{-1},\mathtt{Tape}_{+1},\mathtt{NTape}} \cap \Phi^{-1}((\bigsqcup_{a\in\mathbb{F}_2} \mathcal{A}_{wa})^{\mathbb{Z}}).$$

Let $b \in (\bigsqcup_{a\in\mathbb{F}_2} \mathcal{A}_{wa})^{\mathbb{Z}}$ be such that $c \in \Phi^{-1}(b)$. The problem is that we still cannot know that $\pi_{\texttt{MHist}}(b)$ is the same in all cells, because line 8 only checks that at every cell i, $\pi_{\texttt{MHist}}(c) = w$ (which we know that it is constant) is a prefix of $\pi_{\texttt{MHist}}(b_i)$. However, we could still have that $\pi_{\texttt{MHist}}(b_i) = w0$ and $\pi_{\texttt{MHist}}(b_j) = w1$, for some $i \neq j$. This is why we need to take $2T$ steps instead of T steps.

Indeed, since $F_w^{2T}(c)$ exists, this means that $F^2(b)$ exists, and line 1 ensures that $\pi_{\texttt{MHist}}(b_i) = \pi_{\texttt{MHist}_{+1}}(b_i) = \pi_{\texttt{MHist}}(b_j)$, for all $i, j \in \mathbb{Z}$ The argument for this is similar to the argument used in the proof of Lemma 25.Therefore, $b \in \bigsqcup_{a\in\mathbb{F}_2} \mathcal{A}_{wa}^{\mathbb{Z}}$ and this concludes the proof of the Lemma. □

6.5.1 Satisfying the inequalities

Remark 50. *We can find* $\mathbf{k} \in (\mathbb{N}^{14})^{\mathbb{N}}$ *and* $\mathfrak{S}, \mathfrak{T}, \mathfrak{U} \in \mathbb{N}_1^{\mathbb{N}}$ *such that the inequalities of Lemma 49 are satisfied and* $\prod_{i<n} S_i/T_i = 0$.

Proof. The proof is almost identical to the proof of Remark 46 and is omitted. We just make a few comments:

First of all, the inequalities depend on $w \in \mathbb{F}_2^*$, but in fact, if $|w| = |w'|$, then we have exactly the same inequalities for w and w', so that actually the inequalities can be translated to a set of inequalities that depend on n.

Second, notice that line 3 is computable in polynomial time, and since the program p' is fixed in advance, its contribution to $\|\mathcal{A}_w\|$, t_p and $|p|$ is constant and does not depend on the choice of parameters.

Finally, we can choose $k_{w,\texttt{MHist}} := n$, (where $n := |w|$) which means that this field only contributes a polynomial of n to the various inequalities, so that it can be "incorporated" into the polynomials and the problem can be reduced to the cases that have already been dealt with. □

6.5.2 Realizing computational degrees

The statement of Lemma 49 falls exactly into the situation described in Lemma 19. For all $n \in \mathbb{N}$, let $\mathcal{B}_n = \mathbb{F}_2$. Then, for all $w \in \mathbb{F}_2^n (= \prod_{i<n} \mathcal{B}_i)$, we have defined S_w, T_w, F_w and Φ_w such that F_w exactly, completely $(S_w, T_w, 0)$-simulates $\bigsqcup_{b\in\mathcal{B}_n} F_{ub}$.

The check of line 3 forces that if $z \in \prod_{i\in\mathbb{N}} \mathcal{B}_i$, then $\tilde{\mathcal{D}}_z^{\infty}(\Phi) \neq \emptyset$ if and only if $\mathcal{H}_{p'}^{\infty}(z)$ is true, or in other words, $z \in \mathfrak{Z}_p$. Indeed, we have that $\mathcal{D}(F_{z_{[\![0,n[\![}}) = \emptyset$ for some n if and only if $\mathcal{H}_{p'}^{\infty}(z)$ is not true, or in other words, if and only if p' halts over z within n steps.

Therefore, Lemma 19 implies that $\Omega_{F_\epsilon} = \tilde{\mathcal{D}}_Z^{\infty}(\Phi) = \bigsqcup_{z\in\mathfrak{Z}_p} \tilde{\mathcal{D}}_z^{\infty}(\Phi)$.

Lemma 51. *For any effectively closed subset* $Z \subset \mathbb{F}_2^{\mathbb{N}}$*, there exists an extremely expansive RPCA* F *and a computable, left-invertible map from* Ω_F *onto the Cartesian product* $Z \times \mathbb{F}_2^{\mathbb{N}}$.

One could even prove that the computable map is two-to-one, and almost one-to-one for any reasonable (topological or measure-theoretical) notion. Also, this SFT can be effectively constructed from Z.

Proof. We construct $SyncComput[14, \nu_{SyncComput}, \mathbf{k}, \mathfrak{S}, \mathfrak{T}, \mathfrak{U}, \pi_{\texttt{MHist}}, p']$ for some sequences that satisfy the inequalities and a program p' that recognizes Z. In addition, assume that $\prod_{i<n} S_i/T_i = 0$.

It follows from Lemma 19 that

$$\Omega_{F_\epsilon} = \bigsqcup_{z\in\mathfrak{Z}_p} \tilde{\mathcal{D}}_z^\infty(\Phi) = \bigsqcup_{\substack{z\in\mathfrak{Z}_p \\ \mathfrak{t}\in\prod_{i\in\mathbb{N}}[\![0,T_i[\![\\ \mathfrak{s}\in\prod_{i\in\mathbb{N}}[\![0,S_i[\![}} \bigcap_{n\in\mathbb{N}} \sigma^{\overline{s_{[\![0,n[\![}}{}^{\mathfrak{S}} F_\epsilon^{\overline{t_{[\![0,n[\![}}{}^{\mathfrak{T}}} \Phi_{z_0}^{-1}\cdots\Phi_{z_{[\![0,n[\![}}^{-1}(\Omega_{F_n}).$$

Consider the map that associates, to each configuration $x \in \Omega_{F_\epsilon}$, the unique triple $(z, \mathfrak{s}, \mathfrak{t})$ such that $x \in \bigcap_{n\in\mathbb{N}} \sigma^{\overline{s_{[\![0,n[\![}}{}^{\mathfrak{S}}} F_0^{\overline{t_{[\![0,n[\![}}{}^{\mathfrak{T}}} \Phi_{z_0}^{-1}\cdots\Phi_{z_{[\![0,n[\![}}^{-1}(\Omega_{F_n})$. This map is computable, since, for all $n \in \mathbb{N}$, S_n is for instance given by $\underline{\pi_{\texttt{Clock}}\Phi_0\Phi_1\cdots\Phi_{n-1}}$ and $z_{[\![0,n[\![}$ is given by $\pi_{\texttt{MHist}}\Phi_0\Phi_1\cdots\Phi_{n-1}$.

Conversely, from the tripe $(z, \mathfrak{s}, \mathfrak{t})$, one can build construct a configuration in Ω_{F_ϵ}, as explained in [10, Proposition 4.2].

The result follows from the obvious computable homeomorphisms between Ω_F and $\mathcal{O}_F$, and between $\prod_{i\in\mathbb{N}} [\![0, S_i[\![\times [\![0, T_i[\![$ and $\mathbb{F}_2^{\mathbb{N}^2}$.

Finally, $\mathcal{N}(F_\epsilon) = \{0\}$, because $\mathcal{O}_{F_\epsilon} = \bigsqcup_{z\in\mathfrak{Z}_p} \mathcal{O}_{F_{\tilde{\mathcal{D}}_z^\infty(\Phi)}}$ and $\mathcal{N}(\mathcal{O}_{F_{\tilde{\mathcal{D}}_z^\infty(\Phi)}}) = \{0\}$, due to the exact complete simulation and $\prod_{i<n} S_i/T_i = 0$. □

The following was proven in [38] for general 2D SFT. Here, we can also restrict the set of non-expansive directions.

Theorem 52. *For any effectively closed subset X, there exists a Medvedev-equivalent extremely expansive 2D SFT whose Turing degrees are the cones above the Turing degrees of X.*

A fortiori, all Medvedev (and Mučnik) degrees contain an extremely expansive SFT.

Proof. It is enough to notice that Ω_F and $\mathcal{O}_F$ are computably homeomorphic. □

The second component in the computable homeomorphism cannot easily be taken out: it is pointed in [21] that all aperiodic subshifts admit a cone of Turing degrees (that is one degree and all degrees above it).

Let us make some final comments: In this chapter, we are inspired and draw mainly on the work of Durand, Romashchenko and Shen [8]. Reading that paper, one has the feeling that the construction of that paper can be done in a reversible way, except for the exchange of information. Working

out the details needed to make that intuition work is (as proven by this chapter) messy and even tedious, sometimes, but we manage to obtain results for which there is no known alternative proof. We also feel that our construction can also shed some light on the construction of Durand, Romashchenko and Shen. More specifically, we always write explicitly the inequalities that need to be satisfied, and for each one we explain at least once why it is needed. Also, we construct "once and for all" the rules and then prove that they have the desired behaviour, instead of using their more informal approach where some rule is created and then it is modified, resulting in a new rule, for which it is taken for granted that the previous argumentation still holds.

Chapter 7

Expansive directions

In Lemma 10, we described a necessary condition for the set of non-expansive directions of an SFT: if X is an SFT, then $\mathcal{N}(X)$ is effectively closed. In this section, we are going to show that this is in fact a characterization of sets of non-expansive directions of SFTs.

