On the Expressive Power of Quasiperiodic SFT*

Bruno Durand[1] **and Andrei Romashchenko**[2]

1 **Univ. Montpellier & LIRMM, Montpellier, France**
2 **CNRS, Paris, France & LIRMM, Montpellier, France**
on leave from IITP RAS, Moscow, Russia

Abstract

In this paper we study the *shifts*, which are the shift-invariant and topologically closed sets of configurations over a finite alphabet in $\mathbb{Z}^d$. The *minimal shifts* are those shifts in which all configurations contain exactly the same patterns. Two classes of shifts play a prominent role in symbolic dynamics, in language theory and in the theory of computability: the *shifts of finite type* (obtained by forbidding a finite number of finite patterns) and the *effective shifts* (obtained by forbidding a computably enumerable set of finite patterns). We prove that every effective minimal shift can be represented as a factor of a projective subdynamics on a minimal shift *of finite type* in a bigger (by 1) dimension. This result transfers to the class of minimal shifts a theorem by M. Hochman known for the class of all effective shifts and thus answers an open question by E. Jeandel. We prove a similar result for quasiperiodic shifts and also show that there exists a quasiperiodic shift of finite type for which Kolmogorov complexity of all patterns of size $n \times n$ is $\Omega(n)$.

1998 ACM Subject Classification F.4.1 Computability theory, G.2.0 Discrete mathematics, G.2.1 Combinatorics

Keywords and phrases minimal SFT, tilings, quasiperiodicity

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.5

1 Introduction

The study of symbolic dynamics was initially motivated as a discretization of classic dynamical systems, [9]. Later, the focus of attention in this area shifted towards the questions related to computability theory. The central notion of symbolic dynamics is a *shift* (a.k.a. *subshift*), which is a set of configurations in $\mathbb{Z}^d$ over a finite alphabet, defined by a set of forbidden patterns. Two major notions – two classes of shifts – play now a crucial role in symbolic dynamics: shifts of *finite type* (SFT, the shift defined by a finite set of forbidden patterns) and *effective* shifts (a.k.a. *effectively closed* – shifts with an enumerable set of forbidden patterns). These classes are distinct: every SFT is effective, but in general the reverse implication does not hold. However, the differences between these classes is surprisingly subtle. It is known that every effective shift can be simulated in some sense by an SFT of higher dimension. More precisely, every effective shift in $\mathbb{Z}^d$ can be represented as a factor of the projective subdynamics of an SFT of dimension increased by 1, see [10, 6, 1].

Usually, the proofs of computability results in symbolic dynamics involve sophisticated algorithmic gadgets embedded in dynamical systems. The resulting constructions are typically intricate and somewhat artificial. So, even if the shifts (effective or SFT) in general are proven to have a certain algorithmic property, the known proof may be inappropriate for "natural"

* Supported by ANR-15-CE40-0016-01 RaCAF grant

42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017).
Editors: Kim G. Larsen, Hans L. Bodlaender, and Jean-Francois Raskin; Article No. 5; pp. 5:1–5:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

dynamical systems. Thus, it is interesting to understand the limits of the known algorithmic techniques and find out whether the remarkable properties of algorithmic complexity can be extended to “simple” and mathematically “natural” types of shifts.

One of the classic natural types of dynamical systems is the class of *minimal shifts*. Minimal shifts are those containing no proper shift, or equivalently the shifts where all configurations have exactly the same patterns. The role minimal shifts play in symbolic dynamics is similar to the role simple groups play in group theory (in particular every nonempty shift contains a nonempty minimal shift, see a discussion in [4]). Notice that all minimal shifts are quasiperiodic (but the converse is not true). Intuitively it seems that the structure of a minimal shift must be simple (in terms of dynamical systems). Besides, minimal shifts cannot be “too complex” in algorithmic terms. Indeed, it is known that every effective minimal shift has a computable language of patterns, and it contains at least one computable configuration [10] (which is in general not the case for effective shifts and even for SFT). Nevertheless, minimal shifts can have quite nontrivial algorithmic properties [13, 11].

We have mentioned above that every effective shift $\mathcal{S}$ can be represented as a factor of a projective subdynamics of an SFT $\mathcal{S}'$ (of higher dimension). In the previously known proofs of this result [10, 6, 1], even if $\mathcal{S}$ is minimal, the structure of the corresponding SFT $\mathcal{S}'$ (that simulates by its projective subdynamics the given $\mathcal{S}$) can be very sophisticated (and far from being minimal). So, a natural question arises (E. Jeandel, [12]): is it true that every effective minimal (or quasiperiodic) shift can be represented as a factor of a projective subdynamics on a minimal (respectively, quasiperiodic) SFT of higher dimension? In this paper we give a positive answer to that questions.

The full proof of the main result of this paper is rather cumbersome for the following reason: we use the technique of self-simulating tilings (e.g., [6, 7, 16]) combined with some combinatorial lemmas on quasiperiodic configurations. Unfortunately, there is no clean separation between the generic technique of self-simulating tilings and the supplementary features embedded in this type of tilings, so we cannot use the (previously known) technique of self-simulation as a “black box”. We have to re-explain the core techniques of fixed-point programming embedded in tilings and adjust the supplementary features within the construction. While explaining the proofs, we have to balance clarity with formality, and given the usual space limits of the conference paper we have to sketch some standard parts of the proof. An extended version of this paper is published on `arXiv:1705.01876`.

1.1 Notation and basic definitions

Let Σ be a finite set (an alphabet). Fix an integer $d > 0$. A Σ-*configuration* is a mapping $\mathbf{f} : \mathbb{Z}^d \to \Sigma$. A $\mathbb{Z}^d$-*shift* (or just a *shift* if d is clear from the context) is a set of configuration that is (i) shift-invariant (with respect to the translations along each coordinate axis), and (ii) closed in Cantor's topology.

