Fixed-point tile sets and their applications

Bruno Durand, Andrei Romashchenko, Alexander Shen

Fixed-point tile sets and their applications

Bruno Durand[*], Andrei Romashchenko[†], Alexander Shen[‡]

September 17, 2010

Abstract

An aperiodic tile set was first constructed by R. Berger while proving the undecidability of the domino problem. It turned out that aperiodic tile sets appear in many topics ranging from logic (the Entscheidungsproblem) to physics (quasicrystals).

We present a new construction of an aperiodic tile set that is based on Kleene's fixed-point construction instead of geometric arguments. This construction is similar to J. von Neumann self-reproducing automata; similar ideas were also used by P. Gács in the context of error-correcting computations.

This construction it rather flexible, so it can be used in many ways: we show how it can be used to implement substitution rules, to construct strongly aperiodic tile sets (any tiling is far from any periodic tiling), to give a new proof for the undecidability of the domino problem and related results, characterize effectively closed 1D subshift it terms of 2D shifts of finite type (improvement of a result by M. Hochman), to construct a tile set which has only complex tilings, and to construct a "robust" aperiodic tile set that does not have periodic (or close to periodic) tilings even if we allow some (sparse enough) tiling errors. For the latter we develop a hierarchical classification of points in random sets into islands of different ranks.

Finally, we combine and modify our tools to prove our main result: there exists a tile set such that all tilings have high Kolmogorov complexity even if (sparse enough) tiling errors are allowed.

Some of these results were included in the DLT extended abstract [10] and in the ICALP extended abstract [11].

[*]Laboratoire d'Informatique Fondamentale de Marseille, CNRS & Univ. Aix–Marseille. Supported in part by ANR Sycomore and NAFIT ANR-08-EMER-008-01grants.

[†]Laboratoire d'Informatique Fondamentale de Marseille, CNRS & Univ. Aix–Marseille. On leave from IITP RAS, Moscow. Supported in part by ANR Sycomore, NAFIT ANR-08-EMER-008-01, and RFBR 09-01-00709-a grants.

[‡]Laboratoire d'Informatique Fondamentale de Marseille, CNRS & Univ. Aix–Marseille. On leave from IITP RAS, Moscow. Supported in part by ANR Sycomore, NAFIT ANR-08-EMER-008-01, and RFBR 09-01-00709-a grants.

Contents

1 Introduction

In this paper, *tiles* are unit squares with colored sides. Tiles are considered as prototypes: we may place translated copies of the same tile into different cells of a cell paper (rotations are not allowed). Tiles in the neighbor cells should match (common side should have the same color in both).

Formally speaking, we consider a finite set C of *colors*. A *tile* is a quadruple of colors (left, right, top and bottom ones), i.e., an element of C^4. A *tile set* is a subset $\tau \subset C^4$. A *tiling* of the plane with tiles from τ (τ-*tiling*) is a mapping $U : \mathbb{Z}^2 \to \tau$ that respects the color matching condition. A tiling U is *periodic* if it has a *period*, i.e., a non-zero vector $T \in \mathbb{Z}^2$ such that $U(x+T) = U(x)$ for all $x \in \mathbb{Z}^2$. Otherwise the tiling is *aperiodic*. The following classical result was proved by in [3], where this construction was used as a main tool to prove *Berger's theorem*: the *domino problem* (to find out whether a given tile set has tilings or not) is undecidable.

Theorem 1. *There exists a tile set τ such that τ-tilings exist and all of them are aperiodic.* [3]

The first tile set of Berger was rather complicated. Later many other constructions were suggested. Some of them are simplified versions of the Berger's construction ([29], see also the expositions in [1, 8, 22]). Some others are based on polygonal tilings (including famous Penrose and Ammann tilings, see [15]). An ingenious construction suggested in [19] is based on the multiplication in a kind of positional number system and gives a small aperiodic set of 14 tiles (in [6] an improved version with 13 tiles is presented). Another nice construction with a short and simple proof (based explicitly on ideas of self-similarity) was recently proposed by N. Ollinger [27].

In this paper we present yet another construction of aperiodic tile set. It does not provide a small tile set; however, we find it interesting because:

- The existence of an aperiodic tile set becomes a simple application of a classical construction used in Kleene's fixed point (recursion) theorem, in von Neumann's self-reproducing automata [26] and, more recently, in Gács' reliable cellular automata [12, 13]; we do not use any geometric tricks. The construction of an aperiodic tile set is not only an interesting result but an important tool (recall that it was invented to prove that domino problem is undecidable); our construction makes this tool easier to use.

- The construction is rather general, so it is flexible enough to achieve some additional properties of the tile set. We illustrate this flexibility providing new proof for several known results and proving new results; these new results add robustness (resistance to sparse enough errors) to known results about aperiodic tile sets and tile sets that have only complex tilings.

It is not clear whether this kind of robustness can be achieved for previously known constructions of tile sets; on the other hand, robustness properties look important. For example, a mathematical model for processes like quasicrystals' growth or DNA-computation should take errors into account. Note that our model (independent choice of places where errors are allowed) has no direct physical meaning; it is just a simple mathematical model that can be used as a playground to develop tools for estimating the consequences of tiling errors.

The paper is organized as follows. In Section 2 we present the fixed-point construction of an aperiodic tile set (new proof of Berger's theorem). Then we illustrate the flexibility of this construction by several examples:

- In Section 3 we show that any 'uniform' substitution rule can be implemented by a tile set (thus providing a new proof for this rather old result). Then in Section 4 we use substitutions to show that there are strongly aperiodic tile sets (this means that any tiling is strongly aperiodic, i.e., any shift changes at least some fixed fraction of tiles).

- Fixed-point construction of Section 2 provides a self-similar tiling: blocks of size $n \times n$ ("macro-tiles") behave exactly as individual tiles, so on the next level we have $n^2 \times n^2$ blocks made of $n \times n$ macro-tiles that have the same behavior, etc. In Section 5 we make some changes in our construction that allow us to get variable zoom factors (the numbers of tiles in macro-tiles increase as the level increases).

  Variable zoom factor tilings can be used for simulating computations (higher levels perform more computation steps); we use them to give a simple proof of the undecidability of the domino problem (main technical difficulty in the standard proof was to synchronize computations on different levels, now this is not needed at all); we show also that other undecidability results can be obtained in this way.

- This technique can be used to push the strong aperiodicity to its limits: the distance between any tiling and any periodic one (or between any tiling and its nontrivial shift) can be made arbitrarily close to 1, not only separated from 0. This is done in Section 6 using an additional tool: error-correcting codes.

- In [7] a tile set was constructed such that every tiling has maximal Kolmogorov complexity of fragments ($\Omega(n)$ for $n \times n$ squares); all tilings for this tile set are non-computable (so it implies a classical result of Hanf [17] and Myers [25] as a corollary). The construction in [7] was rather complicated and was based on a classical construction of an aperiodic tile set. In Section 7 we provide another proof of the same result that uses variable zoom factors. It is simpler in some respects and can be generalized to produce robust tile sets with complex tiling, which is our main result (Section 13).

  Further in Section 8 we use the same technique to give a new proof for some results by S. Simpson [32] and M. Hochman [18] about effectively closed subshifts: every 1-dimensional effectively closed subshift can be obtained as a projection of configurations of some 2-dimensional subshift of finite type (in an extended alphabet). Our construction provides a solution of Problem 9.1 from [18]. (Another solution, based on the classical Robinson-type construction, was independently suggested by N. Aubrun and M. Sablik, see [2].)

- To prove the robustness of tile sets against sparse errors we use a hierarchical classification of the elements of random sets into islands of different levels (a method that goes back to Gács [13, 14]). This method is described in Section 9.1.

- In Section 9.2 we give definitions and establish some probabilistic results about islands that are used to prove robustness: we show that a sparse random set on $\mathbb{Z}^2$ with probability 1 (for Bernoulli distribution) can be represented as a union of 'islands' of different ranks. The higher is the rank, the bigger is the size of an island; the islands are well isolated from each other (in some neighborhood of an island of rank k there is no other islands of rank $\geq k$). Then in Section 9.3 we illustrate these tools using standard results of percolation theory as a model example. In Section 9.4 we modify the definition of an island allowing two (but not three!) islands of the same rank to be close to each other. This more complicated definition is necessary to obtain the most technically involved result of the paper in Section 13 but can be skipped if the reader is interested in the other results.

- In Section 10 we use fixed-point construction to get an aperiodic tile set that is robust in the following sense: if a tiling has a "hole" of size n, then this hole can be patched by changing only $O(n)$-size zone around it. Moreover, an $O(n)$ zone (with bigger constant in O-notation) around the hole is enough for this (we don't need to have the entire plane covered). In Section 11 we explain how to get a robust aperiodic tile sets with variable zoom factors. Again, this material is used in Section 13 only.

- In Section 12 we combine the developed techniques to establish one of our main results: there exists a tile set such that every tiling of a plane except a sparse set of random points is far from every periodic tiling.

- Finally, the Section 13 contains our most technically difficult result: a robust tile set such that all tilings, even with sparsely placed errors, have linear complexity of fragments. To this end we need to combine all our techniques: fixed-point construction with variable zoom factors, splitting of a random set into doubled islands, and "robustification" with filling of doubled holes.

2 Fixed-point aperiodic tile set

2.1 Macro-tiles and simulation

Fix a tile set τ and an integer $N > 1$ (*zoom factor*). A *macro-tile* is an $N \times N$ square tiled by τ-tiles matching each other (i.e., a square block of N^2 tiles with no color conflicts inside). We can consider macro-tiles as "pre-assembled" blocks of tiles: instead of tiling the plane with individual tiles, we may use macro-tiles. To get a correct τ-tiling in this way, we need only to ensure that neighbor macro-tiles have matching *macro-colors*, so there are no color mismatches on the borders between macro-tiles. More formally, by macro-color we mean a sequence of N colors on the side of a macro-tile. Each macro-tile has four macro-colors (one for each side). We always assume that macro-tiles are placed side-to-side, so the plane is split into $N \times N$-squares by vertical and horizontal lines.

In the sequel we are interested in the situation when τ-tilings can be split uniquely into macro-tiles that behave like tiles from some other tile set ρ. Formally, let us define the notion of a

simulation.

Let τ and ρ be two tile sets, and let $N > 1$ be an integer. By *simulation of ρ by τ with zoom factor N* we mean a mapping S of ρ-tiles into $N \times N$ τ-macro-tiles such that:

- S is injective (different tiles are mapped into different macro-tiles).
- Two tiles r_1 and r_2 match if and only if their images $S(r_1)$ and $S(r_2)$ match. This means that the right color of r_1 equals the left color of r_2 if and only if the right macro-color of $S(r_1)$ equals the left macro-color of $S(r_2)$, and the same is true in the vertical direction.
- Every τ-tiling can be split by vertical and horizontal lines into $N \times N$ macro-tiles that belong to the range of S, and such a splitting in unique.

The second condition guarantees that every ρ-tiling can be transformed into a τ-tiling by replacing each tile $r \in \rho$ by its image, macro-tile $S(r)$. Taking into account other conditions, we conclude that every τ-tiling can be obtained in this way, and the positions of grid lines as well as the corresponding ρ-tiles can be reconstructed uniquely.

Example 1 (negative). Assume that τ consists of one tile with four white sides. Fix some $N > 1$. There exists a single macro-tile of size $N \times N$. Does it mean that τ simulates itself (when its only tile is mapped to the only macro-tile)? No: the first and second conditions are true, but the third one is false: the placement of cutting lines is not unique.

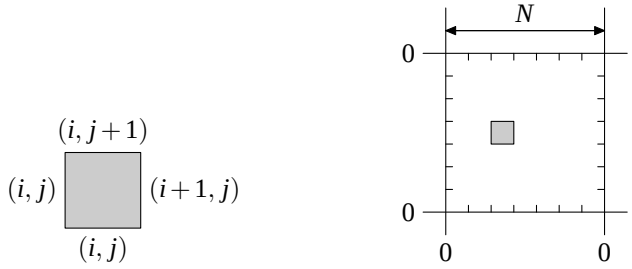

Figure 1: Tiles and macro-tiles for Example 2

Example 2 (positive). In this example ρ consists of one tile with all white sides. The tile set τ consists of N^2 tiles indexed by pairs (i, j) of integers modulo N. A tile from τ has colors on its sides as shown on Fig. 1. The simulation maps ρ-tile to a macro-tile that has colors $(0,0), \ldots, (0, N-1)$ and $(0,0), \ldots, (N-1, 0)$ on its vertical and horizontal borders respectively (see Fig. 1).

Definition. *A* self-similar *tile set is a tile set that simulates itself.*

The idea of self-similarity is used (more or less explicitly) in most constructions of aperiodic tile sets ([19, 6] are exceptions). However not all of these constructions provide literally self-similar tile sets in our sense.

It is easy to see that self-similarity guarantees aperiodicity.

Proposition 1. *A self-similar tile set τ may have only aperiodic tilings.*

Proof. Let S be a simulation of τ by itself with zoom factor N. By definition, every τ-tiling U can be uniquely cut into $N \times N$-macro-tiles from the range of S. So every period T of U is a multiple of N (since the T-shift of a cut is also a cut, the shift should respect borders between macro-tiles). Replacing each macro-tile by its S-preimage, we get a τ-tiling that has period T/N. Therefore, T/N is again a multiple of N. Iterating this argument, we conclude that T is divisible by N^k for every k, so $T = 0$. $\square$

Note also that every self-similar tile set has arbitrarily large finite tilings: starting with some tile, we apply S iteratively and get a big tiled square. The standard compactness argument guarantees the existence of a tiling of the entire plane. So to prove the existence of aperiodic tile sets it is enough to construct a self-similar tile set.

Theorem 2. *There exists a self-similar tile set τ.*

Theorem 2 was explicitly formulated and proven by N. Ollinger [27]; in his proof a self-similar tile set is constructed explicitly (it consists of 104 tiles). This tile set is then used to implement substitution rules (cf. Theorem 3 below). Another example of a self-similar tile set (with many more tiles) is given in [8]. (Note that the definition of self-similarity used in [8] is a bit stronger.)

We prefer a less specific and more flexible argument, which is based on the fixed-point idea. Our proof works for a vast class of tile sets (though we cannot provide explicitly an aperiodic tile set of a reasonably small size). The rest of this section is devoted to our proof of Theorem 2. Before we prove this result, we explain some technique used in our construction and show how to simulate a given tile set by embedding computations.

2.2 Simulating a tile set

Let us start with some informal discussion. Assume that we have a tile set ρ whose colors are k-bit strings ($C = \{0,1\}^k$) and the set of tiles $\rho \subset C^4$ is presented as a predicate $R(c_1, c_2, c_3, c_4)$ with four k-bit arguments. Assume that we have some Turing machine $\mathscr{R}$ that computes R. Let us show how to simulate ρ using some other tile set τ.

This construction extends Example 2, but simulates a tile set ρ that contains not a single tile but many tiles. We keep the coordinate system modulo N embedded into tiles of τ; these coordinates guarantee that all τ-tilings can be uniquely cut into blocks of size $N \times N$ and every tile "knows" its position in the block (as in Example 2). In addition to the coordinate system, now each tile in τ carries supplementary colors (from a finite set specified below) on its sides. These colors form a new "layer" superimposed with the old one, i.e., the set of colors is now a Cartesian product of the old one and the set of colors used in this layer.

On the border of a macro-tile (i.e., when one of the coordinates is zero) only two supplementary colors (say, 0 and 1) are allowed. So the macro-color encodes a string of N bits (where N is the size of macro-tiles). We assume that N is much bigger than k and let k bits in the middle of macro-tile sides represent colors from C. All other bits on the sides are zeros (this is a restriction on tiles: each tile "knows" its coordinates so it also knows whether non-zero supplementary colors are allowed).

Now we need additional restrictions on tiles in τ that guarantee that macro-colors on the sides of each macro-tile satisfy relation R. To achieve this, we ensure that bits from the macro-tile sides

are transferred to the central part of the tile where the checking computation of $\mathscr{R}$ is simulated (Fig. 2).

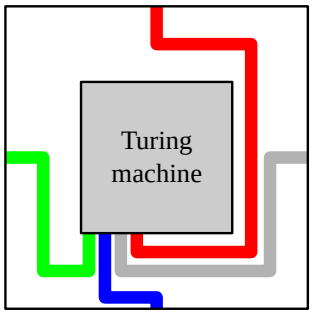

Figure 2: Wires and processing zones; wires appear quite narrow since $N \gg k$

For that we need to fix which tiles in a macro-tile form "wires" (this can be done in any reasonable way; let us assume that wires do not cross each other) and then require that each of these tiles carries equal bits on two sides (so some bit propagates along the entire wire); again it is easy to arrange since each tile knows its coordinates.

Then we check R by a local rule that guarantees that the central part of a macro-tile represents a time-space diagram of $\mathscr{R}$'s computation (the tape is horizontal, time goes up). This is done in a standard way: the time-space diagram (tableau) of a Turing machine computation can be described by local rules, and these rules can be embedded into a tile set (see, e.g., [1, 15]). We require that computation terminates in an accepting state: if not, the tiling cannot be formed.

To make this construction work, the size of macro-tile (N) should be large enough: we need enough space for k bits to propagate and enough time and space (=height and width) for all accepting computations of $\mathscr{R}$ to terminate.

In this construction the number of supplementary colors depends on the machine $\mathscr{R}$ (the more states it has, the more colors are needed in the computation zone). To avoid this dependency, we replace $\mathscr{R}$ by a fixed universal Turing machine $\mathscr{U}$ that runs a *program* simulating $\mathscr{R}$. Let us agree that the tape of the universal Turing machine has an additional read-only layer. Each cell carries a bit that is not changed during the computation; these bits are used as a program for the universal machine U (Fig. 3). In terms of our simulation, the columns of the computation zone carry unchanged bits (considered as a program for U), and the tile set restrictions guarantee that the central zone represents the *record* (time-space diagram) of an accepting computation of U (with this program). In this way we get a tile set τ that simulates ρ with zoom factor N using $O(N^2)$ tiles. (Again we need N to be large enough, but the constant in $O(N^2)$ does not depend on N.)

2.3 Simulating itself

We know how to simulate a given tile set ρ (represented as a program for the universal TM) by another tile set τ with a large enough zoom factor N. Now we want τ to be identical to ρ (then Proposition 1 guarantees aperiodicity). For this we use a construction that follows the proof of Kleene's recursion (fixed-point) theorem.