Theorem 53. *If $\mathcal{N}_0 \subseteq \mathbb{P}$ is effectively closed, then there exists an SFT $X \subseteq \mathcal{A}^{\mathbb{Z}^2}$ such that $\mathcal{N}(X) = \mathcal{N}_0$.*

This is mentioned as Open Problem 11.1 in Mike Boyle's Open Problems for Symbolic Dynamics [5]. We only answer the first part of that problem, since our constructions do not have any SFT direction. The second part of the problem, concerning 2D SFT with an SFT direction is much more difficult to answer, since it is inextricably related to the expansiveness of RCA.

It is enough to prove this for sets of non-expansive directions that are included in $[-1, 1]$, or even $[0, r]$, for some $0 < r < 1$, because we can cover the set of directions with a finite number of rotations of $[-1, 1]$ (and $[0, r]$). Therefore, even though the fact that we are using PPA might seem problematic (since $\mathcal{N}(F) \subseteq [-1, 1]$ in this case), this is not the case.

The key idea consists in constructing subshifts with a unique direction of non-expansiveness through a nested simulation of RPCA, so that we can use Lemma 23. This idea was introduced in [19], in a non-effective way; we will try to emphasize the obstacle that has to be overcome when trying to "SFTize" this construction.

7.1 Directive encoding

Proposition 23 states that, if we manage to implement a certain kind of nested simulation, then we will obtain a subshift with a unique direction of non-expansiveness, equal to $\overline{\mathfrak{D}}^{\mathfrak{S}/\mathfrak{T}}$, where $\mathfrak{S}, \mathfrak{T} \in \mathbb{N}_1^{\mathbb{N}}$ and $\mathfrak{D} \in \mathbb{Z}^{\mathbb{N}}$. [19,

Lemma 5.6] shows that all directions can be written in this form (when the sequences $\mathfrak{S}, \mathfrak{T}, \mathfrak{D}$ are allowed to be chosen without any constraints). But the sequences of nested simulations that are possible with our SFT construction are more constrained: for example, the sequences $\mathfrak{S}$ and $\mathfrak{T}$ must be polynomially checkable.This immediately imposes some restrictions, since, for example, it implies that S_i cannot grow like an exponential tower of height i. This is not excluded from the construction of Hochman, since he takes S "sufficiently large", in order to make some "error term" sufficiently small. A large part of our construction is to show that we can satisfy these restrictions at the same time, or, in other words, that the error terms can be made sufficiently small even if $\mathfrak{S}$ grows relatively slowly. At the same time, we have to take care of some technical details.

Let us begin the construction by giving some additional necessary definitions:

To any vector $\varepsilon \in \mathbb{R}_+^n$ and any **directive word** $\mathbf{d} := (D_i, W_i)_{0 \leq i < n} \in (\mathbb{N}^2 \setminus \{(0,0)\})^n$, where $n \in \mathbb{N}$, we associate the direction interval $\Theta_\varepsilon(\mathbf{d}) := (\prod_{0 \leq i < n} R_i)[-1,1] + \overline{\mathbf{D}}^{\mathbf{R}} \subset \mathbb{P}$, where $R_i := 1/(D_i + W_i + 1 + \varepsilon_i) \leq 1/2$; recall that $\overline{\mathbf{D}}^{\mathbf{R}} := \sum_{0 \leq i < n} D_i \prod_{0 \leq j \leq i} R_j$. It follows immediately by the definition that $\Theta_\varepsilon(\mathbf{d}) = R_0(\Theta_{\varepsilon_{|[\![1,n[\![}}(\mathbf{d}_{|[\![1,n[\![}) + D_0)$ for any $\mathbf{d} = (D_i, W_i)_{0 \leq i < n}$.

We extend these definitions for infinite sequences in the natural way: To any sequence $\varepsilon \in \mathbb{R}_+^{\mathbb{N}}$ and any **directive sequence** $\mathfrak{d} := (D_n, W_n)_{n \in \mathbb{N}} \in (\mathbb{N}^2 \setminus \{(0,0)\})^{\mathbb{N}}$ we associate the direction $\theta_\varepsilon(\mathfrak{d}) := \overline{\mathfrak{D}}^{\mathfrak{R}} \in \mathbb{R}$, the unique element of $\bigcap_{n \in \mathbb{N}} \Theta_{\varepsilon_{|[\![0,n[\![}}((D_i, W_i)_{0 \leq i < n})$ (uniqueness follows from the fact that $R_i \leq 1/2$, for all $i \in \mathbb{N}$).

If $F_0 \underset{S_0, T_0, D_0 S_0}{\succeq} F_1 \underset{S_1, T_1, D_1 S_1}{\succeq} \dots$ and for all $n \in \mathbb{N}$, $T_n = (D_n + W_n + 1 + \epsilon_n) S_n$, then observe that Proposition 23 can be seen as saying that $\mathcal{N}(F_0) = \{\theta_\varepsilon(\mathfrak{D}, \mathfrak{W})\}$. This is point of contact between this section and the rest of the thesis.

[19, Lemma 5.6] can now be reformalized as the following.

Lemma 54. *For all $x \in [0,1]$, there exist a sequence $\varepsilon \in \mathbb{R}_+^{\mathbb{N}}$ and a directive sequence $\mathfrak{d}$ such that $\theta_\varepsilon(\mathfrak{d}) = x$.*

We refine this statement in two ways: first, we will show that the sequence ε can be fixed and second, we will restrict the alphabet of acceptable directive sequences to $\mathcal{B} := \{(0,1), (1,1), (1,0)\}$. By doing that, we will "lose" a small part on the right endpoint of the interval $[0,1]$.

Lemma 55. *Let $\varepsilon \in [0, \sqrt{2} - 1]^{\mathbb{N}}$ be a sequence. Then,*

$$\theta_\varepsilon(\mathcal{B}^{\mathbb{N}}) = [0, \overline{111\dots}^{(1/(2+\varepsilon_i)_i)}].$$

Though the interval $[0,1[$ cannot be covered fully with a fixed, non-trivial sequence, the convergence of the sequence $(\varepsilon_n)_{n \in \mathbb{N}}$ to 0 can be sped up suitably, in order to realise any number arbitrarily close to 1.

Proof. Let us prove by induction over $n \in \mathbb{N}$ that for any such sequence ε, we have that

$$\theta_{\varepsilon_{|[\![0,n[\![}}(\mathcal{B}^n) = [-\prod_{i<n} \frac{1}{2+\varepsilon_i}, \overline{1\dots12}^{(1/(2+\varepsilon_i))_{i<n}}] \supset [0, \frac{1}{\sqrt{2}}].$$

The base of the induction follows from $\epsilon_0 \leq \sqrt{2}-1$. Let us assume that the inductive hypothesis is true for some $n \in \mathbb{N}$, and prove it for $n+1$. Note that $\overline{1\dots12}^{(1/(2+\varepsilon_i))_{i<n+1}} = \frac{1}{2+\varepsilon_0}\left(1+\overline{1\dots12}^{(1/(2+\varepsilon_i))_{1\leq i<n+1}}\right)$ By the induction hypothesis (which we can apply to the truncated sequence $(\epsilon_n)_{n\geq 1}$, since it satisfies the assumption, too) and $\varepsilon_0 \leq \sqrt{2}-1$, we have $\overline{1\dots12}^{(1/(2+\varepsilon_i))_{i<n+1}} \geq \frac{1}{1+\sqrt{2}}\left(1+\frac{1}{\sqrt{2}}\right) = \frac{1}{\sqrt{2}}$.

The set $\theta_{\varepsilon_{|[\![0,n+1[\![}}(\mathcal{B}^{n+1})$ can be decomposed, in terms of the first directive letter, into a union of three intervals

$$\frac{1}{2+\varepsilon_0}\theta_{\varepsilon_{|[\![1,n]\!]}}(\mathcal{B}^n) \cup \frac{1}{3+\varepsilon_0}(\theta_{\varepsilon_{|[\![1,n]\!]}}(\mathcal{B}^n)+1) \cup \frac{1}{2+\varepsilon_0}(\theta_{\varepsilon_{|[\![1,n]\!]}}(\mathcal{B}^n)+1)\ .$$

Following the induction hypothesis, the first interval is equal to

$$\frac{1}{2+\varepsilon_0}[-\prod_{1\leq i\leq n}\frac{1}{2+\varepsilon_i}, \overline{1\dots12}^{(1/(2+\varepsilon_i))_{1\leq i\leq n}}] =$$
$$= [-\prod_{0\leq i\leq n}\frac{1}{2+\varepsilon_i}, \frac{1}{2+\varepsilon_0}\overline{1\dots12}^{(1/(2+\varepsilon_i))_{1\leq i\leq n}}].$$

The second interval is equal to

$$\frac{1}{3+\varepsilon_0}(1+[-\prod_{1\leq i\leq n}\frac{1}{2+\varepsilon_i}, \overline{1\dots12}^{(1/(2+\varepsilon_i))_{1\leq i\leq n}}]) =$$
$$= [\frac{1}{3+\varepsilon_0}(1-\prod_{1\leq i\leq n}\frac{1}{2+\varepsilon_i}), \frac{1}{3+\varepsilon_0}\overline{1\dots12}^{(1/(2+\varepsilon_i))_{1\leq i\leq n}}].$$

The third interval is equal to

$$\frac{1}{2+\varepsilon_0}(1+[-\prod_{1\leq i\leq n}\frac{1}{2+\varepsilon_i}, \overline{1\dots12}^{(1/(2+\varepsilon_i))_{1\leq i\leq n}}]) =$$
$$= [\frac{1}{2+\varepsilon_0} - \prod_{0\leq i\leq n}\frac{1}{2+\varepsilon_n}, \overline{1\dots12}^{(1/(2+\varepsilon_i))_{0\leq i\leq n}}].$$

It is clear that the smallest point of these three intervals is $-\prod_{0\leq i\leq n}\frac{1}{2+\varepsilon_i}$ (the last two intervals are in $\mathbb{R}_+$), and the largest is $\overline{1\dots12}^{(1/(2+\varepsilon_i))_{0\leq i\leq n}}$

(the first interval is obtained through a translation by -1 of the third one, or through a homothecy by $\frac{2+\varepsilon_0}{3+\varepsilon_0}$ of the second one).

