The Undecidability of the Domino Problem

Emmanuel Jeandel, Pascal Vanier

The Undecidability of the Domino Problem

Emmanuel Jeandel and Pascal Vanier

One of the most fundamental problem in tiling theory is to decide, given a surface, a set of tiles and a tiling rule, whether there exist a way to tile the surface using the set of tiles and following the rules. As proven by Berger [7] in the 60's, this problem is undecidable in general.

When formulated in terms of tilings of the discrete plane $\mathbb{Z}^2$ by unit tiles with colored constraints, this is called the Domino Problem and was introduced by Wang [51] in an effort to solve satisfaction problems for $\forall\exists\forall$ formulas by translating the problem into a geometric problem.

There exist a few different proofs of this result. The most well-known proof is probably the proof by Robinson [47] which is a variation on the proof of Berger. A relatively new proof by Kari [32] has some nice ramifications for tilings of surfaces and groups. In terms of ingredients, one can divide the proofs in 4 categories. The remaining two categories are given by the proof of Aanderaa and Lewis[1] and the Fixed point method of Durand, Romashchenko and Shen [14].

In this course, we will give a brief description of the problem and to the meaning of the word "undecidable", and then give the four different proofs. As we will explain soon, the undecidability of the Domino Problem has as a consequence the existence of an aperiodic tileset. All four sections will be organized in such a way that the interested reader can first extract from the proof the aperiodic tileset into consideration, before we go into more details to actually prove the undecidability of the problem.

Emmanuel Jeandel
Université de Lorraine, CNRS, Inria, LORIA, F 54000 Nancy, France
e-mail: emmanuel.jeandel@loria.fr

Pascal Vanier
Laboratoire d'Algorithmique, Complexité et Logique
Université de Paris-Est, LACL, UPEC, France
e-mail: pascal.vanier@lacl.fr

1 Statement and Consequences

1.1 Definitions and Statement

Definition 1. Let C be a finite set, called the set of colors. A Wang tile t over C is a map from $\{N,S,E,W\}$ to C.

A tileset τ is a finite set of Wang tiles.

A first example is given in Figure 1. C can be thought as a set of colors, patterns, symbols or integers.

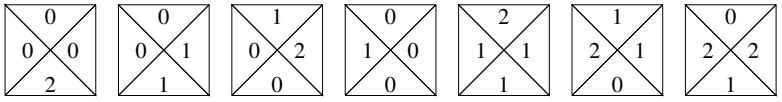


Fig. 1 A tileset τ_1 composed of 7 tiles

Definition 2. Let $P \subseteq \mathbb{Z}^2$, a coloring x of support P assigns to each element of P a tile of τ. When $P = \mathbb{Z}^2$, x is a *configuration*.

A tiling of P by τ assigns to each element of P a tile from τ such that colors of adjacents tiles agree on their common border. Formally, a tiling is a map $x : P \to \tau$ such that:

- If $(i,j) \in P$ and $(i+1,j) \in P$ then $x(i,j)(E) = x(i+1,j)(W)$
- If $(i,j) \in P$ and $(i,j+1) \in P$ then $x(i,j)(N) = x(i,j+1)(S)$

When $P = \mathbb{Z}^2$, it is a *tiling of the plane*. We say that τ *tiles* P if there exists a tiling of P by τ.

An example of a tiling of a finite set by the tileset τ of Figure 1 is given in Figure 2.

Although this particular tileset, as evidenced by the Figure, can be used to tile a large rectangle, it turns out that it doesn't tile the entire plane. In fact, it cannot tile a square of size 15. In general, knowing if a tileset τ tiles the plane is a hard problem:

Theorem 1 (Berger [7]). *There is no algorithm that, given a set of Wang tiles τ, decides if τ tiles the plane.*

In the rest of these notes, we will explain what this theorem means, and how to prove it.

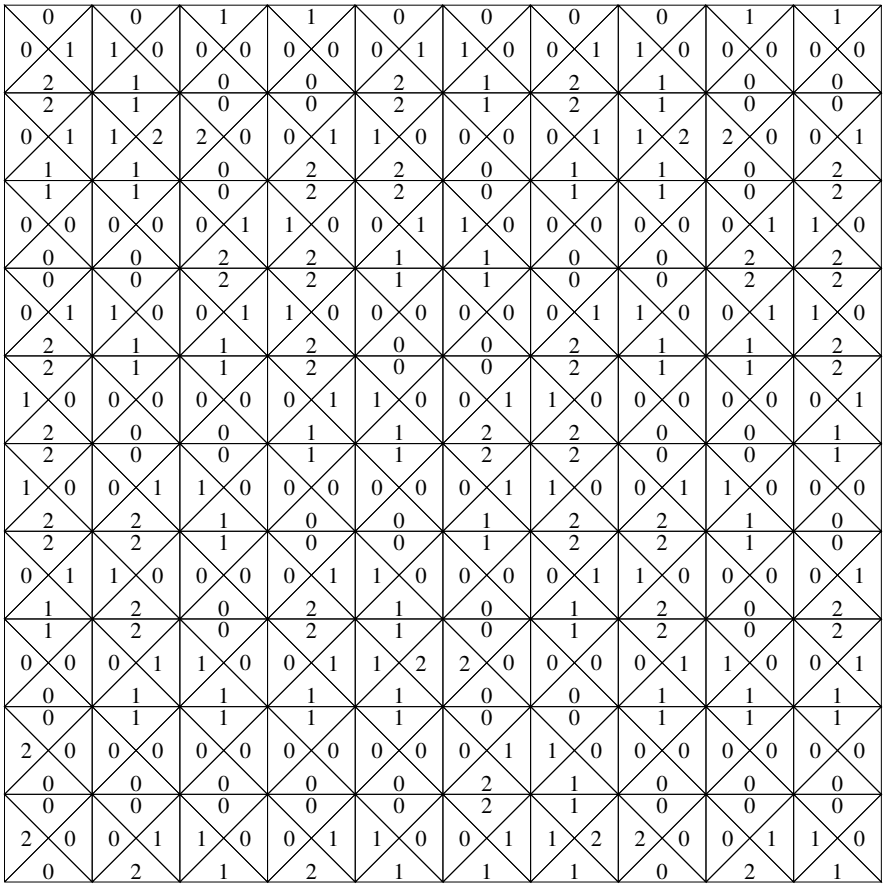


Fig. 2 A tiling of a finite rectangle by τ_1

1.2 Algorithmic Consequences

In this section, we want to investigate what the theorem of Berger actually means in practice when one studies tilings.

Let τ be a set of Wang tiles. τ falls in three different cases:

- τ does not tile the plane
- τ tiles the plane, and a periodic tiling by τ exists
- τ tiles the plan, but no periodic tiling exists. We will say that τ is *aperiodic*.

There are various possible definitions of what a periodic tiling is, but they are all equivalent in our case. We will say that a tiling x is periodic if there exists p s.t $x(i,j) = x(i \mod p, j \mod p)$.

From the algorithmic point of view, it is easy to see if τ does not tile the plane. Indeed

Proposition 1 (Compactness (folklore)). *τ tiles the plane iff for all n, the tileset τ tiles a square of size $n \times n$.*

This proposition gives us an algorithm to prove that τ does not tile the plane: For all n, find a way to tile with τ an $n \times n$ square. If it is not possible for one value of n, then accept.

The second case of the trichotomy is also easy to test algorithmically: For all n, try to find a way to tile a $n \times n$ square in a periodic manner. If this succeeds for some n, then accept.

As the Domino Problem is undecidable, this means there is no algorithm to test for the third case: Indeed, combining the three algorithms together would solve the Domino Problem. Intuitively, this means that there exists aperiodic tilesets τ for which we cannot prove that they are aperiodic. This intuition is made a bit more precise if we look at what Berger proved exactly:

Theorem 2 (Berger [7]). *We can transform effectively any program p into a tileset τ s.t.*

- *If p halts, then τ does not tile the plane.*
- *If p does not halt, then τ tiles the plane aperiodically.*

It is well known that no algorithm can decide if a program halts (see Theorem 5 for a proof), and therefore this theorem implies the undecidability of the Domino problem. It also means in terms of provability that, starting from a program that tries to prove some mathematical statement P, we could build a tileset τ s.t. proving that τ is aperiodic is equivalent to proving the property P.

