Domino Problem Under Horizontal Constraints

Nathalie Aubrun
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP, F-69342, LYON Cedex 07, France
https://perso.ens-lyon.fr/nathalie.aubrun/
nathalie.aubrun@ens-lyon.fr

Julien Esnay
Institut de Mathématiques de Toulouse, Université Paul Sabatier, 118 route de Narbonne, F-31062 Toulouse Cedex 9, France
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP, F-69342, LYON Cedex 07, France
julien.esnay@ens-lyon.fr

Mathieu Sablik
Institut de Mathématiques de Toulouse, Université Paul Sabatier, 118 route de Narbonne, F-31062 Toulouse Cedex 9, France
http://www.math.univ-toulouse.fr/~msablik/index.html
Mathieu.Sablik@math.univ-toulouse.fr

Abstract

The Domino Problem on $\mathbb{Z}^2$ asks if it is possible to tile the plane with a given set of Wang tiles; it is a classical decision problem which is known to be undecidable. The purpose of this article is to parameterize this problem to explore the frontier between decidability and undecidability. To do so we fix horizontal constraints H on the tiles and consider a new Domino Problem DP_H: given a vertical constraint, is it possible to tile the plane? We characterize the nearest-neighbor horizontal constraints where DP_H is decidable using graphs combinatorics.

2012 ACM Subject Classification Theory of computation → Models of computation; Mathematics of computing → Combinatoric problems

Keywords and phrases Dynamical Systems, Symbolic Dynamics, Subshifts, Wang tiles, Undecidability, Domino Problem, Combinatorics, Tilings, Subshifts of Finite Type

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding *Nathalie Aubrun*: Supported by the ANR project CoCoGro (ANR-16-CE40-0005)
Julien Esnay: Supported by the ANR project CoCoGro (ANR-16-CE40-0005) and the Labex CIMI (project "Computational Complexity of Dynamical Systems")
Mathieu Sablik: Supported by the Labex CIMI (project "Computational Complexity of Dynamical Systems")

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Introduction

The Domino Problem is a classical decision problem introduced by Wang [20] to study satisfaction procedures for some fragments of first-order logic. Considering a finite set of tiles that are squares with colored edges, called Wang tiles, we ask if it is possible to tile the plane with shifted copies of these tiles so that contiguous edges have the same color. This question is also central in symbolic dynamics. A $\mathbb{Z}^d$-subshift of finite type is a set of colorings of $\mathbb{Z}^d$ by a finite alphabet, called configurations, and a finite set of patterns that are forbidden to appear in those configurations. The set of tilings obtained when we tile the plane with a Wang tile set is an example of $\mathbb{Z}^2$-subshift of finite type. In this setting, the Domino Problem becomes: given a finite set of forbidden patterns, is the associated subshift of finite type empty?

On $\mathbb{Z}$-subshifts of finite type, the Domino Problem is easily shown to be decidable. On those over $\mathbb{Z}^2$, Wang conjectured the Domino Problem was decidable too, and produced an algorithm of decision relying on the hypothetical fact that all subshifts of finite type contained some periodic configuration. However, his claim was disproved by Berger [5] who proved that the Domino Problem over any $\mathbb{Z}^d, d \geq 2$ is algorithmically undecidable. The key of the proof is the existence of a $\mathbb{Z}^d$-subshift of finite type containing only aperiodic configurations, on which computations are implemented. In the decades that followed, many alternative proofs of this fact were provided [19, 17, 14].

The exact conditions to cross this frontier between decidability and undecidability have been intensively studied under different points of view during the last decade. To explore the difference of behavior between $\mathbb{Z}$ and $\mathbb{Z}^2$, the Domino Problem has been extended on discrete groups [7, 11, 6, 1, 2] and fractal structures [4] in order to determine which types of structures can implement computation. The frontier is also studied by restraining complexity (number of patterns of a given size) [15] or bounding the difference between numbers of colors and tiles [13]. Additional dynamical constraints are also considered, such as block gluing property [18, Lemma 3.1] or minimality [9].

In this article we propose a new approach. We fix horizontal constraints H which define a $\mathbb{Z}$-subshift of finite type and consider the decision problem DP_H that given vertical contraints V asks if it is possible to tile the plane. In other words, is the $\mathbb{Z}$-subshift of finite type defined by V compatible with the one defined by H? The purpose is to determine for which horizontal constraints H the decision problem DP_H is decidable.

This point of view has various motivations. First, one could eliminate horizontal constraints that necessarily yield periodic configurations to perform a more efficient computer proof of the smallest aperiodic Wang tile set (reached with 11 Wang tiles in [12]). Second, a classical result is that every effective $\mathbb{Z}$-subshift can be realized as an horizontal projection on $\mathbb{Z}$ of a $\mathbb{Z}^2$-subshift of finite type [10, 3, 8]. However, in these constructions, the subshift in the vertical direction is trivial. We can ask which other vertical subshift can be compatible with a given horizontal restriction: the result presented here is the first step to understand this. Finally, looking for the undecidable frontier under horizontal constraints helps us understand how to transfer information in order to implement computation.

Section 1 recalls the notions needed and formalizes the problem. Section 2 presents three categories of horizontal constraints that yield a decidable Domino Problem. Finally Section 3 proves that all others have an undecidable Domino Problem. This is shown by reduction: we prove that for any Wang tile set W, it is possible to add vertical constraints to these H so that the resulting subshift simulates W. These vertical constraints can control the horizontal transfer of information and implement any Wang tile set, and by extension computation. The proof faces combinatorial explosion into subcases and is based on a careful dichotomy.

1 Definitions

As a preliminary note, *any interval mentioned in this article will be an interval of integers*, unless explicitly stated otherwise.

1.1 Symbolic Dynamics

For a given finite set $\mathcal{A}$ called the alphabet, $\mathcal{A}^{\mathbb{Z}^d}$ is called the *d-dimensional full shift* over $\mathcal{A}$. Any $x \in \mathcal{A}^{\mathbb{Z}^d}$, called a *configuration*, can be seen as a function from $\mathbb{Z}^d$ to $\mathcal{A}$ and we write $x_{\vec{v}} := x(\vec{v})$. For any $\vec{k} \in \mathbb{Z}^d$ define the *shift map* $\sigma^{\vec{k}} : \mathcal{A}^{\mathbb{Z}^d} \to \mathcal{A}^{\mathbb{Z}^d}$ such that $\sigma^{\vec{k}}(x)_{\vec{v}} = x_{\vec{v}+\vec{k}}$. The product topology on $\mathcal{A}^{\mathbb{Z}^d}$ is generated by the metric $d(x,y) = 2^{-\inf\{|\vec{v}| | \vec{v} \in \mathbb{Z}^d, x_{\vec{v}} \neq y_{\vec{v}}\}}$, and makes $\mathcal{A}^{\mathbb{Z}^d}$ a compact space. A *pattern* p is a finite configuration $p \in \mathcal{A}^{P_p}$ where $P_p \subset \mathbb{Z}^d$ is finite. We say that a pattern $p \in \mathcal{A}^{P_p}$ *appears* in a configuration $x \in \mathcal{A}^{\mathbb{Z}^d}$ – or that x *contains* p – if there exists $\vec{k} \in \mathbb{Z}^d$ such that for every $\vec{\ell} \in P_p$, $\sigma^{\vec{k}}(x)_{\vec{\ell}} = p_{\vec{\ell}}$.