We proved earlier that $\overline{1\dots12}^{(1/(2+\varepsilon_i))_{i<n+1}} \geq \frac{1}{1+\sqrt{2}}\left(1+\frac{1}{\sqrt{2}}\right) = \frac{1}{\sqrt{2}}$. It follows from this that the upper bound $\frac{1}{2+\varepsilon_0}\overline{1\dots12}^{(1/(2+\varepsilon_i))_{1\leq i\leq n}}$ of the first interval is larger than $\frac{1}{(2+\varepsilon_0)\sqrt{2}}$, while the smaller bound of the second interval is lower than $\frac{1}{3+\varepsilon_0}$, which is less than $\frac{1}{(2+\varepsilon_0)\sqrt{2}}$, as can be easily verified.In other words, there is no hole between these two intervals.

Using the same arguments, one can easily see that the upper bound of the second interval is $\frac{1}{3+\varepsilon_0}(1+\overline{1\dots12}^{(1/(2+\varepsilon_i))_{1\leq i\leq n+1}})$, which is larger than $\frac{1+1/\sqrt{2}}{3+\varepsilon_0}$ while the lower bound of the third interval is smaller than $\frac{1}{2+\varepsilon_0}$, which is smaller than $\frac{1+1/\sqrt{2}}{3+\varepsilon_0}$, since $\sqrt{2}-1+\varepsilon_0\frac{1}{\sqrt{2}} > 0$. There is no hole here either, and globally, we get the full interval $\left[-\prod_{0\leq i\leq n}\frac{1}{2+\varepsilon_i}, \overline{1\dots12}^{(1/(2+\varepsilon_i))_{0\leq i\leq n}}\right]$.

The statement is easily derived from the fact that $\prod_{0\leq i<n}\frac{1}{2+\varepsilon_i} \to 0$ and $\overline{1\dots12}^{(1/(2+\varepsilon_i)_{i<n})} \to \overline{1\dots}^{(1/(2+\varepsilon_i)_{i\in\mathbb{N}})}$. □

7.2 Computing directions

Lemma 10 stated that the set of non-expansive directions of an SFT is *effectively* closed, which means that there exists a program which takes as (infinite) input the description of a direction in $\mathbb{P}$ and halts (after having read finitely many bits of the input) if and only if the direction is expansive. In Subsection 2.2.2, it was suggested that the good way to represent directions in order to compute with them was by the two coordinates of some intersection with the unit circle. Each slope then has two (opposite) valid representations. When restricting to closed subsets of $\mathbb{R} \subseteq \mathbb{P}$ (*i.e.*, when we are not talking about the horizontal direction), the notion of effectively closed set of direction is the same with the above representation as with the usual definition of $\mathbb{R}$. This is due to the facts that the functions sin and cos and their inverses are computable and that the function $x \to 1/x$ is uniformly continuous away from 0.

The following remark states that directive sequences give another, equivalent representation for directions.

Remark 56. *Let $\varepsilon \in \mathbb{R}^{\mathbb{N}}$ be computable. Then, θ_ε is a computable function.*

The computation is actually uniform in ε, in the sense that it could be considered as part of the input.

Proof. This follows from the fact that the diameter of $\Theta_{\varepsilon_{|[\![0,n[\![}}(\mathbf{d})$ is at most 2^{-n}, since $R_i \leq 1/2$, for all $i \in \mathbb{N}$ and directive sequence $\mathbf{d}$.

□

Remark 56 implies that effectively closed sets of slopes can be equivalently described by an effectively closed set of directive sequences. This is the computational description of directive sequences that we are going to use in the next chapter.

7.3 Realization of sets of non-expansive directions

Let $\mathcal{C}_{Reali} = \mathcal{C}_{Simulate} \sqcup [\mathtt{MHist}, \mathtt{MShift}, \mathtt{MShift}_{+1}, \mathtt{Prog}, \mathtt{RProg}]$.

The following permutation will also be parametrized by an effectively closed set $\mathcal{N}_0 \subseteq [0, 1/2]$, which is represented by the program p' of the TM that recognizes $\mathcal{N}_0$ as a set of directive sequences. $\mathcal{N}_0$ is the set of non-expansive directions that we are trying to realize.

We identify the set $\mathcal{B} := \{(0,1),(1,1),(1,0)\}$ with $\mathbb{F}_3$ (through any bijection). If $a \in \mathcal{B}$, then D_a, W_a will denote the projection of a onto the first and second coordinate, respectively.

In the following algorithm, p' is the program of a TM that recognizes some set of non-expansive directions.

$Reali[M, \nu, p', \mathbf{k}, \mathfrak{S}, \mathfrak{U}]$

1: $Check[\pi_{\mathtt{MShift}} = \pi_{\mathtt{MShift}_{+1}}]$
2: **if** $\underline{\pi_{\mathtt{Clock}}} = 0$ **then**
3: $\quad Check[\mathcal{H}_{p'}^{|\mathtt{MHist}|}(\mathtt{MHist})]$
4: $\quad Check[\mathcal{S}^{\epsilon}_{\mathtt{Head}_{-1},\mathtt{Head}_{+1},\mathtt{Tape}_{-1},\mathtt{Tape}_{+1},\mathtt{NTape}}]$
5: $\quad CkAlph[M, \underline{\pi_{\mathtt{Addr}}}, \pi_{\mathtt{Tape}}, \mathbf{k}_{|\mathtt{MHist}|+1}]$
6: $\quad HCheck[M, \underline{\pi_{\mathtt{Addr}}}, \pi_{\mathtt{Tape}}, \mathbf{k}_{|\mathtt{MHist}|+1}, \mathtt{Prog}, \pi_{\mathtt{Prog}}]$
7: $\quad HCheck[M, \underline{\pi_{\mathtt{Addr}}}, \pi_{\mathtt{Tape}}, \mathbf{k}_{|\mathtt{MHist}|+1}, \mathtt{RProg}, \pi_{\mathtt{RProg}}]$
8: $\quad HCheck[M, \underline{\pi_{\mathtt{Addr}}}, \pi_{\mathtt{Tape}}, \mathbf{k}_{|\mathtt{MHist}|+1}, \mathtt{MHist}, \pi_{\mathtt{MHist}}\pi_{\mathtt{MShift}}]$
9: **end if**
10: **if** $\underline{\pi_{\mathtt{Clock}}} = 0$ **then**
11: $\quad Swap[\mathtt{Tape}, \mathtt{Tape}_{+1}]$
12: **end if**
13: **if** $\underline{\pi_{\mathtt{Clock}}} = D_{\mathtt{MShift}} S_{|\mathtt{MHist}|}$ **then**
14: $\quad Swap[\mathtt{Tape}, \mathtt{Tape}_{+1}]$
15: **end if**
16: $Simulate[M,\ \nu,\ \mathbf{k}_{|\mathtt{MHist}|},\ \underline{\pi_{\mathtt{Addr}}},\ \underline{\pi_{\mathtt{Clock}}} - D_{\mathtt{MShift}} S_{|\mathtt{MHist}|},\ S_{|\mathtt{MHist}|},\ S_{|\mathtt{MHist}|}(D_{\mathtt{MShift}} + W_{\mathtt{MShift}} + 1) + 4U_{|\mathtt{MHist}|}\ , U_{|\mathtt{MHist}|},\ \pi_{\mathtt{Prog}},\ \pi_{\mathtt{RProg}}]$
17: $Grid[\mathfrak{S}_{|\mathtt{MHist}|}, \mathfrak{S}_{|\mathtt{MHist}|}(D_{\mathtt{MShift}} + W_{\mathtt{MShift}} + 1) + 4\mathfrak{U}_{|\mathtt{MHist}|}]$

This is like the simulation of the computation degrees, only that we keep a more complicated register in the `MHist` field and we use the values of

`MShift` to perform a "macro-shift" before the simulation. We also note that $\mathfrak{T}$ is not given as a parameter of the construction. Instead, it is determined by the sequences $\mathfrak{S}, \mathfrak{U}$ and the value of field `MShift`.

Lemma 57. *Let $\mathfrak{S}, \mathfrak{U}$ be polynomially checkable sequences of integers and p' be the program of a TM. Let us fix the field list $\mathcal{C}_{Reali} := [0, \dots, 14]$, the corresponding direction vector ν_{Reali} and a polynomially checkable sequence of 15-uples $\mathbf{k} \in (\mathbb{N}^{14})^{\mathbb{N}}$. Let F be the IPPA with directions ν_{Reali} and permutation*

$$Reali[14, \nu_{Reali}, p', \mathbf{k}, \mathfrak{S}, \mathfrak{U}]$$

and p, p^{-1} be the programs for this permutation and its inverse, respectively.