A *pattern* is a mapping from a finite subset in $\mathbb{Z}^d$ to Σ (a coloring of a finite set of $\mathbb{Z}^d$). Every shift can be defined by a set of forbidden finite patterns F (a configuration belongs to the shift if and only if it does not contain any pattern from F). A shift is called *effective* (or *effectively closed*) if it can be defined by a computably enumerable set of forbidden patterns. A shift is called a *shift of finite type* (SFT), if it can be defined by a finite set of forbidden patterns.

A special class of a 2-dimensional SFT is defined in terms of *Wang tiles*. In this case we interpret the alphabet Σ as a set of *tiles* – unite squares with colored sides, assuming that all colors belong to some finite set C (we assign one color to each side of a tile, so technically Σ is a subset of C^4). A (valid) *tiling* is a set of all configurations where every two neighboring tiles match, i.e., share the same color on the adjacent sides. Wang tiles are powerful enough to simulate any SFT in a very strong sense: for each SFT $\mathcal{S}$ there exists a set of Wang tiles τ such that the set of all τ-tilings is isomorphic to $\mathcal{S}$. In this paper we focus on tilings since Wang tiles perfectly suit the technique of self-simulation.

A shift $\mathcal{S}$ (in the full shift $\Sigma^{\mathbb{Z}^d}$) can be interpreted as a dynamical system. There are d shifts along each of the coordinates, and each of these shifts map $\mathcal{S}$ to itself. So, the group $\mathbb{Z}^d$ naturally acts on $\mathcal{S}$.

For any shift $\mathcal{S}$ on $\mathbb{Z}^d$ and for any k-dimensional sublattice L in $\mathbb{Z}^d$, the *L-projective subdynamics* $\mathcal{S}_L$ of $\mathcal{S}$ is the set of configurations of $\mathcal{S}$ restricted on L. The L-projective subdynamics of a $\mathbb{Z}^d$-shift can be understood as a $\mathbb{Z}^k$-shift (notice that L naturally acts on $\mathcal{S}_L$). In particular, for every $d' < d$ we have a *standard* $\mathbb{Z}^{d'}$-projective subdynamics on the shift $\mathcal{S}$ generated by the lattice spanned on the first d' coordinate axis. In the proofs of Theorems 1-2 we deal with the standard $\mathbb{Z}^{(d-1)}$-projective subdynamics on $\mathbb{Z}^d$-shifts.

A configuration ω is called *recurrent* if every pattern that appears in ω at least once, must then appear in this configuration infinitely often. A configuration ω is called *quasiperiodic* (or *uniformly recurrent*) if every pattern P that appears in ω at least once, must appear in every pattern Q large enough in ω. Notice that every periodic configuration is also quasiperiodic. It is easy to see that if a shift $\mathcal{S}$ is minimal, then every $\omega \in \mathcal{S}$ is quasiperiodic.

For a quasiperiodic configuration ω, its *function of a quasiperiodicity* is a mapping $\varphi : \mathbb{N} \to \mathbb{N}$ such that every finite pattern of diameter n either never appears in ω, or it appears in every pattern of size $\varphi(n)$ in ω, see [4]. Similarly, a shift $\mathcal{S}$ has a function of quasiperiodicity φ, if φ is a function of a quasiperiodicity for every configuration in $\mathcal{S}$.

If a shift $\mathcal{S}$ is minimal, then all configurations in $\mathcal{S}$ have exactly the same finite patterns. For every minimal shift $\mathcal{S}$, the function of quasiperiodicity is finite (for every n) and even computable. Moreover, for an effective minimal shift, the set of all finite patterns (that can appear in any configuration) is computable, see [10, 3]. From this fact it follows that every effective and minimal shift contains a computable configuration.

1.2 The main results

Our first theorem claims that every effective quasiperiodic $\mathbb{Z}^d$-shift can be simulated by a quasiperiodic SFT in $\mathbb{Z}^{d+1}$.

▶ **Theorem 1.** *Let $\mathcal{A}$ be an effective quasiperiodic $\mathbb{Z}^d$-shift over some alphabet Σ_A. Then there exists a quasiperiodic SFT $\mathcal{B}$ (over another alphabet Σ_B) of dimension $d+1$ such that $\mathcal{A}$ is isomorphic to a factor of a d-dimensional projective subdynamics on $\mathcal{B}$.*

A similar result holds for effective minimal shifts:

▶ **Theorem 2.** *For every effective minimal $\mathbb{Z}^d$-shift $\mathcal{A}$ there exists a minimal SFT $\mathcal{B}$ in $\mathbb{Z}^{d+1}$ such that $\mathcal{A}$ is isomorphic to a factor of a d-dimensional projective subdynamics on $\mathcal{B}$.*

Theorem 1 implies the following somewhat surprising corollary (a quasiperiodic $\mathbb{Z}^2$-SFT can have highly "complex" languages of patterns):

▶ **Corollary 3.** *There exists a quasiperiodic SFT $\mathcal{A}$ of dimension 2 such that Kolmogorov complexity of every $(N \times N)$-pattern in every configuration of $\mathcal{A}$ is $\Omega(N)$.*

▶ Remark. A standalone pattern of size $N \times N$ over an alphabet Σ (with at least two letters) can have a Kolmogorov complexity up to $\Theta(N^2)$. However, this density of information cannot be enforced by local rules, because in every SFT in $\mathbb{Z}^2$ there exists a configuration such that Kolmogorov complexity of all $N \times N$-patterns is bounded by $O(N)$, [5]. Thus, the lower bound $\Omega(N)$ in Corollary 3 is optimal in the class of all SFT.

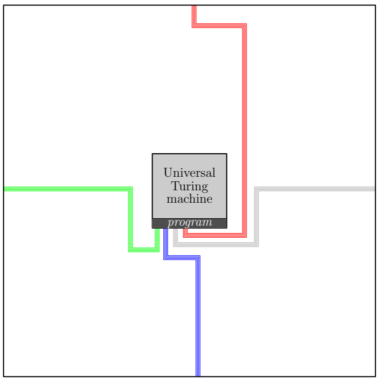

Figure 1 The structure of a macro-tile.