Berger's theorem tells us that we cannot decide whether a set τ tiles the plane. However, it doesn't say anything about what the tilings, if they exist, look like.

Recall that Berger's proof, starting from a program p, builds a set of tiles τ s.t. τ tiles the plane iff p does not halt. It turns out that the tilings by τ, if they exist, are easy to build: There exists a tiling x by τ and a program that on input (n,m) outputs in finite time the tile of x at position (n,m). This program doesn't even need to know

if there exists a tiling by τ: If no such tiling exists, it will output "error" for some value of n and m.

The situation as described by Berger is therefore not as hopeless as it could be. Later results show however that there exist tilesets that are much more complicated that the ones Berger built.

Theorem 3 (Hanf-Myers [21, 41]). *There exists a tileset τ s.t. tilings by τ exist but no tilings by τ can be produced by a program.*

We also mention the following curiosity:

Theorem 4 (Levin [35]). *There exists a tileset τ s.t.:*

- *Tilings by τ exist*
- *No tilings by τ can be produced by a program.*
- *There exists a probabilistic program (i.e. with a random coin) that will produce a tiling by τ with probability at least $1/2$*

The constant $1/2$ can be replaced by $1-\varepsilon$ for any $\varepsilon > 0$. This result is stated in [35] but no formal proof is given. One could obtain the result using e.g. the results of Simpson [49] and some basic notions of algorithmic randomness theory.

1.3 Algorithms to prove a tileset does not tile the plane

While the theorem of Berger prevents any algorithm to succeed in deciding that a tileset tiles the plane in the general case, it is still interesting to find algorithms that can do it in particular cases. As evidenced by the previous discussion, the easy part is to prove that a tileset does not tile the plane: it is sufficient to find an integer N s.t. no tilings of a square of size N exists.

From the point of view of complexity, better algorithms exist and we will investigate them in this section. Of course, none of these algorithms work: Some of them will not always terminate, some of them may answer "I don't know".

1.3.1 The first algorithm

An easy algorithm is therefore to test, for all N, if there exists a tiling of a square of size $N \times N$. While this is the easiest algorithm to describe, it is not easy to implement in a reasonable manner. It is important for this algorithm to not only test if there exists a tiling of a square of size $N \times N$, but to enumerate all of them: This information will indeed be useful when trying to tile squares of size $(N+1) \times (N+1)$: it suffices to take all squares of size $N \times N$ and to find all ways of completing them into squares of bigger sizes.

Of course there might be a large number of possible tilings: If we denote by $|\tau|$ the number of tiles of τ, we see that we can have at most $|\tau|^{N \times N}$ tilings of a $N \times N$

square. This means generating all of them might be costly, and that our algorithm will be of complexity $2^{O(N^2)}$). We can do a bit better by realizing that we do not need to know what the tilings of the square look like, but only what they look like on the *border*. Indeed, the only information we need to complete a tiling of a $N \times N$ square into a tiling of a $(N+1) \times (N+1)$ square is the colors on the border of the square. Doing this reduces the complexity of the whole operation to $2^{O(N)}$. See Figure 3 for an example.

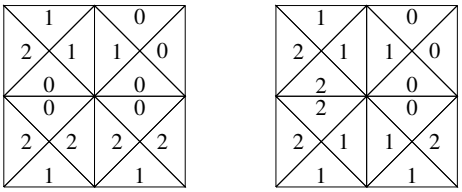


Fig. 3 As the two colorings have the same border, they can be considered to be equal when listing all tilings of a 2×2 square

For the tileset τ_1 of Figure 2, it can be proven that there is no tiling of a 15×15 square, so that the tileset τ_1 indeed does not tile the plane.

1.3.2 Graphs and automata

It is possible for a tileset τ to tile a square of size $N \times N$ but for it to not even tile an entire row. We might expect therefore better results by looking at tilings of N consecutive rows rather than tilings of squares. Before explaining how this can be done, we remark the following:

Proposition 2. *Let τ be a tileset consisting of $q = |\tau|$ Wang tiles.*
If τ tiles a horizontal strip of height N, τ tiles a square of size $N \times N$.
If τ tiles a square of size $N \times N$, it tiles a horizontal strip of height $\lfloor \log_q N \rfloor$.

Proof. The first statement is obvious. For the second statement, suppose that τ tiles a square of size $N \times N$. In particular, τ tiles a rectangle of height $\lfloor \log_q N \rfloor$ and length N.

We look at each column of colors we see inside this rectangle. As the rectangle is of length N, there are $N+1$ different positions. As the rectangle is of height $\lfloor \log_q N \rfloor$, at most $q^{\lfloor \log_q N \rfloor} < N+1$ different associations of color can appear.

By the pigeonhole principle, some column of colors appear at least twice. We can therefore obtain a tiling of an horizontal strip by repeating periodically a sub-rectangle. See Figure 4. □

The bound we obtain is not tight.

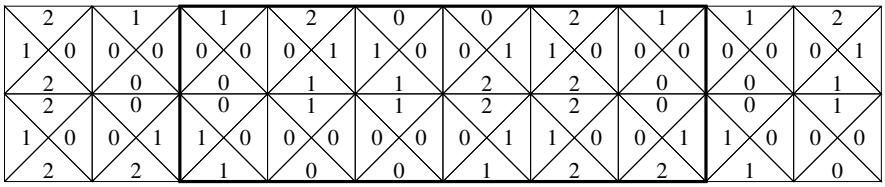


Fig. 4 This rectangle of large width contains (by pigeonhole) a smaller rectangle (in bold) with the same colors on the east side and the west side. This rectangle can be used in a periodic fashion to obtain a tiling of a horizontal strip of height 2.

It remains to explain how to test efficiently if there is a tiling of a horizontal strip of height N.

The key is to represent the tileset as a finite automaton (more accurately finite transducers). The representation is as follows: The states of our finite automaton are the colors C. For each tile $t \in \tau$, there is an edge from $t[W]$ to $t[E]$ labelled with $(t[S], t[N])$.

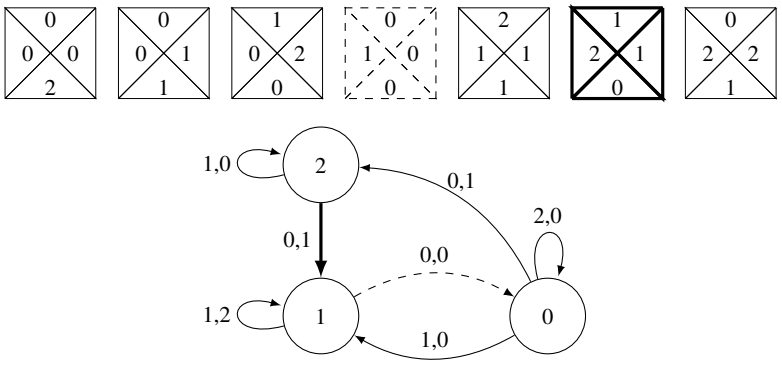


Fig. 5 The tileset τ_1 and its representation as an automaton. The bold (resp. dashed) transition correspond to the bold (resp. dashed) tile.

Figure 5 represents τ_1 as an automaton. In this representation, it is easy to see if τ_1 tiles a infinite strip of height 1. Indeed, an infinite strip of height 1 represents one (biinfinite) run of the automaton.

Proposition 3. *τ_1 tiles an infinite strip of height 1 iff the automaton that represents τ_1 has a directed cycle.*

A strip of height 2 represents therefore two infinite runs of the automaton, s.t. the north labels of the automaton at the bottom agree with the south labels of the automaton on top. This suggest the following definition:

Proposition 4. *Let $\mathscr{A}$ and $\mathscr{B}$ be two automata, with states Q_A and Q_B. The product of $\mathscr{A}$ and $\mathscr{B}$ is the automaton with states $Q_A \times Q_B$ where there is an edge from*

(q_a, q_b) to (q'_a, q'_b) labeled (s_a, n_b) iff there is a color c and an edge from q_a to q'_a labeled (s_a, c) in $\mathscr{A}$ and an edge from q_b to q'_b labeled (c, n_b) in $\mathscr{B}$.