For all $w \in \mathcal{B}^*$ and $a \in \mathcal{B}$, let $\mathbf{k}_{w,a} := \mathbf{k}_{|w|}$, $S_{w,a} := S_{|w|}$, $U_{w,a} := U_{|w|}$, $T_{w,a} := S_{w,a}(D_a + W_a + 1) + 4U_{w,a}$ and $F_{w,a}$ be the restriction of F to the subalphabet

$$\mathcal{A}_{w,a} := \mathbb{F}_5^{\mathbf{k}_{|w|}} \cap \mathcal{S}^{w}_{\texttt{MHist}} \cap \mathcal{S}^{a}_{\texttt{MShift},\texttt{MShift}_{+1}} \cap \mathcal{S}^{p}_{\texttt{Prog}} \cap \mathcal{S}^{p^{-1}}_{\texttt{RProg}}.$$

Assume that the following inequalities hold, for all $w \in \mathcal{B}^*$ and $a, a' \in \mathcal{B}$:

$$\begin{cases} U_{w,a} \geq \max\{t_p(\mathbb{F}_5^{\mathbf{k}_{wa,a'}}), t_{p^{-1}}(\mathbb{F}_5^{\mathbf{k}_{wa,a'}})\} \\ S_{w,a} \geq \max\{2U_{w,a}, \left|\chi\left(\mathbb{F}_5^{\mathbf{k}_{wa,a'}}\right)\right|\} \\ k_{w,a,\texttt{Addr}}, k_{w,a,\texttt{Addr}_{+1}} \geq \|S_{w,a}\| \\ k_{w,a,\texttt{Clock}}, k_{w,a,\texttt{Clock}_{+1}} \geq \|T_{w,a}\| \\ k_{w,a,\texttt{Head}_{-1}}, k_{w,a,\texttt{Head}_{+1}} \geq \max\left|\chi\left(\mathbb{F}_4 \times (Q_p \cup Q_{p^{-1}}) \times \{-1, +1\}\right)\right| \\ k_{w,a,\texttt{Tape}}, k_{w,a,\texttt{NTape}}, k_{w,a,\texttt{Tape}_{-1}}, k_{w,a,\texttt{Tape}_{+1}} \geq 1 \\ k_{w,a,\texttt{Prog}} \geq |p| \\ k_{w,a,\texttt{RProg}} \geq \left|p^{-1}\right| \\ k_{w,a,\texttt{MHist}} \geq |w| \\ k_{w,a,\texttt{MShift}} = k_{w,a,\texttt{MShift}_{+1}} \geq 1\ . \end{cases}$$

Then, $F_{w,a}$ completely exactly simulates $\bigsqcup_{a' \in \mathcal{B}} F_{wa,a'}$ with parameters $(S_{w,a}, T_{w,a}, D_a S_{w,a}, \Phi_{w,a})$, where $\Phi_{w,a} = \tilde{\Phi}_{\texttt{Tape}|\Sigma_{w,a}}$ and

$$\Sigma_{w,a} := \mathcal{A}^{\mathbb{Z}}_{w,a} \cap \Sigma^{0,0,S_{w,a},T_{w,a}} \cap \mathcal{S}^{\epsilon}_{\texttt{Head}_{-1},\texttt{Head}_{+1},\texttt{Tape}_{-1},\texttt{Tape}_{+1},\texttt{NTape}} \cap \\ \cap\, \Phi^{-1}\Big(\bigsqcup_{a' \in \mathcal{B}} \mathcal{A}^{\mathbb{Z}}_{wa,a'}\Big).$$

The only difference between this proof and the proof of Lemma 49 is that we shift all the encodings $D_a S_{w,a}$ cells (equivalently, D_a macro-cells) to the right before starting the simulation.

Also, it is not difficult to see that the usual inequalities hold: $t_p(\mathcal{A}_n) = t_{p^{-1}}(\mathcal{A}_n) = P(|\mathcal{A}_n| + t_{\mathbf{k}}(n) + t_{\mathfrak{S}}(n) + t_{\mathfrak{T}}(n) + t_{\mathfrak{U}}(n))$ and $|p|, \left|p^{-1}\right| = O(|p_{\mathbf{k}}| + |p_{\mathfrak{S}}| + |p_{\mathfrak{T}}| + |p_{\mathfrak{U}}|)$. Recall that p' is fixed in advance so it is a constant in what matters complexity.

Proof. If p' halts on input w within $|w|$ steps, then the check of line 3 will reject every configuration, which means that $F_{w,a} = \emptyset$. But, in this case, wa will also be rejected by p' within $|wa|$ steps, for all $a \in \mathcal{B}$, so that $\bigsqcup_{a' \in \mathcal{B}} F_{wa,a'} = \emptyset$, too. By definition, the empty PCA strongly, completely simulates itself for all possible choices of the simulating parameters, so that the claim is true in this case.

Suppose, then, that p' does not halt on input w within $|w|$ steps. Then, the check of line 3 is always true, so that we can ignore it in the rest of the proof. As in the previous proofs, we have to show three things: that $F_{w,a}$ $(S_{w,a}, T_{w,a})$ simulates $\bigsqcup_{a \in \mathbb{F}_2} F_{wa,a'}$ with decoding function $\Phi_{w,a}$ (simulation), that $\Phi_{w,a}$ is injective (exactness) and that $\Omega_{F_{w,a}} \subseteq \mathcal{D}\Phi_{w,a}$ (completeness).

For the simulation, it is easy to see that if $b \in \mathcal{A}_{wa,a'}^{\mathbb{Z}}$, where $a' \in \mathbb{F}_2$ and $c \in \Phi_{w,a}^{-1}(b)$, then c is not rejected by the checks of lines 1,4,5,6, 7 and 8.

Then, line 11 copies *all* the info bits onto $\texttt{Tape}_{+1}$. During the next $S_{w,a}D_a$ steps, no permutation is applied. The only thing happening to the configuration is that the encodings that are in $\texttt{Tape}_{+1}$ travel to the right at the speed of one cell per time step. After $S_{w,a}D_a$ steps, they are copied back to the `Tape` tape by line 14. Every letter has travelled exactly $S_{w,a}D_a$ cells to the right, which corresponds to D_a macro-cells. Formally, $\Phi(F^{D_a S_{w,a}}(c)) = \sigma^{-D_a}(b)$.

Then, from Fact 34 and since the only rule applied from $\texttt{Clock} = D_a S_{w,a}$ is $Grid \circ Simulate$, we obtain that $\Phi(F^{D_a S_{w,a} + S_{w,a} + 4U_{w,a}}(c)) = F_{wa,a'}(\sigma^{-D_a}(b))$. After $\texttt{Clock} = D_a S_{w,a} + S_{w,a} + 4U_{w,a}$, nothing else changes in the configuration until `Clock` becomes 0 again. Line 17 ensures that `Clock` goes from 0 to $(D_a + W_a + 1)\mathfrak{S}_{w,a} + 4\mathfrak{U}_{w,a}$. This concludes the proof of the simulation part.

Exactness of the simulation is easy to see. The values of all the fields of $c \in \Phi_{w,a}^{-1}(b)$ are uniquely determined by b and $\Sigma_{w,a}$.

For the completeness, we show that if $c \in \Sigma^{0,0,S_{w,a},T_{w,a}} \cap F_{w,a}^{-2T_{w,a}}(\mathcal{A}_{w,a}^{\mathbb{Z}})$, then $c \in \Phi_{w,a}^{-1}(\bigsqcup_{a' \in \mathcal{B}} \mathcal{A}_{wa,a'}^{\mathbb{Z}})$.

Indeed, if $c \in \Sigma^{0,0,S_{w,a},T_{w,a}} \cap F_{w,a}^{-T_{w,a}}(\mathcal{A}_{w,a}^{\mathbb{Z}})$, then lines 4,6,7 and 8 ensure that

$$c \in \mathcal{A}_{w,a}^{\mathbb{Z}} \cap \Sigma^{0,0,S_{w,a},T_{w,a}} \cap \mathcal{S}^{\epsilon}_{\texttt{Head}_{-1},\texttt{Head}_{+1},\texttt{Tape}_{-1},\texttt{Tape}_{+1},\texttt{NTape}} \cap \cap \Phi^{-1}((\bigsqcup_{a' \in \mathcal{B}} \mathcal{A}_{wa,a'})^{\mathbb{Z}}).$$

Let $b \in (\bigsqcup_{a' \in \mathcal{B}} \mathcal{A}_{wa,a'})^{\mathbb{Z}}$ be such that $c \in \Phi^{-1}(b)$. We still cannot know that $\pi_{\texttt{MShift}}(b_i)$ is the same for all $i \in \mathbb{Z}$.

We deal with this problem in a similar way as in Section 6.5, since $F_{w,a}^{2T}(c)$ exists, this means that $F^2(b)$ exists, and line1 ensures that $\pi_{\texttt{MShift}}(b_i) =$

$\pi_{\mathtt{MShift}_{+1}}(b_i) = \pi_{\mathtt{MShift}}(b_j)$, for all $i, j \in \mathbb{Z}$. Therefore, $b \in \bigsqcup_{a' \in \mathbb{F}_2} \mathcal{A}^{\mathbb{Z}}_{wa,a'}$ and this concludes the proof. □

7.3.1 Satisfying the inequalities

Unlike the previous cases, the set of inequalities that we want to satisfy does not depend on n, but instead on a word $w \in \mathcal{B}^*$ and $a, a' \in \mathcal{B}$. However, we will now see that the inequalities can be translated to some inequalities about the polynomially computable sequences $\mathfrak{S}, \mathfrak{U}$.