▶ Remark. Every effective (effectively-closed) minimal shift $\mathcal{A}$ is *computable* (given a pattern, we can algorithmically decide whether it belongs to the configurations of the shift). Patterns of high Kolmogorov complexity cannot be found algorithmically. So Corollary 3 cannot be extended to the class of minimal SFT.

To simplify notation and make the argument more visual, in what follows we focus on the case $d = 1$. The proofs extend to any $d > 1$ in a straightforward way, *mutatis mutandis*.

2 The general framework of self-simulating SFT

In what follows we extensively use the technique of *self-simulating tilesets* from [6] (this technique goes back to [8]). We use the idea of self-simulation to enforce a kind of *self-similar* structure in a tiling. In this section we remind the reader of the principal ingredients of this construction.

Let τ be a tileset and $N > 1$ be an integer. We call a τ*-macro-tile* an $N \times N$ square correctly tiled by tiles from τ. Every side of a τ-macro-tile contains a sequence of N colors (of tiles from τ); we refer to this sequence as a *macro-color*. A tileset τ *simulates* another tileset ρ, if there exists a set of τ-macro-tiles T such that

- there is one-to-one correspondence between ρ and T (the colors of two tiles from ρ match if and only if the macro-colors of the corresponding macro-tiles from T match),
- for every τ-tiling there exists a unique lattice of vertical and horizontal lines that splits this tiling into $N \times N$ macro-tiles from T, i.e., every τ-tiling represents a unique ρ-tiling.

For a large class of sufficiently "well-behaved" sequence of integers N_k we can construct a family of tilesets τ_k ($i = 0, 1, \ldots$) such that each τ_{k-1} simulates the next τ_k with the zoom N_k (and, therefore, τ_0 simulates every τ_k with the zoom $L_k = N_1 \cdot N_2 \cdots N_k$).

If a k-level macro-tile M is a "cell" in a $(k + 1)$-level macro-tile M', we refer to M' as a *father* of M; we call the $(k + 1)$-level macro-tiles neighboring M' *uncles* of M.

In our construction each tile of τ_k "knows" its coordinates modulo N_k in the tiling: the colors on the left and on the bottom sides should involve (i, j), the color on the right side should involve $(i + 1 \mod N_k, j)$, and the color on the top side, respectively, involves $(i, j + 1 \mod N_k)$. So every τ_k-tiling can be uniquely split into blocks (macro-tiles) of size $N_k \times N_k$, where the coordinates of cells range from $(0, 0)$ in the bottom-left corner to $(N - 1, N - 1)$ in top-right corner. Intuitively, each tile "knows" its position in the corresponding macro-tile.

In addition to the coordinates, each tile in τ_k has some supplementary information encoded in the colors on its sides (the size of the supplementary information is always bounded by $O(1)$). In the middle of each side of a macro-tile we allocate $s_k \ll N_k$ positions where an array of s_k bits represents a color of a tile from τ_{k+1} (these s_k bits are embedded in colors on the sides of s_k tiles of a macro-tile, one bit per a cell). We fix some cells in a macro-tile that serve as “communication wires” and then require that these tiles carry the same (transferred) bit on two sides (so the bits of “macro-colors” are transferred from the sides of macro-color towards its central part). The central part of a macro-tile (of size, say $m_k \times m_k$, where $m_k = \text{poly}(\log N_k)$) is a *computation zone*; it represents a space-time diagram of a universal Turing machine (the tape is horizontal, time goes up), see Fig. 1.

The first line of the computation zone contains the following *fields* of the *input data*:

(i) the program of a Turing machine π that verifies that a quadruple of macro-colors correspond to one valid macro-color,

(ii) the binary expansion of the integer rank k of this macro-tile,

(iii) the bits encoding the macro-colors – the position inside the “father” macro-tile of rank $(k+1)$ (two coordinates modulo N_{k+1}) and $O(1)$ bits of the supplementary information assigned to the macro-colors.

We require that the simulated computation terminates in an accepting state (if not, no correct tiling can be formed). The simulated computation guarantees that macro-tiles of level k are isomorphic to the tiles of τ_{k+1}. Notice that on each level k of the hierarchy we simulate in macro-tiles a computation of one and the same Turing machine π. Only the inputs for this machine (including the binary expansion of the rank number k) varies on different levels of the hierarchy.

This construction of a tileset can be implemented using the standard technique of self-referential programming, similar to the Kleene recursion theorem, as it is shown in [6]. The construction works if the size of a macro-tile (the zoom factor N_k) is large enough. First, we need enough space in a macro-tile to “communicate” s_k bits from each macro-colors to the computation zone; second, we need a large enough computation zone, so all accepting computations terminate in time m_k and on space m_k. In what follows we assume that $N_k = 3^{C^k}$ for some large enough k.

3 Embedding a bi-infinite sequence into a self-simulating tiling

In this section we adapt the technique from [6] and explain how to “encode” in a self-simulating tiling a bi-infinite sequence, and provide to the computation zones of macro-tiles of all ranks an access to the letters of the embedded sequences.

We are going to embed in our tiling a bi-infinite sequence $\mathbf{x} = (x_i)$ over an alphabet Σ. To this end we assume that each τ-tile “keeps” a letter from Σ that propagates without change in the vertical direction. Formally speaking, a letter from Σ should be a part of the top and bottom colors of every τ-tile (the letters assigned to both sides of a tile must be equal to each other). We want to guarantee that a Σ-sequence can be embedded in a τ-tiling, if and only if it belongs to some fixed effective $\mathcal{A}$ (so far quasiperiodicity is not assumed).