Intuitively, $\mathscr{A}$ is the automaton on the bottom and $\mathscr{B}$ the automaton on the top.

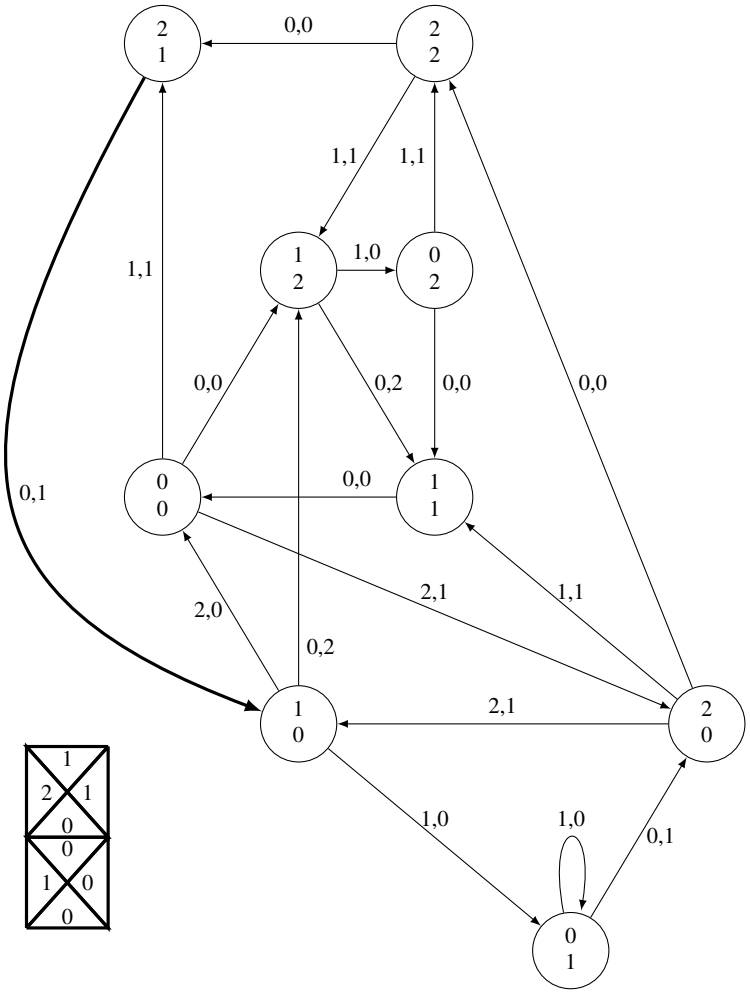


Fig. 6 The composition of the automaton corresponding to τ_1 with itself. The bold edge correspond to the bold and dashed edges in the previous picture, and to the 2×1 pattern on the bottom left

An example is given in Figure 6. Let A be the automaton corresponding to the tileset τ and A^n its n-times composition. Intuitively, an edge in A^n corresponds to a well-tiled pattern of size $1 \times n$, as shown in the figure for $n = 2$. As a consequence, a path in A^n exists iff τ tiles an infinite strip of height n.

We therefore obtain again an algorithm to semi-test if a tileset tiles the plane by computing A^n for all values of n.

In terms of efficiency, it is easy to see that we obtain a complexity similar to the previous algorithm, although we are actually testing for a stronger property. Of course, one can do this both in the vertical and the horizontal direction. Doing this we can prove that τ_1 does not tile a vertical strip of width 8, although it can tile a 14×14 square.

The main interest however is in the fact that translating the problem into the theory of automata means we can use all the tools available in the theory of automata. Indeed, the translation of the fact that τ tiles the plane is about the iteration of the automaton A, which makes it clear that it doesn't really depend on τ, but on the language of infinite words encoded by τ. In particular, we can *minimize* in some way the automaton A (and actually all the automata A^n) under consideration, without changing the problem. We leave [26] as an additional reference about finite automata. These techniques have been used successfully to prove that the smallest aperiodic tileset has 11 tiles [27].

1.3.3 A semi-algorithm based on the Anderson-Putnam complex

We will briefly present here a method that can prove, in some cases, that a tileset τ does not tile the plane without even trying to tile a square with τ.

This method is based on the Anderson-Putnam complex and its second singular homology group, and can be transformed into an "iff" condition [10]. However, we will use here a down to earth approach.

To understand the idea, first suppose that τ has a periodic tiling, say, τ tiles a square of size $p \times p$.

Let $c \in C$ be a color that can appear horizontally. If we look at a square that is tiled periodically by τ, it is obvious that in this square the color c should appear exactly the same number of times on the east side as on the west side. If we denote by x_i the number of times the i-th tile of τ appear, this gives us an equation relating all tiles that contain the color c:

Definition 3. Let $\tau = \{t_1 \dots t_n\}$ be a tileset. Let $x_1 \dots x_n$ be n real variables, where x_i is associated to the tile t_i of τ.

Let c be a color.

The horizontal equation for c is the equation :

$$\sum_{t_i \in E(c)} x_i = \sum_{t_i \in W(c)} x_i$$

where $E(c)$ is the set of all tiles that have the color c in their east side, and $W(c)$ the set of all tiles that have the color c in their west side.

We define similarly the vertical equation for c.

Basically the horizontal equation for c states that the number of times the color c appear somewhere on the east side in the periodic tiling should be equal to the number of times c appear on the west side.

Rather than considering the number of times the tiles appear, we will consider their density. We therefore have an additional equation, stating that the sum of all densities should be one:

Definition 4. The set of equations for τ is the combination of all horizontal equations, all vertical equations, and the unit equation :

$$\sum_i x_i = 1$$

Proposition 5. *If τ tiles a square periodically, then the set of equations for τ has a nonnegative solution.*

Proof. Take x_i to be the density of the tile t_i. □

What is interesting is that this is actually true more generally:

Proposition 6. *If τ tiles the plane, then the set of equations for τ has a nonnegative solution.*

This is true even if τ has no periodic tiling.

Proof. Suppose that τ tiles the plane. Let $n \in \mathbb{N}$. By hypothesis, τ tiles an $n \times n$ square. Let $p_{i,n}$ be the number of times the tile t_i appear in this square and $x_{i,n} = p_{i,n}/n^2$ the density of the tile t_i.

The unit equation is obviously satisfied by the $x_{i,n}$.

Let c be a color. It is easy to see that

$$\sum_{t_i \in E(c)} p_{i,n} - \sum_{t_i \in W(c)} p_{i,n} = O(n)$$

Indeed the difference between these quantities is bounded by the number of times the color c appear in the border of the square.

We therefore get

$$\sum_{t_i \in E(c)} x_{i,n} - \sum_{t_i \in W(c)} x_{i,n} = O(1/n)$$

We now extract from the sequence $(x_{i,n})_{n\in\mathbb{N}} \in [0,1]^{|\tau|}$ a converging subsequence to get the result. □

In our proof we see that the solution x_i we obtain to the system of equations is the limit of densities. Using more complex results from ergodic theory, we can prove that we can take x_i to actually be the density of the tile t_i in a specific tiling of the plane.

This is of course only a necessary condition for τ to tile the plane. Let's look for example at the tileset τ_1 from Figure 2.

We obtain the following set of equations:

$$x_1 + x_2 + x_3 = x_1 + x_4 \quad (1)$$
$$x_4 + x_5 = x_2 + x_5 + x_6 \quad (2)$$
$$x_6 + x_7 = x_3 + x_7 \quad (3)$$
$$x_1 + x_2 + x_4 + x_7 = x_3 + x_4 + x_6 \quad (4)$$
$$x_3 + x_6 = x_2 + x_5 + x_7 \quad (5)$$
$$x_5 = x_1 \quad (6)$$
$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 = 1 \quad (7)$$

The first three equations are the horizontal equations, the following three equations correspond to the vertical constraints.