Remark 58. *We can find* $\mathbf{k} \in (\mathbb{N}^{15})^{\mathbb{N}}$ *and* $\mathfrak{S}, \mathfrak{U} \in \mathbb{N}^{\mathbb{N}}$ *such that the inequalities of Lemma 57 are satisfied.*

In addition, we can have $\varepsilon_n := 4U_n/S_n < \sqrt{2} - 1$*, for all* $n \in \mathbb{N}$ *and* $\theta_\varepsilon(\mathcal{B}^{\mathbb{N}}) \supseteq [0, 1/2]$.

Proof. Let $w \in \mathcal{B}^*$ and $a, a' \in \mathcal{B}$. Let $n := |w|$ be the length of w. Let us write again the inequalities:

$$\begin{cases} U_{w,a} \geq \max\{t_p(\mathbb{F}_5^{\mathbf{k}_{wa,a'}}), t_{p^{-1}}(\mathbb{F}_5^{\mathbf{k}_{wa,a'}})\} \\ S_{w,a} \geq \max\{2U_{w,a}, \left|\chi\left(\mathbb{F}_5^{\mathbf{k}_{wa,a'}}\right)\right|\} \\ k_{w,a,\mathtt{Prog}} \geq |p| \\ k_{w,a,\mathtt{RProg}} \geq \left|p^{-1}\right| \\ k_{w,a,\mathtt{Head}_{-1}}, k_{w,a,\mathtt{Head}_{+1}} \geq \left|\chi\left(\mathbb{F}_4 \times (Q_p \cup Q_{p^{-1}}) \times \{-1, +1\}\right)\right| \\ k_{w,a,\mathtt{Addr}}, k_{w,a,\mathtt{Addr}_{+1}} \geq \|S_{w,a}\| \\ k_{w,a,\mathtt{Clock}}, k_{w,a,\mathtt{Clock}_{+1}} \geq \|T_{w,a}\| \\ k_{w,a,\mathtt{Tape}}, k_{w,a,\mathtt{NTape}}, k_{w,a,\mathtt{Tape}_{-1}}, k_{w,a,\mathtt{Tape}_{+1}} \geq 1 \\ k_{w,a,\mathtt{MHist}} \geq |w| \\ k_{w,a,\mathtt{MShift}} = k_{w,a,\mathtt{MShift}_{+1}} \geq 1\ . \end{cases}$$

According to the definition, $\mathbf{k}_{w,a} = \mathbf{k}_n$, $S_{w,a} = S_n$, $U_{w,a} = U_n$ and $T_{w,a} = S_n(D_a + W_a + 1) + 4U_n$: Therefore, we can write the above inequalities as follows:

$$\begin{cases} U_n \geq \max\{t_p(\mathbb{F}_5^{\mathbf{k}_{n+1}}), t_{p^{-1}}(\mathbb{F}_5^{\mathbf{k}_{n+1}})\} \\ S_n \geq \max\{2U_n, \left|\chi\left(\mathbb{F}_5^{\mathbf{k}_{n+1}}\right)\right|\} \\ k_{n,\mathtt{Prog}} \geq |p| \\ k_{n,\mathtt{RProg}} \geq \left|p^{-1}\right| \\ k_{n,\mathtt{Head}_{-1}}, k_{n,\mathtt{Head}_{+1}} \geq \left|\chi\left(\mathbb{F}_4 \times (Q_p \cup Q_{p^{-1}}) \times \{-1, +1\}\right)\right| \\ k_{n,\mathtt{Addr}}, k_{n,\mathtt{Addr}_{+1}} \geq \|S_n\| \\ k_{n,\mathtt{Clock}}, k_{n,\mathtt{Clock}_{+1}} \geq \|S_n(D_a + W_a + 1) + 4U_n\| \\ k_{n,\mathtt{Tape}}, k_{n,\mathtt{NTape}}, k_{n,\mathtt{Tape}_{-1}}, k_{n,\mathtt{Tape}_{+1}} \geq 1 \\ k_{n,\mathtt{MHist}} \geq n \\ k_{n,\mathtt{MShift}} = k_{n,\mathtt{MShift}_{+1}} \geq 1\ . \end{cases}$$

Unlike the previous proofs, we cannot choose $\mathbf{k}_{\mathfrak{S},\mathfrak{T}} \in \mathbb{N}^{\mathbb{N}}$ such that all of the inequalities except the first two are satisfied as equalities. This is because the inequalities about $\texttt{Clock}$ and $\texttt{Clock}_{+1}$ depend on $a \in \mathcal{B}$ and not only on n. However, $D_a + W_a \leq 2$, for all $a \in \mathcal{B}$, so that we can replace these inequalities with $k_{n,\texttt{Clock}}, k_{n,\texttt{Clock}_{+1}} \geq \|3S_n + 4U_n\|$ and show that this new set of inequalities can be satisfied. (Here, it is essential that $\mathcal{B}$ is a finite set. Bounding $D_a + W_a$ from above is one of the reasons that we had to do a little more work with the directive sequences.)

The rest of the proof follows the usual pattern. We choose $S_n = Q^{n+n_0}$ and $U_n = (n + n_0)^r$ for some suitable values of n_0, r and Q.

For the second claim, let us recall more specifically in which order n_0, r and Q are chosen. In the proof of Remark 42, we showed that there exists n_0 and r that work for *every* Q. Therefore, by choosing $S_n = Q^{n+n_0}$ and $U_n = (n + n_0)^r$, for some sufficiently large Q, we can make $\varepsilon := 4U_n/S_n$ smaller than $\sqrt{2} - 1$ and $\overline{111\ldots}^{(1/(2+\varepsilon_i)_i)}$ larger than $1/2$. □

For all w, a, we have $T_{w,a} = (D_a + W_a + 1)S_{w,a} + 4U_{w,a} = (D_a + W_a + 1 + 4U_{w,a}/S_{w,a})S_{w,a} = (D_a + W_a + 1 + \epsilon_{w,a})S_{w,a}$, where $\epsilon_{w,a} < \sqrt{2} - 1$, so that we are in the situation described in Lemma 55 and $\theta_\varepsilon(\mathcal{B}^{\mathbb{N}}) \supseteq [0, 1/2]$. Since $\mathcal{N}_0 \subseteq [0, 1/2]$ by assumption, this means that every direction in $\mathcal{N}_0$ is representable as a sequence in $\theta_\varepsilon(\mathcal{B}^{\mathbb{N}})$.

7.3.2 Realization

For all $z \in \mathfrak{Z}_{p'}$, we have a sequence of complete, exact simulations given by

$$F_{z_{[\![0,n[\![},z_n} \underset{S_{z_{[\![0,n[\![},z_n},T_{z_{[\![0,n[\![},z_n},D_{z_n}S_{z_{[\![0,n[\![},z_n}}}{\succeq} F_{z_{[\![0,n+1[\![},z_{n+1}},$$

for all $n \in \mathbb{N}$.

Therefore, according to a Lemma 19, we have that $\Omega_F = \bigsqcup_{z\in\mathfrak{Z}_{p'}} \tilde{\mathcal{D}}^\infty_z(\Phi)$, and $\mathcal{O}_{F_\epsilon} = \bigsqcup_{z\in\mathfrak{Z}_{p'}} \mathcal{O}_{F_{\tilde{\mathcal{D}}^\infty_z(\Phi)}}$. In addition, we know that $\mathcal{N}(\mathcal{O}_{F_{\tilde{\mathcal{D}}^\infty_z(\Phi)}}) = \theta(z)$, by Lemma 22 and that every direction in $[\![0, 1/2]\!]$ can be represented in such a way, by Lemma 55. Finally, we know by Lemma 11 that $\mathcal{N}(\mathcal{O}_{F_\epsilon}) = \bigsqcup_{z\in\mathfrak{Z}_{p'}} \mathcal{N}(\mathcal{O}_{F_{\tilde{\mathcal{D}}^\infty_z(\Phi)}}) = \bigsqcup_{z\in\mathfrak{Z}_{p'}} \theta(z) = \mathcal{N}_{p'} = \mathcal{N}_0$.

Therefore, for every effectively closed set of directions which is included in $[0, 1/2]$, recognized by a TM with program p', we have constructed a 2D SFT with exactly this set as set of non-expansive directions. According to our previous discussions, this is enough to realize arbitrary effectively closed sets of directions as the set of non-expansive directions of 2D SFT. This concludes the proof of Theorem 53.

Conclusion and Open Questions

We have provided a general method for constructing extremely-expansive 2D SFT's of finite type and we have shown that this class of 2D SFT's has very rich computational, dynamical and geometrical properties. At the same time, our method throws some light on the essence of self-similar and hierarchical constructions and we hope that it might help to better understand previous works with hierarchical constructions, especially [8]. (On the other hand, the difficulty of [10] only partly comes from the hierarchical simulation, so our work is certainly not sufficient to explain this construction better.)

Regarding future work, we believe that the following questions about extremely-expansive 2D SFTs are very natural: First of all, can (a variant of) our method produce a *minimal* extremely-expansive SFT? Is the emptiness problem undecidable for *minimal* extremely-expansive SFT's? Recently, Durand and Romashchenko [7] described a method for constructing minimal (but not extremely-expansive) SFT's and answer the second question positively. It seems that their technique can be readily generalized to our framework. Second, is it possible to realize all effective subshifts, in the sense of [18, 8, 1] with 2D extremely-expansive SFT's? This would be an improvement with of the result of [8, 1] since it would (in some sense) further lower the dimension of the realizing subshift by one. Third, is it possible to construct extremely expansive SFT covers for square substitutions? This question goes back to the construction of Mozes [33], which constructs SFT covers for square substitutions without any directions of expansivess. Recently, Ollinger and Legloannec [30] constructed 4-way deterministic covers. We believe that the answer to this question is also positive. Finally, and this is certainly the most interesting, but also difficult question, is it possible to use our method in order to construct reversible, self-simulating CA, *i.e.*, is it possible to turn the partial rules, with which we have been working in this thesis, to complete rules, while at the same time keeping the good properties of self-simulation? This could find an application to the problem of the undecidability of expansiveness for reversible CA.