We want to “delegate” the factors of the embedded sequence to the computation zones of macro-tiles, where these factors will be validated (that is, we will check that they do not contain any forbidden subwords). While using tilings with growing zoom factor, we can guarantee that the size of the computation zone of a k-rank macro-tile grows with the rank k. So we have at our disposal the computational resources suitable to run all necessary validation tests on the embedded sequence. It remains to organize the propagation of the letters of the embedded sequence to the “conscious memory” (the computation zones) of macro-tiles of all ranks. In what follows we explain how this propagation is organized.

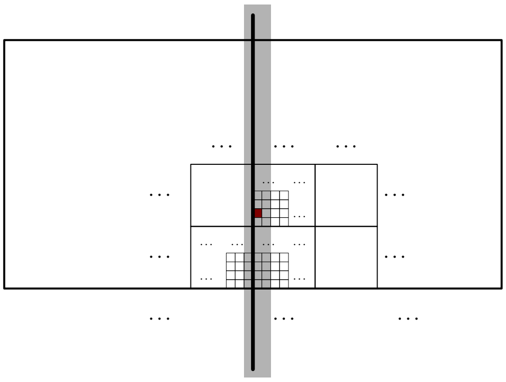

Figure 2 The zone of responsibility (the grey vertical stripe) for a macro-tile (the red square) is 3 times wider than the macro-tile itself.

Zone of responsibility of macro-tiles. In our construction, a macro-tile of level k is a square of size $L_k \times L_k$, with $L_k = N_1 \cdot N_2 \cdot \ldots \cdot N_k$ (where N_i is the zoom factor on level i of the hierarchy of macro-tiles). We say that a k-level macro-tile is *responsible* for the letters of the embedded sequence $\mathbf{x}$ assigned to the columns of (ground level) tiles of this macro-tile as well as to the columns of macro-tiles of the same rank on its left and on its right. That is, the *zone of responsibility* of a k-level macro-tile is a factor of length $3L_k$ from the embedded sequence, see Fig. 2. (The zones of responsibility of any two horizontally neighboring macro-tiles overlap.)

Letters assignment: The computation zone of a k-level macro-tile (of size $m_k \times m_k$) is too small to contain all letters from its zone of responsibility. So we require that the computation zone obtains as an input a (short enough) chunk of letters from its zone of responsibility. Let us say, that it is a factor of length $l_k = \log\log L_k$ from the stripe of $3L_k$ columns constituting the zone of responsibility of this macro-tile. We say that this chunk is *assigned* to this macro-tile.

The infinite stripe of vertically aligned k-level macro-tiles share the same zone of responsibility. However, different macro-tiles in such a stripe will obtain different assigned chunks. The choice of the assigned chunk varies from 0 to $(3L_k - l_k)$. We need to choose a position of a factor of length l_k in a word of length L_k. Let us say for certainty that for a macro-tile M of rank k the first position of the assigned chunk (in the stripe of length $3L_k$) is defined as the vertical position of M in the bigger macro-tile of rank $(k+1)$ (modulo $(3L_k - l_k)$).

▶ Remark. We have chosen the zoom factors N_k so that $N_{k+1} \gg 3L_k$. Hence, every chunk of length l_k from a stripe of width $3L_k$ is assigned to some of the macro-tiles “responsible” for these $3L_k$ letters. Since the zones of responsibility of neighboring k-level macro-tiles overlap by more than l_k, every finite factor of length l_k in the embedded sequence $\mathbf{x}$ is assigned to some k-level macro-tile (even if it involves columns of two macro-tiles of rank k).

Implementing the letters assignment by self-simulation. In the *letters assignment* paragraph above we presented some requirements – how the data must be propagated from the ground level (individual tiles) to k-level macro-tiles. Technically, for each k-level macro-tile $\mathcal{M}$ we specified which chunk of the embedded sequence should be a part of the data fields on the computation zone of $\mathcal{M}$. So far we have not explained how the assigned chunks arrive to the high-level data fields. Now, we are going to explain how to implement the desired scheme of letter assignment in a self-simulating tiling. Technically, we append to the input data of the computation zones of macro-tiles some supplementary data fields:

(v) the block of l_k letters from the embedded sequence assigned to this macro-tile,
(vi) three blocks of bits of l_{k+1} letters of the embedded sequence assigned to this "father" macro-tile, and two "uncle" macro-tiles (the left and the right neighbors of the "father"),
(vii) the coordinates of the "father" macro-tile in the "grandfather" (of rank $(k+2)$).

Informally, each k-level macro-tile must check that the data in the fields (iv), (v) and (vi) is consistent. That is, if some letters from the fields (iv) and (v) correspond to the same vertical column (in the zone of responsibility), then these letters must be equal to each other. Also, if a k-level macro-tile plays the role of cell in the computation zone of the $(k+1)$-level father, it should check the consistency of its (v) and (vi) with the bits displayed in father's computation zone. Finally, we must ensure the coherence of the fields (v) and (vi) for each pair of neighboring k-level macro-tiles; so this data should make a part of the macro-colors.

Notice that the data from "uncles" macro-tiles is necessary to deal with the letters from the columns that physically belong to the neighboring macro-tiles. So the consistency of the fields (v) is imposed also on neighboring k-level macro-tiles that belong to different $(k+1)$-level fathers (the boarder line between these k-level macro-tiles is also the boarder line between their fathers).

The computations verifying the coherence of the new fields can be performed in polynomial time, and the required update of the construction fits the constraints on the parameter. See a more detailed discussion on "letter delegation" in [6, Section 7].

Final remarks: testing against forbidden factors. To guarantee that the embedded sequence $\mathbf{x}$ contains no forbidden patterns, each k-level macro-tile should allocate some part of its computation zone to enumerate (within the limits of available space and time) the forbidden pattern, and verify that the block of l_k letters assigned to this macro-tile contains none of the found forbidden factors.

The time and space allocated to enumerating the forbidden words grow as a function of k. To ensure that the embedded sequence contains no forbidden patterns, it is enough to guarantee that each forbidden pattern is found by macro-tiles of high enough rank, and every factor of the embedded sequence is compared (on some level of the hierarchy) with every forbidden factor. Thus, we have a general construction of a 2D tiling that simulates a given, effective 1D shift. In the next sections we explain how to make these tilings quasiperiodic in the case when the simulated 1D shift is also quasiperiodic.