Even though τ_1 does not tile the plane, the system admits solutions. Among them, there is for example: $x_3 = x_4 = x_6 = 1/5$ and $x_7 = 2/5$, the other variables being 0.

This solution corresponds to the tiling of the pattern in Figure 7. Each color appear the same number of times in the east and west side of this Figure, but this pattern cannot be used to tile the plane periodically. However, it can be used to tile *some* surface: Just glue the north side and the south side of the pattern together and do the same in the east and west side. One can prove that solutions of the system of equations can always be used to tile some surface but that the genre of the surface will have to grow linearly in the number of tiles if the tileset τ does not have a tiling. This is in essence the result of [10].

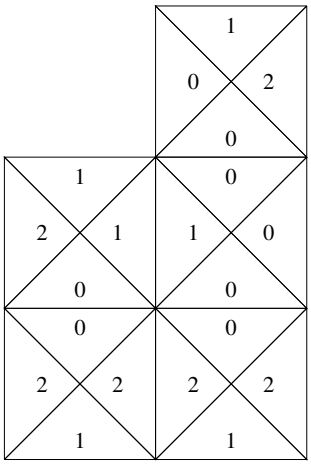


Fig. 7 A tiling of a finite pattern by τ_1 where colors appear the same number of times in the east (resp. north) side and the west (resp. south) side.

2 Main ingredients in the proofs

There are basically four vastly different proofs of the undecidability of the Domino Problem: The proof by Berger [7, 8] later simplified by Robinson [47], the proof by Aanderaa and Lewis [1], best presented in the book of Lewis [36], the proof by Kari [32], and the proof by Durand, Romashchenko and Shen [14]. All other proofs of the result the authors are aware of can be roughly characterized as variants of the construction of Berger.

It is interesting to note that Kari's proof relies on the proof by Hooper [25] of the undecidability of the immortality problem of Turing machines (more on this later). As a consequence, three of the different proofs were obtained by, or use results of, PhD students of Hao Wang (Robert Berger, Philip Hooper and Stål Aanderaa), working on the undecidability of the $\forall\exists\forall$ fragment of first order logic.

We will focus in this section on the common elements in all the proofs. First, there is a need to define precisely what it is meant by "undecidable", for which we need a formal definition of an algorithm (or a program). Then we will prove the undecidability of the fixed domino problem, which is the first brick in most of the proofs. We will then explain why the previous proof cannot be adapted directly to obtain the main result, and what is needed to accomplish it.

2.1 Turing machines

The theorem by Berger states that no algorithm can decide, starting from a set of Wang tiles τ, if τ tiles the plane.

To prove such a theorem, one needs a formal definition of a program. Since 1930, various equivalent definitions, due to Church, Turing or Herbrand-Gödel [11, 50, 34] have been given.

For our purpose, the definition by Turing will be the most suitable, although some results have been obtained [33] using the concept of 2-counter machines introduced by Minsky [39].

We refer the reader to [37, 39, 44] for more information about Turing machines. The book [26] offers in particular an introduction to both Turing machines and finite automata.

A Turing machine is a formal definition of a computation device. This device takes as input a word in some given alphabet, and outputs, after several steps, a word in some other alphabet.

The machine uses the concept of a *tape*, which is an infinite or biinfinite array of cells. Each position of the cell contains a symbol, and one specific position of the tape is marked by the *head*. The Turing machine makes a decision at each step depending on the symbol it can read on the tape at the position of the head, and its internal *state*. Based on these informations, the machine can shift the position of the head, change the symbol of the tape, and change its internal state.

Definition 5. A Turing Machine is given by:

- A finite set of states Q. One distinguishes in Q a specific state $q_0 \in Q$, called the initial state, and two special states q_a, q_r, the accepting and refusing state[1].
- An input alphabet Σ
- A work alphabet $\Gamma \supset \Sigma$, it contains a special symbol B (not present in Σ), called the blank symbol.
- A transition function: $\delta : \Gamma \times Q \to \Gamma \times Q \times \{-1, 1\}$

Intuitively $\delta(a,q) = (b,q',d)$ means that, if the Turing machine reads the symbol a on its head and is currently in state q, then it will write the symbol b instead of a, change its internal state to q', and move its head in direction d.

The transition function is best depicted with a graph. Vertices represent states of the machine, and there is an edge from q to q' labeled with $a/b,d$ if $\delta(a,q) = (b,q',d)$. It is also customary to write $+1,-1$ as $\to$ and $\leftarrow$ as they denote movement of the head.

An example of a Turing machine is given in Figure 8. The machine works as follows: we first write the input on the tape (see the first row of Figure 9). The Turing machine starts with initial state q_0. At each step, the Turing machine looks at its internal state and the symbol at the position of the head. The transition function then tells us what symbol should be written at the position of the head, and what is the new state and the new position of the head. Figure 9 contains the first steps of the execution of a Turing Machine.

Definition 6. An *instantaneous description* (also called a *configuration*) is a tuple $w = (c,h,q) \in \Gamma^{\mathbb{Z}} \times \mathbb{Z} \times Q$. The content of the tape is represented by c, while h contains the position of the head on the tape, and q the internal state of the machine.

For $w = (c,h,q)$ an ID, the successor of w is the ID $w' = (c',h',q')$ defined by:

- Let $\delta(c_h, q) = (b,q',d)$
- $c'_j = c_j$ for $j \neq h$ and $c'_h = b$.
- $h' = h + d$

We write $M(w)$ for the successor of w and $M^n(w)$ is defined in the usual way.

Definition 7. Let M be a Turing machine.

Let u be a word. The initial ID is $w_u = (c,h,q)$ where

- $c_{i+1} = u_i$ when it make sense and $c_i = B$ otherwise (the word u is represented on the tape, starting at position 1.
- $h = 0$ (The head is initially at position 0)
- $q = q_0$ (The initial state is q_0).

We say that u is accepted if there exists n s.t. $M^n(w_u) = (c',h',q_a)$. We say that a Turing machine accepts a language L if on every input the Turing machine eventually reaches either the accepting state q_a or the rejecting state q_r, and L is exactly the set of accepted words.

[1] The set of states is called Q rather than S for some forgotten historical reason. Alan M. Turing uses the vocabulary "configuration" rather than "state", but the word "state" has become more common nowadays.

Notice that there are three possible behaviours of a Turing machine on input u: Either it eventually reaches the accepting or the rejecting state, or it keeps running forever.

If the Turing machine is defined with an infinite array of cells rather than an biinfinite array (i.e. configurations are in $\Gamma^{\mathbb{N}} \times \mathbb{N} \times Q$), a fourth possibility arises: the Turing machine may try to go left when the head is in the leftmost cell, in which case it crashes. It is easy to impose safeguards so that the last case never happens.

If we look at what is written on the tape of the Turing machine when it reaches an accepting state, we can define a *function* computed by a Turing machine. We leave the details to the reader.

A first example of a Turing machine is depicted on Figure 8, with the first 15 steps of the machine represented in Figure 9.

This machine has not been chosed arbitrarily, and we can explain what it does. The input of the machine has to be understood as an integer coded in binary, written from least to most significant bit, i.e. the integer 6 is written 011. The state 1 has two different outgoing transitions, depending on whether the first symbol is a 0 or a 1. This means we have two different behaviours depending if the integer is even or odd.

We will first focus on the case where the integer is even. When in state 2, the head keeps going right, until it reaches the end of the word (state 3) and then we go to state 4 or 5 depending on the last symbol. States 4 and 5 essentially shift the input to the left. This means replacing each symbol with the symbol previously seen; the state 4 (resp. 5) means the previously seen symbol was a 0 (resp. 1). When we reach the begining of the word, we go again to state 1. We have said that, if an integer is even, we will delete its first bit, meaning that states 2 to 5 essentially divide an integer by 2.

States $7,8,9$ basically multiply an integer by 3. Multiplying an integer by 3 in binary may produce a carry of value 0, 1 and 2, which correspond respectively to the three states $7,8,9$. As an example, suppose the carry is currently 1, and the integer to multiply by 3 begins with a 1. As $3 \times 1 + 1 = 0 + 2 \times 2$, this means the first bit of the output should be a 0 and our new carry is equal to 2. This corresponds to the transition from state 8 to state 9. The state 6 is essentially there to kickstart the process, and detect if the output is the integer 1. The state A rewinds the head so that it is again at the initial position. In other words, states 6 to A, starting from an integer n, produces the integer $3n+1$, except if $n = 1$ in which case the Turing machine stops.