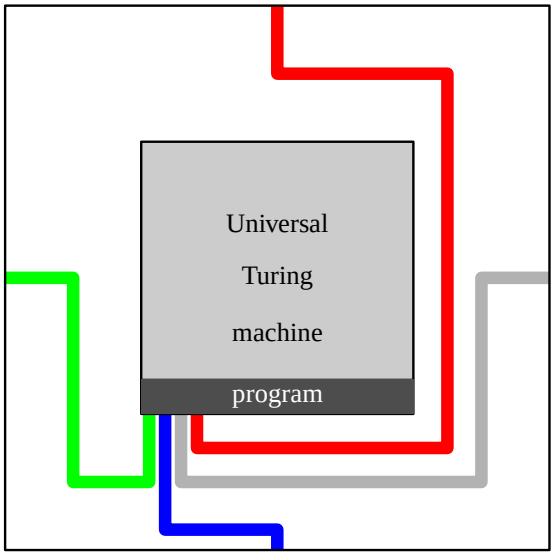

Figure 3: Checking tiles with a universal TM

Warning: we cannot refer here to the *statement* of the theorem; we need to recall its proof and adapt it to our framework. Kleene's theorem [20] says that for every computable transformation π of programs one can find a program p such that p and $\pi(p)$ are equivalent, i.e., produce the same output (for simplicity we consider programs with no input, but this restriction does not really matter). In other terms, there is no guaranteed way to transform a given program p into some other program $\pi(p)$ that produces different output. Proof sketch: first we note that the statement is language-independent since we may use translations in both directions before and after π. So without loss of generality we may assume that the programming language has some special properties. First, we assume that it has a function `GetText()` that returns the text of the program (or a pointer to a memory address where the program text is kept). Second, we assume that the language contains an interpreter function `Execute(string s)` that interprets the content of its string argument *s* as a program written in the same language. It is not difficult to develop such a language and write an interpreter for it. Indeed, the interpreter can access the program text anyway, so it can copy the text into some string variable. The interpreted also can recursively call itself with another program as an argument when it sees the `Execute` call. If our language has these properties, it is easy to construct the fixed point for π: just take the program `Execute(`π`(GetText()))`.

This theorem shows that a kind of "vicious circle" when we write the program as if its full text is already given to us, is still acceptable. A classical example is a program that prints its own text. The proof shows a way how to do this by using a computation model where the immutable text of the program is accessible to it.

Constructing a self-similar tiling, we have the same kind of problems. We have already seen how to construct a tile set τ that simulates a given tile set ρ. [Counterpart: it is easy to write a program that prints any given text.] What we need is to construct a tile set that simulates *itself*. [Counterpart: what we need is to write a program that prints *its own* text.]

Let us look again at our construction that transforms the description of ρ (a Turing machine that computes the corresponding predicate) into a tile set τ that simulates ρ. Note that most rules of τ do not depend on the program for $\mathscr{R}$, dealing with information transfer along the wires, the vertical propagation of unchanged program bits, and the space-time diagram for the universal TM in the computation zone. Making these rules a part of ρ's definition (we let $k = 2\log N + O(1)$ and encode $O(N^2)$ colors by $2\log N + O(1)$ bits), we get a program that checks that macro-tiles behave like τ-tiles in this respect. Macro-tiles of the second level ("macro-macro-tiles") made of them would have correct structure, wires that transmit bits to the computation zone and even the record of some computation in this zone, but this computation could be arbitrary. So at the third level all the structure is lost.

What do we need to add to our construction to close the circle and get self-simulation? The only remaining part of the rules for τ (not implemented yet at the level of macro-tiles) is the hardwired program. We need to ensure that macro-tiles carry the same program as τ-tiles do. For that our program (for the universal TM) needs to access the bits of its own text. As we have discussed, this self-referential action is in fact quite legal: the program is written on the tape, and the machine can read it. The program checks that if a macro-tile belongs to the first line of the computation zone, this macro-tile carries the correct bit of the program.

How should we choose N (hardwired in the program)? We need it to be large enough so the computation described above (it deals with $O(\log N)$ bits) can fit in the computation zone. Note that the computation never deals with the list of tiles in τ, or a truth table of the corresponding 4-ary relation on bit strings; all these objects are represented by programs that describe them. The computation needs to check simple things only: that numbers in the $0 \ldots N-1$ range on four sides are consistent with each other, that rules for wires and computation time-space diagram are observed, that program bits on the next level coincide with actual program bits, etc. All these computations are rather simple. They are polynomial in the input size, i.e., in $O(\log N)$), so for large N they easily fit in $O(N)$ available time and space.

This finishes the construction of a self-similar aperiodic tile set.

Remark. Let us also make a remark that would be useful later. We defined tile set as a subset of C^4 where C is a set of colors. Using this definition, we do not allow different tiles to have the same colors on their sides: the only information carried by the tile is kept on its sides. However, sometimes a more general definition is preferable. We can define a tile set as a finite set T together with a mapping of T into C^4. Elements of T are tiles, and the mapping tells for each tile which colors it has on its four sides.

One can easily extend the notions of macro-tiles and simulation to this case. In fact, macro-tiles are well suited to this definition since they already may carry some information that is not reflected in the side macro-colors. The construction of self-similar tile set also can be adapted. For example, we can construct a self-similar tile sets where each tile carries an auxiliary bit, i.e., exists in two copies having the same side colors. Since the tile set is self-similar, every macro-tile at every level of the hierarchy also carries one auxiliary bit, and the bits at different levels and in different macro-tiles are not related to each other. Note that the total density of information contained in a tiling is still finite, since the density of information contained in auxiliary bits assigned to high level macro-tiles decreases with level as a geometric sequence.

3 Substitution rules implemented

The construction of a self-similar tiling is rather flexible and can be easily augmented to get a self-similar tiling with additional properties. Our first illustration is the simulation of substitution rules.

Let A be some finite alphabet and $m > 1$ be an integer. A *substitution rule* is a mapping $s\colon A \to A^{m\times m}$. This mapping can be naturally extended to A-configurations. By *A-configuration* we mean an integer lattice filled with A-letters, i.e., a mapping $\mathbb{Z}^2 \to A$ considered modulo translations. A substitution rule s applied to a configuration X produces another configuration $s(X)$ where each letter $a \in A$ is replaced by an $m \times m$ matrix $s(a)$.

We say that a configuration X is *compatible* with substitution rule s if there exists an infinite sequence

$$\ldots \xrightarrow{s} X_3 \xrightarrow{s} X_2 \xrightarrow{s} X_1 \xrightarrow{s} X,$$

where X_i are some configurations. This definition was proposed in [27]. The classical definition (used, in particular, in [24]) is slightly different: configuration $X : \mathbb{Z}^2 \to A$ is said compatible with a substitution rule s if every finite part of X occurs inside of some $s^{(n)}(a)$ (for some $n \in \mathbb{N}$ and some $a \in A$). We prefer the first approach since it looks more natural in the context of tilings. However, all our results can be reformulated and proven (with some technical efforts) for the other version of the definition; we do not go into details here.

Example 3. Let $A = \{0,1\}$,

$$s(0) = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right), \quad s(1) = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right).$$

It is easy to see that the only configuration compatible with s is the chess-board coloring where zeros and ones alternate horizontally and vertically.

Example 4 (Fig. 4). Let $A = \{0,1\}$,

$$s(0) = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right), \quad s(1) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right).$$

One can check that all configurations that are compatible with this substitution rule (called *Thue – Morse configurations* in the sequel) are aperiodic. We will prove in Section 4 a stronger version of this fact, but it is not difficult anyway; one may note, for example, that every configuration compatible with this substitution rule can be uniquely decomposed into disjoint 2×2 blocks $\left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ by vertical and horizontal lines; since neighbor cells of the same color should be separated by one of those lines, the position of lines is unique. Then we can apply the argument from Proposition 1 (with $N = 2$).

The following theorem goes back to Mozes [24]. It says that every substitution rule can be enforced by a tile set.

Theorem 3. *Let A be an alphabet and let s be a substitution rule over A. Then there exists a tile set τ and a mapping $e\colon \tau \to A$ such that*

(a) *e-image of any τ-tiling is an A-configuration compatible with s;*

(b) *every A-configuration compatible with s can be obtained in this way.*

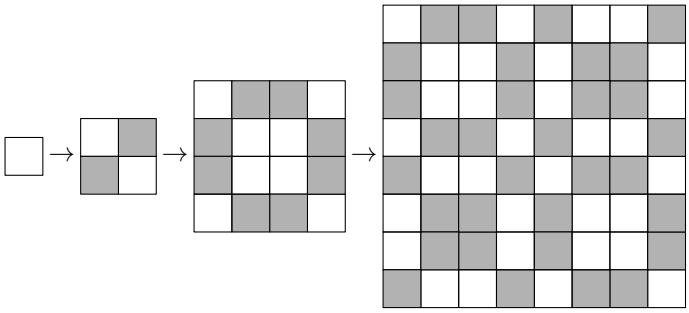

Figure 4: Three steps of Thue–Morse substitution

A nice proof of this result for 2×2-substitutions is given in [27], where an explicit construction of a tile set τ for every substitution rule s is provided. We prove this theorem using our fixed-point argument, avoiding explicit construction of τ (the tile sets that can be extracted from our proof contain a huge number of tiles).

Proof. Let us modify the construction of the tile set τ (with zoom factor N) taking s into account. First consider a very special case when

- the substitution rule maps each A-letter into an $N \times N$-matrix (i.e., $m = N$).
- the substitution rule is easy to compute: given a letter $u \in A$ and (i, j), we can compute the (i, j)-th letter of $s(u)$ in time which is much less than N.

In this case we proceed as follows. In our basic construction every tile knows its coordinates in the macro-tile and some additional information needed to arrange "wires" and simulate calculations of the universal TM.[1] Now in addition to this basic structure each tile keeps two letters of A: the first is the label of a tile itself, and the second is the label of the $N \times N$-tile it belongs to. This means that we keep additional $2\log|A|$ bits in each tile, i.e., multiply the number of tiles by $|A|^2$ (the restrictions will reduce the size of the tile set, see the discussion below). It remains to explain how the local rules work. We add two requirements:

(i) *The second letter is the same for neighbor tiles* (*unless they are separated by a border of some $N \times N$ macro-tile*). This constraint can be easily enforced by colors on sides of tiles. We multiply the number of colors in our basic construction by $|A|$; now each color of the new construction is a pair: its 1st component is a color from the basic construction and its 2nd component is a letter of A. The second component of the new color guarantees that every two neighbor tiles keep the same "father" letter (unless these tiles are separated by a border and do not belong to the same father macro-tile; we do not exhibit the letter to those borders).

[1] We use this "anthropomorphic" terminology in the hope it makes the proof more intuitive. Saying "each tile knows its coordinates", we mean that the tile set is split into disjoint N^2 groups; each of the groups corresponds to tiles that appear in one of the N^2 positions in macro-tiles. The correct positioning of the tiles is ensured (as we have seen) by side colors. The self-similarity guarantees that the same is true for macro-tiles, where the group (i.e., the coordinates in a macro-tile of next level) is determined by the content of the computation zone and corresponding bits (macro-colors) on sides.

(ii) *The first letter in a tile is determined by the second letter and the coordinates of the tile inside the macro-tile, according to the substitution rule.* Indeed, each tile "knows" its coordinates in a macro-tile. So, its first letter must appear in $s(\text{second letter})$ at the corresponding position. We do not need to extend the set of colors to enforce this property. This requirement is only a restriction on tiles: it explains which combinations

$$\langle \text{coordinates in the father macro-tile}, \text{first letter}, \text{second letter} \rangle$$

can be combined in one tile of our tile set.[2]

We want the new tile set to be self-similar. So we should guarantee that the requirements above hold also for macro-tiles. Fortunately, both requirements are easy to integrate in our basic self-referential construction. In each macro-tile, two letters of A are encoded by strings of bits in some specially reserved locations on the tape of the Turing machine (simulated in the computation zone of this macro-tile). The requirement (i) is enforced by adding extra $\log|A|$ bits to macro-colors; to achieve (ii), a macro-tile should check that its first letter appears in $s(\text{second letter})$ at the required position. It is possible when s is easy to compute. (Knowing the coordinates and the second letter, the program computes the required value of the first letter and then compares it with the actual value.)

The requirements (i) and (ii) ensure that if we take first letters from A assigned to each tile, we get an A-configuration which is an s-image of some other configuration. Also (due to self-similarity) we have the same property on the level of macro-tiles. But this is not enough: we need to guarantee that the first letter on the level of macro-tiles is identical to the second letter on the level of tiles. This is also achievable: the first letter of a macro-tile is encoded by bits in its computation zone, and we can require that those bits match the second letter of the tiles at that place (recall that second letter is the same across the tiles that constitute one macro-tile; note also that each tile "knows" its coordinates and can determine whether it is in the zone for the first letter in the macro-tile and which bit should be there). By self-similarity, the same arguments work for macro-tiles of all levels.

It is easy to see that now the tile set τ has the required properties (each tiling projects into a configuration compatible with τ and vice versa).

However, this construction assumes that N (the zoom factor) is equal to the matrix size in the substitution rule, which is usually not the case (m is given, and N we have to choose, and it needs to be large enough). To overcome this difficulty, we let N be equal to m^k for some k, and use the substitution rule s^k, i.e., the kth iteration of s (a configuration is compatible with s^k if and only if it is compatible with s). Now we do not need s to be easily computed: for every s, if k is large enough, the computation of s^k will fit into the available space (exponential in k). □

[2] A natural question arises: what does it mean to add a letter that is determined by other information? Adding a letter means that we create $|A|$ copies of the same tile (with different letters) — but then the restriction prohibits all of them except one, so is there any change at all? In fact, the actual change is happening on higher levels: we want the macro-tiles have both letters written on the tape as binary strings (in some pre-arranged places). This is important for checking consistency between levels.

4 Thue–Morse lemma and strongly aperiodic tile sets

Let $\alpha > 0$ be a real number. A configuration $U\colon \mathbb{Z}^2 \to A$ is *α-aperiodic* if for every nonzero vector $T \in \mathbb{Z}^2$ there exists N such that in every square whose side is at least N the fraction of points x such that $U(x) \neq U(x+T)$ exceeds α.

Remark. If U is α-aperiodic, then Besicovitch distance between U and any periodic pattern is at least $\alpha/2$. (The Besicovitch distance between two configurations is defined as $\limsup_N d_N$ where d_N is the fraction of points where two configurations differ in the $N \times N$ centered square. It is easy to see that the distance does not depend on the choice of the center point.)

Theorem 4. *There exists a tile set τ such that τ-tilings exist and every τ-tiling is α-aperiodic for every $\alpha < 1/4$.*

Proof. This is obtained by applying Theorem 3 to the Thue–Morse substitution rule T (Example 4). Let C be a configuration compatible with T. We have to show that C is α-aperiodic for every $\alpha < 1/4$. It is enough to prove that every configuration compatible with the Thue–Morse substitution rule is α-aperiodic.

Informally, we can reduce the statement to one-dimensional case, since Thue–Morse substitution is a XOR-combination of two one-dimensional substitutions. Here are the details.

Consider a one-dimensional substitution system with two rules $0 \to 01$ and $1 \to 10$. Applying these rules to 0 and 1, we get

$$0 \to 01 \to 0110 \to 01101001 \to \ldots$$
$$1 \to 10 \to 1001 \to 10010110 \to \ldots$$

Let a_n and b_n be nth terms in these sequences ($a_0 = 0$, $a_1 = 01$,..., $b_0 = 1$, $b_1 = 10$, etc.); it is easy to see that $a_{n+1} = a_n b_n$ and $b_{n+1} = b_n a_n$.

For some n we consider the `xor` combination of these strings, where i-j-th bit is `xor` of ith bit in the first string and jth bit in the second string. Since b_n is a bitwise negation of a_n, we get only two different combinations (one obtained from two copies of a_n or two copies of b_n, and the other obtained from different strings) which are bitwise opposite. It is easy to see (e.g., by induction) that these two square patterns are images of 0 and 1 after n steps of two-dimensional Thue–Morse substitution.

To prove the statement for aperiodicity of Thue–Morse configuration, we start with an estimate for (one-dimensional) aperiodicity of a_n and b_n:

Lemma 1 (folklore)**.** *For any integer $u > 0$ and for any n such that $u \leq |a_n|/4$ the shift by u steps to the right changes at least $|a_n|/4$ positions in a_n and leaves unchanged at least $|a_n|/4$ positions. (Formally, in the range $1 \ldots (2^n - u)$ there exist at least $(1/4) \cdot 2^n$ positions i such that ith and $(i+u)$th bits in a_n coincide and at least $(1/4)2^n$ positions where these bits differ.*)

Proof of the Lemma: a_n can be represented as *abbabaab* where $a = a_{n-3}$ and $b = b_{n-3}$. One may assume without loss of generality that $u \geq |a|$ (otherwise we apply Lemma separately to the two halves of a_n). Note that *ba* appears in the sequence twice and once it is preceded by *a* and

once by b. Since a and b are opposite, the shifted bits match in one of the cases and do not match in the other one. The same is true for ab that appears preceded both by a and b. □

Now consider a large $N \times N$ square in two-dimensional Thue–Morse configuration, and some shift vector T. We assume that N is much bigger than components of T (we are interested in the limit behavior as $N \to \infty$). Moreover, we may assume that some power of 2 (let us call it m) is small compared to N and large compared to T. Then $N \times N$ square consists of a large number of $m \times m$ Thue–Morse blocks and some boundary part (which can be ignored by changing α slightly). Then we can consider each $m \times m$ block separately to estimate the fraction of positions that are changed by T-shift. If T is horizontal or vertical, we can use the statement of the lemma directly: at least one fourth of all positions are changed. If not (shift has two non-zero components), we are interested in the probability of some event that is a xor-combination of two *independent* events with probabilities in the interval $(1/4, 3/4)$. It is easy to check that such an event also has probability in $(1/4, 3/4)$ (in fact, even in $(3/8, 5/8)$, but we do not need this stronger bound).

Theorem 4 is proved. □

In fact, the bound $1/4$ can be replaced by $1/3$ if we use more professional analysis of Thue–Morse sequence (see, e.g., [33]). But if we want to get a most strong result of this form and make the bound close to 1, this substitution rule does not work. We can use some other rule (in a bigger alphabet) as Pritykin and Ulyashkina have shown [28], but we prefer to give another construction with variable zoom factors, see Section 6.

5 Variable zoom factor

The fixed point construction of aperiodic tile set is flexible enough and can be used in other contexts. For example, the "zoom factor" N could depend on the level. This means that instead of one tile set τ we have a sequence of tile sets $\tau_0, \tau_1, \tau_2, \ldots$, and instead of one zoom factor N we have a sequence of zoom factors $N_0, N_1, \ldots$ The tile set τ_0 simulates τ_1 with zoom factor N_0, the tile set τ_1 simulates τ_2 with zoom factor N_1, etc.

In other terms, τ_0-tilings can be uniquely split (by horizontal and vertical lines) into $N_0 \times N_0$-macro-tiles from some list, and the macro-tiles in this list are in one-to-one correspondence (that respects matching rules) with τ_1. So τ_0-tilings are obtained from τ_1-tilings by replacing each τ_1-tile by the corresponding τ_0-macro-tile, and each τ_0-tiling has unique reconstruction.

Again, every τ_1-tiling can be split into macro-tiles of size $N_1 \times N_1$ that correspond to τ_2-tiles. So after two steps of zooming out every τ_0-tiling looks like a τ_2-tiling; only closer look reveals that each τ_2-tile is in fact a τ_1-macro-tile of size $N_1 \times N_1$, and even closer look is needed to realize that every τ_1-tile in these macro-tiles is in fact a τ_0-macro-tile of size $N_0 \times N_0$.

This is what we want to achieve (together with other things needed to get the tile set with desired properties, see the discussion below). How do we achieve this? Each macro-tile should "know" its level: macro-tile that simulates τ_k-tile and is made of τ_{k-1}-tiles, should have k in some place on the tape of TM simulated in this macro-tile. To make this information consistent between neighbors, k is exhibited as a part of macro-color at all four sides. The value of k is used for the computations: macro-colors on the sides of a macro-tile encode the coordinates of this macro-tile inside its father, and the computation should check that they are consistent modulo N_{k+1} (the x-

coordinate on the right side should be equal to x-coordinate on the left side plus 1 modulo N_{k+1}, etc.). This means that N_{k+1} should be computable from k, moreover, it should be computable fast enough to fit into the computation zone (which carries only $\Theta(N_k)$ steps of computation). After N_{k+1} is computed, there should be enough time to perform the arithmetic operations modulo N_{k+1}, and so on.