4 Combinatorial lemmas: the direct product of quasiperiodic and periodic sequences

The technique from [6] allows to embed in a self-similar tiling a 1-dimensional sequence and handle factors of this sequence. However, the previously known constructions cannot guarantee minimality or even quasiperiodicity of the resulting tiling, even if the embedded sequences have very simple combinatorial structure. To achieve the property of quasiperiodicity we will need some new techniques. The new parts of the argument begins with two simple combinatorial lemmas concerning quasiperiodic sequences.

▶ **Lemma 4.** (*see [2, 15]*) *Let* $\mathbf{x}$ *be a bi-infinite recurrent sequence,* w *be a finite factor in* $\mathbf{x}$*, and* q *be a positive integer number. Then there exists an integer* $t > 0$ *such that another copy of* w *appears in* $\mathbf{x}$ *with a shift* $q \cdot t$*. In other words, there exists another instance of the same* *factor w with a shift divisible by q. Moreover, if $\mathbf{x}$ is quasiperiodic, then the gap $q \cdot t$ between neighboring appearances of w is bounded by some number L that depends on $\mathbf{x}$ and w (but not on a specific instance of the factor x in the sequence).*

Notation: For a configuration $\mathbf{x}$ (over some finite alphabet) we denote with $\mathcal{S}(\mathbf{x})$ the shift that consists of all configurations $\mathbf{x}'$ containing only patterns from $\mathbf{x}$. If a shift $\mathcal{T}$ is minimal, then $\mathcal{S}(\mathbf{x}) = \mathcal{T}$ for all configurations $\mathbf{x} \in \mathcal{T}$.

▶ **Lemma 5.** *(a) Let $\mathcal{T}$ be an effective minimal shift. Then for every $\mathbf{x} = (x_i)$ from $\mathcal{T}$ and every periodic configuration $\mathbf{y} = (y_i)$ the direct product $\mathbf{x} \otimes \mathbf{y}$ (the bi-infinite sequence of pairs (x_i, y_i) for $i \in \mathbb{Z}$) generates a minimal shift, i.e., $\mathcal{S}(\mathbf{x} \otimes \mathbf{y})$ is minimal. (b) If in addition the sequence $\mathbf{x}$ are computable, then the set of patterns in $\mathcal{S}(\mathbf{x} \otimes \mathbf{y})$ is also computable.*

▶ Remark. In general, different configurations $\mathbf{x} \in \mathcal{T}$ in the product with one and the same periodic $\mathbf{y}$ can result in different shifts $\mathcal{S}(\mathbf{x} \otimes \mathbf{y})$.

Lemma 5 can be deduced from the fact that for every effective minimal shift the function of quasiperiodicity is computable, [10], and Lemma 4.

5 Towards quasiperiodic SFT

In this section we combine the combinatorial lemmas from the previous section with the technique of enforcing quasiperiodicity from [7], and prove our main results.

5.1 When macro-tiles are clones of each other

To show that (some) self-simulating tilings enjoy the property of quasiperiodicity, we need a tool to prove that every pattern in a tiling has "clones" (equal patterns) in each large enough fragment of this tiling. In our tiling every finite pattern is covered by a block of (at most) four macro-tiles of high enough rank, so we can focus on the search for "clones" in macro-tiles. The following lemma gives a natural characterization of the equality of two macro-tiles in a tiling: they must have the same information in their "conscious memory" (the data written on the tape of the Turing machine in the computation zone) and the same information hidden in their "deep subconscious" (the fragments of the embedded 1D sequence corresponding to the responsibility zones of these macro-tiles must be identical).

▶ **Lemma 6.** *Two macro-tiles of rank k are equal to each other if and only if they (a) contain the same bits in the fields (i) - (vi) in the input data on the computation zone, and (b) the factors of the encoded sequence corresponding to the zones of responsibility of these macro-tiles (in the corresponding vertical stripes of width $3N_k$) are equal to each other.*

Proof. Induction by the rank k. For the macro-tile of rank 1 the statement follows directly from the construction. For a pair of macro-tiles M_1 and M_2 of rank $(k+1)$ with identical data in the fields (i) - (vi) we observe that the corresponding "cells" in M_1 and M_2 (which are macro-tiles of rank k) contain the same data in their own fields (i) - (vi), since the communication wires of M_1 and M_2 carry the same information bits, their computation zones represent exactly the same computations, etc. If the factors (of length $3L_k$) from the encoded sequences in the zones of responsibility of M_1 and M_2 are also equal to each other, we can apply the inductive assumption. ◀

5.2 Supplementary features: constraints that can be imposed on the self-simulating tiling

The tiles involved in our self-simulating tiles set (as well as all macro-tile of each rank) can be classified into three types:

(a) the "skeleton" tiles that keep no information except for their coordinates in the father macro-tile; these tiles work as building blocks of the hierarchical structure;

(b) the "communication wires" that transmit the bits of macro-colors from the border line of the macro-tile to the computation zone;

(c) the tiles of the computation zone (intended to simulate the space-time diagram of the Universal Turing machine).

Each pattern that includes only "skeleton" tiles (or "skeleton" macro-tiles of some rank k) reappears infinitely often in all homologous position inside all macro-tiles of higher rank. Unfortunately, this property is not true for the patterns that involve the "communication zone" or the "communication wires". Thus, the general construction of a fixed-point tiling does not imply the property of quasiperiodicity. To overcome this difficulty we need some new technical tricks.