A $\mathbb{Z}^d$*-subshift* associated to a set of patterns $\mathcal{F}$, called set of *forbidden patterns*, is defined by

$$X_{\mathcal{F}} = \{x \in \mathcal{A}^{\mathbb{Z}^d} \mid \forall p \in \mathcal{F}, \forall \vec{k} \in \mathbb{Z}^d, \exists \vec{\ell} \in P_p, \sigma^{\vec{k}}(x)_{\vec{\ell}} \neq p_{\vec{\ell}}\}$$

that is, $X_{\mathcal{F}}$ is the set of all configurations that do not contain any pattern from $\mathcal{F}$. Note that there can be several sets of forbidden patterns defining the same subshift X. A subshift can equivalently be defined as a closed set under both the topology and the shift map. If $X = X_{\mathcal{F}}$ with $\mathcal{F}$ finite, then X is called a *Subshift of Finite Type*, SFT for short.

For $\mathbb{Z}$-subshifts, we will talk about *nearest-neighbor* SFTs if $\mathcal{F} \subset \mathcal{A}^{\{0,1\}}$. For $\mathbb{Z}^2$-subshifts, the most well-known are the *Wang shifts*, defined by a finite number of squared tiles with colored edges that must be placed matching colors called Wang tiles. Formally, these tiles are quadruplets of symbols (t_e, t_w, t_n, t_s). A Wang shift is described by a finite Wang tile set, and local rules $x(i,j)_e = x(i+1,j)_w$ and $x(i,j)_n = x(i,j+1)_s$ for all integers i, j.

Note that Wang shifts are enough to encode any $\mathbb{Z}^2$-SFT, albeit changing the underlying local rules and alphabet, so that a $\mathbb{Z}^2$-SFT is empty if and only if the corresponding Wang shift is.

1.2 One-dimensional SFTs as graphs

As explained in [16], a nearest-neighbor SFT $X = X_{\mathcal{F}} \subset \mathcal{A}^{\mathbb{Z}}$, with $\mathcal{F} \subset \mathcal{A}^2$, can be described as an oriented graph $\mathcal{G} = (\mathcal{V}, \vec{E})$ with $\mathcal{V} = \mathcal{A}$ and $(a,b) \in \vec{E} \Leftrightarrow ab \notin \mathcal{F}$. This graph, that encodes the allowed patterns, depends on $\mathcal{A}$ and $\mathcal{F}$, thus two different descriptions of the same SFT will yield different graphs.

However one graph can be canonically associated to a nearest neighbor SFT from any other description of said SFT. It is the only one obtained by iterated suppression of all vertices with no incoming edge or no outgoing edge, and so there only remain biinfinite paths in the graph, that correspond to proper tilings of the line. This graph, denoted by $\mathcal{G}(X)$, is called the *Rauzy graph* (of order 1) of X. Note that a Rauzy graph can be made of one or several *strongly connected components*, *SCC* for short. In case it has several SCCs it can also contain *transient vertices*, that are vertices with no path from themselves to themselves.

▶ **Example 1.** The subshifts $X = X_{\{10,20,21,11,30,31,32,33\}} \subset \{0,1,2,3\}^{\mathbb{Z}}$ and $Y = Y_{\{10,20,21,11\}} \subset \{0,1,2\}^{\mathbb{Z}}$ are the same SFT. They have the same Rauzy graph made of two SCCs $\{0\}$ and $\{2\}$, and one transient vertex 1 (vertex 3 has been deleted from $\mathcal{G}(X)$ else it would be of out-degree 0).

This technique that algorithmically associates a graph to an SFT will be of great use in Section 3, because it means that our proof can mostly focus on combinatorics over graphs to describe all nearest-neighbor one-dimensional SFTs.

1.3 The Domino Problem

Define $DP(\mathbb{Z}^d) = \{< H >| \ H \text{ is a nonempty } \mathbb{Z}^d\text{-SFT}\}$ where $< H >$ is the encoding of the SFT H using a finite alphabet and a finite set of forbidden patterns. $DP(\mathbb{Z}^d)$ is a language called the *Domino Problem* on $\mathbb{Z}^d$. As for any language, we can ask if it is algorithmically decidable, i.e. recognizable by a Turing Machine. Said otherwise, is it possible to find a Turing Machine that takes as input any *finite* set of patterns $\mathcal{F} \subset \mathcal{A}^{\mathbb{Z}^d}$ of rules and answers `YES` if $X_{\mathcal{F}}$ contains at least one configuration, and `NO` if it is empty?

It is widely known that $DP(\mathbb{Z})$ is decidable, because the problem can be reduced to the emptiness of nearest-neighbor $\mathbb{Z}$-SFTs, and finding a valid configuration in a nearest-neighbor SFT is equivalent, via what precedes, to finding a biinfinite path − hence a cycle − in a finite oriented graph. On the contrary, $DP(\mathbb{Z}^2)$ is undecidable [5, 19, 17, 14], and so is any $DP(\mathbb{Z}^d)$ for $d \geq 2$ by reduction to the undecidability of $DP(\mathbb{Z}^2)$.

1.4 Framework

▶ **Definition 2.** *Let $H, V \subset \mathcal{A}^{\mathbb{Z}}$ be subshifts. The two-dimensional subshift*

$$X_{H,V} := \{x \in \mathcal{A}^{\mathbb{Z}^2} \mid \forall i, j \in \mathbb{Z}, (x_{k,j})_{k\in\mathbb{Z}} \in H \text{ and } (x_{i,\ell})_{\ell\in\mathbb{Z}} \in V\}$$

is called the combined subshift *of H and V, and uses H as horizontal rules and V as vertical rules.*

▶ Remark. The projection of the horizontal configurations that appear in $X_{H,V}$ does not necessarily recover all of H; we may simply have a subset of it. Indeed, all configurations in H will not necessarily appear because some of them may not be legally extended vertically.