Bibliography

[1] Nathalie Aubrun and Mathieu Sablik. Simulation of effective subshifts by two-dimensional subshifts of finite type. *Acta Applicandae Mathematicae*, 126(1):35–63, 2013.

[2] Alexis Ballier. Universality in symbolic dynamics constrained by medvedev degrees. *CoRR*, abs/1304.5418, 2013.

[3] C. H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17(6):525–532, November 1973.

[4] R. Berger. *The Undecidability of the Domino Problem*. American Mathematical Society memoirs. American Mathematical Society, 1966.

[5] Mike Boyle. Open problems in symbolic dynamics. *Contemporary mathematics*, 469:69–118, 2008.

[6] Mike Boyle and Douglas Lind. Expansive subdynamics. *Transactions of the American Mathematical Society*, 349(1):55–102, 1997.

[7] Bruno Durand and Andrei Romashchenko. Quasiperiodicity and non-computability in tilings. *CoRR*, abs/1504.06130, 2015.

[8] Bruno Durand, Andrei Romashchenko, and Alexander Shen. Fixed-point tile sets and their applications. *Journal of Computer and System Sciences*, 78(3):731 – 764, 2012.

[9] Peter Gács. Reliable computation with cellular automata. *Journal of Computer and System Sciences*, 32(1):15–78, 1986.

[10] Peter Gács. Reliable cellular automata with self-organization. *Journal of Statistical Physics*, 102(1–2):45–267, 2001.

[11] Peter Gács, Mathieu Hoyrup, and Cristóbal Rojas. Randomness on computable probability spaces—a dynamical point of view. *Theory of Computing Systems*, 48(3):465–485, 2011.

[12] Lawrence Gray. A reader's guide to Gács's "Positive Rates" paper. *Journal of Statistical Physics*, 103(1–2):1–44, 2001.

[13] Pierre Guillon, Jarkko Kari, and Charalampos Zinoviadis. On determinism in subshifts. Unpublished manuscript, 2015.

[14] Pierre Guillon and Charalampos Zinoviadis. Densities and entropies in cellular automata. In *How the World Computes - Turing Centenary Conference and 8th Conference on Computability in Europe, CiE 2012, Cambridge, UK, June 18-23, 2012. Proceedings*, pages 253–263, 2012.

[15] G.A. Hedlund. Endomorphisms and automorphisms of the shift dynamical system. *Mathematical systems theory*, 3(4):320–375, 1969.

[16] Peter G. Hinman. A survey of mučnik and medvedev degrees. *Bull. Symbolic Logic*, 18(2):161–229, 06 2012.

[17] Michael Hochman. A note on universality in multidimensional symbolic dynamics. *Discrete and Continuous Dynamical Systems - Series S*, 2(2):301–314, 2009.

[18] Michael Hochman. On the dynamics and recursive properties of multidimensional symbolic systems. *Inventiones Mathematicæ*, 176(1):131–167, April 2009.

[19] Michael Hochman. Expansive directions for $\mathbb{Z}^2$ actions. *Ergodic Theory & Dynamical Systems*, 31(1):91–112, 2011.

[20] Michael Hochman and Tom Meyerovitch. A characterization of the entropies of multidimensional shifts of finite type. *Annals of Mathematics*, 171(3):2011–2038, 2010.

[21] Emmanuel Jeandel and Pascal Vanier. Turing degrees of multidimensional SFTs. *Theor. Comput. Sci.*, 505:81–92, 2013.

[22] J. Kari and P. Papasoglu. Deterministic aperiodic tile sets. *Geometric and Functional Analysis GAFA*, 9(2):353–369, 1999.

[23] Jarkko Kari. The nilpotency problem of one-dimensional cellular automata. *SIAM Journal on Computing*, 21(3):571–586, 1992.

[24] Jarkko Kari. Representation of reversible cellular automata with block permutations. *Mathematical Systems Theory*, 29(1):47–61, 1996.

[25] Jarkko Kari. A small aperiodic set of wang tiles. *Discrete Mathematics*, 160(1-3):259–264, 1996.

[26] Jarkko Kari. On the undecidability of the tiling problem. In *SOFSEM 2008: Theory and Practice of Computer Science, 34th Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, January 19-25, 2008, Proceedings*, pages 74–82, 2008.

[27] G. L. Kurdyumov. An example of a non-ergodic one-dimensional homogeneous random medium with positive transition probabilities. *Soviet Mathematics Doklady*, 19(1), 1978.

[28] Grégory Lafitte and Michael Weiss. Computability of tilings. In *IFIP*, volume 273 of *International Federation for Information Processing*, pages 187–201, 2008.

[29] Grégory Lafitte and Michael Weiss. Tilings: simulation and universality. *Mathematical Structures in Computer Science*, 20(5):813–850, 2010.

[30] Bastien Le Gloannec and Nicolas Ollinger. Substitutions and strongly deterministic tilesets. In *How the World Computes*, pages 462–471. Springer, 2012.

[31] Ville Lukkarila. The 4-way deterministic tiling problem is undecidable. *Theoretical Computer Science*, 410(16):1516 – 1533, 2009. Theory and Applications of Tilings.

[32] Kenichi Morita. Computation-universality of one-dimensional one-way reversible cellular automata. *Information Processing Letters*, 42(6):325 – 329, 1992.

[33] Shahar Mozes. Tilings, substitution systems and dynamical systems generated by them. *Journal d'analyse mathématique*, 53:139–186, 1988.

[34] Masakazu Nasu. Textile systems and one-sided resolving automorphisms and endomorphisms of the shift. *Ergodic Theory and Dynamical Systems*, 28:167–209, 2 2008.

[35] Nicolas Ollinger. The intrinsic universality problem of one-dimensional cellular automata. In Helmut Alt and Michel Habib, editors, *STACS 2003*, volume 2607 of *Lecture Notes in Computer Science*, pages 632–641. Springer Berlin Heidelberg, 2003.

[36] Nicolas Ollinger. Two-by-two substitution systems and the undecidability of the domino problem. In Arnold Beckmann, Costas Dimitracopoulos, and Benedikt Löwe, editors, *CiE'2008*, volume 5028 of *LNCS*, pages 476–485, Athens, Greece, June 2008. Springer Berlin / Heidelberg.

[37] Raphael M. Robinson. Undecidability and nonperiodicity for tilings of the plane. *Inventiones Mathematicæ*, 12(3), 1971.

[38] Stephen G. Simpson. Medvedev degrees of 2-dimensional subshifts of finite type, 2007.

[39] Guillaume Theyssier. How common can be universality for cellular automata? In *STACS 2005*, volume 3404 of *Lecture Notes in Computer Science*, pages 121–132. 2005.

[40] Hao Wang. Proving theorems by pattern recognition ii. *Bell System Technical Journal, The*, 40(1):1–41, Jan 1961.

[41] Charalampos Zinoviadis. Hierarchy and expansiveness in 2d subshifts of finite type. In *Language and Automata Theory and Applications - 9th International Conference, LATA 2015, Nice, France, March 2-6, 2015, Proceedings*, pages 365–377, 2015.

Turku Centre for Computer Science
TUCS Dissertations

1. **Marjo Lipponen**, On Primitive Solutions of the Post Correspondence Problem
2. **Timo Käkölä**, Dual Information Systems in Hyperknowledge Organizations
3. **Ville Leppänen**, Studies on the Realization of PRAM
4. **Cunsheng Ding**, Cryptographic Counter Generators
5. **Sami Viitanen**, Some New Global Optimization Algorithms
6. **Tapio Salakoski**, Representative Classification of Protein Structures
7. **Thomas Långbacka**, An Interactive Environment Supporting the Development of Formally Correct Programs
8. **Thomas Finne**, A Decision Support System for Improving Information Security
9. **Valeria Mihalache**, Cooperation, Communication, Control. Investigations on Grammar Systems.
10. **Marina Waldén**, Formal Reasoning About Distributed Algorithms
11. **Tero Laihonen**, Estimates on the Covering Radius When the Dual Distance is Known
12. **Lucian Ilie**, Decision Problems on Orders of Words
13. **Jukkapekka Hekanaho**, An Evolutionary Approach to Concept Learning
14. **Jouni Järvinen**, Knowledge Representation and Rough Sets
15. **Tomi Pasanen**, In-Place Algorithms for Sorting Problems
16. **Mika Johnsson**, Operational and Tactical Level Optimization in Printed Circuit Board Assembly
17. **Mats Aspnäs**, Multiprocessor Architecture and Programming: The Hathi-2 System
18. **Anna Mikhajlova**, Ensuring Correctness of Object and Component Systems
19. **Vesa Torvinen**, Construction and Evaluation of the Labour Game Method
20. **Jorma Boberg**, Cluster Analysis. A Mathematical Approach with Applications to Protein Structures
21. **Leonid Mikhajlov**, Software Reuse Mechanisms and Techniques: Safety Versus Flexibility
22. **Timo Kaukoranta**, Iterative and Hierarchical Methods for Codebook Generation in Vector Quantization
23. **Gábor Magyar**, On Solution Approaches for Some Industrially Motivated Combinatorial Optimization Problems
24. **Linas Laibinis**, Mechanised Formal Reasoning About Modular Programs
25. **Shuhua Liu**, Improving Executive Support in Strategic Scanning with Software Agent Systems
26. **Jaakko Järvi**, New Techniques in Generic Programming – C++ is more Intentional than Intended
27. **Jan-Christian Lehtinen**, Reproducing Kernel Splines in the Analysis of Medical Data
28. **Martin Büchi**, Safe Language Mechanisms for Modularization and Concurrency
29. **Elena Troubitsyna**, Stepwise Development of Dependable Systems
30. **Janne Näppi**, Computer-Assisted Diagnosis of Breast Calcifications
31. **Jianming Liang**, Dynamic Chest Images Analysis
32. **Tiberiu Seceleanu**, Systematic Design of Synchronous Digital Circuits
33. **Tero Aittokallio**, Characterization and Modelling of the Cardiorespiratory System in Sleep-Disordered Breathing
34. **Ivan Porres**, Modeling and Analyzing Software Behavior in UML
35. **Mauno Rönkkö**, Stepwise Development of Hybrid Systems
36. **Jouni Smed**, Production Planning in Printed Circuit Board Assembly
37. **Vesa Halava**, The Post Correspondence Problem for Market Morphisms
38. **Ion Petre**, Commutation Problems on Sets of Words and Formal Power Series
39. **Vladimir Kvassov**, Information Technology and the Productivity of Managerial Work
40. **Frank Tétard**, Managers, Fragmentation of Working Time, and Information Systems