We can enforce the following additional properties (p1) - (p4) of a tiling with only a minor modification of the construction:

(p1) In each macro-tile, the size of the computation zone m_k is much less than the size of the macro-tile N. In what follows we need to reserve free space in a macro-tile to insert $O(1)$ (some constant number) of copies of each 2×2 pattern from the computation zone (of this macro-tile), right above the computation zone. This requirement is easy to meet. We assume that the size of a computation zone in a k-level macro-tile of size $N_k \times N_k$ is only $m_k = \text{poly}(\log N_k)$. So we can reserve an area of size $\Omega(m_k)$ above the computation zone, which is free of "communication wires" or any other functional gadgets (so far this area consisted of only skeleton tiles).

(p2) We require that the tiling inside the computation zone satisfies the property of 2×2-*determinacy*. If we know all the colors on the borderline of a 2×2-pattern inside of the computation zone (i.e., a tuple of 8 colors), then we can uniquely reconstruct the 4 tiles of this pattern. Again, to implement this property we do not need new ideas; this requirement is met if we represent the space-time diagram of a Turing machine in a natural way.

(p3) The communication channels in a macro-tile (the wires that transmit the information from the macro-color on the borderline of this macro-tile to the bottom line of its computation zone) must be isolated from each other. The distance between every two wires must be greater than 2. That is, each 2×2-pattern can touch at most one communication wire. Since the width of the wires in a k-level macro-tile is only $O(\log N_{k+1})$, we have enough free space to lay the "communication cables", so this requirement is easy to satisfy.

▶ Remark. Property (p3) is a new feature, it was not used in [7] or any other preceding constructions of self-simulating tilings.

(p4) In our construction the macro-colors of a k-level macro-tile are encoded by bit strings of some length $r_k = O(\log N_{k+1})$. We assumed that this encoding is natural in some way. So far the choice of encoding was of small importance; we only required that some natural manipulations with macro-colors can be implemented in polynomial time. Now, we add another (seemingly artificial) requirement: that each of r_k bits encoding the macro-colors (on the top, bottom, left and right sides of a macro-color) was equal to 0 and to 1 for quite a lot of macro-tiles (so the fact that some bit of some macro-color has this or that value, must not be unique in a tiling). Technically, we require an even stronger property: at every position $s = 1, \dots, r_k$ and for every $i = 0, \dots, N_{k+1} - 1$ there must exist j_0, j_1 such that the s-th bit in the top, the left and the right macro-colors of the k-level macro-tile at the positions (i, j_0) and (i, j_1) in the $(k+1)$-level father macro-tile is equal to 0 and 1 respectively.

There are many (more or less artificial) ways to realize this constraint. For example, we may subdivide the array of r_k bits in three equal zones of size $r_k/3$ and require that for each macro-tile only one of these three zones contains the "meaningful" bits, and two other zones contain only zeros and ones respectively; we require then that the "roles" of these three zones cyclically exchange as we go upwards along a column of macro-tiles.

5.3 Enforcing quasiperiodicity

To achieve the property of quasiperiodicity, we should guarantee that every finite pattern that appears once in a tiling, must appear in each large enough square. If a tileset τ is self-similar, then in every τ-tiling each finite pattern can be covered by at most 4 macro-tiles (by a 2×2-pattern) of an appropriate rank. Thus, it is enough to show that every 2×2-block of macro-tiles of any rank k that appears in at least one τ-tiling, actually appears in this tiling in every large enough square.

Case 1: skeleton tiles. For a 2×2-block of four "skeleton" macro-tiles of level k this is easy. Indeed, we have exactly the same blocks with every vertical shift multiple of L_{k+1} (we have there a similar block of k-level "skeleton" macro-tiles within another macro-tile of rank $(k+1)$). A vertical shift does not change the embedded letters in the zone of responsibility, so we can apply Lemma 6.

To find a similar block of k-level "skeleton" macro-tiles with a different abscissa coordinate, we need a horizontal shift Q which is divisible by L_{k+1} (to preserve the position in the father macro-tile) and at the same time does not change the letters embedded in the zone of responsibility. This is possible due to Lemma 4, if the embedded sequence is quasiperiodic. Given a suitable horizontal shift, we can again apply Lemma 6.

Case 2: communication wires. Let us consider the case when a 2×2-block of k-level macro-tiles involves a part of a communication wire. Due to the property (p3) we may assume that only one wire is involved. The bit transmitted by this wire is either 0 or 1; in both cases, due to the property (p4) we can find another similar 2×2-block of k-level macro-tiles (at the same position within the father macro-tile of rank $(k+1)$ and with the same bit included in the communication wire) in every macro-tile of level $(k+2)$. In this case we need a vertical shift longer than in Case 1: we can find a duplicate of the given block with a vertical shift of size $O(L_{k+2})$.

As in Case 1, any vertical shift does not change the letters embedded in the zone of responsibility of the involved macro-tiles, and we can apply Lemma 6 immediately. If we are looking for a horizontal shift, we again use quasiperiodicity of the simulated shift and apply Lemma 4: there exists a horizontal shift that is divisible by L_{k+2} and does not change the letters embedded in the zone of responsibility. Then we again apply Lemma 6.

Case 3: computation zone. Now we consider the most difficult case: when a 2×2-block of k-level macro-tiles touches the computation zone. In this case we cannot obtain the property of quasiperiodicity for free, and we have to make one more (the last one) modification of our general construction of a self-simulating tiling.

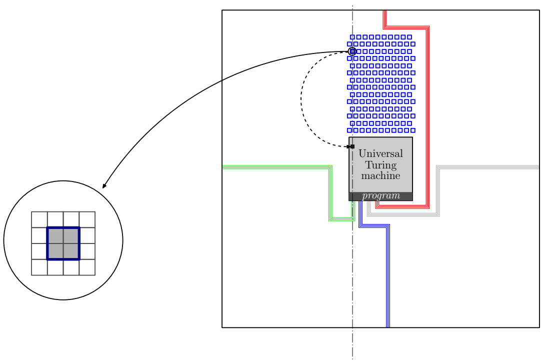

Figure 3 Positions of the slots for patterns from the computation zone.