Let us look at these restrictions more closely. We need to keep both k and the coordinates (modulo N_{k+1}) on the tape of level k macro-tiles, and $\log k + O(\log N_{k+1})$ bits are required for that. Both $\log k$ and $\log N_{k+1}$ should be much less than N_k, so all the computations could fit in the available time frame. This means that N_k should not increase too fast or too slow. Say, $N_k = \log k$ is too slow (in this case k occupies almost all available space, and we do not have enough time even for simple computations), and $N_{k+1} = 2^{N_k}$ is too fast ($\log N_{k+1}$ is too large compared to time and space available on the computation zone in a macro-tile of rank k). Also we need to compute N_{k+1} when k is known, so we assume that not only the size of N_{k+1} (i.e., $\log N_{k+1}$) but also the time needed to compute it (given k) are small compared to N_k. These restrictions still allow many possibilities, say, N_k could be proportional to $\sqrt{k}$, k, 2^k, $2^{(2^k)}$, $k!$ etc. Note that we say "proportional" since N_k needs to be reasonably large even for small k (we need some space in macro-tile for wires, all our estimates for computation time are not precise but only asymptotic, so we need some reserve for small k, etc.).

There is one more problem: it is not enough to ensure that the value of k is the same for neighbor macro-tiles. We also need to ensure that this value is correct, i.e., is 1 for level 1 macro-tiles made of τ_0-tiles, is 2 for level 2 macro-tiles made of τ_1-tiles, etc. To guarantee this, we need to compare somehow the level information that is present in a macro-tile and its sons. Using the anthropomorphic terminology, we say that each macro-tile "knows" its level, since it is explicitly written on its tape, and this is, so to say, a "conscious" information processed by a computation in the computation region of the macro-tile. One may say also that a macro-tile of any level contains "subconscious" information ("*existing in the mind but not immediately available to consciousness*", as the dictionary says [34]): this is the information that is conscious for its sons, grandsons and so on (all the way down to the ground level). The problem is that the macro-tile cannot check consistency between conscious and subconscious information since the latter is unavailable (the problem studied by psychoanalysis in a different context).

The solution is to check consistency in the son, not in the father. Every tile knows its level and also knows its position in its father. So it knows whether it is in the place where father should keep level bits, and can check whether indeed the level bit that father keeps in this place is consistent with the level information the tile has. (In fact we used the same trick when we simulate a substitution rule: a check that the father letter of a tile coincides with the letter of the father tile, is done in the same way.) The careful reader will also note here that now the neighbor tiles will automatically have the same level information, so there is no need to check consistency between neighbors.

This kind of a "self-similar" structure with variable zoom factors can be useful in some cases. Though it is not a self-similar according to our definition, one can still easily prove that any tiling is aperiodic. Note that now the computation time for the TM simulated in the central part increases with level, and this can be used for a simple proof of undecidability of domino problem. The problem in the standard proof (based on the self-similar construction with fixed zoom factor) is

that we need to place computations of unbounded size into this self-similar structure, and for that we need special geometric tricks (see [1, 3]). With our new construction, if we want to reduce an instance of the halting problem (some machine M) to the domino problem, we add to the program embedded in our construction the parallel computation of M on the empty tape; if it terminates, this destroys the tiling.

In a similar way we can show that the existence of a periodic tiling is an undecidable property of a tile set, and, moreover, the tile sets that admit periodic tilings and tile sets that have no tilings form two inseparable sets (this is another classical result, see [16]). Recall that two sets A and B are called (computably) *inseparable* if there is no computable set C such that $A \subset C$ and $B \cap C = \emptyset$.

Here is an example of a more exotic version of the latter result (that has probably no interest in itself, just an illustration of the technique). We say that a tile set τ is *m-periodic* if τ-tilings exist and for each of them the set of periods is the set of *all* multiples of m, in other words, if the group of periods is generated by $(0,m)$ and $(m,0)$. Let E [resp. O] be all m-periodic tile sets for all even m [resp. odd m].

Theorem 5. *The sets E and O are inseparable enumerable sets.*

Proof. It is easy to see that the property "to be an m-periodic tile set" is enumerable (both the existence of an m-periodic tiling and enforcing periods $(m,0)$ and $(0,m)$ are enumerable properties).

It remains to reduce some standard pair of inseparable sets (say, machines that terminate with output 0 and 1) to (E,O). It is easy to achieve using the technique explained above. Assume that the numbers N_k increase being odd integers as long as the computation of a given machine does not terminate. When and if it terminates with output 0 [resp. 1], we require periodicity with odd [resp. even] period at the next level. □

Another application of a variable zoom factor is the proof of the following result obtained by Lafitte and Weiss (see [21]) using Turing machine simulation inside Berger–Robinson construction.

Theorem 6. *Let f be a total computable function whose arguments and values are tile sets. Then there exists a tile set τ that simulates a tile set $f(\tau)$.*

Here we assume that some computable encoding for tile sets is fixed. Since there are no restrictions on the computation complexity of f, the choice of the encoding is not important.

Proof. Note that for identity function f this result provides a self-simulating tile set of Section 2.3. To prove it we may use the same kind of a fixed-point technique. However, there is a problem: the computation resources inside a tile are limited (by its size) while time needed to compute f can be large (and, moreover, depends on the tile size).

The solution is to postpone the simulation to large levels: if a tile set τ_0 simulates τ_1 that simulates τ_2 that simulates etc. up to τ_n, then τ_0 simulates τ_n, too. Therefore we may proceed as follows.

We use the construction explained above with a variable zoom factor. Additionally, at each level the computation starts with a preliminary step that may occupy up to (say) half of the available time. On this step we read the program that is on the tape and convert it into the tile set (recall that

each program determines some tile set τ_0 such that τ_0-tilings can be uniquely split into macro-tiles, and this program is written on a read-only part of the tape simulated in the computation zone of all macro-ties, as it was explained in Section 2.2). Then we apply f to the obtained tile set.

This part of the computation checks also that it does not use more than half of the available time and that the output is small enough compared to the tile size. If this time turns out to be insufficient or the output is too big, this part is dropped and we start a normal computation for variable zoom factor, as explained above. In this case zoom factor on the next level should be greater than zoom factor on the current level (e.g., we may assume $N_k = Ck$ for some large enough constant C). However, if the time is enough and result (the list of tiles that corresponds to f's output) is small compared to the tile size, we check that macro-tile (of the current level) belongs to the tile set computed. The hierarchy of macro-tiles stops at this level. The behavior of macro-tiles at this level depends on f: they are isomorphic to $f(\tau)$-tiles. Since the program is the same at all levels and the computation of f should be finite (though may be very long), at some (big enough) level the second possibility is activated, and we get a tile set isomorphic to $f(\tau)$ where τ is the tile set on the ground level. □

Another application of the variable zoom factor technique is the construction of tile sets with any given computable density. Assume that a tile set is given and, moreover, all tiles are divided into two classes, say, A-tiles and B-tiles. We are interested in a fraction of A-tiles in a tiling of an entire plane or its large region. If the tile set is flexible enough, this fraction can vary. However, for some tile sets this ratio tends to a limit value when the size of a tiled region increases. This phenomenon is captured in the following definition: we say that tile set τ divided into A- and B-tiles *has a limit density* α if for every $\varepsilon > 0$ there exists N such that for any $n > N$ the fraction of A-tiles in any tiling of the $n \times n$ square is between $\alpha - \varepsilon$ and $\alpha + \varepsilon$.

Theorem 7. **(i)** *If a tile set has a density α, then α is a computable real number in* $[0,1]$. **(ii)** *Any computable real number $\alpha \in [0,1]$ is a density of some tile set.*

Proof. The first part is a direct corollary of the definitions. For each n we can consider all tilings of the $n \times n$ square and look for the minimal and maximal fractions of A-tiles in them. Let us denote them by m_n and M_n. It is easy to see that the limit frequency (if exists) is in the interval $[m_n, M_n]$. Indeed, in a large square split into squares of size $n \times n$ the fraction of A-tiles is between m_n and M_n being at the same time arbitrarily close to α. Therefore, α is computable (to get its value with ε-precision, we increase n until the difference between M_n and m_n becomes smaller than ε).

It remains to prove (ii). Since α is computable, there exist two computable sequences of rational numbers l_i and r_i that converge to α in such a way that

$$[l_1, r_1] \supset [l_2, r_2] \supset [l_3, r_3] \supset \ldots$$

Our goal will be achieved if macro-tiles of the first level have density either l_1 or r_1, macro-macro-tiles of the second level have density either l_2 or r_2, and so on. Indeed, each large square can be split into macro-tiles (and the border that does not change the density much), so in any large square the fraction of A-tiles is (almost) in $[l_1, r_1]$. The same argument works for macro-macro-tiles, etc.

However, this plan cannot be implemented directly: the main difficulty is that the computation of l_i and r_i may require a lot of time while the computation abilities of macro-tiles of level i are limited (we use variable zoom factors, e.g., we may define the kth zoom factor as $N_k = Ck$, but they cannot grow too fast).

The solution is to postpone the switch from densities l_i and r_i to densities l_{i+1} and r_{i+1} to the higher level of the hierarchy where the computation has enough time to compute all these four rational numbers and find out in which proportion l_i- and r_i-tiles should be mixed in l_{i+1}- and r_{i+1}-tiles. (This proportion is restricted by our construction: the denominator should be the number of i-level macro-tiles in $(i+1)$-level macro-tile, but this restriction can be always satisfied by a slight change in l_i and r_i which leaves α unchanged.) So, we allocate, say, the first half of the available time for controlled computation of all these values; if the computation does not finish in time, the densities for the next level are the same as for the current level. If the computation terminates in time, we use the result of the computation to have two types of the next level tiles: one with density l_{i+1} and one with density r_{i+1}. They are made by using prescribed amount of l_i- and r_i-tiles (since each tile knows its coordinates, it can find out whether it should be of the first or second type). This finishes the construction. □

6 Strongly aperiodic tile sets revisited

In Section 4 we constructed a tile set such that every tiling is α-aperiodic for every $\alpha < 1/4$. Now we want to improve this result and construct a tile set such that every tiling is, say, 0.99-aperiodic (here 0.99 can be replaced by any constant less than 1). It is easy to see that this cannot be achieved by the same argument, with Thue–Morse substitutions, as well as with any substitutions in a two-letter alphabet; we need a large alphabet to make the constant close to 1.

It is possible to achieve 0.99-aperiodicity with some carefully chosen substitution rule (in a bigger alphabet) recently proposed by Pritykin and Ulyashkina [28], just applying Theorem 3 (similarly to the argument with the Thue-Morse substitution presented in Section 4). In this section we present an alternative proof of this result. We exploit substitution rules with variable zoom factor (and different substitutions on each level) and use an idea of error correcting code.

Instead of one alphabet, A, we now consider a sequence of finite alphabets, A_0, A_1, A_2,…; the cardinality of A_k will grow as k grows. Then we consider a sequence of mappings:

$$s_1 \colon A_1 \to A_0^{N_0 \times N_0}, \quad s_2 \colon A_2 \to A_1^{N_1 \times N_1}, \quad s_3 \colon A_3 \to A_2^{N_2 \times N_2}, \ldots$$

where N_0, N_1, N_2,… are some positive integers (zoom factors); N_k will increase as k increases.

Then we can compose this mappings. For example, a letter z in A_2 can be first replaced by a $N_1 \times N_1$ square $s_2(z)$ filled by A_1-letters. Then each of these letters can be replaced by a $N_0 \times N_0$-square filled by A_0-letters according to s_1 and we get a $N_0N_1 \times N_0N_1$-square filled by A_0-letters; we denote this square by $s_1(s_2(z))$ (slightly abusing the notation).

We call all this (i.e., the sequence of A_k, N_k, s_k) a *substitution family*. Such a family defines a class of A_0-configurations compatible with it (in the same way as in Section 3). Our plan is to construct a substitution family such that:

- every configuration compatible with this family is 0.99-aperiodic;
- there exists a tile set and projection of it onto A_0 such that only compatible configurations (and all compatible configurations) are projections of tilings.

In other words, we use the same argument as before (proving Theorem 4) but use a substitution family instead of one substitution rule. This substitution family will have special properties:

A. Symbols used in different locations are different. This means that A_k-letters that appear in a given position of the squares $s_{k+1}(z)$ for some $z \in A_{k+1}$, never appear in any other places of these squares (for any z); thus, set A_k is split into $N_k \times N_k$ disjoint subsets used for different positions in $N_k \times N_k$ squares.

B. Different letters are mapped to squares that are far away in Hamming distance. This means that if $z, w \in A_{k+1}$ are different, then the Hamming distance between images $s_{k+1}(z)$ and $s_{k+1}(w)$ is large: the fraction of positions in the $N_k \times N_k$ square where $s_{i+1}(z)$ and $s_{i+1}(w)$ have equal letters, does not exceed ε_k.

Here ε_i will be a sequence of positive reals such that $\sum_{i\geq 0} \varepsilon_i < 0.01$.

This implies that composite images of different letters are also far apart. For example, the fraction of positions in the $N_0N_1 \times N_0N_1$ square where $s_1(s_2(z))$ and $s_1(s_2(w))$ coincide does not exceed $\varepsilon_0 + \varepsilon_1 < 0.01$. Indeed, in $s_2(z)$ and $s_2(w)$ we have at most ε_1-fraction of matching letters; these letters generate ε_1-fraction of matching A_0-letters on the ground level; all other (non-matching) pairs add ε_0-fraction. In fact, we get even a stronger bound $1-(1-\varepsilon_0)(1-\varepsilon_1)$.

In the same way, if we take two different letters in A_k and then go down to the ground level and obtain two squares of size $N_0N_1\ldots N_{k-1} \times N_0N_1\ldots N_{k-1}$ filled by A_0-letters, the fraction of coincidences is at most $\varepsilon_0 + \ldots + \varepsilon_{k-1} < 0.01$.

This property of the substitution family implies the desired property:

Lemma 2. *If an A_0-configuration U is compatible with a substitution family having properties* (A) *and* (B)*, then U is* 0.99*-aperiodic.*

Proof. Consider a shift vector T. If T is not a multiple of N_0 (one of the coordinates is not a multiple of N_0), then property (A) guarantees that the original configuration and its T-shift differ everywhere. Now assume that T is a multiple of N_0. Then T induces a T/N_0-shift of an A_1-configuration U_1 that is a s_1-preimage of U. If T is not a multiple of N_0N_1, then T/N_0 is not a multiple of N_1 and for the same reason this T/N_0-shift changes all the letters in U_1. And different letter in A_1 are mapped to $N_0 \times N_0$ squares that coincide in at most ε_0-fraction of positions.

If T is a multiple of N_0N_1 but not $N_0N_1N_2$, we get a $T/(N_0N_1)$ shift of A_2-configuration U_2 that changes all its letters, and different letters give squares that are $1-(\varepsilon_0+\varepsilon_1)$ apart. The same argument works for the higher levels. □

It remains to construct a substitution family that has properties (A) and (B), and can be enforced by a tile set. The property (B) (large Hamming distance) is standard for coding theory, and the classical tool is the Reed–Solomon code.

Let us remind the idea of the Reed-Solomon codes (for details see, e.g., [4]). The codewords of the Reed–Solomon code are tables of (values of) polynomials of bounded degree. More precisely, we fix some finite field $\mathbb{F}_q$ of size q and an integer $d > 0$. Let $p(x) = a_0 + a_1 x + \ldots + a_{d-1}x^{d-1}$ be a polynomial over $\mathbb{F}_q$ of degree less than d. Then the codeword corresponding to $p(x)$ (i.e., the encoding of the sequence $a_0, \ldots, a_{d-1}$) is a vector in $(\mathbb{F}_q)^q$ (i.e., a sequence of q elements of the field), which consists of the values of this polynomial computed at all points $x \in \mathbb{F}_q$. Thus, for given parameters d and q, the code consists of q^d codewords. Since two polynomials of degree less than d can coincide in at most $(d-1)$ points, the distance between any two codewords is at least $q - d + 1$. Of course, this construction can be used even if the desired length of the codewords is not a size of any finite field; we can choose a slightly larger field and use only part of its elements.)

Now we embed these codes in a family of substitution rules. First, let B_k be a finite field (we specify its size below) and A_k be equal to $B_k \times \{0,1,\ldots,N_k-1\} \times \{0,1,\ldots,N_k-1\}$; let us agree that we use letters $\langle b,i,j\rangle$ only in (i,j)-position of an s_{k+1}-image. This trivially implies requirement (A).

Then we construct a code that encodes each A_{k+1}-letter w by a a string of length N_k^2 made of B_k-letters (arranged in a square); adding the coordinates, we get s_{k+1}-image of w. Thus, we need a sequence of codes:

$$
\begin{aligned}
s_1 &: A_1 \to B_0^{N_0 \times N_0}, \quad \text{s.t. } s_1(a_i),\ s_1(a_j) \text{ coincide at most in } \varepsilon_0\text{-fraction of all positions (if } i \neq j)\\
s_2 &: A_2 \to B_1^{N_1 \times N_1}, \quad \text{s.t. } s_2(a_i),\ s_2(a_j) \text{ coincide at most in } \varepsilon_1\text{-fraction of all positions (if } i \neq j)\\
&\ldots
\end{aligned}
$$

To satisfy requirement (B), we need a code with the Hamming distance (between every two codewords) at least $(1-\varepsilon_k)N_k^2$. The Reed–Solomon codes works well here. The size of the field can be equal to the length of the codeword, i.e., N_k^2. Let us decide that N_k is a power of 2 and the size of the field B_k is exactly N_k^2. (There are fields of size 2^t for every $t = 1,2,3,\ldots$; we could use also $\mathbb{Z}/p\mathbb{Z}$ for prime p of an appropriate size.) To achieve the required code distance, we use polynomials of degree less than $\varepsilon_k N_k^2$. The number of codewords (polynomials of degree less than $\varepsilon_k N_k^2$) is at least $2^{\varepsilon_k N_k^2}$ (even if we use only polynomials with coefficients 0 and 1). This is enough if

$$|A_{k+1}| \leq 2^{\varepsilon_k N_k^2}.$$

Recalling that $|A_{k+1}| = |B_{k+1}| \cdot N_{k+1}^2$ and that B_{k+1} is a field of size N_{k+1}^2, we get the inequality

$$N_{k+1}^4 \leq 2^{\varepsilon_k N_k^2}, \text{ or } 4\log N_{k+1} \leq \varepsilon_k N_k^2.$$

Now let $N_k = 2^{k+c}$ for some constant c; we see that for large enough c this inequality is satisfied for ε_k with sum less than 0.01 (or any other constant), since the left-hand side is linear in k while the right-hand side is exponential.