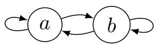
An easy instance of this is choosing $\mathcal{A} = \{0, 1\}$, H nearest-neighbor and forbidding 00 and 11, and V non-nearest-neighbor and forcing to alternate a 0 and two 1s: the resulting $X_{H,V}$ is empty, although neither H nor V are. In some sense, said H and V are incompatible.

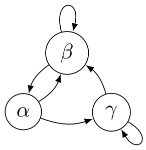
The previous remark motivates the main problem we will study in the rest of this article: understanding when two one-dimensional SFTs are compatible to build a two-dimensional SFT, and by extension where the frontier between decidability and undecidability lies. This question is notably reflected by the following adapted version of the Domino Problem:

▶ **Definition 3.** *Let $H \subset \mathcal{A}^{\mathbb{Z}}$ be an SFT. The* Domino Problem depending on H *is the language*

$$DP_H := \{< V >| \ V \subset \mathcal{A}^{\mathbb{Z}} \text{ is an SFT and } X_{H,V} \neq \emptyset\}.$$

▶ Remark. It is important to understand that this Domino Problem is defined for a given H, and its decidability depends on such a H we choose beforehand. Subshifts can be conjugate, this being defined as a continuous bijection that commutes with the shift maps. Although it allows to identify structurally identical subshifts from a dynamical point of view, these subshifts remain different in how they can encode information. Indeed, some SFT H_1 and H_2 may be conjugate with DP_{H_1} decidable but DP_{H_2} undecidable. Consider for instance the following Rauzy graphs and applications on finite words (extensible to biinfinite words):





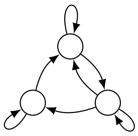
$$\phi : \begin{cases} aa \mapsto \gamma \\ ab \mapsto \beta \\ ba \mapsto \alpha \\ bb \mapsto \beta \end{cases} \quad \psi : \begin{cases} \alpha \mapsto a \\ \beta \mapsto b \\ \gamma \mapsto a \end{cases}$$

These graphs describe conjugate SFTs through these applications, with $\phi(x)_{i+1} = \phi(x_i x_{i+1})$. However, as we will see in Section 2, the first graph has decidable DP_H and the second has not.

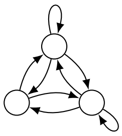
2 Main result: frontier between decidability and undecidability

With all the tools of Section 1 and the following reasoning, we are able to state Theorem 6, which contains the precise conditions under which DP_H is decidable for nearest-neighbor H. To understand where these conditions stem from, we study three examples:

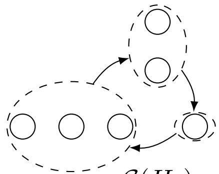
▶ **Example 4.** Consider the following Rauzy graphs:



$\mathcal{G}(H_1)$



$\mathcal{G}(H_2)$



$\mathcal{G}(H_3)$

Let us consider a "vertical" $\mathbb{Z}$-SFT V. As long as one can build a single column respecting the rules of V, this column can legally be juxtaposed with itself in $X_{H_1,V}$ since any element of H_1 can be horizontally juxtaposed with itself due to the self-loop at each vertex. Conversely, if $X_{H_1,V}$ contains a configuration, then in particular it contains a single column respecting the rules of V. Hence checking if $X_{H_1,V}$ is empty is tantamount to checking if V is empty. Since $DP(\mathbb{Z})$ is decidable, DP_{H_1} is easily decidable in this case.

The same reasoning can be applied to the two other cases: for H_2 any pair of columns, and for H_3 any triplet of columns, can be juxtaposed with itself. This finite number of columns makes the decidability of our Domino Problem at stake depend on the decidability of $DP(\mathbb{Z})$. The extensive proof of this fact is located below Theorem 6.

With these three examples, we briefly saw how these three kinds of graphs yielded a decidable DP_H. The rest of this article will produce a proper proof of this and, more importantly, will show that these three rather natural categories are in fact the only ones where decidability appears.

▶ **Definition 5.** *We say that an oriented graph $\mathcal{G} = (\mathcal{V}, \vec{E})$ verifies condition D (for "Decidable") if all its SCCs have a type in common among the following list. A SCC S can be of none, one or several of these types:*

- $\forall v \in S, (v,v) \in \vec{E}$) *(we say that S is of* reflexive type*);*
- $\forall v \neq w \in S$ *s.t.* $(v,w) \in \vec{E}, \bar{e} = (w,v) \in \vec{E}$ *(we say that S is of* symmetric type*; note that $S = \{v\}$ a single vertex with a loop is also symmetric);*
- $S = \bigsqcup V_i$ *such that* $\forall v \in V_i, [(v,w) \in \vec{E} \Leftrightarrow w \in V_{i+1}]$ *with i meant modulo the number of classes (we say that S is of* state-split cycle type *in reference to a term used in [16]; note that a partition with one unique class V_0 causes S to be a single vertex with self-loop).*

▶ **Theorem 6.** *Let H be a nearest-neighbor $\mathbb{Z}$-SFT.*

$$DP_H \text{ is decidable} \Leftrightarrow \mathcal{G}(H) \text{ verifies condition } D.$$

Proof. Proof of $\Leftarrow$: assume $\mathcal{G}(H)$ verifies condition D. Then its SCCs share a common type, be it reflexive, symmetric, or state-split cycle. For each of these three cases, we produce an algorithm that takes as input a $\mathbb{Z}$-SFT $V \subset \mathcal{A}^{\mathbb{Z}}$, and that returns `YES` if $X_{H,V}$ is nonempty, and `NO` otherwise.

Let M be the maximal size of forbidden patterns in $\mathcal{F}_V$ (since V is an SFT, such an integer exists).