As a whole, our Turing Machine is essentially mimicking the Collatz sequence: Starting from an integer n written in binary, we do the operation $n \mapsto n/2$ or $n \mapsto 3n+1$ until n is equal to 1. Knowing if this particular Turing machine halts on all inputs that ends with the symbol 1 is therefore equivalent to the Collatz Conjecture. (For simplicity, the Turing machine we designed does not work if the input ends with a symbol 0, i.e. if it codes the integer 0, or if an integer is coded with additional symbols 0 at the end).

This somewhat easy example should convince the reader that Turing machines are indeed able to represent arbitrary computations. We have somehow coded tradi-

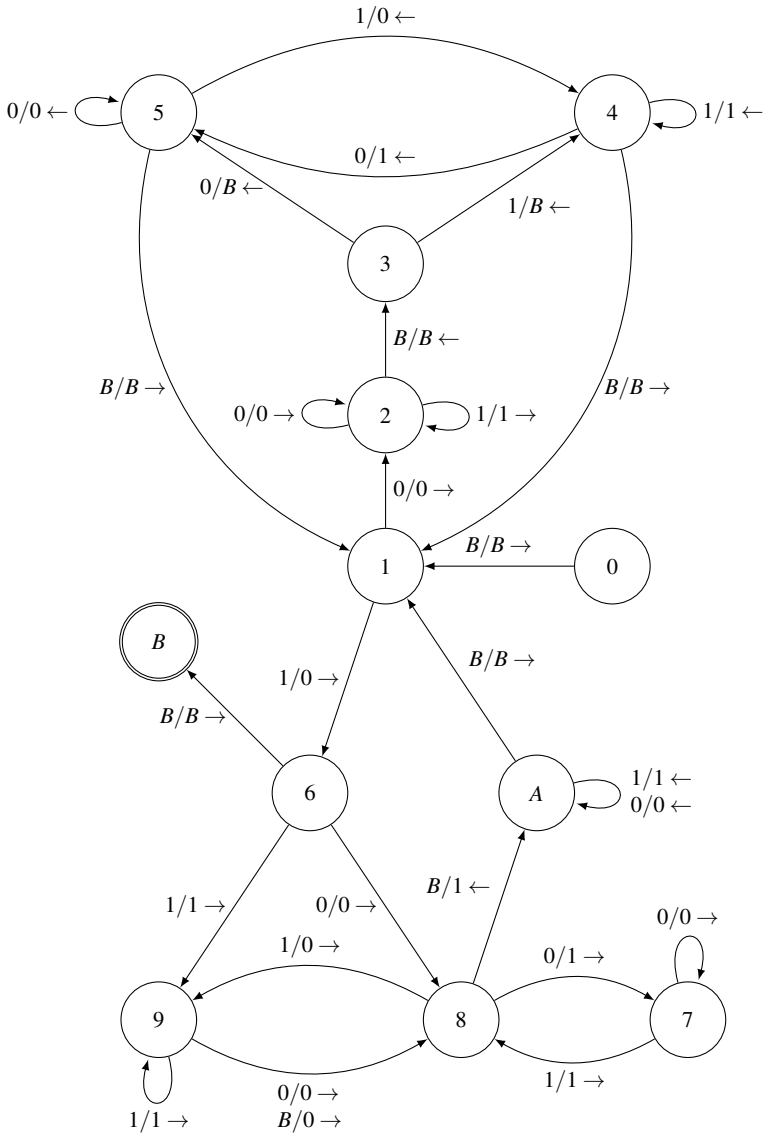


Fig. 8 An example of a Turing machine. The initial state is $q_0 = 0$. The accepting state is state $q_a = B$.

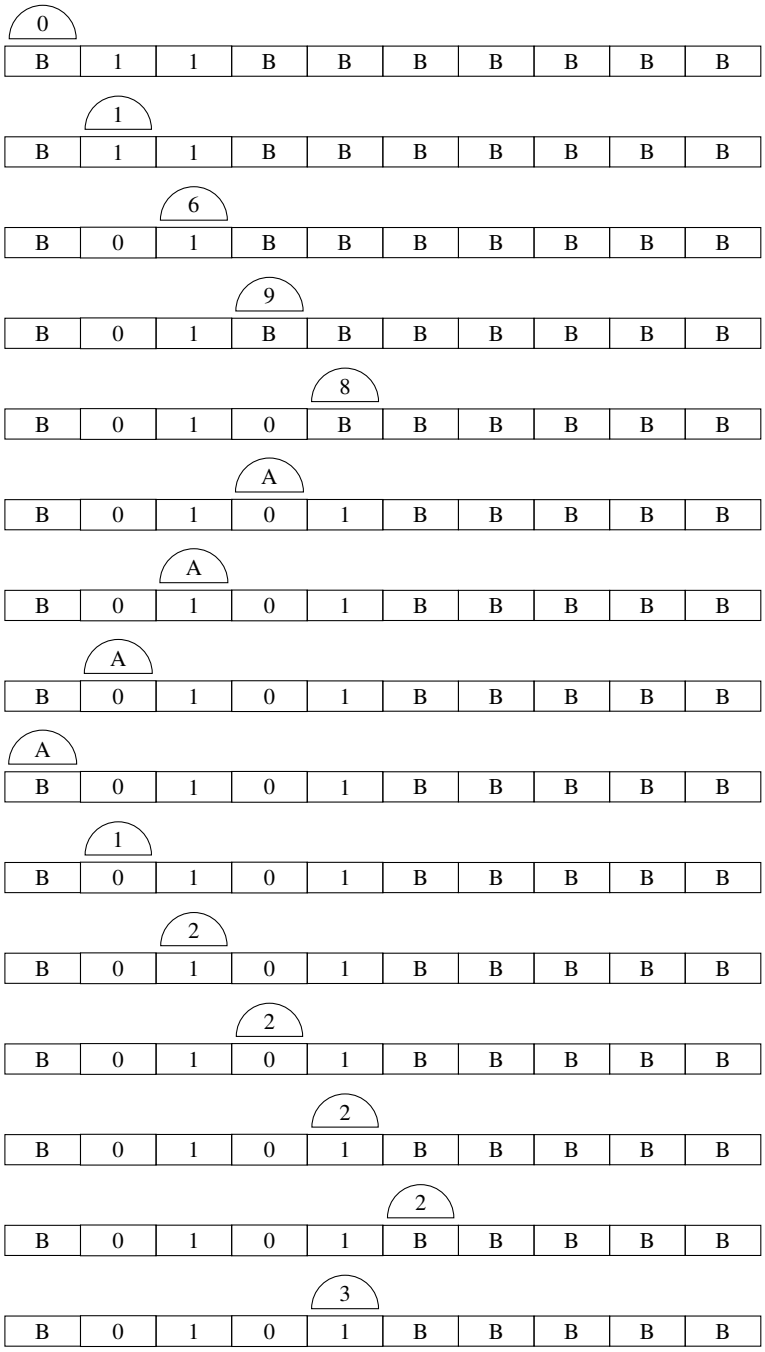


Fig. 9 The first 15 steps of the Turing machine of Figure 8 on input `11`

tional constructions from programming languages: We have a `while` loop that ends when the integer n is equal to 1 (this is done by states 1 and 6), and we have used *subroutines* that divides by 2 or multiply by 3 respectively (states $3/4$ and states $7/8/9$).

One can prove that any program in your favorite programming language may be translated into an equivalent Turing machine. The *Church-Turing* thesis essentially states that all reasonable models of computation we can devise are equivalent, in the sense that they define exactly the same computable functions. In fact, almost all models also use, up to a polynomial, the same ressources in time and memory, this is the extent of the extended Church-Turing thesis.

With the vocabulary of Turing machines, we can give the first formal example of an undecidable problem:

Theorem 5. *There is no algorithm that, given a Turing machine M and a word u, decides if M halts on input u.*

This problem is called the "Halting problem".