Now it remains to implement all this scheme using tiling rules. As we have discussed, the zoom factor $N_k = 2^{k+c}$ is OK for the construction. This factor leaves enough space to keep two substitution letters (for the tile itself and its father tile), since these letters require linear size (in k). Moreover, we have enough time to perform the computations in the finite fields needed to

construct the error correction code mappings. Indeed, in a k-level macro-tile we are allowed to use exponential (in the bit size of the field element) time. Remind that one can operate with elements in the field of size 2^r using polinomial (in r) time; to this end we need to construct some irreducible polynomial p of degree r over the field of two elements and then perform arithmetic operations (on polynomials) modulo p. All this operations can be done by deterministic algorithms in polynomial time, see, e.g., [23]. Thus, we can reuse here the construction of the proof of Theorem 3.

Remark. We can also get an 0.99-aperiodic tile set as a corollary of the result of next section; indeed, we construct there a tile set such that any tiling embeds a horizontal sequence with high complexity substrings, and such a sequence cannot match itself well after a shift (in fact, to get 0.99-aperiodicity we would need to replace a binary alphabet by a larger finite alphabet in this argument). We can superimpose this with a similar 90°-rotated construction; then any non-zero translation will shift either vertical or horizontal sequence and therefore change most of the positions. Note that in this way we can also get a tile set that is 0.99-far from every periodic pattern (a slightly different approach to define "strong aperiodicity").

However, we preferred to present in this section a more explicit (and simpler) construction that does not refer to (rather complicated) arguments in Section 7.

7 Tile set that has only complex tilings

In this section we provide a new proof of the following result from [7]:

Theorem 8. *There exists a tile set τ and constants $c_1 > 0$ and c_2 such that τ-tilings exist and in every τ-tiling T every $N \times N$-square has Kolmogorov complexity at least $c_1 N - c_2$.*

Here Kolmogorov complexity of a tiled square is the length of the shortest program that describes this square. We assume that programs are bit strings. Formally speaking, Kolmogorov complexity of an object depends on the choice of programming language (consult [31] for the definition and properties of Kolmogorov complexity). However, in our case the choice of programming language does not matter, and you may think of Kolmogorov complexity of an object as the length of the shortest program in your favorite programming language that prints out this object. We need to keep in mind only two important properties of Kolmogorov complexity. First, the Kolmogorov complexity function is not computable, but is *upper semicomputable*. This means that there is an algorithm that for a given n enumerates all objects that have complexity less than n. It can be done by the brute force search over all short descriptions: we cannot say in advance which programs stop with some output and which do not; but we can run all programs of length less than n in parallel, and enumerate the list of their outputs, as some programs stop. Second, any computable transformation (e.g., the change of encoding) changes Kolmogorov complexity at most by $O(1)$. We refer to [7] for the discussion of Theorem 8 (why it is optimal, why the exact value of c_1 does not matter etc.) and other related results.

7.1 A biinfinite bit sequence

Proof. We start the proof in the same way as in [7]: we assume that each tile keeps a bit that propagates (unchanged) in the vertical direction. Then any tiling contains a biinfinite sequence of bits ω_i (where $i \in \mathbb{Z}$). Any $N \times N$ square contains an N-bit substring of this string, so if (for large enough N) every N-bit substring of ω has complexity at least $c_1 N$ for some fixed c_1, we are done.

We say that a sequence ω has *Levin's property* if every N-bit substring x of ω has complexity $\Omega(N)$. Such a biinfinite sequence indeed exists (see [7]; another proof can be obtained by using Lovasz local lemma, see [30]). So our goal is to formulate tiling rules in such a way that a correct tiling "ensures" that the biinfinite sequence embedded in it indeed has this property.

The set of all "forbidden" binary strings, i.e., strings x such that $K(x) < c_1|x| - c_2$ (here $K(x)$ stands for Kolmogorov complexity of x and $|x|$ stands for the length of x) is enumerable: there is an algorithm that generates the list of all forbidden substrings. It would be nice to embed into the tiling a computation that runs this algorithm and compares its output strings with the substrings of ω; such a computation blows up (creates a tiling error) if a forbidden substring is found.

However, there are several difficulties.

- First of all, our self-similar tiling contains only finite computations; the duration depends on the zoom factor and may increase as the level increases (bigger macro-tiles keep longer computations), but at any level the computations are finite. This is a problem since or a given string x we do not know *a priori* how much time the shortest program for x uses, so we never can be sure that Kolmogorov complexity of x is large. Hence, each substring of ω should be examined in computations somehow distributed over infinitely many macro-tiles.

- The computation at some level deals with bits encoded in the cells of that level, i.e., with macro-tile states. So the computation cannot achieve the bits of the sequence (that are "deep in the subconscious") directly and some mechanism to dig them out is needed.

Let us explain how to overcome these difficulties.

7.2 Bits delegation

A macro-tile of level k is a square whose side is $L_k = N_0 \cdot N_1 \cdot \ldots \cdot N_{k-1}$, so there are L_k bits of the sequence that intersect this macro-tile. Let us delegate each of these bits to one of the macro-tiles it intersects. Note that macro-tile of the next level is made of $N_k \times N_k$ macro-tiles of level k. We assume that N_k is much bigger than L_k (more about choice of N_k later); this guarantees that there are enough macro-tiles of level k (in the next level macro-tile) to serve all bits that intersect them. Let us decide that ith (from bottom to top) macro-tile of level k in a $(k+1)$-level macro-tile serves (consciously knows, so to say) ith bit (from the left) in its zone, see Fig. 5. Since $N_k \gg L_k$, we have much more macro-tiles of level k (inside some macro-tile of level $k+1$) than needed to serve all bits. So some k-level macro-tiles remain unused.

So each bit (each vertical line) has a representative on every level — a macro-tile that consciously knows this bit. However, we need some mechanisms that guarantee that this information

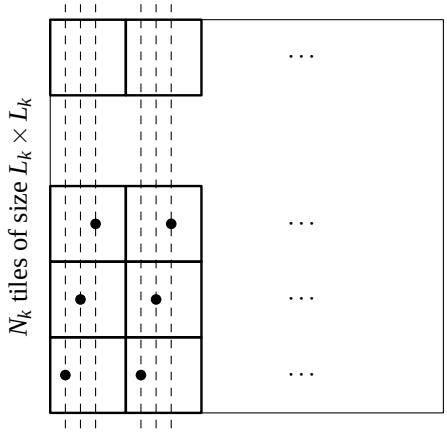

Figure 5: Bit delegation

is indeed true (i.e., consistent on different levels). On the bottom level it is easy, since the bits are available on the same level.

To guarantee the consistency we use the same trick as in Section 3: at each level macro-tile keeps not only its own bit but also its father's bit and makes necessary consistency checks. Namely, each macro-tile knows (has on its computation tape):

- the bit delegated to this macro-tile;
- the coordinates of this macro-tile in its father macro-tile (that are already used in the fixed-point construction); the y-coordinate at the same time is the position of the bit delegated to this macro-tile (relative to the left boundary of the macro-tile).
- the bit delegated to the father of this macro-tile;
- the coordinates of the father macro-tile in the grandfather macro-tile.

This information is subject to consistency checks:

- the information about the father macro-tile should coincide with the same information in neighbor tiles (unless they have a different father, i.e., one of the coordinates is zero).
- if it happens that the bit delegated to the father macro-tile is the same bit as delegated for the tile, these bits should match;
- it can happen that the macro-tile occupies a place in its father macro-tile where some bits of father's coordinates (inside grandfather macro-tile) or the bit delegated to the father are kept; then this partial information on the father level should be checked against the information about father coordinates and bit.

These tests guarantee that the information about father is the same in all brothers, and some of these brothers (that are located on the father tape) can check it against actual father information; at the same time some other brother (that has the same delegated bit as the father) checks the consistency of the delegated bits information.

Note that this scheme requires that not only $\log N_k$ but also $\log N_{k+1}$ is much less than N_{k-1}. This requirement, together with the inequality $L_k = N_0 \cdot N_1 \cdot \ldots N_{k-1} \leq N_k$ (discussed earlier) is satisfied if $N_k = Q^{c^k}$ where Q is a large enough constant (this is needed also to make macro-tiles of the first level large enough) and $c > 2$ (so $1 + c + c^2 + \ldots + c^{k-1} < c^k$).

Later, in Section 13, the choice of c has to be reconsidered: we need $2 < c < 3$ to achieve error correction, but for our current purposes this does not matter.

7.3 Bit blocks checked

We explained how macro-tile of any level can have a true information about one bit (delegated to it). However, we need to check not bits, but substrings (and throw a tiling error if a forbidden string appears). Note that it is OK to test only very short substrings compared to the macro-tile size (N_k): if this test is done on all levels, this restriction does not prevent us from detecting any violation. (Recall that short forbidden substrings can appear very late in the generation process, so we need computation at arbitrary high levels for this reason, too.)

So we need to provide more information to macro-tiles. It can be done in the following way. Let us require that a macro-tile contains not one bit but a group of bits that starts at the delegated bit and has length depending on the level k (and growing very slowly with k, say, $\log\log\log k$ is slow enough). If this group goes out of the region occupied by a macro-tile, we truncate it.

Similarly, a macro-tile should have this information for the father macro-tile (even if the bits are outside its own region), this information should be the same for brothers and needs to be checked against the delegated bits on the macro-tile level and pieces of information on the father level.

The computation in the computation zone runs the process that generates the list of all forbidden strings (strings that have too small Kolmogorov complexity) and checks the forbidden strings that appear against all the substrings of the group of bits available to this macro-tile. This process is time- and space-bounded, but this does not matter since every string is considered on a high enough level.

Our construction has some kind of duplication: we first guarantee the consistency of information for individual bits, and then do the same for substrings. The first part of the construction is still needed, since we need arbitrary long substrings to be delegated to macro-tiles (of high enough level), so delegation of substrings cannot start from the ground level where the tile size is limited, so we need to deal with bits separately.

7.4 Last correction

The argument explained above still needs some correction. We claim that every forbidden string will be detected at some level where it is short enough compared to the level parameters. However, there could be strings that never become a part of one macro-tile. Imagine that there is some

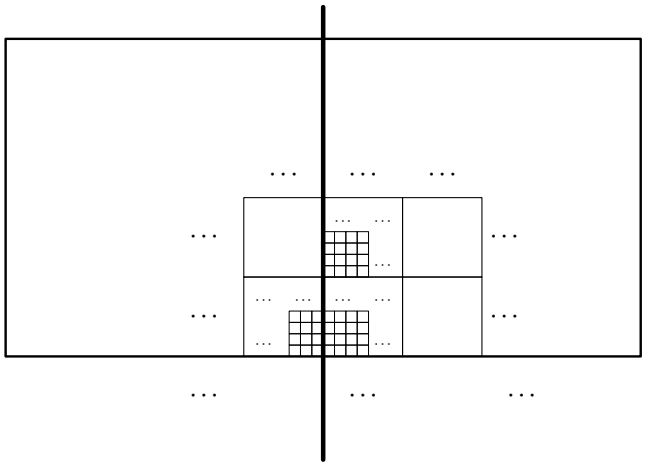

Figure 6: Degenerate case: An infinite vertical line is a boundary between macro-tiles of all levels.

vertical line that is a boundary between macro-tiles of all levels (so we have bigger and bigger tiles on both sides, and this line is still the boundary between them, see Fig. 6). Then a substring that crosses this line will be never checked and therefore we cannot guarantee that it is not forbidden.

There are several ways to get around this problem. One can decide that each macro-tile contains information not only about blocks inside its father macro-tile but in a wider regions (say, three times wider, including uncle macro-tiles); this information should be checked for consistency between cousins, too. This trick (extension of zones of responsibility for macro-tiles) will be used later in Section 8.

But to prove Theorem 8 a simpler solution is enough. Note that even if a string on the boundary is never checked, its parts (on both sides of the boundary) are, so their complexity is proportional to their length. And one of the parts has length at least half of the original length, so we still have a complexity bound, just the constant is twice smaller.

This finishes the proof of Theorem 8. □

8 Subshifts

The analysis of the proof in the previous section shows that it can be divided into two parts. We defined *forbidden* strings as bit strings that are sufficiently long and have complexity at most $\alpha \cdot (\text{length})$. We started by showing that biinfinite strings without forbidden factors (substrings) exist. Then we constructed a tile set that contains such a biinfinite string in any tiling.

The second part can be separated from the first one, and in this way we get new proofs for some results of S. Simpson [32] and M. Hochman [18] about effectively closed subshifts.

Fix some alphabet A. Let F be a set of A-strings. Consider a set S_F of all biinfinite A-sequences that have no factors (substrings) in F. This is a *closed* 1*-dimensional subshift* over A, i.e., a closed shift-invariant subset of the space of all biinfinite A-sequences. If the set F is (computably) enumerable, S_F is called an *effectively closed* 1*-dimensional subshift* over A. If F is finite, S_F is

called a *subshift of finite type*.

We can define 2-dimensional subshifts in a similar way. More precisely, let F be a set of two-dimensional patterns (squares filled with A-letters). Then we can consider a set S_F of all A-configurations (= mappings $\mathbb{Z}^2 \to A$) that do not contain any pattern from F. This is a closed shift-invariant set of A-configurations (= 2-dimensional closed subshift over A). If F is (computably) enumerable, S_F is called *a* 2*-dimensional effectively closed subshift* over A. If F is finite, S_F is called *a* 2*-dimensional subshift of finite type*.

All (non-empty) 1-dimensional subshifts of finite type always contain periodic sequences. Berger's theorem says that for two-dimensional subshifts it is not the case. Indeed, a tile set can be transformed into a subshift since color matching condition is local, and there exist aperiodic tile sets. Moreover, 2-dimensional subshifts of finite type are powerful enough to simulate any effectively closed 1-dimensional subshift in the following sense:

Theorem 9. *Let A be some alphabet and let S be a* 1*-dimensional effectively closed subshift over A. Then there exists an alphabet B, a mapping $r\colon B \to A$, and a* 2*-dimensional subshift S' of finite type over B such that r-images of configurations in S' are* (*exactly*) *elements of S extended vertically* (*vertically aligned cells contain the same A-letter*).

(As we have mentioned, this result was independetly obtained by Aubrun and Sablik using Robinson-style aperiodic tilings [2].)

Proof. The proof uses the same argument as in Theorem 8. Each cell now contains an A-letter that propagates vertically. Computation zones in macro-tiles generate (in available space and time) elements of the enumerable set of forbidden A-substrings and compare them with A-substrings that are made available to them. It remains to note that tiling requirements (matching colors) are local, i.e., they define a finite type 2-dimensional subshift.

Note that now the remark of Section 7.4 (the trick of extension of zones of responsibility for macro-tiles) becomes crucial, since otherwise the image of S'-configuration may be a concatenation of two sequences (a left-infinite one and a right-infinite one); each sequence does not contain forbidden patterns but forbidden patterns may appear near the meeting point. □

A similar argument shows that every 2-dimensional effectively closed subshift can be represented as an image of a 3-dimensional subshift of finite type (after a natural extension along the third dimension), any 3-dimensional effectively closed subshift is an image of a 4-dimensional subshift of finite type, etc.

This result is an improvement of a similar one proved by M. Hochman (Theorem 1.4 in [18], where the dimension increases by 2), thus providing a solution of Problem 9.1 in this paper. Note also that it implies the result of S. Simpson [32] where 1-dimensional sequences are embedded into 2-dimensional tilings but in some weaker sense (defined in terms of Medvedev degrees).

One can ask whether a dimension reduction is essential here. For example, is it true that every 2-dimensional effectively closed subshift is an image of some 2-dimensional subshift of finite type? The answer for this question (and related questions in higher dimensions) is negative. This follows from an upper bound in [7] saying that every tile set (unless it has no tilings at all) has a tiling such that all $n \times n$ squares in it have complexity $O(n)$ (this result immediately translates for subshifts of

finite type) and a result from [30] that shows that some non-empty effectively closed 2-dimensional subshift has $n \times n$ squares of complexity $\Omega(n^2)$. Therefore the latter cannot be an image of the first one (complexity can only decrease when we apply an alphabet mapping).

9 Random errors

9.1 Motivation and discussion

In what follows we discuss tilings with faults. This means that there are some places (faults) where colors of neighbor tiles do not match. We are interested in "robust" tile sets which maintain some structure (for example, can be converted into an error-free tiling by changing a small fraction of tiles) if faults are sparse.

There are two almost equivalent ways to define faulty tilings. We can speak about *errors* (places where two neighbor tiles do not match) or *holes* (places without tiles). Indeed, we can convert a tiling error into a hole (by deleting one of two non-matching tiles) or convert a 1-tile hole (one missing tile) into a small number (at most 4) errors by placing an arbitrary tile there. Holes look more naturally if we start with a set of holes and then try to tile the rest; on the other hand, if we imagine some process similar to crystallization when a tiling tries to become correct by some trial-and-error procedure, it is more natural to consider tiling errors. Since it does not make serious difference from the mathematical point of view, we use both metaphors.

We use a hierarchical approach to hole patching that goes back to P. Gacs who used it in a much more complicated situation [13]. This means that first we try to patch small holes that are not too close to each other (by changing small neighborhoods around them). This (if we are lucky enough) makes larger (and still unpatched) holes more isolated since there are less small holes around. Some of these larger holes (that are not too large and not too close to each other) can be patched again. Then the same procedure can be repeated again for the next level. Of course, we need some conditions (that guarantee that holes are not too dense) to make this procedure successful. These conditions are described later in full details, but the important question is: How do we ensure that these conditions are reasonable (i.e., general enough)? Our answer is: we prove that if holes are generated at random (each position becomes a hole independently of other positions with small enough probability ε) then the generated set satisfies these conditions with probability 1.

From the physics viewpoint, this argument sounds rather weak: if we imagine some crystallization process, errors in different positions are not independent at all. However, this approach could be a first approximation until a more adequate one is found.

Note that patching holes in a tiling could be considered as a generalization of the percolation theory. Indeed, let us consider a simple tile set made of two tiles: one has all black sides and the other has all white sides. Then the tiling conditions reduce to the following simple condition: each connected component of the complement to the holes set is either completely black or completely white. We want to make small corrections in the tiling that patch the holes (and therefore make the entire plane black or white). This means that initially either we have small black "islands" in a white ocean or vice versa, which is exactly what percolation theory says (it guarantees that if

holes are generated at random independently with small probability, the rest consists of one large connected component and many small islands.)

This example shows also that simple conditions like small density (in Besicovitch sense) of the holes set are not enough: a regular grid of thin lines can have small density but still splits the plane into non-connected squares; if half of these squares are black and the others are white, no small correction can patch the holes.

One can define an appropriate notion of a sparse set in the framework of algorithmic randomness (Martin-Löf definition of randomness) considering individual random sets (with respect to Bernoulli distribution B_ε) and their subsets as "sparse". Then we can prove that any sparse set (in this sense) satisfies the conditions that are needed to make the iterative patching procedure works. This algorithmic notion of "sparseness" is discussed in [5]. However, in the current paper we do not assume that reader is familiar with algorithmic randomness and restrict ourselves to the classical probability theory.

So our statements become quite lengthy and use probabilistic quantifiers "for almost all" (=with probability 1). The order of quantifiers (existential, universal and probabilistic) is important here. For example, the statement "a tile set τ is robust" means that *there exists* some $\varepsilon > 0$ such that *for almost all* E (with probability 1 with respect to the distribution where each point independently belongs to E with probability ε) the following is true: *for every* (τ, E)-tiling U *there exists* a τ-tiling U' (of the entire plane) that is "close" to U. Here by (τ, E)-tiling we mean a tiling of $\mathbb{Z}^2 \setminus E$ (where existing pairs of neighbor tiles match).