- If $\mathcal{G}(H)$ has state-split cycle type SCCs: let L be the LCM of the number of V_is in each component. If there is no rectangle of size $L \times M(|\mathcal{A}|^{LM}+1)$ (width $\times$ height) respecting local rules of $X_{H,V}$ and containing no transient element, then answer `NO`. Indeed, any configuration in $X_{H,V}$ contains valid rectangles as large as we want that do not contain transient elements. If there is such a rectangle R, then by the pigeonhole principle it contains at least twice the same rectangle R' of size $L \times M$. To simplify the writing, we assume that the rectangle that repeats is the one of coordinates $[1, L] \times [1, M]$ inside R where $[1, L]$ and $[1, M]$ are intervals of integers, and that it can be found again with coordinates $[1, L] \times [k, k+M-1]$. Else, we simply truncate a part of R so that it becomes true.
  Define $P := R|_{[1,L]\times[1,k+M-1]}$. Since V has forbidden patterns of size at most M, and since R respects our local rules and begins and ends with R', P can be vertically juxtaposed with itself (overlapping on R').
  P can also be horizontally juxtaposed with itself (without overlap). Indeed, one line of P uses only elements of one SCC of H (since elements of two different SCCs cannot be juxtaposed horizontally, and we banned transient elements). Since L is a multiple of the length of all cycle classes, the first element in a given line can follow the last element in the same line. Hence all lines of P can be juxtaposed with themselves.
  As a conclusion, P is a valid patch that can tile $\mathbb{Z}^2$ periodically. Therefore, $X_{H,V}$ is nonempty; return `YES`.
- If $\mathcal{G}(H)$ has symmetric type SCCs the construction is similar, but this time build a rectangle R of size $2 \times M(|\mathcal{A}|^{2M}+1)$. Either we cannot find one and return `NO`; or we can find one and from it extract a patch that tiles the plane periodically and return `YES`.
- Finally, if $\mathcal{G}(H)$ has reflexive type SCCs, the construction is even simpler than before. Build a rectangle R of size $1 \times M(|\mathcal{A}|^{M}+1)$; the rest of the reasoning is identical.

Proof of $\Rightarrow$ is postponed to Section 3, and is done by contraposition. If $\mathcal{G}(H)$ does not verify condition D, then for any Wang shift W we can algorithmically build some $\mathbb{Z}$-SFT V_W such that X_{H,V_W} reproduces all configurations in W. If we were able to solve DP_H, then there would exist a Turing Machine $\mathcal{M}$ able to tell us if $X_{H,V}$ is empty for any $\mathbb{Z}$-SFT V. But then we could build a Turing Machine $\mathcal{N}$ taking as input any Wang shift W, building the corresponding V_W after the following construction, and by running $\mathcal{M}$, $\mathcal{N}$ would be able to tell us if X_{H,V_W} is empty or not. Then it could answer if W is empty or not; but determining the emptiness or nonemptiness of every Wang shift is equivalent to $DP(\mathbb{Z}^2)$ being decidable, which is false. Hence, since $DP(\mathbb{Z}^2)$ is undecidable, DP_H is too. ◀

3 Encoding a Wang shift under horizontal constraints

We begin this section by presenting the core idea of our algorithmic construction to prove the direct implication by contraposition. Then, we introduce the vertical patterns needed to achieve it in a generic case, said generic case being based on a set of conditions C. Finally, we show that most graphs that do not verify condition D do verify condition C, and those which don't only need a slight adaptation of our generic construction.

3.1 Core idea

We have a one-dimensional nearest-neighbor SFT $H \subset \mathcal{A}^{\mathbb{Z}}$ ("horizontal") that does not verify condition D, and we fix a Wang shift W with a set of N tiles $\tau = \{\tau_1, ..., \tau_N\}$.

The idea is to introduce a well-chosen one-dimensional SFT $V \subset \mathcal{A}^{\mathbb{Z}}$ ("vertical") depending on W so that $X_{H,V}$ encodes the full shift on N elements. Then, we refine V by adding conditions on forbidden patterns, thus encoding exactly the configurations in W. Such a construction is done with the use of two main parts, that we will obtain by some carefully chosen forbidden patterns in V.

First, there are parts of synchronization, also called *sync parts*, that give some rigidity to our tilings. They precise where the actual coding parts can be, which letters of the alphabet can be used and where in these coding parts, and they ensure that you cannot glue patches together in an unexpected way. They are the frame of our construction. Second comes the filling: the *coding parts*. A given coding part simply codes a number between 1 and N, possibly several times.

In Figure 1a (our first rough attempt to encode a full shift on an alphabet of size N), we suppose that our sync parts properly maintain this global structure. We notice that it offers an interesting opportunity to transmit information vertically. Since our coding parts are exactly aligned, once we have encoded the full shift over an alphabet of size N, it will suffice to add vertical conditions to our V to precise whether a coding part can be above another one.

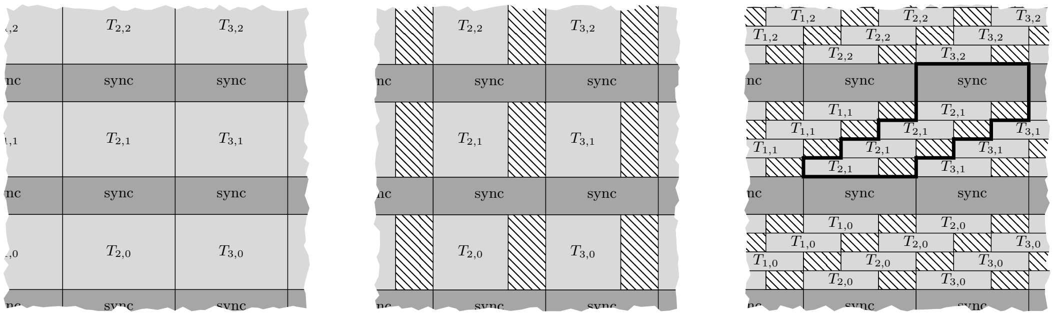
However, horizontally Figure 1a overlooks two problems:

- Since we must respect the horizontal conditions given by H, we cannot put any coding part next to any other one if we do not put some kind of buffer between the two;
- Even with this, we have no control on the horizontal transfer of information. The idea is to transmit this horizontal information vertically, since we can add vertical constraints.

We can fix the first problem by setting a buffer (see Figure 1b) between two coding parts, a column that contains no coding and which can be next to any coding part. Of course, we must ensure that this buffer cannot be anywhere in a configuration but obediently remains between two coding parts. The sync parts will be designed to handle this.

However, this does not solve our need of horizontal transmission of information. Hence a new idea: altering our coding parts so that they transmit information diagonally. We put several consecutive lines of them, shifted little by little, as illustrated in Figure 1c. That way, we can encode horizontal forbidden patterns vertically, because we can see vertically which coding part is on the right of the one we are considering. For instance, by looking vertically we can know that the encoding of $T_{1,1}$ is next to $T_{2,1}$ and above $T_{1,0}$, and thus restrict the content of these codings.

In what follows, we will build Figure 1c in details, although some technicalities will be needed to preserve the integrity of our sync parts and to ensure that the coding of a tile of W is well transmitted. This construction will indeed encode the full shift over τ, the tile set of W. Then, it can easily be refined by adding vertical rules so that the local rules of W are ensured. Consequently, our newly built $X_{H,V}$ will properly simulate all configurations of W, allowing us to perform the rest of the proof of Theorem 6.