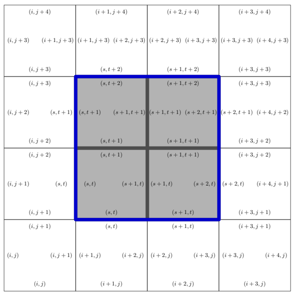

Figure 4 A slot for a 2×2-pattern from the computation zone.

Notice that for each 2×2-window that touches the computation zone of a macro-tile there exist only $O(1)$ ways to tile them correctly. For each possible position of a 2×2-window in the computation zone and for each possible filling of this window by tiles, we reserve a special 2×2-*slot* in a macro-tile, which is essentially a block of size 2×2 in the "free" zone of a macro-tile. It must be placed far away from the computation zone and from all communication wires, but in the same vertical stripe as the "original" position of this block, see Fig. 3. We have enough free space to place all necessary slots due to the property (p1). We define the neighbors around this slot in such a way that only one specific 2×2 pattern can patch it (here we use the property (p2)).

In our construction the tiles around this slot "know" their real coordinates in the bigger macro-tile, while the tiles inside the slot do not (they "believe" they are tiles in the computation zone, while in fact they belong to an artificial isolated *diversity preserving* "slot" far outside of any real computation), see Fig. 3 and Fig. 4. The frame of the slot consists of 12 "skeleton" tiles (the white squares in Fig. 4), they form a slot a 2×2-pattern from the computation zone (the grey squares in Fig. 4). In the picture we show the "coordinates" encoded in the colors on the sides of each tile. Notice that the colors of the bold lines (the blue lines between white and grey tiles and the bold black lines between grey tiles) should contain some information beyond coordinates – these colors involve the bits used to simulate a space-time diagram of the universal Turing machine. In this picture, the "real" coordinates of the bottom-left corner of this slot are $(i+1, j+1)$, while the "natural" coordinates of the pattern (when it appears in the computation zone) are (s, t).

We choose the positions of the "slots" in the macro-tile so that coordinates can be computed with a short program in time polynomial in $\log N$. We require that all slots are isolated from each other in space, so they do not damage the general structure of "skeleton" tiles building the macro-tiles.

Through construction, each of these slots is aligned with the "natural" position of the corresponding 2×2-block in the computation zone. This guarantees that the tiles in the computation zone and their "sibling" in the artificial slots share the same bits of the embedded sequences in the corresponding zone of responsibility. We have defined the slots so that the "conscious memory" of the tiles in the computation zone and in the corresponding slots is the same. Thus, we can apply Lemma 6 and conclude that a 2×2-blocks in diversity preserving slots are exactly equal to the corresponding 2×2-patterns in the computation zone. For a horizontal shift, similarly to the Cases 1–2 above, we use quasiperiodicity of the embedded sequences and apply Lemma 4.

▶ Remark (Concluding Remark). Formally speaking, we proved Lemma 6 before we introduced the last upgrades of our tileset. However, it is easy to verify that the updates of the main construction discussed in this Section do not affect the proof of that lemma.

Thus, we constructed a tileset τ such that every $L_k \times L_k$ pattern that appears in a τ-tiling must also appear in every large enough square in this tiling. So, the constructed tileset satisfies the requirements of Theorem 1.

The proof of Corollary 3. To prove Corollary 3 we only need to combine Theorem 1 with a fact from [14]: there exists a 1D shift $\mathcal{S}$ that is quasiperiodic, and for every configuration $\mathbf{x} \in \mathcal{S}$ the Kolmogorov complexity of all factors is linear, i.e., $K(x_i x_{i+1} \dots x_{i+n}) = \Omega(n)$ for all i. ◀

The proof of Theorem 2. First of all we notice that the proof of Theorem 1 discussed above does not imply Theorem 2. If we take an effective minimal 1D-shift $\mathcal{A}$ and plug it into the construction form the proof of Theorem 1, we obtain a tileset τ (simulating $\mathcal{A}$) which is *quasiperiodic* but not necessary *minimal.* The property of minimality can be lost even for a *periodic* shift $\mathcal{A}$. Indeed, assume that the minimal period $t > 0$ of the configurations in $\mathcal{A}$ is a factor of the size N_k of k-level macro-tiles in our self-simulating tiling, then we can extract from the resulting SFT τ nontrivial shifts T_i, $i = 0, 1, \dots, t-1$ corresponding to the position of the embedded 1D-configuration with respect to the grid of macro-tiles. To overcome this obstacle we will superimpose some additional constraints on the embedding of the simulated $\mathbb{Z}$-shift in a $\mathbb{Z}^2$-tiling. Roughly speaking, we will enforce only "standard" positioning of the embedded 1D sequences with respect to the grid of macro-tiles. This will not change the class of configurations that can be simulated (we still get all configurations from a given minimal shift $\mathcal{A}$), but the classes of all valid tilings will reduce to some minimal $\mathbb{Z}^2$-SFT.

The standardly aligned grid of macro-tiles: In general, the hierarchical structure of macro-tiles permits non-countably many ways of cutting the plane in macro-tiles of different ranks. We fix one particular version of this hierarchical structure and say that a grid of macro-tiles is *standardly aligned*, if for each level k the point $(0, 0)$ is the bottom-left corner of a k-level macro-tile. This means that the tiling is cut into k-level macro-tiles of size $L_k \times L_k$ by vertical lines with abscissae $x = L_k \cdot t'$ and ordinates $y = L_k \cdot t''$, with $t', t'' \in \mathbb{Z}$ (so the vertical line $(0, *)$ and the horizontal line $(*, 0)$ serve as separating lines for macro-tiles of all ranks). Of course, this structure of macro-tiles is computable.