9.2 Islands of errors

In this section we develop the notion of "sparsity" based on the iterative grouping of errors (or holes) and prove its properties.

Let $E \subset \mathbb{Z}^2$ be a set of points; points in E are called *dirty*; other points are *clean*. Let $\beta \geq \alpha > 0$ be integers. A non-empty set $X \subset E$ is an (α, β)-*island* in E if:

(1) the diameter of X does not exceed α;

(2) in the β-neighborhood of X there is no other point from E.

(Diameter of a set is a maximal distance between its elements; the distance d is defined as l_∞, i.e., the maximum of distances along both coordinates; β-neighborhood of X is a set of all points y such that $d(y,x) \leq \beta$ for some $x \in X$.)

It is easy to see that two (different) islands are disjoint (and the distance between their points is greater than β).

Let (α_1, β_1), (α_2, β_2),... be a sequence of pairs of integers and $\alpha_i \leq \beta_i$ for all i. Consider the following iterative "cleaning" procedure. At the first step we find all (α_1, β_1)-islands (*rank* 1 *islands*) and remove all their elements from E (thus getting a smaller set E_1). Then we find all (α_2, β_2)-islands in E_1 (*rank* 2 *islands*); removing them, we get $E_2 \subset E_1$, etc. Cleaning process is *successful* if every dirty point is removed at some stage.

At the ith step we also keep track of the β_i-neighborhoods of islands deleted during this step. A point $x \in \mathbb{Z}^2$ is *affected* during the ith step if x belongs to one of these neighborhoods.

The set E is called *sparse* (for a given sequence α_i, β_i) if the cleaning process is successful, and, moreover, every point $x \in \mathbb{Z}^2$ is affected at finitely many steps only (i.e., x is far from islands

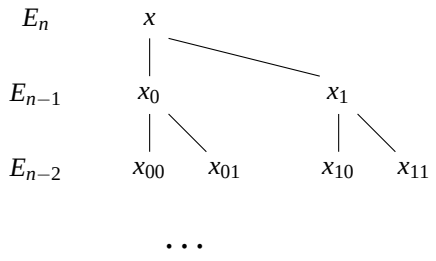

Figure 7: Explanation tree; vertical lines connect different names for the same points.

of sufficiently large ranks).

The values of α_i and β_i should be chosen in such a way that for sufficiently small $\varepsilon > 0$ a B_ε-random set is sparse with probability 1. (As we have said, this justifies that our notion of sparsity is not unreasonably restrictive.) The sufficient conditions are provided by the following statement:

Lemma 3. *Assume that*

$$8\sum_{k<n}\beta_k < \alpha_n \le \beta_n \quad \textit{for every } n \textit{ and} \quad \sum_i \frac{\log\beta_i}{2^i} < \infty.$$

Then for all sufficiently small $\varepsilon > 0$ a B_ε-random set is sparse with probability 1.

Proof of Lemma 3. Let us estimate the probability of the event "x is not cleaned after n steps" for a given point x (this probability does not depend on x). If $x \in E_n$, then x belongs to E_{n-1} and is not cleaned during the nth step (when (α_n,β_n)-islands in E_{n-1} are removed). Then $x \in E_{n-1}$ and, moreover, there exists some other point $x_1 \in E_{n-1}$ such that $d(x,x_1)$ is greater than $\alpha_n/2$ but not greater than $\beta_n + \alpha_n/2 < 2\beta_n$. Indeed, if there were no such x_1 in E_{n-1}, then the $\alpha_n/2$-neighborhood of x in E_{n-1} is an (α_n,β_n)-island in E_{n-1} and x would be removed.

Each of the points x_1 and x (that we denote also x_0 to make the notation uniform) belongs to E_{n-1} because it belongs to E_{n-2} together with some other point (at the distance greater than $\alpha_{n-1}/2$ but not exceeding $2\beta_{n-1}$). In this way we get a tree (Figure 7) that "explains" why x belongs to E_n.

The distance between x_0 and x_1 in this tree is at least $\alpha_n/2$ while the diameter of the subtrees starting at x_0 and x_1 does not exceed $\sum_{i<n} 2\beta_i$. Therefore, the Lemma's assumption guarantees that these subtrees cannot intersect and, moreover, that all the leaves of the tree are different. Note that all 2^n leaves of the tree belong to $E = E_0$. As every point appears in E independently from other points, such an "explanation tree" is valid with probability ε^{2^n}. It remains to estimate the number of possible explanation trees for a given point x.

To specify x_1 we need to specify horizontal and vertical distance between x_0 and x_1. Both distances do not exceed $2\beta_n$, therefore we need about $2\log(4\beta_n)$ bits to specify them (including the sign bits). Then we need to specify the distances between x_{00} and x_{01} as well as distances between x_{10} and x_{11}; this requires at most $4\log(4\beta_{n-1})$ bits. To specify the entire tree we therefore need

$$2\log(4\beta_n) + 4\log(4\beta_{n-1}) + 8\log(4\beta_{n-2}) + \ldots + 2^n\log(4\beta_1)$$

bits, and that is (reversing the sum and taking out the factor 2^n) equal to $2^n(\log(4\beta_1)+\log(4\beta_2)/2+\ldots)$. Since the series $\sum \log\beta_n/2^n$ converges by assumption, the total number of explanation trees for a given point (and given n) does not exceed $2^{O(2^n)}$, so the probability for a given point x to be in E_n for a B_ε-random E does not exceed $\varepsilon^{2^n}2^{O(2^n)}$, which tends to 0 (even super-exponentially fast) as $n\to\infty$, assuming that ε is small enough.

We conclude that the event "x is not cleaned" (for a given point x) has zero probability; the countable additivity guarantees that with probability 1 all points in $\mathbb{Z}^2$ are cleaned.

It remains to show that every point with probability 1 is affected by finitely many steps only. Indeed, if x is affected by step n, then some point in its β_n-neighborhood belongs to E_n, and the probability of this event is at most

$$O(\beta_n^2)\varepsilon^{2^n}2^{O(2^n)} = 2^{2\log\beta_n+O(2^n)-\log(1/\varepsilon)2^n};$$

the convergence conditions guarantees that $\log\beta_n = o(2^n)$, so the first term is negligible compared to others, the probability series converges (for small enough ε) and the Borel–Cantelli lemma gives the desired result. □

Our next step: by definition a sparse set is split into a union of islands of different ranks; now we prove that these islands together occupy only a small part of the plane. To make this statement formal, we use the notion of Besicovitch size (density) of a set $E\subset\mathbb{Z}^2$. Let us recall the definition. Fix some point O of the plane and consider squares of increasing size centered at O. For each square consider the fraction of points in this square that belong to E. The lim sup of these frequencies is called *Besicovitch density* of E. (Note that the choice of the center point O does not matter, since for any two points O_1 and O_2 large squares of the same size centered at O_1 and O_2 share most of their points.)

By definition the distance between two rank k islands is at least β_k. Therefore the $\beta_k/2$-neighborhoods of these islands are disjoint. Each of the islands contains at most α_k^2 points (it can be placed in a rectangle that has sides at most α_k). Each neighborhood has at least β_k^2 points (since it contains a $\beta_k\times\beta_k$-square centered at any point of the island). Therefore the union of all rank k islands has Besicovitch density at most $(\alpha_k/\beta_k)^2$. Indeed, for a large square the islands near its border can be ignored, and all other islands are surrounded by disjoint neighborhoods where their density is bounded by $(\alpha_k/\beta_k)^2$.

One would like to conclude that the overall density of all islands (of all ranks) does not exceed $\sum_k(\alpha_k/\beta_k)^2$. However, the Besicovitch density is in general not countably semi-additive (for example, the union of finite sets having density 0 may have density 1). But in our case the second condition of the definition of a sparse set (each point is covered by only finitely many neighborhoods of islands) helps.

Lemma 4. *Let E be a sparse set for a given family of α_k and β_k. Then Besicovitch density of E is $O(\sum(\alpha_k/\beta_k)^2)$.*

Proof of Lemma 4. Let O be a center point used in the definition of Besicovitch density. By definition of sparsity, this point is not covered by β_k-neighborhoods of rank k islands if k is greater than some K. Now we split the set E into two parts: one ($E_{\le}$) is formed by islands of rank at most K and other ($E_{>}$) is formed by all islands of bigger ranks. As we have just seen, in a large

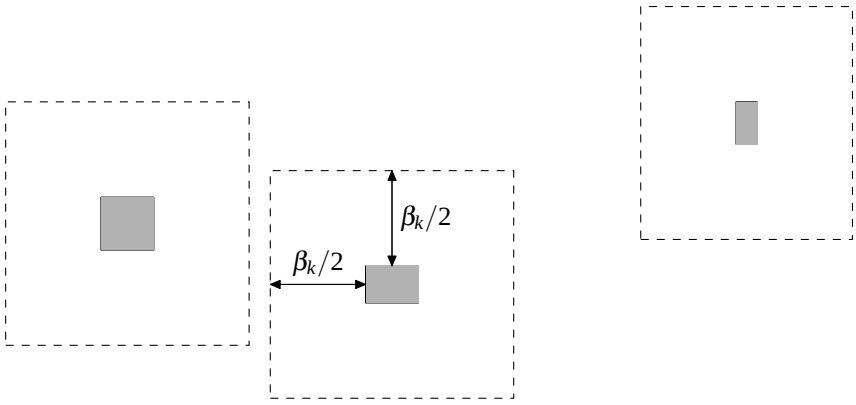

Figure 8: Rank k islands form a set of a small density. (In this picture each island is shown as a rectangle, which is not always the case.)

square the share of $E_{\leq}$ is bounded by $\sum_{k\leq K}(\alpha_k/\beta_k)^2$ up to negligible (as the size goes to infinity) boundary effects (we consider separately each $k \leq K$ and then sum over all $k \leq K$). The similar bound is valid for rank k islands with $k > K$, though the argument is different and a constant factor appears. Indeed, such an island I has β_k-neighborhood that does not contain the center point O. Therefore, any square S centered at O that intersects the island, contains also a significant part of its $\beta_k/2$-neighborhood N: the intersection of N and S contains at least $(\beta_k/2)^2$ elements.

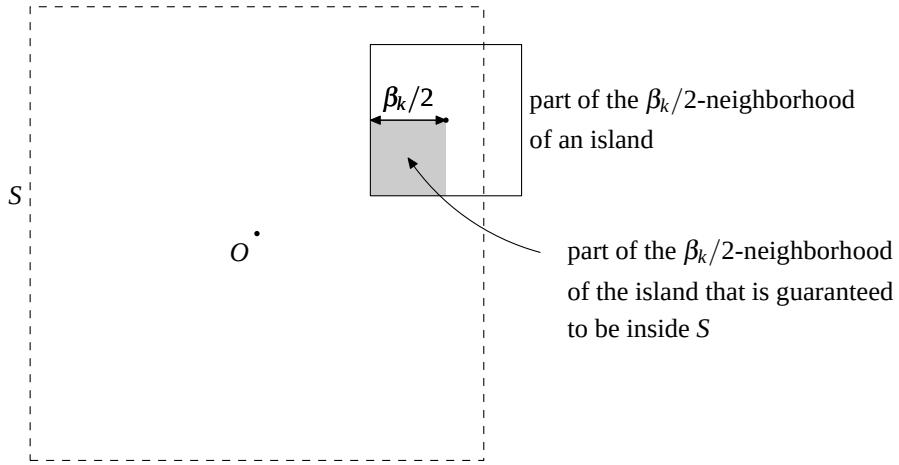

Figure 9: Together with a point in a rank k island, a square S contains at least $(\beta_k/2)^2$ points of its $(\beta_k/2)$-neighborhood.

Therefore, the share of $E_{>}$ in S is bounded by $4\sum_{k>K}(\alpha_k/\beta_k)^2$. □

Remark. It is easy to choose α_k and β_k satisfying the conditions of Lemma 3 and having arbitrarily small $\sum(\alpha_k/\beta_k)^2$ (take geometric sequences that grow fast enough). Therefore we get

the following well known result as a corollary of Lemmas 3 and 4: for every $\alpha > 0$ there exists $\varepsilon > 0$ such that with probability 1 a B_ε-random set has Besicovitch density less than α. (In fact, a much stronger result is well known: by the law of large numbers B_ε-random set has Besicovitch density ε with probability 1.)

In fact we will need a slightly more complicated version of Lemma 4. We are interested not only in the Besicovitch density of a sparse set E but also in the Besicovitch density of a larger set: the union of γ_k-neighborhoods of rank k islands in E. Here γ_k are some numbers (in most applications $\gamma_k = c\alpha_k$ for some constant c). The same argument gives the bound $4\sum((\alpha_k + 2\gamma_k)/\beta_k)^2$. Assuming that $\gamma_k \geq \alpha_k$, we can rewrite this bound as $O(\sum(\gamma_k/\beta_k)^2)$. So we arrive at the following statement:

Lemma 5. *Let E be a sparse set for a given family of α_k and β_k and let $\gamma_k \geq \alpha_k$ be some integers. Then the union of γ_k-neighborhoods of level k islands* (*over all k and all islands*) *has Besicovitch density $O(\sum(\gamma_k/\beta_k)^2)$.*

9.3 Islands as a tool in percolation theory

Let us show how some basic results of percolation theory can be proved using the island technique.

Theorem 10. *For some α_k and β_k satisfying Lemma 3 the complement of any sparse set E contains exactly one infinite connected component C; the complement of C has Besicovitch density $O(\alpha_k/\beta_k)^2$.*

Proof. Let $\gamma_k = 2\alpha_k$. (The choice of α_k and β_k will be discussed later.) For every k and for every rank k island fix a point in this island and consider the γ_k-neighborhood of this point. It is a square containing the entire island plus an additional security zone of width α_k and contained in the γ_k-neighborhood of the island.

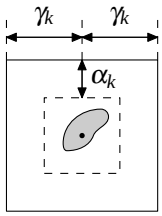

Figure 10: A point in a rank k island, its γ_k-neighborhood and the security zone of width α_k.

It is enough to prove the following three statements:

- *The union U of all these squares* (*for all ranks*) *contains the set E and has Besicovitch density $O(\sum(\alpha_k/\beta_k)^2)$.*
- *The complement of U is connected.*

- *There are no other infinite connected component in the complement of E.*

The first statement is a direct corollary of Lemma 5 above.

To prove the second statement consider two points x and y that lie outside U. We need to prove that x and y can be connected by a path that is entirely outside U. Let us connect x and y by some path (say, one of the shortest paths) and then push this path out of U. Consider squares of maximal rank that intersect this path. For each of them consider the first moment when the path gets into the square and the last moment when the path goes out, and connect these two points by a path outside the square:

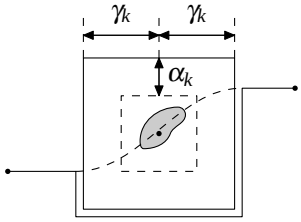

Figure 11: Pushing a path out of the square.

Let us assume that $\beta_k > 2\gamma_k$; then the new path is α_k-separated from rank k islands. Note also that the shift (the distance between the original path and the shifted one) does not exceed $3\gamma_k$.

Then we can do the same for islands of rank $k-1$ (pushing the path out of surrounding squares). Note that since the shift is bounded by $3\gamma_{k-1}$, we will not bump into islands of rank k assuming that $3\gamma_{k-1}$ is less than the width of the security zone, α_k.

Repeating this process for decreasing k, we finally get a path that connects x and y and goes entirely outside U. For this we need only that the total shift on the smaller levels, the sum $3\sum_{i<k}\gamma_i$ is less than α_k. (This is easy to achieve if α_k, β_k and γ_k are suitable geometric sequences.)

It remains to show that every infinite connected set intersects the complement of U. To show this, let us take a big circle centered at the origin and then push it out of U as described above. Since the center is outside β_k-neighborhoods of islands for large enough k, we may assume that the size of islands that intersect this circle are small compared to its radius (say, less than 1% of it; this can be guaranteed if the geometric sequences α_k, β_k and γ_k grow fast enough). Then after the change the circle will still encircle a large neighborhood of the origin, so any infinite connected component should cross such a circle. □

9.4 Bi-islands of errors

In the proof of our main result (Section 13) we need a more delicate version of the definition of islands. In fact we need such a definition that some counterpart of Lemma 3 could be applied even if the sequence $\log\beta_n$ grows much faster than 2^n (e.g., for $\beta_n = c^{(2.5)^n}$). In this section we define bi-islands (that generalize the notion of islands from Section 9.2) and prove bi-islands versions of Lemma 3, Lemma 4, and Lemma 5. The reader can safely skip this section for now and return here before reading Section 13.

Let $E \subset \mathbb{Z}^2$ be a set of points. As in Section 9.2, we call points in E *dirty*, and the other points are *clean*. Let $\beta \geq \alpha > 0$ be integers. A non-empty set $X \subset E$ is an (α,β)-*bi-island* in E if X can be covered by the union of some sets X_0, X_1 such that:

(1) the diameters of X_0 and X_1 do not exceed α;

(2) in the β-neighborhood of $X_0 \cup X_1$ there are no points from $E \setminus (X_0 \cup X_1)$.

(3) the distance between X_0 and X_1 does not exceed β.

(See Fig. 12.) In particular, an (α,β)-island is a special case of an (α,β)-bi-island (let X_1 be

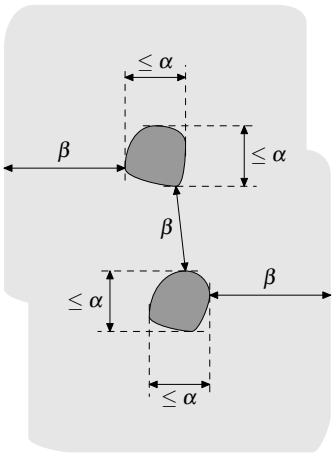

Figure 12: Bi-island is a union of two "islands" that are close to each other.

empty).

Note that one may split the same bi-island into X_0 and X_1 in different ways.

Obviously, every two different bi-islands are disjoint. Moreover, the distance between them is greater than β. The diameter of a bi-island is at most $(2\alpha+\beta)$.

Let (α_1,β_1), (α_2,β_2),... be a sequence of pairs of integers and $\alpha_i \leq \beta_i$ for all i. We define an iterative cleaning procedure for bi-islands. At the first step we find all (α_1,β_1)-bi-islands and remove all their elements from E (getting a smaller set E_1). Then we find in E_1 all (α_2,β_2)-bi-islands; removing them, we get $E_2 \subset E_1$, etc. Cleaning process is *successful* if every dirty point is removed at some stage.

Similarly to the case of islands, we say that a point $x \in \mathbb{Z}^2$ is *affected* during step i if x belongs to the β_i-neighborhood of one of bi-islands of rank i.

The set E is called *bi-sparse* (for a given sequence α_i,β_i) if the cleaning process defined above is successful, and, moreover, every point $x \in \mathbb{Z}^2$ is affected at finitely many steps only (that means that x is far from bi-islands of sufficiently large ranks).

We choose the values of α_i and β_i in such a way that for sufficiently small $\varepsilon > 0$ a B_ε-random set is bi-sparse with probability 1. The main achievement here is that the convergence condition is now weaker than in the corresponding statement for islands (Lemma 3):

Lemma 6. *Assume that*

$$12\sum_{k<n}\beta_k < \alpha_n \leq \beta_n \text{ for every } n, \text{ and } \quad \sum_i \frac{\log\beta_i}{3^i} < \infty.$$

Then for all sufficiently small $\varepsilon > 0$ a B_ε-random set is bi-sparse with probability 1.