(a) Basic depiction of sync parts and coding parts that represent tiles of W (not to scale; actually unrealizable).

(b) Same construction adding "buffers" between codings of tiles to be realizable.

(c) Improved construction: we encode vertically the horizontal restrictions between tiles of W. A tile of W (here $T_{2,1}$) is represented in bold.

Figure 1 Steps of the core idea to reach the generic construction. Not to scale: the sync part will be much bigger than the code part.

3.2 Generic construction

In this section, we describe a set of conditions on a nearest-neighbor SFT H that allows to build formally what we described informally in Section 3.1. In all that follows, *we denote elements of cycles with an index that is written modulo the length of the corresponding cycle.*

▶ **Definition 7.** *Let C^1 and C^2 be two cycles in an oriented graph $\mathcal{G}$, with elements denoted respectively c_i^1 and c_j^2. Let $M := \operatorname{lcm}(|C^1|, |C^2|)$.*

We say that the cycles C^1 and C^2 contain a good pair *if there is a pair (i, j) and an integer $1 < l < M-1$ such that $c_i^1 \neq c_j^2, c_{i+1}^1 \neq c_{j+1}^2, \dots, c_{i+l}^1 \neq c_{j+l}^2$ and $c_{i+(l+1)}^1 = c_{j+(l+1)}^2, \dots, c_{i+(M-1)}^1 = c_{j+(M-1)}^2$.*

▶ **Definition 8.** *Let $H \subset \mathcal{A}^{\mathbb{Z}}$ be a one-dimensional nearest-neighbor SFT. We say that H* verifies condition C *if $\mathcal{G}(H) = (\mathcal{V}, \vec{E})$ contains two cycles C^1 and C^2, of elements denoted respectively c_i^1 and c_j^2, with the following properties:*

- **(i)** *C^1 and C^2 contain a good pair;*
- **(ii)** *(there is no uniform shortcut neither in C^1 nor in C^2) There does not exist any $k \in \{0, 2, ..., |C^1|-1\}$ such that for any $c_i^1 \in C^1, (c_i^1, c_{i+k}^1) \in \vec{E}$; and there does not exist any $k \in \{0, 2, ..., |C^2|-1\}$ such that for any $c_j^2 \in C^2, (c_j^2, c_{j+k}^2) \in \vec{E}$;*
- **(iii)** *(there is no cross-bridge between C^1 and C^2) There are no $i \in \{0, ..., |C^1|-1\}$ and $j \in \{0, ..., |C^2|-1\}$ with $c_i^1 \neq c_j^2$ and $c_{i+1}^1 \neq c_{j+1}^2$ such that $(c_i^1, c_{j+1}^2) \in \vec{E}$ and $(c_j^2, c_{i+1}^1) \in \vec{E}$;*
- **(iv)** *(there cannot be both an attractive vertex and a repulsive vertex for C^1) Either there is no $r \in C^1 \cup C^2$ such that for all $c \in C^1, (c, r) \in \vec{E}$, or there is no $r \in C^1 \cup C^2$ such that for all $c \in C^1, (r, c) \in \vec{E}$.*

▶ **Proposition 9.** *If $\mathcal{G}(H)$ verifies condition C, then DP_H is undecidable.*

The rest of the subsection is devoted to proving this result.

Let H with $\mathcal{G}(H)$ verifying condition C. We focus on encoding a full shift on an alphabet τ of cardinality N. Then, the possibility to add vertical rules will allow us to encode any Wang shift W using this alphabet, that is, to simulate the configurations of W as described in Section 3.1.

For the rest of the construction, we will name $M := \mathrm{lcm}(|C^1|, |C^2|)$ and $K := 2|C^1| + |C^2| + 3$. We suppose that $N \geq 2$ and do not focus on the trivial instance of W being a monotile Wang shift.

We refer to Figure 2 in all that follows. We use the term *slice* as a truncation of a column: it is a part of width 1 and of finite height. We use the following more specific denominations for the various scales of our construction:

- A macro-slice is of height KMN. Any column is merely made of a succession of some specific macro-slices called ordered macro-slices (see below).
- A meso-slice is of height MN. An ordered macro-slice is made of various meso-slices that ensure it carries information and is correctly aligned with neighboring ordered macro-slices (of neighboring columns).
- A micro-slice is of height N. This subdivision is used inside code meso-slices (see below).

Although any scale of slice could denote any truncation of column of the right size, we will only focus on specific slices that are meaningful because of what they contain, so that we can assemble them precisely. They are:

- A (i, j) k-coding micro-slice is a micro-slice composed of $N - 1$ symbols c_i^1 and one symbol c_j^2 at position k. It encodes the kth tile of alphabet τ.
- A (i_0, j_0)-code meso-slice is made of M successive coding micro-slices starting with a (i_0, j_0) coding micro-slice (that can encode anything). We add the vertical constraint that if $c_i^1 \neq c_j^2$, then the (i, j) k-coding micro-slice is vertically followed by the $(i + 1, j + 1)$ k-coding micro-slice. If $c_i^1 = c_j^2$, the (i, j) k-coding micro-slice can be vertically followed by any $(i+1, j+1)$ l-coding micro-slice. Note that there can be at most one such rupture in the coding since C^1 and C^2 contain a good pair; k is then called the *main-coded tile*, and l the *side-coded tile*.
- A i-border meso-slice is made of $\frac{M}{|C^1|}N$ times the vertical repetition of elements of the cycle C^1, starting at c_i^1.
- A c_i^1 meso-slice is made of NM times the vertical repetition of element c_i^1. Same for a c_j^2 meso-slice.
- The succession of a c_i^1 meso-slice, then a c_{i+1}^1 meso-slice, ..., then a c_{i-1}^1 meso-slice is called a i C^1-slice (of height $MN|C^1|$). Similarly, we define a j C^2-slice (of height $MN|C^2|$).
- Finally, a (i, j)-ordered macro-slice is the succession of a i-border meso-slice, a i C^1-slice, a second i C^1-slice, a j C^2-slice, a i-border meso-slice, and finally a (i, j)-code meso-slice.

Now, the patterns we authorize in V are exactly the cyclic permutations of $(i_0 + k, j_0 + k)$-ordered macro-slices with some good pair (i, j) and $k \in \{0, \ldots, M - 1\}$. We prove below that it is enough for our resulting $X_{H,V}$ to simulate a full shift on τ.