41. **Jan Manuch**, Defect Theorems and Infinite Words
42. **Kalle Ranto**, Z_4-Goethals Codes, Decoding and Designs
43. **Arto Lepistö**, On Relations Between Local and Global Periodicity
44. **Mika Hirvensalo**, Studies on Boolean Functions Related to Quantum Computing
45. **Pentti Virtanen**, Measuring and Improving Component-Based Software Development
46. **Adekunle Okunoye**, Knowledge Management and Global Diversity – A Framework to Support Organisations in Developing Countries
47. **Antonina Kloptchenko**, Text Mining Based on the Prototype Matching Method
48. **Juha Kivijärvi**, Optimization Methods for Clustering
49. **Rimvydas Rukšėnas**, Formal Development of Concurrent Components
50. **Dirk Nowotka**, Periodicity and Unbordered Factors of Words
51. **Attila Gyenesei**, Discovering Frequent Fuzzy Patterns in Relations of Quantitative Attributes
52. **Petteri Kaitovaara**, Packaging of IT Services – Conceptual and Empirical Studies
53. **Petri Rosendahl**, Niho Type Cross-Correlation Functions and Related Equations
54. **Péter Majlender**, A Normative Approach to Possibility Theory and Soft Decision Support
55. **Seppo Virtanen**, A Framework for Rapid Design and Evaluation of Protocol Processors
56. **Tomas Eklund**, The Self-Organizing Map in Financial Benchmarking
57. **Mikael Collan**, Giga-Investments: Modelling the Valuation of Very Large Industrial Real Investments
58. **Dag Björklund**, A Kernel Language for Unified Code Synthesis
59. **Shengnan Han**, Understanding User Adoption of Mobile Technology: Focusing on Physicians in Finland
60. **Irina Georgescu**, Rational Choice and Revealed Preference: A Fuzzy Approach
61. **Ping Yan**, Limit Cycles for Generalized Liénard-Type and Lotka-Volterra Systems
62. **Joonas Lehtinen**, Coding of Wavelet-Transformed Images
63. **Tommi Meskanen,** On the NTRU Cryptosystem
64. **Saeed Salehi**, Varieties of Tree Languages
65. **Jukka Arvo**, Efficient Algorithms for Hardware-Accelerated Shadow Computation
66. **Mika Hirvikorpi**, On the Tactical Level Production Planning in Flexible Manufacturing Systems
67. **Adrian Costea**, Computational Intelligence Methods for Quantitative Data Mining
68. **Cristina Seceleanu**, A Methodology for Constructing Correct Reactive Systems
69. **Luigia Petre**, Modeling with Action Systems
70. **Lu Yan**, Systematic Design of Ubiquitous Systems
71. **Mehran Gomari**, On the Generalization Ability of Bayesian Neural Networks
72. **Ville Harkke**, Knowledge Freedom for Medical Professionals – An Evaluation Study of a Mobile Information System for Physicians in Finland
73. **Marius Cosmin Codrea**, Pattern Analysis of Chlorophyll Fluorescence Signals
74. **Aiying Rong**, Cogeneration Planning Under the Deregulated Power Market and Emissions Trading Scheme
75. **Chihab BenMoussa**, Supporting the Sales Force through Mobile Information and Communication Technologies: Focusing on the Pharmaceutical Sales Force
76. **Jussi Salmi**, Improving Data Analysis in Proteomics
77. **Orieta Celiku**, Mechanized Reasoning for Dually-Nondeterministic and Probabilistic Programs
78. **Kaj-Mikael Björk**, Supply Chain Efficiency with Some Forest Industry Improvements
79. **Viorel Preoteasa**, Program Variables – The Core of Mechanical Reasoning about Imperative Programs
80. **Jonne Poikonen**, Absolute Value Extraction and Order Statistic Filtering for a Mixed-Mode Array Image Processor
81. **Luka Milovanov**, Agile Software Development in an Academic Environment
82. **Francisco Augusto Alcaraz Garcia**, Real Options, Default Risk and Soft Applications
83. **Kai K. Kimppa**, Problems with the Justification of Intellectual Property Rights in Relation to Software and Other Digitally Distributable Media
84. **Dragoş Truşcan**, Model Driven Development of Programmable Architectures
85. **Eugen Czeizler**, The Inverse Neighborhood Problem and Applications of Welch Sets in Automata Theory

86. **Sanna Ranto**, Identifying and Locating-Dominating Codes in Binary Hamming Spaces
87. **Tuomas Hakkarainen**, On the Computation of the Class Numbers of Real Abelian Fields
88. **Elena Czeizler**, Intricacies of Word Equations
89. **Marcus Alanen**, A Metamodeling Framework for Software Engineering
90. **Filip Ginter**, Towards Information Extraction in the Biomedical Domain: Methods and Resources
91. **Jarkko Paavola**, Signature Ensembles and Receiver Structures for Oversaturated Synchronous DS-CDMA Systems
92. **Arho Virkki**, The Human Respiratory System: Modelling, Analysis and Control
93. **Olli Luoma**, Efficient Methods for Storing and Querying XML Data with Relational Databases
94. **Dubravka Ilić**, Formal Reasoning about Dependability in Model-Driven Development
95. **Kim Solin**, Abstract Algebra of Program Refinement
96. **Tomi Westerlund**, Time Aware Modelling and Analysis of Systems-on-Chip
97. **Kalle Saari**, On the Frequency and Periodicity of Infinite Words
98. **Tomi Kärki**, Similarity Relations on Words: Relational Codes and Periods
99. **Markus M. Mäkelä**, Essays on Software Product Development: A Strategic Management Viewpoint
100. **Roope Vehkalahti**, Class Field Theoretic Methods in the Design of Lattice Signal Constellations
101. **Anne-Maria Ernvall-Hytönen**, On Short Exponential Sums Involving Fourier Coefficients of Holomorphic Cusp Forms
102. **Chang Li**, Parallelism and Complexity in Gene Assembly
103. **Tapio Pahikkala**, New Kernel Functions and Learning Methods for Text and Data Mining
104. **Denis Shestakov**, Search Interfaces on the Web: Querying and Characterizing
105. **Sampo Pyysalo**, A Dependency Parsing Approach to Biomedical Text Mining
106. **Anna Sell**, Mobile Digital Calendars in Knowledge Work
107. **Dorina Marghescu**, Evaluating Multidimensional Visualization Techniques in Data Mining Tasks
108. **Tero Säntti**, A Co-Processor Approach for Efficient Java Execution in Embedded Systems
109. **Kari Salonen**, Setup Optimization in High-Mix Surface Mount PCB Assembly
110. **Pontus Boström**, Formal Design and Verification of Systems Using Domain-Specific Languages
111. **Camilla J. Hollanti**, Order-Theoretic Mehtods for Space-Time Coding: Symmetric and Asymmetric Designs
112. **Heidi Himmanen**, On Transmission System Design for Wireless Broadcasting
113. **Sébastien Lafond**, Simulation of Embedded Systems for Energy Consumption Estimation
114. **Evgeni Tsivtsivadze**, Learning Preferences with Kernel-Based Methods
115. **Petri Salmela**, On Commutation and Conjugacy of Rational Languages and the Fixed Point Method
116. **Siamak Taati**, Conservation Laws in Cellular Automata
117. **Vladimir Rogojin**, Gene Assembly in Stichotrichous Ciliates: Elementary Operations, Parallelism and Computation
118. **Alexey Dudkov**, Chip and Signature Interleaving in DS CDMA Systems
119. **Janne Savela**, Role of Selected Spectral Attributes in the Perception of Synthetic Vowels
120. **Kristian Nybom**, Low-Density Parity-Check Codes for Wireless Datacast Networks
121. **Johanna Tuominen**, Formal Power Analysis of Systems-on-Chip
122. **Teijo Lehtonen**, On Fault Tolerance Methods for Networks-on-Chip
123. **Eeva Suvitie**, On Inner Products Involving Holomorphic Cusp Forms and Maass Forms
124. **Linda Mannila**, Teaching Mathematics and Programming – New Approaches with Empirical Evaluation
125. **Hanna Suominen**, Machine Learning and Clinical Text: Supporting Health Information Flow
126. **Tuomo Saarni**, Segmental Durations of Speech
127. **Johannes Eriksson**, Tool-Supported Invariant-Based Programming