Proof of Lemma 6 is very similar to the proof of Lemma 3. At first we estimate the probability of the event "x is not cleaned after n steps" for a given point x. If $x \in E_n$, then x belongs to E_{n-1} and is not cleaned during the nth step (when (α_n, β_n)-bi-islands in E_{n-1} are removed). Then $x \in E_{n-1}$. Moreover, we show that there exist *two other* points $x_1, x_2 \in E_{n-1}$ such that the three distances $d(x,x_1)$, $d(x,x_2)$, $d(x_1,x_2)$ are all greater than $\alpha_n/2$ but not greater than $2\beta_n + 2(\alpha_n/2) < 3\beta_n$.

Let X_0 be the $\alpha_n/2$-neighborhood of x in E. If X_0 were an island, it would be removed. Since it does not happen, there is a point x_1 outside X_0 but in the β_n-neighborhood of X_0.

Let X_1 be the $\alpha_n/2$-neighborhood of x_1 in E. Again X_0 and X_1 do not form a bi-island. Both sets X_0 and X_1 have diameter at most α_n, and the distance between them is at most β_n. So the only reason why they are not a bi-island is that there exists a point $x_2 \in E$ outside $X_0 \cup X_1$ but in the β_n-neighborhood of it. The points x_1, and x_2 have the required properties (the distances $d(x,x_1)$, $d(x,x_2)$, $d(x_1,x_2)$ are greater than $\alpha_n/2$ but not greater than $3\beta_n$).

To make the notation uniform, we denote x by x_0. Each of the points x_0, x_1, x_2 belongs to E_{n-1}. This means that each of them belongs to E_{n-2} together with a pair of other points (at the distance greater than $\alpha_{n-1}/2$ but not exceeding $3\beta_{n-1}$). In this way we get a 3-ary tree that "explains" why x belongs to E_n.

The distance between every two points among x_0, x_1, and x_2 in this tree is at least $\alpha_n/2$ while the diameters of the subtrees starting at x_0, x_1, and x_2 do not exceed $\sum_{i<n} 3\beta_i$. Thus, the Lemma's assumption guarantees that these subtrees cannot intersect and that all the leaves of the tree are different. The number of leaves in this 3-ary tree is 3^n, and they all belong to $E = E_0$. Every point appears in E independently from other points; hence, one such an "explanation tree" is valid with probability ε^{3^n}. It remains to count the number of all explanation trees for a given point x.

To specify x_1 and x_2 we need to specify horizontal and vertical distance between x_0 and x_1, x_2. These distances do not exceed $3\beta_n$, therefore we need about $4\log(6\beta_n)$ bits to specify them (including the sign bits). Then we need to specify the distances between x_{00} and x_{01}, x_{02} as well as the distances between x_{10} and x_{11}, x_{12}, and between x_{20} and x_{21}, x_{22}. This requires at most $12\log(6\beta_{n-1})$ bits. To specify the entire tree we therefore need

$$4\log(6\beta_n) + 12\log(6\beta_{n-1}) + 36\log(6\beta_{n-2}) + \ldots + 4\cdot 3^{n-1}\log(6\beta_1),$$

which is equal to $4\cdot 3^{n-1}(\log(6\beta_1) + \log(6\beta_2)/3 + \ldots)$. The series $\sum \log\beta_n/3^n$ converges by assumption; so, the total number of explanation trees for a given point (and given n) does not exceed $2^{O(3^n)}$. Hence, the probability for a given point x to be in E_n for a B_ε-random E does not exceed $\varepsilon^{3^n} 2^{O(3^n)}$, which tends to 0 as $n \to \infty$ (assuming that ε is small enough).

We conclude that the event "x is not cleaned" (for a given point x) has zero probability; hence, with probability 1 *all* points in $\mathbb{Z}^2$ are cleaned.

It remains to show that every point with probability 1 is affected by finitely many steps only. Indeed, if x is affected by step n, then some point in its β_n-neighborhood belongs to E_n, and the probability of this event is at most

$$O(\beta_n^2)\varepsilon^{3^n} 2^{O(3^n)} = 2^{2\log\beta_n + O(3^n) - \log(1/\varepsilon)3^n}.$$

From the convergence conditions we have $\log \beta_n = o(3^n)$, so the first term is negligible compared to others. The probability series converges (for small enough ε) and the Borel–Cantelli lemma gives the result. $\square$

By definition, a bi-sparse set is split into a union of bi-islands of different ranks. Such bi-islands occupy only a small part of the plane:

Lemma 7. *Let E be a bi-sparse set for a given family of α_k and β_k. Then Besicovitch density of E is $O(\sum(\alpha_k/\beta_k)^2)$.*

Proof of Lemma 7 repeats the proofs of Lemma 4. $\square$

Recalling Lemma 5, we may consider a sequence of numbers γ_k such that $\gamma_k \geq \alpha_k$. Then the Besicovitch density of the union of γ_k-neighborhoods of rank k bi-islands (for all k and for all islands) is bounded by $O(\sum(\gamma_k/\beta_k)^2)$.

However, this statement is not enough for us. In Section 13 we will need a kind of "closure" of γ_k-neighborhood of a bi-island:

Definition. *Let S be an k-level bi-island. We say that $(x,y) \in \mathbb{Z}^2$ belongs to the* extended γ-*neighborhood of S if there exist two points $(x,y'),(x,y'') \in \mathbb{Z}^2$ (with the same first coordinate) such that* $\operatorname{dist}(S,(x,y')) \leq \gamma$, $\operatorname{dist}(S,(x,y'')) \leq \gamma$, *and $y' \leq y \leq y''$, see Fig. 13.*

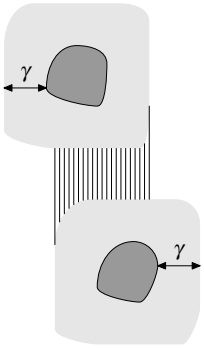

Figure 13: An extended neighborhood of a bi-island consists of the neighborhoods of its two parts and a zone between them.

The meaning of the last definition is quite simple: we take not only the points that are close to S but also those points that are placed somehow between the neighborhoods of S_0 and S_1.

Lemma 8. *Let E be a bi-sparse set for a given family of α_k and β_k satisfying the conditions of Lemma 6. Let γ_k be a sequence of numbers such that $\alpha_k < \gamma_k$, and the series $\sum(\gamma_k/\beta_k)$ converges. Then the Besicovitch density of the union of extended γ_k-neighborhoods of rank k bi-islands in E is bounded by $O(\sum(\gamma_k/\beta_k))$.*

Proof: Arguments are similar to the proof of Lemma 5. An extended γ_k-neighborhood of a k-level island can be covered by a rectangle of width $O(\gamma_k)$ and height $O(\beta_k + \gamma_k)$; so its area is

$O(\gamma_k\beta_k)$ (since $\gamma_k \le \beta_k$)). The distance between any two bi-islands of rank k is at least β_k. Hence, the fraction of *extended* γ_k-neighborhoods of islands is $O(\sum \gamma_k/\beta_k)$ (we get it instead of the bound $O(\sum(\gamma_k/\beta_k)^2)$, which holds for simple γ_n-neighborhoods). □

Lemmas 6–8 will be used in Section 13. (The arguments of Sections 10–12 do not refer to bi-islands.) These lemmas will be used for α_k, β_k such that $\log \alpha_k \sim q^k$ for $q > 2$, $\beta_k \sim \alpha_{k+1}$, and $\gamma_k = O(\alpha_k)$ or $\gamma_k = O(\alpha_k^2)$. Note that we cannot apply Lemmas 3 and 4 (about *islands*) for these parameters because $\log \beta_k$ grows faster than 2^k. So there we need to deal with bi-islands.

10 Robust tile sets

In this section we construct an aperiodic tile set where isolated defects can be healed.

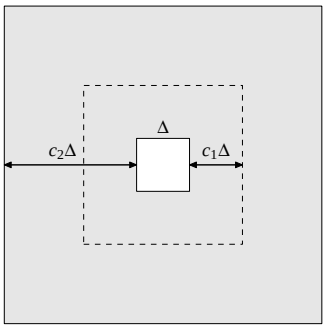

Figure 14: Patching holes

Let $c_1 < c_2$ be positive integers. We say that a tile set τ is (c_1, c_2)-*robust* if the following holds: For every Δ and for every τ-tiling U of the $c_2\Delta$-neighborhood of a square $\Delta \times \Delta$ excluding the square itself there exists a tiling V of the entire $c_2\Delta$-neighborhood of the square (including the square itself) that coincides with U outside of the $c_1\Delta$-neighborhood of the square (see Fig. 14).

Theorem 11. *There exists a self-similar tile set that is* (c_1, c_2)*-robust for some* c_1 *and* c_2.

Proof. For every tile set μ it is easy to construct a "robustified" version μ' of μ, i.e., a tile set μ' and a mapping $\delta\colon \mu' \to \mu$ such that: (a) δ-images of μ'-tilings are exactly μ-tilings; (b) μ' is "5-robust": every μ'-tiling of a 5×5 square minus 3×3 hole (see Fig. 15) can be uniquely extended to the tiling of the entire 5×5 square. (One can replace 5 by 4 in our argument using more careful estimates.)

Indeed, it is enough to keep in one μ'-tile the information about, say, 5×5 square in μ-tiling and use the colors on the borders to ensure that this information is consistent in neighbor tiles.

This robustification can be easily combined with the fixed-point construction. In this way we can get a 5-robust self-similar tile set τ if the zoom factor N (which is considered to be fixed in this argument) is large enough. Let us show that this set is also (c_1, c_2)-robust for some c_1 and c_2 (that depend on N, but N is fixed.)

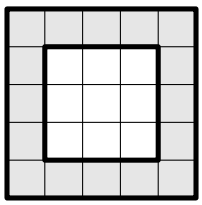

Figure 15: Filling a 3×3-hole

Indeed, assume that a tiling of a large enough neighborhood around an $\Delta \times \Delta$ hole is given. Denote by k the minimal integer such that $N^k \geq \Delta$ (so the k-level macro-tiles are greater than the hole under consideration). Note that the size of k-level macro-tiles is linear in Δ since $N^k \leq N \cdot \Delta$.

In the tiling around the hole, an $N \times N$ block structure is correct except for the N-neighborhood of the central $\Delta \times \Delta$ hole. Indeed, the colors encode coordinates, so in every connected tiled region coordinates are consistent. For similar reasons $N^2 \times N^2$-structure is correct except for the $N + N^2$-neighborhood of the hole, etc. Hence, for the chosen k we get a k-level structure that is correct except for (at most) $9 = 3 \times 3$ squares of level k, so we can delete everything in these squares and use 5-robustness to replace them with macro-tiles that correspond to replacement tiles.

To start this procedure (and fill the hole), we need a correct tiling only in the $O(N^k)$-neighborhood of the hole (technically, we need to have a correct tiling in the $(3N^k)$-neighborhood of the hole; as $3N^k \leq 3N\Delta$, we let $c_2 = 3N$). The correction procedure involves changes in another $O(N^k)$-neighborhood of the hole (technically, changes touch $(2N^k)$ of the hole; $2N^k \leq 2N\Delta$, so we let $c_1 = 2N$). $\square$

11 Robust tile sets with variable zoom factors

The construction from the previous Section works only for self-similar tilings with a fixed zoom factor. It is enough for simple applications, as we see below in Section 12. However, in the proof of our main result in Section 13 we need variable zoom factor. So here we develop some technique suitable for this case. This Section can be skipped now but it should be read before Section 13.

Now we explain how to get "robust" fixed-point tilings with variable zoom factors N_1, N_2,... As well as in the case of a fixed zoom factor, the idea is that k-level macro-tiles are "responsible" for healing holes of size comparable with this macro-tiles.

Let $\Delta_0 \leq \Delta_1 \leq \Delta_2 \leq \ldots$ be a sequence of integers. Let $c_1 < c_2$ be positive integers. We say that a tile set τ is (c_1, c_2)-*robust against holes of size* $\Delta_0, \Delta_1, \ldots$, if the following holds: For every n and for every τ-tiling U of $c_2\Delta_k$-neighborhood of a square $\Delta_k \times \Delta_k$ excluding the square itself there exists a tiling V of the entire $c_2\Delta_k$-neighborhood of the square (including the square itself) that coincides with U outside of the $c_1\Delta_k$-neighborhood of the square. The difference with the definition from Section 10 is that we take only values $\Delta \in \{\Delta_0, \Delta_1, \ldots\}$ instead of holes of arbitrary size.

Proposition 2. *Assume a sequence of zoom factors N_k grows not too fast and not too slow* (*it is enough to assume that $N_k \geq C \log k$ and $C \log N_{k+1} < N_k$ for a large enough C, cf. discussions in*

Section 5). *Then there exists a tile set with variable zoom factors N_k (k-level macro-tiles are of size $L_k = N_0 \cdot \ldots \cdot N_{k-1}$) that is (c_1, c_2)-robust (for some c_1 and c_2) against holes of size $L_0, L_1, \ldots$*

Proof. First, we apply the fixed-point construction from Section 5 and get a tile set which is "self-similar" with variable zoom factors $N_1, N_2, \ldots$ Denote by μ_k the family of k-level macro-tiles corresponding to this tile set.

Further we make a "robustified" version of this tile set. To this end we basically repeat the arguments from Section 10 (the proof of Theorem 11). The difference in the argument is that now we deal with variable zoom factors, and sizes of holes are taken from the sequence $L_0, L_1, \ldots$.

Denote by μ'_k the family of k-level macro-tiles for the new tiling. We need that there exists a mapping $\delta\colon \mu'_k \to \mu_k$ such that: (a) δ-images of μ'_k-tilings are exactly μ_k-tilings; (b) μ'_k is "5-robust": every μ'_k-tiling of a 5×5 square minus 3×3 hole (see again Fig. 15) can be uniquely extended to the tiling of the entire 5×5 square.

To get such a robustification, it is enough to keep in every μ'_k-macro-tile the information about 5×5 square in μ_k-tiling and use the colors on the borders to ensure that this information is coherent in neighbor macro-tiles.

As usual, this robustification can be combined with the fixed-point construction. We get a 5-robust macro-tiles for all levels of our construction. "Self-similarity" guarantees that the same property holds for macro-tiles of all ranks, which implies the required property of generalized robustness.

Indeed, assume that a tiling of a large enough neighborhood around a $\Delta \times \Delta$ hole is given, and $\Delta \le L_k$ for some k. In the tiling around the hole, an $(L_1 \times L_1)$ block structure is correct except only for the L_1-neighborhood of the hole. For similar reasons $(L_2 \times L_2)$-structure is correct except for the $(L_1 + L_2)$-neighborhood, etc. So we get a k-level structure that is correct except for (at most) $9 = 3 \times 3$ squares of size $L_k \times L_k$. Due to 5-robustness, this hole can be filled with k-level macro-tiles. Note that reconstruction of ground level tiles inside a high-level macro-tile is unique after we know its "conscious" memory (this memory is reconstructed from the conscious memory of the neighbor macro-tiles). [For the maximal complexity tile set (Section 7) it is not the case, and the absence of this property will become a problem in Section 13 where we robustify it. To solve this problem, we will need to use error correcting codes.]

To implement the patching procedure (and fill the hole) we need to have a correct tiling in the $O(L_k)$-neighborhood of the hole. The correction procedure involves changes in another $O(L_k)$-neighborhood of the hole. More technically, we need to have a correct tiling in the $(3L_k)$-neighborhood of a hole of size L_k, so we let $c_2 = 3$. Since the correction procedure involves changes in $2L_k$-neighborhood of the hole, we let $c_1 = 2$. $\square$

We can robustify tiling not only against holes, but against *pairs of holes*. To this end we slightly modify our definition of robustness. Let $\Delta_0 \le \Delta_1 \le \Delta_2 \le \ldots$ be an increasing sequence of integers, and $c_1 < c_2$ be positive integers. We say that a tile set τ is (c_1, c_2)-*robust against pairs of holes of size* $\Delta_0, \Delta_1, \ldots$, if the following holds: Let us have two sets $H_1, H_2 \subset \mathbb{Z}^2$, each of them of diameter at most Δ_k (for some $k > 0$). For every τ-tiling U of $c_2\Delta_k$-neighborhood of the union $(H_1 \cup H_2)$ excluding H_1 and H_2 themselves there exists a tiling V of the entire $c_2\Delta_k$-neighborhood of $(H_1 \cup H_2)$ (including H_1 and H_2 themselves) that coincides with U outside of the $c_1\Delta_k$-neighborhood of $(H_1 \cup H_2)$.

A robustification against pairs of holes can be done in the same way as the robustification against a single isolated hole above. Indeed, if these two holes are far apart from each other, we can "correct" them independently; if they are rather closed to each other, we correct them as one hole of (roughly) doubled size. So we can employ the same robustification technique as above; we need only to take a large enough "radius of multiplication" D (and use D-robustness instead of 5-robustness). So we get the following generalization of Proposition 2:

Proposition 3. *Assume a sequence of zoom factors N_k grows not too fast and not too slow* (*e.g., $N_k \geq C\log k$ and $C\log N_{k+1} < N_k$ for a large enough C*). *Then there exists a tile set with zoom factors N_k (k-level macro-tiles should be of size $L_k = N_0 \cdot \ldots \cdot N_{k-1}$) that is (c_1, c_2)-robust* (*for some c_1 and c_2*) *against pairs of holes of size $L_0, L_1, \ldots$ for some c_1 and c_2.*

Of course, similar propositions can be also proven for triples, quadruples and any other sets of holes of bounded cardinality. But in this paper we consider only pairs of holes; this is enough for our argument in Section 13.

12 Strongly aperiodic robust tile set

Now we are ready to apply islands technique to construct a robust strongly aperiodic tile set. We start with a formal definition of a tiling with errors (see motivation and discussion in Section 9.1).

Definition. *For a subset $E \subset \mathbb{Z}^2$ and a tile set τ we call by (τ, E)-tiling any mapping*

$$T \ : \ (\mathbb{Z}^2 \setminus E) \to \tau$$

such that for every two neighbor cells $x, y \in \mathbb{Z}^2 \setminus E$, tiles $T(x)$ and $T(y)$ satisfy the tiling rules (*colors on adjacent sides match*). *We may say that T is a τ-tiling of the plane with errors at points of E.*

Theorem 12. *There exists a tile set τ with the following properties:* (1) *τ-tilings of $\mathbb{Z}^2$ exist;* (2) *for all sufficiently small ε for almost every* (*with respect to B_ε*) *subset $E \subset \mathbb{Z}^2$ every (τ, E)-tiling is at least $1/10$ Besicovitch-apart from every periodic mapping $\mathbb{Z}^2 \to \tau$.*

Remark 1. Since the tiling contains holes, we need to specify how we treat the holes when defining Besicovitch distance. We do *not* count points in E as points where two mappings differ; this makes our statement stronger.

Remark 2. The constant $1/10$ is not optimal and can be replaced by any other constant $\alpha < 1$.

Proof. Consider a tile set τ such that (a) all τ-tilings are α-aperiodic for every $\alpha < 1/4$; (b) τ is (c_1, c_2)-robust for some c_1 and c_2. Such a tile set can be constructed by combining the arguments used for Theorem 11 (p. 38) and Theorem 4 (p. 14).