The canonical representative of a minimal shift: A minimal effectively-closed 1D-shift $\mathcal{A}$ is always computable, i.e., the set of finite patterns that appear in configurations of this shift is computable. It follows immediately that $\mathcal{A}$ contains some computable configuration. Let us fix one computable configuration $\mathbf{x}$; in what follows we call it *canonical.*

The standard embedding of the canonical representative: We superimpose the standardly aligned grid of macro-tiles with the canonical representative of a minimal shift $\mathcal{A}$: we take the direct product of the hierarchical structures of the standardly aligned grid of macro-tiles with the canonical configuration $\mathbf{x}$ from $\mathcal{A}$ (that is, each tile with coordinates (i, j) "contains" the letter x_i from the canonical configuration).

Claim 1: Given a pattern w of size $n \le L_k$ and an integer i, we can algorithmically verify whether the factor w appears in the standard embedding of the canonical representative with the shift $(i \mod L_k)$ relative to the grid of k-level macro-tiles. This follows from Lemma 5(b) applied to the superposition of the canonical representative with the periodical grid of k-level macro-tiles).

▶ Remark. This verification procedure is computable, but its computational complexity can be very high. To perform the necessary computation we may need space and time much bigger than the length of w and L_k.

Upgrade of the main construction: Let us update the construction of self-simulating tiling from the proof of Theorem 1. So far we assumed that every macro-tile (of every level k) verifies that the delegated factor of the embedded sequences contains no factors forbidden for the shift $\mathcal{A}$. Now we make the constraint stronger: we require that the delegated factor contains only factors allowed in the shift $\mathcal{A}$ and placed in the positions (relative to the grid of macro-tiles) permitted for factors in the standard embedding of the canonical representative. This property is computable (Claim 1), so every forbidden pattern or a pattern in a forbidden position will be discovered in a computation in a macro-tile of some rank. The computational complexity of this procedure can be very high (see Remark after Claim 1), and we cannot guarantee that the forbidden patterns of small length are discovered by the computation in macro-tiles of small size. But we do guarantee that each forbidden pattern or a pattern in a forbidden position is discovered by a computation in some macro-tile of *high enough* rank.

Claim 2: The new tileset admits correct tilings of the plane. Indeed, at least one tiling is valid by the construction: the standard embedding of the canonical representative corresponds to a valid tiling of the plane, since macro-tiles of all rank never find any forbidden placement of patterns in the embedded sequence.

Claim 3: The new tileset simulates the shift $\mathcal{A}$. This follows immediately from the construction: the embedded sequence must be a configuration from $\mathcal{A}$.

Claim 4: For the constructed tileset τ the set of all tilings is a minimal shift. We need to show that *every* τ-tiling contains all patterns that can appear in *at least one* τ-tiling. Similarly to the proof of Theorem 1, it is enough to prove this property for 2×2-blocks of k-level macro-tile. The difference with the argument in the previous section is that for every 2×2-block of macro-tiles in one tiling T we must find a similar block of macro-tiles in another tiling T', so that this block has exactly the same position with respect to father macro-tile $\mathcal{M}$ of rank $(k + 1)$, and $\mathcal{M}$ and $\mathcal{M}'$ own exactly the same factor of the embedded sequence in their zones of responsibility. This is always possible due to Lemma 5(a) (applied to the canonical representative of $\mathcal{A}$ superimposed with the periodical grid of $(k + 1)$-level macro-tiles). This observation concludes the proof. ◀

Acknowledgments. We are indebted to Emmanuel Jeandel for raising and motivating the questions which led to this work. We are grateful to Gwenaël Richommes and Pascal Vanier for fruitful discussions and to the anonymous reviewers for truly valuable comments.

References

1 Nathalie Aubrun and Mathieu Sablik. Simulation of effective subshifts by two-dimensional subshifts of finite type. *Acta Applicandae Mathematicae*, 128(1):35–63, 2013.

2 Sergey V. Avgustinovich, Dmitrii G. Fon-Der-Flaass, and Anna E. Frid. Arithmetical complexity of infinite words. In *3rd Int. Colloq. on Words, Languages and Combinatorics*, pages 51–62, 2003.

3 Alexis Ballier and Emmanuel Jeandel. Computing (or not) quasiperiodicity functions of tilings. In *2nd Symposium on Cellular Automata* Journées Automates Cellulaires *(JAC 2010)*, pages 54–64, 2010.

4 Bruno Durand. Tilings and quasiperiodicity. *Theoretical Computer Science*, 221(1):61–75, 1999.

5 Bruno Durand, Leonid Levin, and Alexander Shen. Complex tilings. *The Journal of Symbolic Logic*, 73(2):593–613, 2008.

6 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.

7 Brunourand Durand and Andrei Romashchenko. Quasiperiodicity and non-computability in tilings. In *Proc. International Symposium on Mathematical Foundations of Computer Science (MFCS 2015)*, pages 218–230, 2015.

8 Peter Gács. Reliable computation with cellular automata. *Journal of Computer and System Sciences*, 32(1):15–78, 1986.

9 Gustav Hedlund and Marston Morse. Symbolic dynamics. *American Journal of Mathematics*, 60(4):815–866, 1938.

10 Michael Hochman. On the dynamics and recursive properties of multidimensional symbolic systems. *Inventiones mathematicae*, 176(1):131–167, 2009.

11 Michael Hochman and Pascal Vanier. A note on turing degree spectra of minimal shifts. In *The 12th International Computer Science Symposium in Russia*, pages 154–161, 2017.

12 Emmanuel Jeandel. Personal communication. private communication, 2015.

13 Emmanuel Jeandel and Pascal Vanier. Turing degrees of multidimensional sfts. *Theoretical Computer Science*, 505:81–92, 2013.

14 Andrey Rumyantsev and Maxim Ushakov. Forbidden substrings, kolmogorov complexity and almost periodic sequences. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 396–407, 2006.

15 Pavel V. Salimov. On uniform recurrence of a direct product. *Discrete Mathematics and Theoretical Computer Science*, 12(4), 2010.

16 Linda Brown Westrick. Seas of squares with sizes from a Π^0_1 set. *arXiv preprint arXiv:1609.07411*, 2016.