We say that two legally adjacent columns are *aligned* if they are subdivided into ordered macro-slices exactly on the same lines. We say that two adjacent and aligned columns are *synchronized* if any (i, j)-ordered macro-slice of the first one is followed by a $(i + 1, j + 1)$-ordered macro-slice in the second one.

▶ **Proposition 10.** *In this construction, two legally adjacent columns are aligned up to a vertical translation of size at most $2|C^1| - 1$ of one of the columns.*

Proof. If two columns, call them K_1 and K_2, can be legally juxtaposed such that they are not aligned even when vertically shifted by $2|C^1| - 1$ elements, it means that one of the border meso-slices of K_1 has at least $2|C^1|$ vertically consecutive elements that are horizontally followed by something that is not a border meso-slice in K_2 (see Figure 3). Since $2|C^1| < MN$, at least $|C^1|$ successive elements among the ones of the border meso-slice are horizontally followed by elements that are part of the same meso-slice. If this is a code

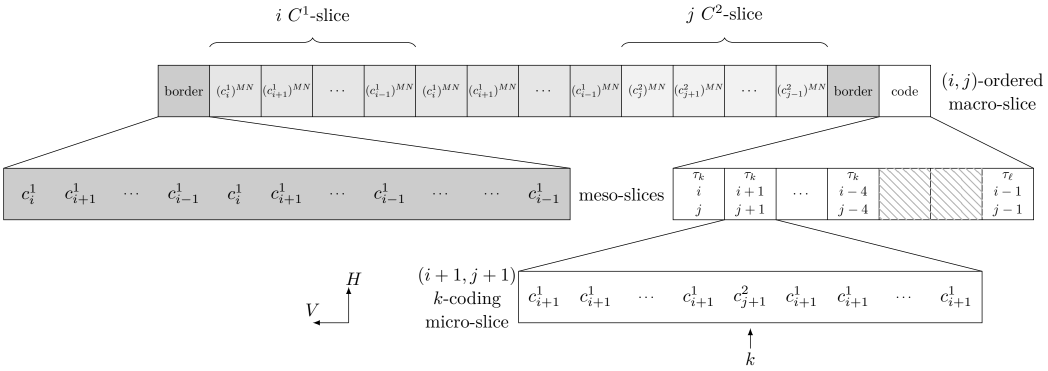


Figure 2 Columns allowed, for (i, j) in the orbit of a good pair. Here $c^1_{i-3} = c^2_{j-3}$ and $c^1_{i-2} = c^2_{j-2}$, forming buffers in the code meso-tile.

meso-slice, simply consider the other border meso-slice of K_1 (the first you can find, above or below, before repeating the pattern cyclically): this one must be in contact with a c^1_i or c^2_j meso-slice instead. Either way, we obtain that a border meso-slice has at least $|C^1|$ successive elements that are horizontally followed by some t meso-slice made of a single element. Hence if we suppose that juxtaposing K_1 and K_2 this way is legal, it means that in H all the elements of C^1 lead to t, i.e t is an attractive vertex. Either this is forbidden, or the "reverse" reasoning where we focus on the borders of K_2 proves that there is also an element p used in a C^j slice of K_1 that leads to every element of C^1; that is, a repulsive vertex. We forbade any graph that had both, hence we reach a contradiction here. We obtain the proposition we announced. ◀

▶ **Proposition 11.** *In this construction, two legally adjacent columns are always aligned and synchronized.*

Proof. Proposition 10 states that two adjacent columns K_1 and K_2 are always, in some sense, approximately aligned (up to a vertical translation of size at most $2|C^1|$). If the two columns are indeed slightly shifted, then any meso-slice of the C^1 slice of K_1 (consisting only of the repetition of some c^1_i) is horizontally followed by two different meso-slices in K_2. Being different, at least one of them is not c^1_{i+1} but some $c^1_{i+k}, k \in \{2, ..., |C^1|\}$. This is true with the same k for all values of i because all meso-slices representing c^1_i are repeated twice so that there is no "border effect". We obtain something that contradicts our assumption that C^1 has no uniform shortcut. Hence there is no vertical shift at all between two consecutive columns. Thus our construction ensures that two adjacent columns are always aligned.

It is easy to see that a meso-slice made only of c^1_i in column K_1 is horizontally followed, because the columns are aligned, by a meso-slice made only of c^1_{i+k} in column K_2. This k is

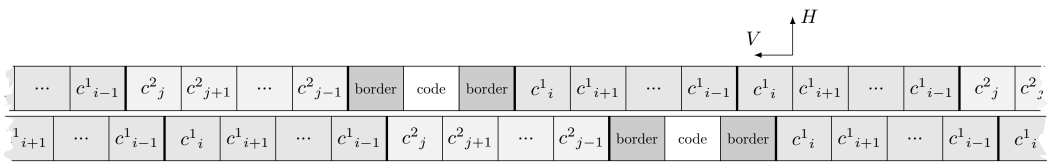


Figure 3 Faulty alignment of adjacent columns (represented as lines).

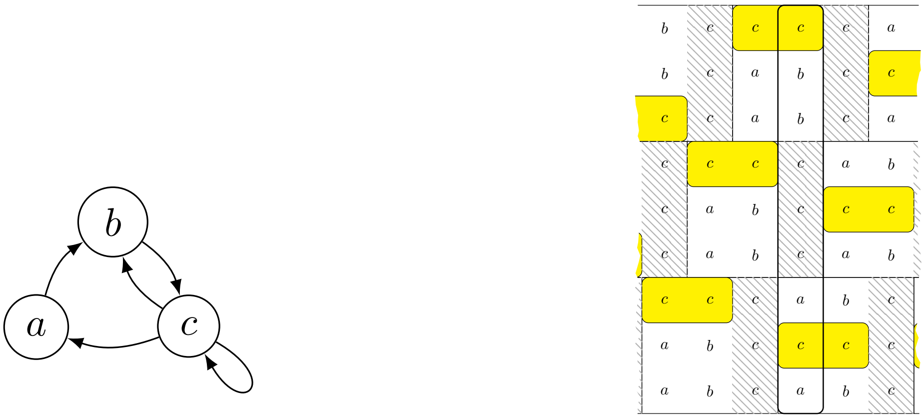


Figure 4 Some Rauzy graph and several associated code meso-slices for $|\tau| = 3$. Here are horizontally successively encoded τ_3, τ_1, τ_2 and τ_2, the number being indicated by the location of the line of c's.

once again independent of the i because inside a macro-slice, meso-slices respect the order of cycle C^1. But because C^1 has no uniform shortcut, we must have $k = 1$. The reasoning is the same for the C^2 slice, and we use the fact that C^2 has no shortcut either. Hence our columns are synchronized. ◀

With these properties, we have ensured that our structure is rigid: our ordered macro-slices are aligned just as we expected. The last fact to check is the transmission of information between horizontally aligned code meso-slices (since, by the structure of a code meso-slice, the vertical transmission is guaranteed).