Our plan is to choose some α_k and β_k such that:

- the conditions of Lemma 3 (p. 30) are satisfied, and therefore a random error set with probability 1 is sparse with respect to these α_k and β_k;

- for every sparse set $E \subset \mathbb{Z}^2$ every (τ, E)-tiling can be iteratively corrected (by changing it in the neighborhoods of islands of all ranks) into a τ-tiling of the entire plane;
- the Besicovitch distance between the tilings before and after correction is small.

Then we conclude that the original (τ, E)-tiling is strongly aperiodic since the corrected tiling is strongly aperiodic and close to the original one.

To implement this plan, we use the following lemma that describes the error correction process.

Lemma 9. *Assume that a tile set τ is (c_1, c_2)-robust, $\beta_k > 4c_2\alpha_k$ for every k and a set $E \subset \mathbb{Z}^2$ is sparse (with parameters α_k, β_k). Then every (τ, E)-tiling can be transformed into a τ-tiling of the entire plane by changing it in the union of $(2c_1\alpha_k)$-neighborhoods of rank k islands (for all islands of all ranks).*

Proof. Note that $\beta_k/2$-neighborhoods of rank k islands are disjoint and large enough to perform the error correction of rank k islands, since $\beta_k > 4c_2\alpha_k$. The definition of a sparse set guarantees also that every point is changed only finitely many times (so the limit tiling is well defined) and that the limit tiling has no errors. □

The Besicovitch density of the changed part of a tiling can be estimated using Lemma 4: here $\gamma_k = 2c_1\alpha_k$ is proportional to α_k, so the Besicovitch distance between the original and corrected tilings (in Lemma 9) does not exceed $O(\sum_k(\alpha_k/\beta_k)^2)$. (Note that the constant in O-notation depends on c_1.)

It remains to chose α_k and β_k. We have to satisfy all the inequalities in Lemmas 3, Lemma 4 and Lemma 9. To satisfy Lemma 4 and Lemma 9, we may let $\beta_k = ck\alpha_k$ for large enough c. To satisfy Lemma 3, we may let $\alpha_{k+1} = 8(\beta_1 + \ldots + \beta_k) + 1$. Then α_k and β_k grow faster than any geometric sequence (like $k!$ multiplied by some exponent in k), but still $\log \beta_k$ is bounded by a polynomial in k and the series in Lemma 3 converges.

With these parameters (taking c large enough) we guarantee that Besicovitch distance between the original (τ, E)-tiling and the corrected τ-tiling does not exceed, say $1/100$. Since the corrected tiling is $1/5$-aperiodic and $1/10 + 2 \cdot (1/100) < 1/5$, we get the desired result. □

13 Robust tile set that enforces complex tilings

In this section we prove the main result of the paper. We construct a tile set that guarantees large Kolmogorov complexity of every tiling, and which is robust with respect to random errors.

Theorem 13. *There exists a tile set τ and constants $c_1, c_2 > 0$ with the following properties:*

(1) *a τ-tiling of $\mathbb{Z}^2$ exists;*

(2) *for every τ-tiling T of the plane, every $N \times N$-square of T has Kolmogorov complexity at least $c_1 N - C_2$;*

(3) *for all sufficiently small ε for almost every (with respect to the Bernoulli distribution B_ε) subset $E \subset \mathbb{Z}^2$ every (τ, E)-tiling is at most $1/10$ Besicovitch-apart from some τ-tiling of the entire plane $\mathbb{Z}^2$;*

(4) *for all sufficiently small ε for almost every B_ε-random subset $E \subset \mathbb{Z}^2$, for every (τ, E)-tiling T Kolmogorov complexity of centered squares of T of size $N \times N$ is $\Omega(N)$.*

The rest of the section is devoted to the proof of this theorem. It combines almost all technique developed in this paper: self-similar tile sets with variable zoom factors, embedding a sequence with Levin's property (i.e., with linear Kolmogorov complexity of all factors) into tilings, bi-sparse sets, incremental error correcting and robustness against doubled holes.

In this section the basic idea of incremental error correcting is applied in a slightly modified form. Here we cannot apply directly the technique of (c_1, c_2)-robustness from Section 10. Instead we use the idea of robustness against holes of some sequence of sizes $\Delta_0, \Delta_1, \Delta_2, \ldots$, as explained in Section 11. More precisely, we do it as follows: we split the set of random errors into bi-islands of different ranks. Then we eliminate them one by one, starting from lower ranks. When we correct an isolated bi-island of rank k, we need a pre-condition (similarly to the argument in Section 10): in a large enough neighborhood of this bi-island there is no other errors. Elimination of a k-level bi-island involves corrections in its *extended* $O(\Delta_k)$-neighborhood (all parameters are specified below).

13.1 The main difficulties and ways to get around them

We want to combine the construction from Section 7 with error correcting methods based on the idea of "islands" of errors. There are two main difficulties in this plan: fast growing zoom factors and gaps in vertical columns. Let us discuss these two problems in some detail.

The first problem is that our construction of tiling with high Kolmogorov complexity from Section 7 requires *variable zoom factors*. What is even worse, zoom factors N_k must increase very fast (their logarithms grow faster than 2^k). Hence, we cannot apply directly the technique of islands from Section 9.2 since it works only when $\sum \frac{\log \beta_k}{2^k} < \infty$ (here β_k is the parameter from the definition of islands; in our construction it must be of the same order as the size of k-level macro-tiles). To overcome this obstacle, we replace islands by bi-islands (the technique developed in Section 9.4).

The second problem is that now it is not enough to know the "conscious" memory of a macro-tile to reconstruct it. The missing information is the sequence of bits assigned to the vertical columns (each vertical column of tiles carries one bit of a high-complexity sequence ω). Random errors make gaps in vertical columns, so now the columns are split into parts, which a priori can carry different bits. To overcome this problem we organize additional information flows between macro-tiles to guarantee that each infinite vertical column carries in most of its tiles one and the same bit value.

13.2 General scheme

Here we explain the general ideas of our proof. First of all, we use macro-tiles with variable zoom factors $N_k = Q^{\lfloor 2.5^k \rfloor}$ for a large enough integer $Q > 0$. This means that every k-level macro-tile is an $(N_{k-1} \times N_{k-1})$-array of $(k-1)$-level macro-tiles. So the size (the number of columns and the number of rows) of a k-level macro-tile is $L_k = N_0 \cdot \ldots \cdot N_{k-1}$, and $L_k < N_k$. (The constant 2.5 in our construction can be replaced by any rational number between 2 and 3.)

To get tilings with high Kolmogorov complexity, we re-use the construction from Section 7 with the zoom factors defined above. Let us remind the idea of that construction (proof of The-

orem 8). In a correct tiling, in the ith column all tiles keep some bit ω_i, and we want that in the corresponding biinfinite sequence ω every N-bits substring has Kolmogorov complexity $\Omega(N)$. To enforce this property we organize some computation on macro-tiles of all levels. The crucial point of the construction is propagation of bits ω_i to the computation zones of macro-tiles of high levels. Let us remind the main points of this construction (following the argument from Section 7):

- We say that for each (infinite) column of tiles in a tiling there is an *assigned* bit ω_i, which is "known" to each tile in the column (in other words, there is a mapping that attributes to each tile the corresponding bit ω_i; vertically neighboring tiles must keep the same value of the bit).

- For a k-level macro-tile (of size $L_k \times L_k$) its *zone of responsibility* is the sequence of L_k bits ω_i assigned to all columns of this macro-tile. Vertically aligned macro-tiles of the same level have the same zone of responsibility.

- For *some* k-level macro-tiles M there is one *delegated bit*; this is a bit ω_i from the zone of responsibility of this macro-tile. This bit must be known to the "consciousness" of the macro-tile, i.e., it must be presented explicitly on the tape in the computation zone of this macro-tile. For technical reasons, we decide that the position of the delegated bit ω_i in the zone of responsibility of M (this position is an integer between 0 and $L_k - 1$) is equal to the position (vertical coordinate) of M in its father macro-tile, see Fig. 5. The father is a macro-tile of level $k+1$, which consists of $N_k \times N_k$ macro-tiles of level k (thus, the vertical coordinate of a k-level macro-tile in its father ranges over $0 \ldots N_k - 1$). In our settings, $N_k > L_{k-1}$. So, if a k-level macro-tiles M has vertical coordinate in its father grater than N_k, then M does not have a delegated bit.

- If a k-level macro-tile M has a delegated bit in its computation zone, it contains also a *group of bits to check* that starts at the delegated bit and has rather small length (say, $\log\log\log k$). If this group of bits goes out of the responsibility zone, we truncate it. The Turing machine simulated in the computation zone of M enumerates the forbidden strings of "too small Kolmogorov complexity" and verifies that the *checked group of bits* does not contain any of them. This process is bounded by time and space allocated to a k-level macro-tile.

The last item requires more comments. Technically, we fix constants $\alpha \in (0,1)$ and c and check that for every string x in zones of responsibility of all macro-tile $K(x) \geq \alpha|x| - c$. To check this property, a macro-tile enumerates all strings x of complexity less than $\alpha|x| - c$. This enumeration requires infinite time, though computations in each macro-tile are time-bounded. But this is not a problem since every such x is checked in macro-tiles of arbitrarily high levels (if x is covered by a macro-tile of rank k, then it is also covered by macro-tiles of all ranks greater than k). Thus, we guarantee the following property:

> *for every k-level macro-tile M ($k = 1, 2, \ldots$), and for every substring x of ω that is contained in M's zone of responsibility* (*its horizontal projection*) *it holds that $K(x) \geq \alpha|x| - c$.* $(*)$

Notice that that $K(x) \geq \alpha|x| - c$ holds only for strings x covered by some macro-tile (i.e., strings that belong to some macro-tile's zone of responsibility). In "degenerate" tilings there can exist an infinite vertical line that is a border line for macro-tiles of *all* levels (see Fig. 6). A string x that intersects this line is not covered by any macro-tile of any rank. Hence, $(*)$ does not guarantee for such a string x that its Kolmogorov complexity is greater than $\alpha|x| - c$. However, as we noticed in Section 7.4, the parts of x on both sides of the boundary are covered by some macro-tile. Hence, it follows from $(*)$ that $K(x) \geq \frac{\alpha}{2}|x| - O(1) = \Omega(|x|)$ for *all* factors x of the bi-infinite string ω.

Thus, we re-use the argument from Section 7, and it works OK if there is no errors. But when we introduce random errors, the old construction is broken. Indeed, vertical columns can be damaged by islands of errors. Now we need to do some efforts to enforce that copies of ω_i consciously kept by different macro-tiles are coherent (at least for macro-tiles that are not seriously damaged by local errors). To this end we will use some checksums, which guarantee that neighbor macro-tiles have coherent conscious and subconscious memory. We discuss it in next section.

To deal with random errors we use the technique of bi-islands (see Section 9.4). Our arguments work if diameters of k-level bi-islands are comparable with the size of k-level macro-tiles. Technically we set $\alpha_k = 13L_{k-1}$ and $\beta_k = L_k$. Remind that $N_k = Q^{\lfloor 2.5^k \rfloor}$ and $L_k = N_0 \cdot \ldots \cdot N_{k-1}$. Note that Lemmas 6, 7 can be used with these values of parameters. We will also employ Lemma 8 with $\gamma_k = O(\alpha_k)$.

13.3 The new construction of the tile set

We take the construction from Section 7 as a starting point and superimpose some new structures on k-level macro-tiles. We introduce these supplementary structures in several steps.

First step (introducing checksums): Every k-level macro-tile M (in a correct tiling) consists of an $N_{k-1} \times N_{k-1}$-array of $(k-1)$-level macro-tiles; each of these $(k-1)$-level macro-tiles may keep one delegated bit. We take in this 2-dimensional array of size $N_{k-1} \times N_{k-1}$ one horizontal row (bits assigned to N_{k-1} macro-tiles of level $k-1$). Denote the corresponding sequence of bits by $\eta_1, \ldots, \eta_{N_{k-1}}$. We introduce a sort of *erasure code* for this string of bits. In other words, we will calculate some checksums for this sequence. These checksums should be suitable to reconstruct all bits $\eta_1, \ldots, \eta_{N_{k-1}}$ *if at most D of these bits are erased* (i.e., if we know values η_i for only $N_{k-1} - D$ positions); here $D > 0$ is a constant (to be fixed later). We want the checksums to be easily computable. Here we use again the checksums of the Reed–Solomon code (discussed in Section 6).

Let us explain this solution in more detail. We take a finite field $\mathbb{F}_k$ of large enough size (the size of $\mathbb{F}_k$ should be greater than $N_{k-1} + D$). Now we calculate a polynomial of degree less than N_{k-1} that takes values $\eta_1, \ldots, \eta_{N_{k-1}}$ at some N_{k-1} points of the field. Then take as checksums the values of this polynomial at some other D points from $\mathbb{F}_k$ (all $(N_{k-1} + D)$ points of the field are fixed in advance). Two polynomials of degree less than N_{k-1} can coincide in at most $(N_{k-1} - 1)$ points. Hence, if D bits from the sequence $\eta_1, \ldots, \eta_{N_{k-1}}$ are erased, we can reconstruct them given the other (the non-erased) bits η_j and the checksums defined above.

These checksums contain $O(\log N_{k-1})$ bits of information. Further we discuss how to compute them.

Second step (calculating checksums): First of all, we explain how to compute the checksums, going from the left to the write along the sequence $\eta_1,\dots,\eta_{N_{k-1}}$. This can be done in a rather standard way as follows.

Let $\eta_1,\dots,\eta_{N_{k-1}}$ be the values of a polynomial $p(x)$ (of degree less than N_{k-1}) at points $x_1,\dots,x_{N_{k-1}}$. Assume we want to reconstruct all coefficients of this polynomial. We can do it by the following iterative procedure. For $i=1,\dots,N_{k-1}$ we calculate polynomials $p_i(x)$ and $q_i(x)$ (of degree $\le (i-1)$ and i respectively) such that

$$p_i(x_j)=\eta_j \text{ for } j=1,\dots,i$$

and

$$q_i(x)=(x-x_1)\dots(x-x_i)$$

It is easy to see that for each i polynomials p_{i+1} and q_{i+1} can be calculated from polynomials p_i, q_i, and values x_{i+1} and η_{i+1}.

If we do not need to know the resulting polynomial $p=p_{N_{k-1}}(x)$ but want to get only the value $p(a)$ at some particular point a, then we can do all these calculations modulo $(x-a)$. Thus, to obtain the value of $p(x)$ at D different points, we run in parallel D copies of this process. At each step of the calculation we need to keep in memory only $O(1)$ elements of $\mathbb{F}_k$, which is $O(\log N_{k-1})$ bits of temporary data (the multiplicative constant in this $O(\cdot)$-notation depends on the value of D).

These calculation can be simulated by a tiling. We embed the explained above procedure into the computation zones of $(k-1)$-level macro-tiles. The partial results of the calculation are transfered from one $(k-1)$-level macro-tile to another one, from the left to the right (in each row of length N_{k-1} in a k-level macro-tile). The final result (for each row) is kept in the conscious memory of the rightmost $(k-1)$-level macro-tile of the row.

To organize this calculations, we need to include into conscious memory of $(k-1)$-level macro-tiles additional $O(\log N_{k-1})$ bits and add the same number of bits to their macro-colors. This fits well our fixed-point construction since zoom factors N_k grow fast, and we have enough room in the computation zone.

Third step (consistency of checksums between macro-tiles): So far, every k-level macro-tile contains $O(N_{k-1}\log N_{k-1})$ bits of checksums, $O(\log N_{k-1})$ bits for every row. We want these checksums to be the same for every two vertical neighbor macro-tiles. It is inconvenient to keep the checksums for all rows only in the rightmost column (since it would create too much traffic in this column if we try to transmit the checksums to the neighbor macro-tiles of level k). So we propagate the checksums of the ith row in a k-level macro-tile M ($i=1,\dots,N_{k-1}$) along the entire ith row and along the entire ith column of M. In other words, these checksums must be "consciously" known to all $(k-1)$-level macro-tiles in the ith row and in the ith column of M. On Fig. 16 we show the area of propagation of checksums for two rows (the ith and the jth rows).

On the border of two neighbor k-level macro-tiles (one above another) we check that in each column $i=1,\dots,N_{k-1}$ all the corresponding checksums calculated in both macro-tiles coincide. This check is redundant if there is no errors in the tiling: the checksums are calculated from delegated bits (which come from the sequence of bits ω encoded into tiles of the ground level), so the corresponding values for all vertically aligned macro-tiles must be equal to each other. However, this redundancy is useful to resist errors, as we show in the sequel.

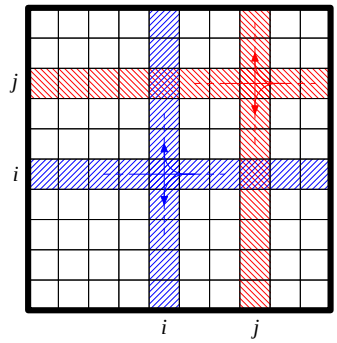

Figure 16: Propagation of checksums inside of a macro-tiles

Fourth step (robustification): The explained above features organized in every k-level macro-tile (bits delegation, calculating and propagating checksums, and all calculations simulated in the computation zone of a macro-tile) are simulated by means of bits kept in "conscious" memory (computation zone) of $(k-1)$-level macro-tiles. Now we fix some constant C and "robustify" this construction in the following sense: each $(k-1)$-level macro-tile M keeps in its consciousness not only "its own" data but also the bits previously assigned to $(k-1)$-level macro-tiles from its $(C \cdot L_{k-1})$-neighborhood (i.e., the $(2C+1) \times (2C+1)$ array of $(k-1)$-level macro-tiles centered at M). So, the content of the conscious memory of each macro-tile is multiplied by some constant factor. Neighbor macro-tiles check that the data in their consciousness are coherent.

We choose the constant C so that any k-level bi-island (that consist of two parts of size α_k) and even the $\gamma_k = O(\alpha_k)$-neighborhood of any k-level bi-island (we specify γ_k below) can involve only a small part of the (CL_{k-1})-neighborhood of any $(k-1)$-level macro-tile. (Note that we speak here about neighborhoods, not about *extended neighborhoods* of bi-islands defined in Section 9.4.)

This robustification allows to reconstruct the conscious memory of a k-level macro-tile and of its $(k-1)$-level sons when this macro-tile is damaged by one k-level bi-island (assuming there is no other errors).

The last remark (the number of bits in the conscious memory): The construction explained above requires that we put into the computation zones of $(k-1)$-level macro-tiles additional $\mathrm{poly}(\log N_{k-1})$ bits of data (the most substantial part is the data used for calculating the checksums). Again, this fits our fixed-point construction because $\mathrm{poly}(\log N_{k-1})$ is much less than N_{k-2}, so we have enough room to keep and process all these data.

The tile set τ is defined. Since there exist ω with Levin's property, it follows that τ-tiling exist, and every $N \times N$-square of such a tiling has Kolmogorov complexity $\Omega(N)$. Further we prove that this τ satisfies also statement (3) of Theorem 13.