Proof. To prove that such an algorithm does not exist, we need to formalize exactly its input. So we need an encoding of descriptions of Turing machines into words. We write machine(u) for the Turing machine whose encoding is u.

Suppose such an algorithm exists. As algorithms can be turned into Turing machines, we therefore have the existence of a Turing machine N s.t., on input u and v, works as follows:

- If machine(u) halts on input v, N accepts
- If machine(u) does not halt on input v, N rejects.

We obtain from N a Turing machine N' that works in the following way: On input v:

- If machine(v) halts on input v, N' does not halt
- If machine(v) of code v does not halt on input v, N' halts.

We then obtain a contradiction by looking at the behaviour of N' on input w s.t. machine$(w) = N'$. □

Corollary 1. *There is no algorithm that, given a Turing machine M, decides if M halts on the empty input.*

Sketch. Given a Turing machine M and an input u, we can easily build a Turing machine N s.t. N on the empty input simulates M on input u: N proceeds by first writing u on its tape, and then "calls" the Turing machine M. Therefore an algorithm that solves the halting problem on the empty input can solve it on any input. □

2.2 Encoding of Turing machines in tilings - The Fixed Domino Problem

Figure 9 shows that the execution of a Turing machine can be written essentially as a two-dimensional object, with each row representing a step of the Turing machine. It is therefore not surprising that this model is quite easy to encode into Wang tiles. There exists multiple equivalent encodings in the litterature.

In the case of Turing machines with an infinite (rather than biinfinite) tape, such an encoding is proposed in Figure 10, and a quarter of plane using these tiles is given in Figure 11.

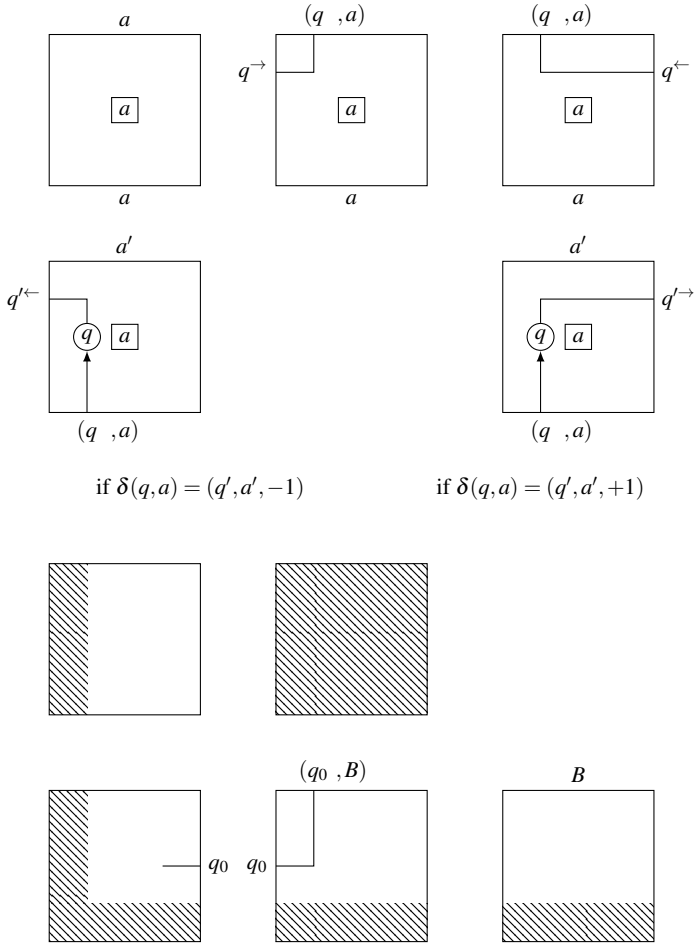


Fig. 10 Coding of a Turing machine by Wang Tiles

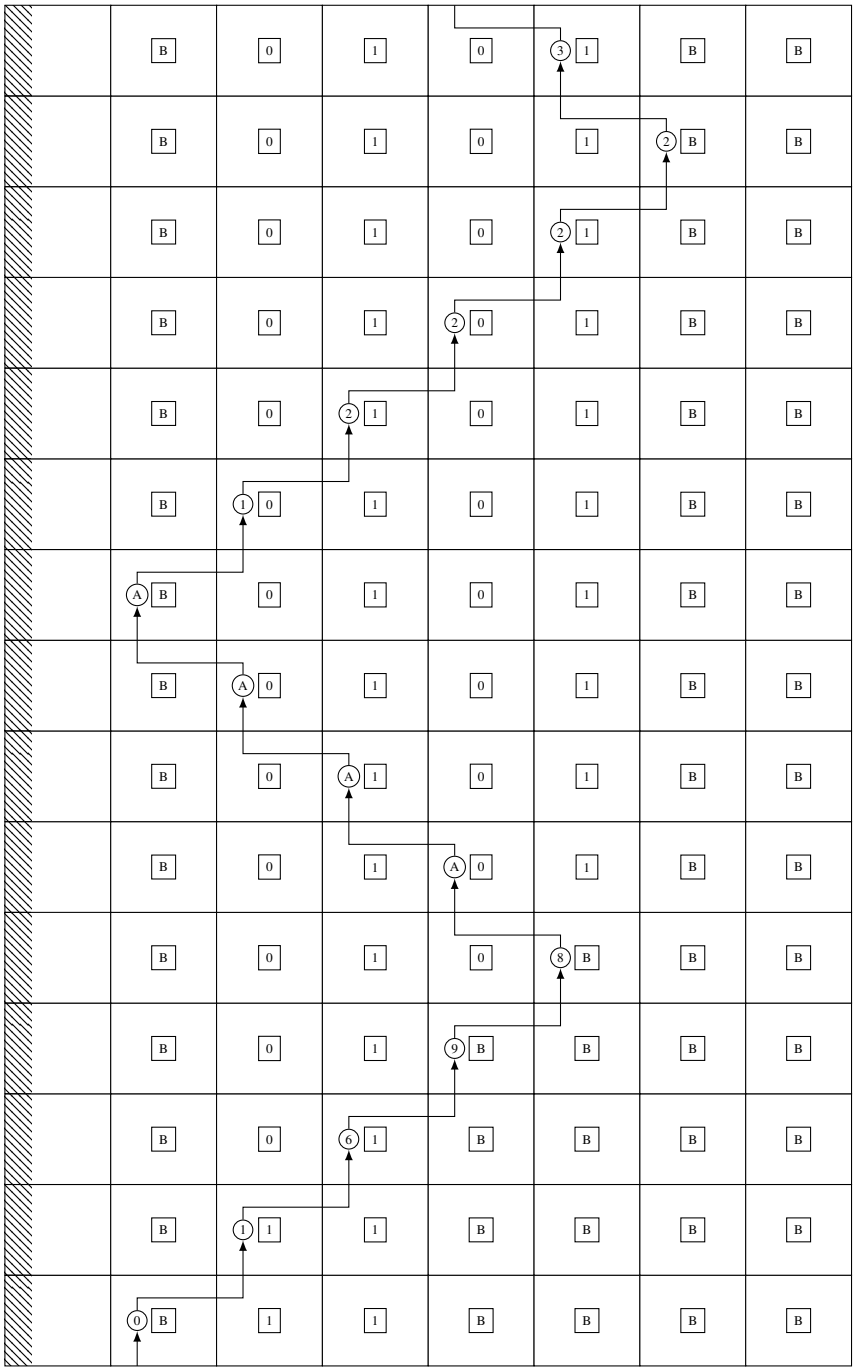


Fig. 11 A tiling of a quarter of plane by the set of Wang tiles of Fig10. The link with Fig. 9 should be obvious.

If we write the input on the bottom left of a quarter of a plane, then the only way to fill up the quarter of a plane is to simulate the execution of the Turing machine. If furthermore, we delete all tiles involving the accepting state q_a, then a tiling is possible iff the Turing machine does not halt.