We have to ensure that, every M horizontally aligned coding micro-slices, we have a succession of buffer ones (a (i, j)-coding micro-slice that does not encode information because $c_i^1 = c_j^2$) then of non-buffer ones. Furthermore, we ask that all the buffers follow each other so that we get some distinct "coding zone" and "buffer zone". This is actually simply deduced from the fact that we only authorized as ordered macro-slices the ones based on the orbit of a good pair (see Definition 7).

Now, in which situation can there be a problem of horizontal transmission of the encoded tile between two micro-slices? Suppose we have a (i, j) micro-slice then a $(i + 1, j + 1)$ micro-slice. A problem of transmission would mean that c_i^1 can be followed by c_{j+1}^2, and c_j^2 by c_{i+1}^1. A problem of transmission also assumes that we transmit something, hence we don't consider buffer micro-slices: necessarily $c_i^1 \neq c_j^2$ and $c_{i+1}^1 \neq c_{j+1}^2$. Then we would contradict the assumption "no cross-bridge" (which was assumed precisely to prevent this case).

As a consequence of all this, every M micro-slices starting with a buffer micro-slice, horizontally successive coding micro-slices encode exactly one element of τ, since there is one single coding zone and the coded tile is correctly transmitted.

In the end, we proved that if we were able to find such C^1 and C^2 complying with condition C, they would be enough to build the construction we desire: a full shift on N elements. Then, to encode only configurations that are valid in W, we forbid the following additional vertical patterns:

- the code meso-slices that would contain both the main-coded kth tile and the side-coded lth tile if tile k cannot be horizontally followed by tile l in W;
- and the vertical succession of two ordered macro-slices that would contain code meso-slices with two main-coded tiles that cannot be vertically successive in W.

With this, we proved Proposition 9.

3.3 Proof of Theorem 6 for one strongly connected component

We suppose that $H \subset \mathcal{A}^{\mathbb{Z}}$ is a one-dimensional nearest-neighbor SFT such that its Rauzy graph does not verify condition D and is made of only one SCC.

Note that $\mathcal{G}(H)$, since it does not verify condition D, contains at least one loopless vertex, and one unidirectional edge.

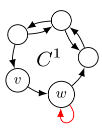
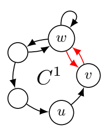
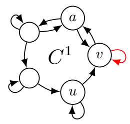
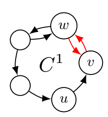
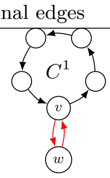
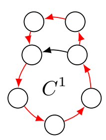
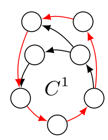
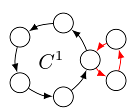
The idea is to divide the possible graphs into various cases, see Table 1. This way, one has a standard procedure to find convenient C^1 and C^2 inside any graph to perform the generic construction. Of course, for some specific cases, we won't meet condition C even if H does not verify condition D. However, we will punctually adapt the generic construction to these specificities.

The division into cases is presented in a disjunctive fashion:

Is there a loop on a vertex?

- If YES: Is there a unidirectional edge $(v, w) \in \vec{E}$ so that v is loopless and w has a loop (or the reverse, which is similar)?
  - If YES: This is Case 1.1. We can find C^1 and C^2 that check condition C with the exception of the possible presence of both an attractive and a repulsive vertices. However, Proposition 10 is verified anyway because choosing the smallest possible cycle containing such v and w, v has in-degree 1, a property that allows for an easy synchronization.
  - If NO: Do unidirectional edges have loopless vertices?
    - If YES: This is Case 1.2. We can find C^1 and C^2 that check condition C, possibly by reducing to a situation encountered in case 2.2.
    - If NO: This is Case 1.3. This one generates some exceptional graphs with 4 or 5 vertices that do not check condition C and must be treated separately. However, technical considerations prove that our generic construction still works.
- If NO: Is there a bidirectional edge?
  - If YES: Is there a cycle of size at least 3 that contains a bidirectional edge?
    - If YES: This is Case 2.1. We can find C^1 and C^2 that verify condition C rather easily.
    - If NO: This is Case 2.2, in which checking condition C is also easy.
  - If NO: Is there a minimal cycle with a path between two *different* elements of it, say c_0^1 and c_k^1, that does not belong to the cycle?
    - If YES: Can we find such a path of length different from k?

Table 1 Table of the main cases, each of them illustrated with an example (the C^2 on which we perform the generic construction is in red).

Loops			No loop				
			Bidirectional edges		No bidirectional edge		
							
Case 1.1	Case 1.2	Case 1.3	Case 2.1	Case 2.2	Case 3.1	Case 3.2	Case 3.3

        - If YES: This is Case 3.1, a rather tedious case, but we can find cycles C^1 and C^2 that verify condition C nonetheless.
        - If NO: This is Case 3.2, which relies heavily on the fact that $\mathcal{G}(H)$ is not of state-split cycle type to find cycles that verify condition C.
    - If NO: This is Case 3.3, an easy case to find cycles that verify condition C.

3.4 Proof of Theorem 6 for multiple strongly connected components

The idea if H has several SCCs is to build one, by products of SCCs, that is none of the three types that constitute condition D. We can then apply what we did in the previous subsections.

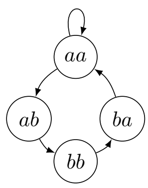
The direct product $S_1 \times S_2$ of two SCCs S_1 and S_2 is made of pairs (s_1, s_2), where an edge exists between two pairs if and only if edges exist in both S_1 and S_2 between the corresponding vertices. It can be used in our construction by forcing pairs of elements $(s_1, s_2) \in S_1 \times S_2$ to be vertically one on top of the other.