13.4 Error correcting procedure

Denote by τ the tile set described in Section 13.3. Let $\varepsilon > 0$ be small enough. Lemma 12 says that B_ε-random set with probability 1 is bi-sparse. Now we assume that $E \subset \mathbb{Z}^2$ is a bi-sparse set (for

the chosen values of α_i and β_i), and T is a τ-tiling of $\mathbb{Z}^2 \setminus E$. Further we explain how to correct errors and convert T into a tiling T' of the entire plane (T' should be close to T).

We follow the usual strategy. The set E is bi-sparse, i.e., it can be represented as a union of isolated bi-islands of different ranks. We correct them one by one, starting from bi-islands of low rank. We need only to explain how to correct bi-island S of rank k assuming that it is well isolated, i.e., in the β_k-neighborhood of this bi-island there are no other (still non-corrected) errors. Eliminating of an isolated k-level bi-island will involve corrections only in the extended γ_k-neighborhood of this bi-island.

Let us recall that a k-level bi-island S is a union of two "clusters" S_0, S_1; diameters of both S_0 and S_1 are at most $\alpha_k = O(L_{k-1})$. Hence the clusters S_0 and S_1 touch only $O(1)$ macro-tiles of level $(k-1)$. The distance between S_0 and S_1 is at most β_k, and the β_k-neighborhood of S is free of other bi-islands of rank k and higher (so we can assume that the β_k-neighborhood of S is already cleaned of errors). Our correction procedure around S will involve only points in the extended γ_k-neighborhood of S, where $\gamma_k = 2\alpha_k$. Since the size of the extended neighborhood of a k-level bi-island is much less than $2L_k$, the correction procedure can involve points of only $O(1)$ macro-tiles of level k (maximum four, if it happens near the corner of a macro-tile).

Let M be one of k-level macro-tiles intersecting the extended γ_k-neighborhood of k-level bi-island S. Basically, we need to reconstruct all $(k-1)$-level macro-tiles in M destroyed by S. First we will reconstruct the conscious memory of all $(k-1)$-level macro-tiles in M. This is enough to get all bits of ω from the "zone of responsibility" of M. Then we will reconstruct in a consistent way all n-level macro-tiles inside M for all $n < k$.

Thus, we start with reconstructing the conscious memory of all $(k-1)$-level macro-tiles M' in M. First of all we remind that the conscious memory (the content of the computation zone) of every $(k-1)$-level macro-tile M' consists of several groups of bits (cf. the outline of the construction on p. 44):

[A] the binary representation of the number $(k-1)$ and coordinates of M' in the father macro-tile M (these coordinates are integers from the range $0 \ldots N_{k-1} - 1$);

[B] the bits used to simulate a Turing machine on the computation zone of M; the bits used to implement "wires" of M;

[C] the bit (from the sequence ω) delegated to M';

[D] the bit (from ω) delegated to M;

[E] bits used to calculate and communicate the checksums for the corresponding row of $(k-1)$-level macro-tiles in M;

[F] a "group of bits to check" from the zone of responsibility of M' (these bits are checked by the macro-tile: M' checks on its computation zone that this "group of bits to check" does not contain factors of low Kolmogorov complexity).

Bits of field [A] in a small isolated group of $(k-1)$-level macro-tiles are trivially reconstructed from the surrounding macro-tiles of the same level. Fields [B,C,D,E] can be reconstructed because

of the robustification on the level of $(k-1)$-level macro-tiles (we organized the robustification on the level of $(k-1)$-level macro-tiles in such a way that we are able to reconstruct these fields for any $C \times C$ group of missing or corrupt $(k-1)$-level macro-tiles). So far the correcting procedure goes absolutely in the same way as in Section 11.

To reconstruct fields [F] of $(k-1)$-level macro-tiles in M, we need to reconstruct all bits of ω from the zone of responsibility of M. We can extract these bits from neighbor k-level tiles above/below M (recall that bi-island S touches only $O(1)$ k-level macro-tiles, and there is a "healthy" zone of k-level macro-tiles around them). However, the problem remains since we are not sure that ω-bit above M, below M, and inside M are consistent. Now we show that this consistency is guaranteed by checksums.

Denote by M_u and M_d the k-level macro-tiles just above and below S that are free of errors, see Fig. 17 (our explanations refer to Fig. 17, where bi-island S touches only one k-level macro-tile; if S touches several k-level macro-tiles, substantially the same arguments work). It is enough to prove that the bits ω_i assigned to corresponding columns of M_u and in M_d are equal to each other.

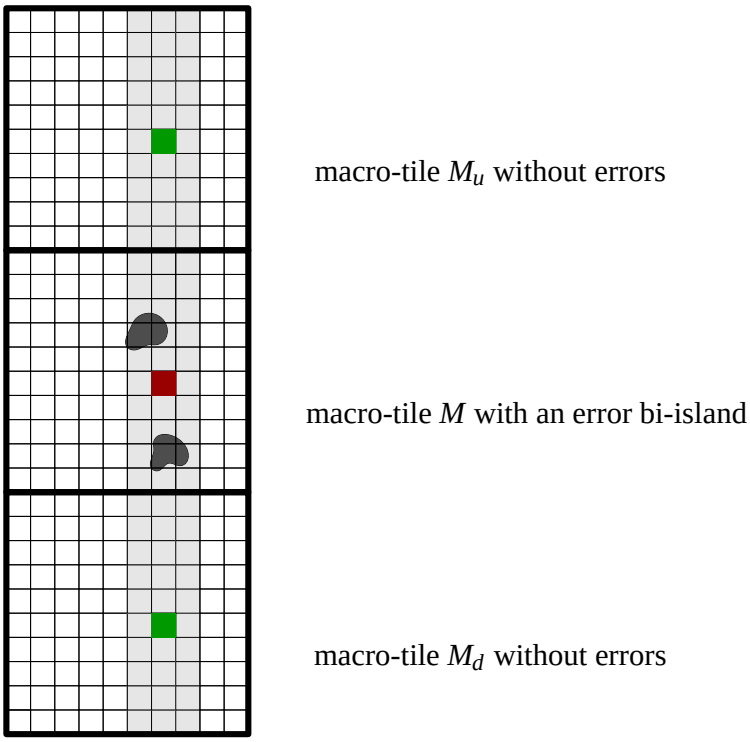

Figure 17: Bi-island of errors in a macro-tile

The macro-tiles M_u and M_d are error-free; so, the sequences of L_k bits ω_i corresponding to the vertical lines intersecting these k-level macro-tiles are well defined. Since there is no errors, the conscious information (including checksums) in all macro-tiles of all the levels inside M_u and M_d is consistent with these bit sequences. So, the L_k bits assigned to the vertical columns are correctly delegated to the corresponding $(k-1)$-level macro-tiles inside M_u and M_d. However, it is not

evident that the sequences of L_k bits embedded in M_u and M_d are equal to each other.

In fact, it is easy to see that bit sequences for M_u and M_d coincide with each other at most positions: they must be equal for all columns (from the range $0 \dots L_k - 1$) that do not intersect bi-island S (in non-damaged columns of tiles on the ground level the assigned bits ω_i correctly spread though macro-tiles M_u, M and M_d). Hence, the bits delegated to the corresponding $(k-1)$-level macro-tiles in M_u and M_d are equal to each other, except for only $(k-1)$-level macro-tiles in the "grey zone" on Fig. 17, which contains the $(k-1)$-level macro-tiles involved in the correction of S and all vertical stripes touching the involved sites (the width of this grey stripe is only $O(1)$ macro-tiles of level $(k-1)$). Hence, for $i = 0, \dots, (N_{k-1} - 1)$, in the ith rows of $(k-1)$-level macro-tiles in M_u and M_d, the sequences of delegated bits are equal to each other except possibly for only $O(1)$ bits (delegated to $(k-1)$-level macro-tiles in the "grey zone").

Robustness property guarantees that all checksums are correctly transmitted through M. Hence, checksums for corresponding rows in M_u and in M_d must be equal to each other.

Thus, for every two corresponding rows of $(k-1)$-level macro-tiles in M_u and in M_d we know that (a) all except for $O(1)$ delegated bits in the corresponding positions are equal to each other; and (b) the checksums are equal to each other. From the property of our erasure code it follows that in fact *all* delegated bits in these rows are equal to each other (the ith bit in M_u is equal to the ith bit in M_d). Therefore, all bits ω_i in M_u and M_d are the same (on the ground level). We can use these bits to reconstruct subconsciousness of M, and get a consistent tiling in M.

We are almost done: bi-island S is corrected, we reconstructed conscious memory for k-level macro-tile M and for all its $(k-1)$-level sons. Now we can reconstruct fields [F] in the damaged $(k-1)$-level macro-tiles inside M. It is trivial: we just take the corresponding bits ω_i from the zone of reponsibility (shared by M, M_u and M_d). It remains to explain why the checking does not fail for these groups of bits (i.e., $(k-1)$-level macro-tiles do not discover in these bit strings any factors of low Kolmogorov complexity). This is true because macro-tiles of levels $(k-1)$ (and also below $(k-1)$) inside M apply exactly all the same checks to exactly the same bits ω_i as the macro-tiles in the corresponding positions in M_u and M_d. Since there is no errors in M_u and M_d, these computations do not come to the contradiction.

Let us observe which tiles are involved in the error correcting process around bi-island S. In $(k-1)$-level macro-tiles outside the "grey zone" we change nothing. Moreover, not all the grey zone needs to be changed: only the part between two clusters of S (and their small neighborhoods) is affected. Indeed, in all tiles of M that are above S the assigned bits ω_i are the same as in the corresponding columns of M_u; in the tiles of M that are below S the assigned bits ω_i are the same as in the corresponding columns of M_u. Hence, there is no need to correct "subconscious memory" of $(k-1)$-level macro-tiles that are above or below S. Only the area between two clusters of S requires corrections. More precisely, the area involved in the correcting procedure is inside the extended neighborhood of S. (In fact, this argument is the motivation of our definition of extended neighborhood.)

Thus, we have proven that step-by-step correcting procedure eliminates all bi-islands of errors, and only extended γ_k-neighborhoods of k-level bi-islands are involved in this process. Now Theorem 13 (part 3) follows from Lemma 8. It remains only to prove part 4 of the theorem. We do it in next section.

13.5 Levin's property for ω embedded into a (τ, E)-tiling

It remains to prove part (4) of Theorem 13. In the previous section we proved that if the set of errors E is bi-sparce, then a (τ, E)-tiling T can be converted into a τ-tiling T' of the entire plane, and the difference between T and T' is covered by extended γ_k-neighbors of k-level bi-islands from E ($k = 0, 1, \ldots$). We want to show that also in the initial tiling T Kolmogorov complexity of centered squares of size $N \times N$ was $\Omega(N)$.

Fix a point O. Since E is bi-sparse, O is covered by β_k-neighborhoods of only finitely many bi-islands. Hence, for large enough Δ, the $\Delta \times \Delta$-square Q_Δ centered at O intersects extended γ_k-neighborhoods of k-level bi-islands only if $\beta_k < \Delta$. (If the extended γ_k-neighborhood of some bi-island intersects Q_Δ and $\beta_k \geq \Delta$, then $\beta_k - \gamma_k > \Delta/2$ and O is covered by β_k-neighborhood of this bi-island.) Therefore, to reconstruct T' in Q_Δ it is enough to correct there all bi-islands of bounded levels (such that $\beta_k < \Delta$).

To reconstruct T' in Q_Δ we need to know the original tiling T in Q_Δ and some neighborhood around it (i.e., in some centered $O(\Delta) \times O(\Delta)$-square $Q_{\Delta'}$, which is only constant time greater than Q_Δ). Indeed, given the tiling T restricted on $Q_{\Delta'}$, we can locally correct there bi-islands of level $1, 2, \ldots, k$ (such that $\beta_k < \Delta$) one by one. Correcting a bi-island of errors in $Q_{\Delta'}$ we obtain the same results as in error correcting procedure on the entire plane $\mathbb{Z}^2$ unless this bi-islands is too close to the boarder of $Q_{\Delta'}$ (and the local correction procedure should involve information outside $Q_{\Delta'}$). Thus, we can reconstruct T'-tiling not in all $Q_{\Delta'}$ but in points that are far enough from the boarder of this square. If $\Delta' = c\Delta$ for large enough c, then $Q_{\Delta'}$ provides enough information to reconstruct T' in Q_Δ.

We know that Kolmogorov complexity of error-free tiling T' in Q_Δ is $\Omega(\Delta)$. Therefore, the Kolmogorov complexity of the original T-tililng in the greater square $Q_{\Delta'}$ is also $\Omega(\Delta)$. Since Δ' is only constant times greater than Δ, we get that Kolmogorov complexity of (τ, E)-tiling T restricted to the centered $(\Delta' \times \Delta')$-square is $\Omega(\Delta')$. $\square$

14 Acknowledgments

The results included in this paper were discussed with many colleagues, including Peter Gács, Leonid Levin, our LIF collaborators Emmanuel Jeandel, Gregory Lafitte, Nicolas Ollinger, and Michael Weiss. We are grateful to all of them and to the participants of seminars (Kolmogorov seminar at Moscow State University, IITP seminar, FRAC) and conferences (DLT 2008, ICALP 2009) where some of these results were presented.

References

[1] C. Allauzen, B. Durand. Appendix A: Tiling Problems. In: E. Börger, E. Grädel, Y. Gurevich, *The Classical Decision Problems*, Springer-Verlag, 1996.

[2] N. Aubrun, M. Sablik, Simulation of recursively enumerable subshifts by two-dimensional SFT and a generalization. Preprint available at the home page of M. Sab-

lik, http://www.latp.univ-mrs.fr/~sablik/article/SimulSRE.pdf (as of August 5, 2010).

[3] R. Berger, The Undecidability of the Domino Problem. *Mem. Amer. Math. Soc.*, **66**, 1966.

[4] E. R. Berlekamp, *Algebraic Coding Theory*, 1984, Laguna Hills, CA: Aegean Park Press.

[5] L. Bienvenu, A. Romashchenko, A. Shen. Sparse sets. *Journées Automates Cellulaires 2008* (*Uzès*), p. 18–28. MCCME Publishers, 2008. Available online: http://hal.archives-ouvertes.fr/docs/00/27/40/10/PDF/18-28.pdf

[6] K. Culik, An Aperiodic Set of 13 Wang Tiles, *Discrete Math.*, **160**, 245–251, 1996.

[7] B. Durand, L. Levin, A. Shen, Complex Tilings. *J. Symbolic Logic*, **73** (2), 593–613, 2008 (See also Proc. *33rd Ann. ACM Symp. Theory Computing*, 732–739, 2001, and www.arxiv.org/cs.CC/0107008 for an earlier version.)

[8] B. Durand, L. Levin, A. Shen, Local Rules and Global Order, or Aperiodic Tilings, *Math. Intelligencer*, **27**(1), 64–68, 2004.

[9] B. Durand, A. Romashchenko, On Stability of Computations by Cellular Automata, In Proc. *European Conf. Compl. Syst.*, Paris, 2005.

[10] B. Durand, A. Romashchenko, A. Shen, Fixed Point and Aperiodic Tilings, *Developments in Language Theory, 12th International Conference, DLT 2008, Kyoto, Japan, September 16–19, 2008. Proceedings*, Springer, Lecture Notes in Computer Science, volume 5257, 2008. ISBN: 978-3-540-85779-2. P. 276–288.

[11] B. Durand, A. Romashchenko, A. Shen, High Complexity Tilings with Sparse Errors, *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5–12, 2009, Proceedings, Part I.* (Lecture Notes in Computer Science, 5555.) Springer-Verlag, 2009, ISBN: 978-3-642-02926-4. P. 403–414.

[12] P. Gács, Reliable Cellular Automata with Self-Organization, In Proc. *38th Ann. Symp. Found. Comput. Sci.*, 90–97, 1997.

[13] P. Gács, Reliable Cellular Automata with Self-Organization, *J. Stat. Phys.*, **103** (1/2), 45–267, 2001.

[14] L. Gray, A Reader's Guide to Gács' Positive Rates Paper. *J. Stat. Phys.*, **103** (1/2), 1–44, 2001.

[15] B. Grünbaum, G.C. Shephard, *Tilings and Patterns*, W.H. Freeman and Company, New York, 1987.

[16] Yu. Gurevich, I. Koryakov, Remarks of Berger's paper on the domino problem, *Siberian Math. Journal*, **13**, 319–321, 1972.

[17] W. Hanf, Nonrecursive tilings of the plane, I, *Journal of Symbolic Logic*, **39**:283–285, 1974.

[18] M. Hochman, On the dynamic and recursive properties of multidimensional symbolic systems. *Inventiones mathematicae*, **176**, 131–167 (2009).

[19] J. Kari, A Small Aperiodic Set of Wang tiles, *Discrete Math.*, **160**, 259–264, 1996.

[20] H. Rogers, *The Theory of Recursive Functions and Effective Computability*, Cambridge, MIT Press, 1987.

[21] G. Lafitte and M. Weiss, *Computability of tilings*. In Proc. *International Federation for Information Processing, Fifth IFIP International Conference on Theoretical Computer Science* (IFIP-TCS 2008), vol. 273, pp. 187–201, 2008.

[22] L. Levin, Aperiodic Tilings: Breaking Translational Symmetry, *Computer J.*, **48**(6), 642–645, 2005. On-line: `http://www.arxiv.org/cs.DM/0409024`

[23] R. Lidl and H. Niederreiter, *Finite Fields*, 1997 (2nd ed.), Cambridge University Press.

[24] S. Mozes, Tilings, Substitution Systems and Dynamical Systems Generated by Them, *J. Analyse Math.*, **53**, 139–186, 1989.

[25] D. Myers. Nonrecursive tilings of the plane, II, *Journal of Symbolic Logic*, **39**:286–294, 1974.

[26] J. von Neumann, *Theory of Self-reproducing Automata*, Edited by A. Burks, University of Illinois Press, 1966.

[27] N. Ollinger, *Two-by-two Substitution Systems and the Undecidability of the Domino Problem*, In Proc. *Computability in Europe*, LNCS **5028**, 476–485, 2008.

[28] Yu. Pritykin, J. Ulyashkina, Aperiodicity Measure for Infinite Sequences, Computer Science — Theory and Applications. Fourth International Computer Science Symposium in Russia, CSR 2009, Novosibirsk, Russia, August 18–23, 2009. (Lecture Notes in Computer Science, v. 5675) Springer, 2009. ISBN: 978-3-642-03350-6. P. 274–285.

[29] R. Robinson, Undecidability and Nonperiodicity for Tilings of the Plane. *Inventiones Mathematicae*, **12**, 177–209, 1971.

[30] An. Rumyantsev, M. Ushakov, Forbidden Substrings, Kolmogorov Complexity and Almost Periodic Sequences, *STACS 2006 Proceedings*, Lecture Notes in Computer Science, Vol. 3884, Springer, 2006.

[31] A. Shen, *Algorithmic Information Theory and Kolmogorov Complexity*, Lecture notes of a course taught at the Uppsala University. Available as Technical Report at `http://www.it.uu.se/research/publications/reports/2000-034/`

[32] S.G. Simpson, Medvedev degrees of 2-dimensional subshifts of finite type, MPIM2007-67 preprint, 2007.

[33] M. Zaks, A.S. Pikovsky, J. Kurths, On the Correlation Dimension of the Spectral Measure for the Thue–Morse Sequence, *J. Stat. Phys.*, **88**(5/6), 1387–1392, 1997.

[34] Merriam-Webster's Medical Dictionary, 08.10.2010, `http://dictionary.reference.com`.