Using the three tiles at the bottom of Figure 10, we can initialize the computation so that it starts with the empty input and state q_0. In fact, any tiling of the quarter plane that contains at the origin the tile that is at the bottom-left of Figure 10 simulates the execution of the Turing machine with empty input. This is in fact a tiling of the entire plane, as the rest can be filled up with the blank tiles. Therefore, we have a tiling of the entire plane, with a specific tile at the origin, iff the Turing machine does not halt on empty input.

We have proven:

Theorem 6 (Undecidability of the Fixed Domino Problem, Wang [52], Büchi [9], Kahr-Moore-Wang [29]). *There is no algorithm that decides, given a set of Wang tiles τ and a predescribed tile $t \in \tau$ whether there exists a tiling of the plane by τ that contains t.*

If we look at the construction, we can obtain

Corollary 2. *There is no algorithm that decides, given a set of Wang tiles τ and a predescribed tile $t \in \tau$ whether there exists a tiling of the quarter of the plane by τ with t at the corner.*

It is possible to adjust slightly the set of tiles so that we can, starting from an input u, produce a finite pattern P s.t. there is a tiling by τ that contains the pattern P iff the Turing machine does not halt with input u. Doing this we obtain

Theorem 7 (Robinson [47]). *There exists a set of Wang tiles τ s.t. there is no algorithm that decides, given a finite pattern P whether there exists a tiling of the plane by τ that contains P.*

Proof sketch. Let M be the Turing machine that on input u simulates machine(u) on the empty input. This machine is indeed implementable, and it is easy to see that there is no algorithm that decides, given an input u, if M halts on input u. We now code this Turing machine as in Figure 10. □

2.3 Towards the Domino Problem

With one notable exception, the coding we presented before is central in all the proofs of the undecidability of the Domino Problem. However this is far from sufficient.

Indeed, the construction gives rise to a lot of tilings if the primordial tile is not used. First, we can have tilings with no head per row, using e.g. only the first tile on the top left of Figure 10, or even using only the blank tile. Second, we can have tilings with more than one head per row. There are easy tricks to stop this problem

from happening, e.g. by having different background colors before and after the head. The third problem is that there can be tilings that correspond to uninitialized computations, i.e. biinfinite computations rather than a computation starting from a finite input and the initial state.

There are no easy solutions to the remaining problems. Three of the methods outlined built a set of Wang tiles in which a quarter of plane appears, in such a way that the corner of the quarter of the plane can be locally identified. It is then a routine exercise to enforce that a specific tile appears in that corner, so that we can use the previous construction inside this quarter of plane.

The last method uses a very different encoding technique, which will enforce that every row codes a configuration of a Turing machine in such a way that the head is always present. The remaining problem is therefore uninitialized computations, which is solved using the following theorem:

Theorem 8 (Hooper [25]). *There is no algorithm that, given a Turing machine M, decides if M halts on all configurations.*

This is still true if we restrict ourselves to Turing machines with a working alphabet of size 2*.*

This last proof of the undecidability of the Domino problem is tremendously simpler, but one could say that part of the complexity of the proof is hidden inside the proof of Theorem 8, which is far from trivial.

It is interesting to remark that Hooper was, like Robert Berger, a PhD Student of Hao Wang. In fact, this theorem was obtained for the same purpose as Berger's, i.e. proving the undecidability of the $\forall\exists\forall$ fragment of first order logic.

3 The Substitutive Method 1/2

3.1 Preliminary discussion

We now start with the first proof of the Undecidability of the Domino Problem. The proof we present here follows closely Berger [7] and Robinson [47] although we will use some recent results and concepts from Ollinger [43].

As we saw in the previous section, it is easy to use Turing machines to prove the undecidability of the Fixed Domino Problem, i.e. whether there exists a tiling by τ that contains some specific tile t. The problem of the construction is, of course, that τ always has some trivial tilings that do not contain t. We need therefore some way to force the tile t to appear.

This suggests a naïve approach to prove the undecidability of the (General) Domino problem : First build a specific tileset τ_s with the property that every tiling of the plane with τ_s contains exactly one copy of a specific tile t_s. Now, given a tileset τ and a tile t, superimpose τ_s with τ by forcing the tile t to appear exactly on top of t_s. Let's call $\tau \times \tau_s$ this new tileset. Then it is clear that $\tau \times \tau_s$ tiles the plane iff there exists a tiling of the plane by τ that contains t. Therefore we cannot decide the General Domino Problem as it would allow us to decide the Fixed Domino Problem, case closed.

However, such a tileset τ_s cannot exist:

Proposition 7. *Let τ_s be a tileset. If every tiling by τ_s contains the tile t_s, then there exists some integer n s.t. every tiling of a square of size n contains the tile t_s.*

Proof. If it were not the case, we would have tilings of arbitrary large squares without the tile t_s. By compactness (Proposition 1), this would imply that there is a tiling without t_s. □

Therefore the tile t_s, on top of which we want to kickstart the computation process of τ appears in every sufficiently large square. This means that our construction would have to contain infinitely many instances of the computation process. How to manage all these different computations in such a way that they do not overlap is not an easy task.

The solution is provided by building tileset τ_s with some specific properties. The general idea is to first build an aperiodic tileset, and then try to embed computation in it. While in theory, the method can be used starting from any aperiodic tileset, the only tilesets that have been used successfully for this method are substitutive tilesets, that we now define.

3.2 Substitutions

Definition 8. Let A be a finite alphabet. A (square) *substitution* is a map ϕ from A to $A^{n\times n}$. If w is a (two-dimensional) word of size $m \times m$, then the image by ϕ of w,

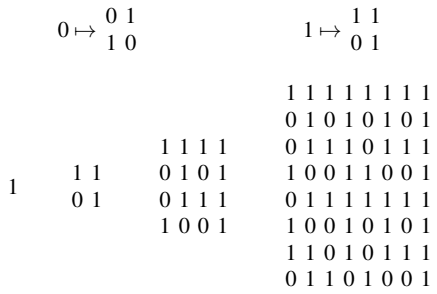


Fig. 12 A substitution ϕ on $A = \{0,1\}$ and $\phi^k(1)$ for $k = 0,1,2,3$

named $\phi(w)$ is the word of size $nm \times nm$ defined by

$$\phi(w)_{(ni_1+i_2,nj_1+j_2)} = \phi(w_{i_1,j_1})_{i_2,j_2}$$

We write $\phi^k(w)$ for the k-th iterate of ϕ. The map ϕ is extended naturally to $w \in A^{\mathbb{Z}\times\mathbb{Z}}$ using the same formula.

The *substitutive subshift* associated to ϕ is the set S_ϕ of all configurations of $A^{\mathbb{Z}\times\mathbb{Z}}$ s.t. $x \in S_\phi$ iff there exists an infinite sequence $(w_i)_{i<0}$ s.t. $\phi(w_i) = w_{i+1}$ and $w_0 = x$.

An example of a substitution is given in Figure 12. This substitution is quite random and has no relevance in the rest of the paper.

Intuitively the substitutive subshift S_ϕ looks like $\phi^k(w)$ at the infinity. The term "subshift" comes from the symbolic dynamics school. We refer to other articles in this volume for details on subshifts.

Definition 9 (Informal definition). A tileset τ is substitutive if the set of tilings by τ is, up to a recoloring, a substitutive subshift.

There are a lot of examples of substitutive tilesets [7, 47]. In fact, one can show that, for any ϕ, S_ϕ can be realized as a tileset τ [40, 20, 17].

Most constructions of substitutive tilesets are hindered by the fact that the tilings are not directly substitutive, but this is only true of their recolorings.

We therefore strive to find a tileset that is intrinsically substitutive:

Definition 10 ([13, 43, 5, 14]). Let τ be a tileset. We say that τ is intrinsically substitutive with factor 2 if there exists a one-to-one substitution $\phi : \tau \to \tau^{2\times 2}$ s.t.:

- For any finite pattern w, $\phi(w)$ is a valid tiling precisely when w is a valid tiling
- For any tiling of the plane x by τ, there exist y s.t. $\phi(y) = x$ upto shift.

The second point ensure that ϕ somehow exhausts all possible patterns that can appear in a tiling of the plane. Notice that, in the second property y is automatically a valid tiling (as every pattern of y is valid by the first property).