128. **Tero Jokela**, Design and Analysis of Forward Error Control Coding and Signaling for Guaranteeing QoS in Wireless Broadcast Systems
129. **Ville Lukkarila**, On Undecidable Dynamical Properties of Reversible One-Dimensional Cellular Automata
130. **Qaisar Ahmad Malik**, Combining Model-Based Testing and Stepwise Formal Development
131. **Mikko-Jussi Laakso**, Promoting Programming Learning: Engagement, Automatic Assessment with Immediate Feedback in Visualizations
132. **Riikka Vuokko**, A Practice Perspective on Organizational Implementation of Information Technology
133. **Jeanette Heidenberg**, Towards Increased Productivity and Quality in Software Development Using Agile, Lean and Collaborative Approaches
134. **Yong Liu**, Solving the Puzzle of Mobile Learning Adoption
135. **Stina Ojala**, Towards an Integrative Information Society: Studies on Individuality in Speech and Sign
136. **Matteo Brunelli**, Some Advances in Mathematical Models for Preference Relations
137. **Ville Junnila**, On Identifying and Locating-Dominating Codes
138. **Andrzej Mizera**, Methods for Construction and Analysis of Computational Models in Systems Biology. Applications to the Modelling of the Heat Shock Response and the Self-Assembly of Intermediate Filaments.
139. **Csaba Ráduly-Baka**, Algorithmic Solutions for Combinatorial Problems in Resource Management of Manufacturing Environments
140. **Jari Kyngäs**, Solving Challenging Real-World Scheduling Problems
141. **Arho Suominen**, Notes on Emerging Technologies
142. **József Mezei**, A Quantitative View on Fuzzy Numbers
143. **Marta Olszewska**, On the Impact of Rigorous Approaches on the Quality of Development
144. **Antti Airola**, Kernel-Based Ranking: Methods for Learning and Performace Estimation
145. **Aleksi Saarela**, Word Equations and Related Topics: Independence, Decidability and Characterizations
146. **Lasse Bergroth**, Kahden merkkijonon pisimmän yhteisen alijonon ongelma ja sen ratkaiseminen
147. **Thomas Canhao Xu**, Hardware/Software Co-Design for Multicore Architectures
148. **Tuomas Mäkilä**, Software Development Process Modeling – Developers Perspective to Contemporary Modeling Techniques
149. **Shahrokh Nikou**, Opening the Black-Box of IT Artifacts: Looking into Mobile Service Characteristics and Individual Perception
150. **Alessandro Buoni**, Fraud Detection in the Banking Sector: A Multi-Agent Approach
151. **Mats Neovius**, Trustworthy Context Dependency in Ubiquitous Systems
152. **Fredrik Degerlund**, Scheduling of Guarded Command Based Models
153. **Amir-Mohammad Rahmani-Sane**, Exploration and Design of Power-Efficient Networked Many-Core Systems
154. **Ville Rantala**, On Dynamic Monitoring Methods for Networks-on-Chip
155. **Mikko Pelto**, On Identifying and Locating-Dominating Codes in the Infinite King Grid
156. **Anton Tarasyuk**, Formal Development and Quantitative Verification of Dependable Systems
157. **Muhammad Mohsin Saleemi**, Towards Combining Interactive Mobile TV and Smart Spaces: Architectures, Tools and Application Development
158. **Tommi J. M. Lehtinen**, Numbers and Languages
159. **Peter Sarlin**, Mapping Financial Stability
160. **Alexander Wei Yin**, On Energy Efficient Computing Platforms
161. **Mikołaj Olszewski**, Scaling Up Stepwise Feature Introduction to Construction of Large Software Systems
162. **Maryam Kamali**, Reusable Formal Architectures for Networked Systems
163. **Zhiyuan Yao**, Visual Customer Segmentation and Behavior Analysis – A SOM-Based Approach
164. **Timo Jolivet**, Combinatorics of Pisot Substitutions
165. **Rajeev Kumar Kanth**, Analysis and Life Cycle Assessment of Printed Antennas for Sustainable Wireless Systems
166. **Khalid Latif**, Design Space Exploration for MPSoC Architectures

167. **Bo Yang**, Towards Optimal Application Mapping for Energy-Efficient Many-Core Platforms
168. **Ali Hanzala Khan**, Consistency of UML Based Designs Using Ontology Reasoners
169. **Sonja Leskinen**, m-Equine: IS Support for the Horse Industry
170. **Fareed Ahmed Jokhio**, Video Transcoding in a Distributed Cloud Computing Environment
171. **Moazzam Fareed Niazi**, A Model-Based Development and Verification Framework for Distributed System-on-Chip Architecture
172. **Mari Huova**, Combinatorics on Words: New Aspects on Avoidability, Defect Effect, Equations and Palindromes
173. **Ville Timonen**, Scalable Algorithms for Height Field Illumination
174. **Henri Korvela**, Virtual Communities – A Virtual Treasure Trove for End-User Developers
175. **Kameswar Rao Vaddina**, Thermal-Aware Networked Many-Core Systems
176. **Janne Lahtiranta**, New and Emerging Challenges of the ICT-Mediated Health and Well-Being Services
177. **Irum Rauf**, Design and Validation of Stateful Composite RESTful Web Services
178. **Jari Björne**, Biomedical Event Extraction with Machine Learning
179. **Katri Haverinen**, Natural Language Processing Resources for Finnish: Corpus Development in the General and Clinical Domains
180. **Ville Salo**, Subshifts with Simple Cellular Automata
181. **Johan Ersfolk**, Scheduling Dynamic Dataflow Graphs
182. **Hongyan Liu**, On Advancing Business Intelligence in the Electricity Retail Market
183. **Adnan Ashraf**, Cost-Efficient Virtual Machine Management: Provisioning, Admission Control, and Consolidation
184. **Muhammad Nazrul Islam**, Design and Evaluation of Web Interface Signs to Improve Web Usability: A Semiotic Framework
185. **Johannes Tuikkala**, Algorithmic Techniques in Gene Expression Processing: From Imputation to Visualization
186. **Natalia Díaz Rodríguez**, Semantic and Fuzzy Modelling for Human Behaviour Recognition in Smart Spaces. A Case Study on Ambient Assisted Living
187. **Mikko Pänkäälä**, Potential and Challenges of Analog Reconfigurable Computation in Modern and Future CMOS
188. **Sami Hyrynsalmi**, Letters from the War of Ecosystems – An Analysis of Independent Software Vendors in Mobile Application Marketplaces
189. **Seppo Pulkkinen**, Efficient Optimization Algorithms for Nonlinear Data Analysis
190. **Sami Pyöttiälä**, Optimization and Measuring Techniques for Collect-and-Place Machines in Printed Circuit Board Industry
191. **Syed Mohammad Asad Hassan Jafri**, Virtual Runtime Application Partitions for Resource Management in Massively Parallel Architectures
192. **Toni Ernvall**, On Distributed Storage Codes
193. **Yuliya Prokhorova**, Rigorous Development of Safety-Critical Systems
194. **Olli Lahdenoja**, Local Binary Patterns in Focal-Plane Processing – Analysis and Applications
195. **Annika H. Holmbom**, Visual Analytics for Behavioral and Niche Market Segmentation
196. **Sergey Ostroumov**, Agent-Based Management System for Many-Core Platforms: Rigorous Design and Efficient Implementation
197. **Espen Suenson**, How Computer Programmers Work – Understanding Software Development in Practise
198. **Tuomas Poikela**, Readout Architectures for Hybrid Pixel Detector Readout Chips
199. **Bogdan Iancu**, Quantitative Refinement of Reaction-Based Biomodels
200. **Ilkka Törmä**, Structural and Computational Existence Results for Multidimensional Subshifts
201. **Sebastian Okser**, Scalable Feature Selection Applications for Genome-Wide Association Studies of Complex Diseases
202. **Fredrik Abbors**, Model-Based Testing of Software Systems: Functionality and Performance
203. **Inna Pereverzeva**, Formal Development of Resilient Distributed Systems
204. **Mikhail Barash**, Defining Contexts in Context-Free Grammars
205. **Sepinoud Azimi**, Computational Models for and from Biology: Simple Gene Assembly and Reaction Systems
206. **Petter Sandvik**, Formal Modelling for Digital Media Distribution

207. **Jongyun Moon**, Hydrogen Sensor Application of Anodic Titanium Oxide Nanostructures
208. **Simon Holmbacka**, Energy Aware Software for Many-Core Systems
209. **Charalampos Zinoviadis**, Hierarchy and Expansiveness in Two-Dimensional Subshifts of Finite Type

ISBN 978-952-12-3337-1
ISSN 1239-1883

Turku
Centre *for*
Computer
Science

http://www. tucs.fi

tucs@abo.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics and Statistics

Turku School of Economics
- Institute of Information Systems Science

Åbo Akademi University
Faculty of Science and Engineering
- Computer Engineering
- Computer Science

Faculty of Social Sciences, Business and Economics
- Information Systems

ISBN 978-952-12-3337-1
ISSN 1239-1883

Charalampos Zinoviadis

Hierarchy and Expansiveness in Two-Dimensional Subshifts of Finite Type