Since H does not verify condition D, it has a non-reflexive SCC S_1, a non-symmetric SCC S_2 and a non-state-split SCC S_3 (two of them being possibly the same). But then:

- Since S_1 is non-reflexive, no SCC of $S_1 \times S_2 \times S_3$ is reflexive. Indeed, since S_1 is strongly connected, all vertices of S_1 are represented in any SCC C of that graph product, meaning that for any $s_1 \in S_1$ there is at least one vertex of the form $(s_1, *, *)$ in C. But if C had loop on all its vertices, then in particular S_1 would be reflexive.
- Similarly, since S_2 is non-symmetric, no SCC of $S_1 \times S_2 \times S_3$ is symmetric.
- Finally, since S_3 is non-state-split, no SCC of $S_1 \times S_2 \times S_3$ is a state-split cycle. Indeed, suppose S is such a state-split SCC of the direct product. It can be written as a collection of classes $(V_i)_{i \in I}$ of elements from $S_1 \times S_2 \times S_3$ that we can project onto S_3, getting new classes $(W_i)_{i \in I}$ with elements of S_3 that possibly appear in several of these. Let c be any vertex in S_3 that appears at least twice *with the least difference of indices between two classes where it appears*; say $c \in W_i$ and $c \in W_{i+k}$. Since S is state-split, all elements in W_{i+1} are exactly the elements of S_3 to which c leads. But it is the same for W_{i+k+1}. Hence $W_{i+1} = W_{i+k+1}$. From this we deduce that $W_i = W_{i+k}$ for any i, using the fact that indices are modulo $|I|$. Since k is the smallest possible distance between classes having a common element, classes from $(W_i)_{i \in \{0,\dots,k-1\}}$ are all disjoint. Now simply consider these classes W_0 to W_{k-1}: you get the proof that S_3 is state-split.

Perspectives

Nearest-neighbor conditions are strong constraints, hence it is rather coherent that apart from some very simple graphs the undecidability of DP_H is systematic. Investigation has begun about non-nearest-neighbor constraints, for which there seem to be more graphs with a decidable Domino Problem, this set of graphs possibly being non-recursive. For instance, the opposite graph is of decidable Domino Problem.



Another perspective would be to generalize these results to $\mathbb{Z}^d$: we fix restrictions on a $\mathbb{Z}^k$-SFT with $k < d$ and look at the consequent decidability for $\mathbb{Z}^d$-SFTs. It is immediate that if $d - k \geq 2$ then we can reduce to $DP(\mathbb{Z}^2)$ so this new problem is always undecidable. However, the $d - k = 1$ case – that is, fixing a hyperplane – is still open.

References

1. Nathalie Aubrun, Sebastián Barbieri, and Emmanuel Jeandel. About the domino problem for subshifts on groups. *Springer International Publishing, , in Valérie Berthé and Michel Rigo, editors, Sequences, Groups, and Number Theory., chapter 9, to appear.*, 2019.
2. Nathalie Aubrun, Sebastián Barbieri, and Etienne Moutot. The domino problem is undecidable on surface groups. In *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany.*, pages 46:1–46:14, 2019. `doi:10.4230/LIPIcs.MFCS.2019.46`.
3. Nathalie Aubrun and Mathieu Sablik. Simulation of Effective Subshifts by Two-dimensional Subshifts of Finite Type. *Acta Appl. Math.*, 126:35–63, 2013. URL: `http://dx.doi.org/10.1007/s10440-013-9808-5`, `doi:10.1007/s10440-013-9808-5`.
4. Sebastián Barbieri and Mathieu Sablik. The domino problem for self-similar structures. In *Pursuit of the universal*, volume 9709 of *Lecture Notes in Comput. Sci.*, pages 205–214. Springer, [Cham], 2016. URL: `https://doi.org/10.1007/978-3-319-40189-8_21`.
5. Robert Berger. *Undecidability of the Domino Problem*, volume 66 of *Memoirs AMS*. American Mathematical Society, 1966.
6. David B. Cohen. The large scale geometry of strongly aperiodic subshifts of finite type. *ArXiv e-prints*, December 2014. `arXiv:1412.4572`.
7. David B. Cohen and Chaim Goodman-Strauss. Strongly aperiodic subshifts on surface groups. *ArXiv e-prints*, October 2015. `arXiv:1510.06439`.
8. Bruno Durand, Andrei E. Romashchenko, and Alexander Shen. Fixed-point tile sets and their applications. *J. Comput. Syst. Sci.*, 78(3):731–764, 2012.
9. Silvère Gangloff and Mathieu Sablik. Simulation of minimal effective dynamical systems on the cantor sets by minimal tridimensional subshifts of finite type. *Prépublication (http://arxiv.org/abs/1806.07799)*, 2018.
10. Michael Hochman. On the dynamics and recursive properties of multidimensional symbolic systems. *Inventiones mathematicae*, 176(1):131, Dec 2008. `doi:10.1007/s00222-008-0161-7`.
11. Emanuel Jeandel. Aperiodic subshifts of finite type on groups. *https://hal.inria.fr/hal-01110211*, 2015.
12. Emmanuel Jeandel and Michaël Rao. An aperiodic set of 11 wang tiles. *CoRR*, abs/1506.06492, 2015. URL: `http://arxiv.org/abs/1506.06492`, `arXiv:1506.06492`.
13. Emmanuel Jeandel and Nicolas Rolin. Fixed parameter undecidability for wang tilesets. In *Proceedings 18th international workshop on Cellular Automata and Discrete Complex Systems and 3rd international symposium Journées Automates Cellulaires, AUTOMATA & JAC 2012, La Marana, Corsica, September 19-21, 2012.*, pages 69–85, 2012. `doi:10.4204/EPTCS.90.6`.
14. Jarkko Kari. The tiling problem revisited. In *MCU*, pages 72–79, 2007.
15. Jarkko Kari and Etienne Moutot. Decidability and periodicity of low complexity tilings. *CoRR*, abs/1904.01267, 2019. URL: `http://arxiv.org/abs/1904.01267`, `arXiv:1904.01267`.
16. Douglas Lind and Brian Marcus. *Symbolic Dynamics and Coding.* Cambridge University Press, 1995.
17. Shahar Mozes. Tilings, substitution systems and dynamical systems generated by them. *Journal d'Analyse Mathématique*, 53(1):139–186, Dec 1989. `doi:10.1007/BF02793412`.
18. Ronnie Pavlov and Michael Schraudner. Entropies realizable by block gluing $\mathbb{Z}^d$ shifts of finite type. *Journal d'Analyse Mathématique*, 126(1):113–174, Apr 2015. `doi:10.1007/s11854-015-0014-4`.
19. Raphael M. Robinson. Undecidability and nonperiodicity for tilings of the plane. *Inventiones mathematicae*, 12(3):177–209, Sep 1971. `doi:10.1007/BF01418780`.
20. Hao Wang. Proving theorems by pattern recognition – II. *The Bell System Technical Journal*, 40(1):1–41, 1961. `doi:10.1002/j.1538-7305.1961.tb03975.x`.