Theorem 9 ([13, 43]). *There exists an intrinsically substitutive aperiodic tileset.*

The tileset obtained by Ollinger is depicted in Figure 13. Although the tiles are decorated for cosmetic reasons, they are really Wang tiles: two tiles can be put together if they match on their common edge. An example of a large square tiled by τ is given in Figure 15. Each tile is composed of three layers, as seen in Figure 14.

The first layer ensures that, in each tiling of the plane, the tiles are grouped in squares of size 2×2 to form a small red square.

The substitution that corresponds to τ_s is given in Figure 16.

Proposition 8. *The tileset τ_s depicted in Figure 13 is intrinsically substitutive and aperiodic.*

Proof. Let ϕ be the substitution defined in Figure 16. It is easy to see that ϕ is one to one, and an easy inspection of ϕ shows that $\phi(w)$ is valid precisely when w is valid.

We say that a tiling of a 2×2 square is well behaved if its first layer consists of a small red square (i.e. similar to Figure 16(a)).

We now look at every possible tiling of a 4×4 square by τ where the central 2×2 square is well behaved. By an exhaustive analysis, we see that every such 2×2 square is of the form $\phi(w)$ for some w. (This is easily seen by writing and running a small program. A detailed proof is also given in [5].)

Now let x be a tiling of the plane. We regroup the cells of x into 2×2 well-behaved squares $(x_{i,j})_{i,j \in \mathbb{Z}} \in (\tau^{2\times 2})^{\mathbb{Z}^2}$. By the previous argument, each of them is of the form $\phi(w_{i,j})$, which means that $x = \phi(w)$ upto shift.

Therefore τ is intrinsically substitutive.

τ should be aperiodic. Indeed, suppose that there exists a tiling x by τ of period p, and choose p as small as possible. As x can be divided into 2×2 squares that are not periodic, p should be even. Now $x = \phi(w)$ for some tiling of the plane w. As x is periodic of period p, $\phi(w)$ should be periodic of period $p/2$, a contradiction. □

The reader familiar with the work of Robinson [47, 28, 19, 2] might recognize something familiar in Figure 15. The tileset of Robinson can indeed be obtained from the tileset τ_s by some identification of colors, see the bibliographic notes below.

3.3 Grids inside tilings

We now explain how to use the tileset we built previously to obtain the undecidability of the Domino Problem. The proof highly depends on properties of this tileset, and it is unknown whether a similar reasoning could be used starting from any substitutive tileset.

First, to understand the tilings, we need to simplify the pictures. We will forget about the first and third layer, and focus only on the wires of color red in the second layer. It is easy to see that they form arbitrarily large squares. We obtain Figure 17 by considering only half of the squares. This is done in practice by changing τ_s to have two different shades of red (say, light red and dark red) and forcing light red

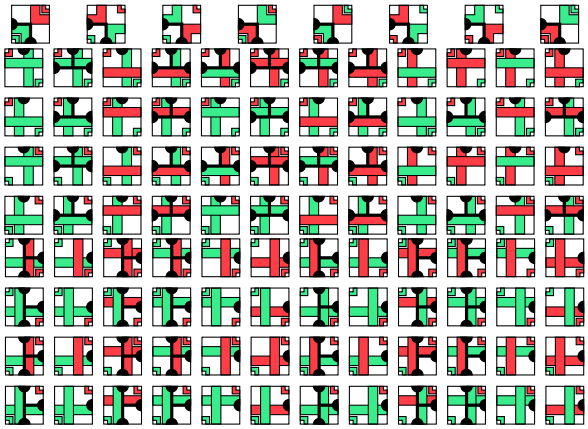


Fig. 13 The tileset τ_s

Fig. 14 The three layers inside a tile

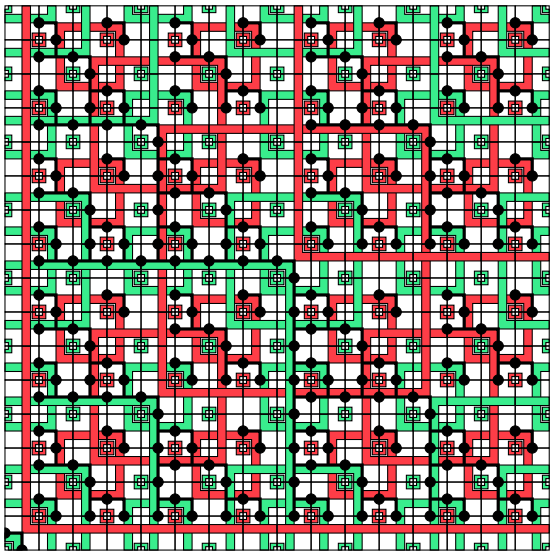


Fig. 15 A portion of a tiling by τ_s

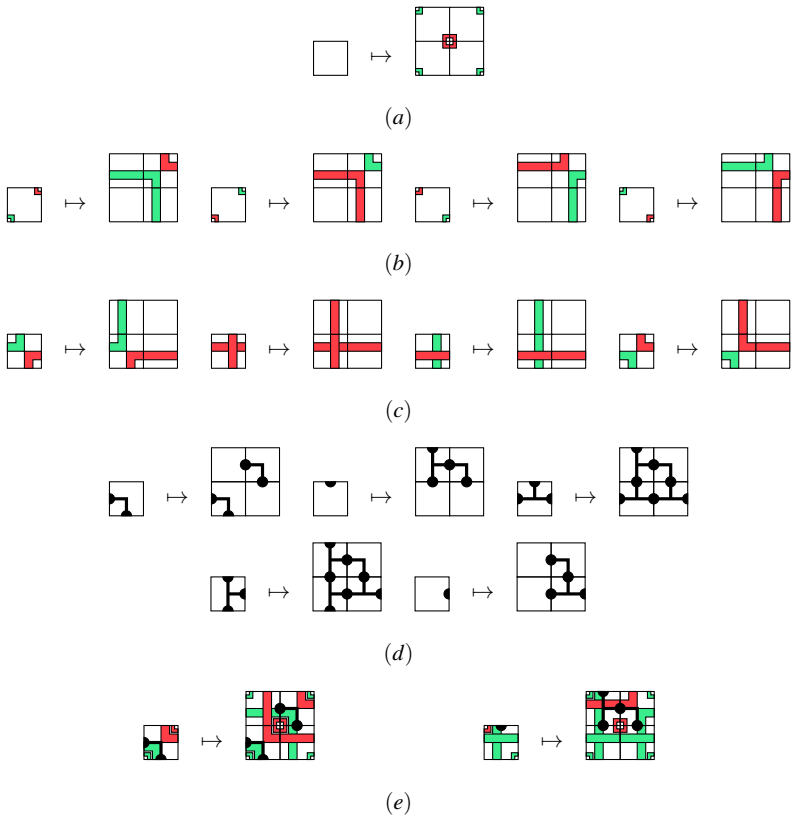


Fig. 16 The substitution. The layer 1 of $\phi(t)$ is the same for all tiles t, as shown in Subfigure (a). The second layer of $\phi(t)$ is cut in half. First, The layer 1 of the tile t is stretched (Subfigure (b)). Second, the wires in the layer 2 of t are extended horizontally and vertically (Subfigure (c)). Finally Subfigure (d) explains how the layer 3 of $\phi(t)$ is obtained from t. Subfigure (e) gives two examples.

wires to cross only dark red wires. Figure 17 then represents only, say, the light red squares.

We now see that tilings by τ_s are formed of squares that do not cross.

Remember that by Corollary 2 there is no algorithm to decide, given a tileset τ and a tile t, whether there exists a tiling of a quarter of plane by τ with t at the corner. By compactness, it means there is no algorithm to decide, given a tileset τ and a tile t whether one can tile arbitrary large squares by τ with t at the bottom-left corner.

Our goal now is clear: starting with a tileset τ and a tile t, we will fill up the squares of τ_s by tilings by τ in such a way that t appear in the bottom-left corner. One difficulty remains: While the squares in τ_s do not cross, they are included one in each other, which means that we cannot use all the space in one square, as some of

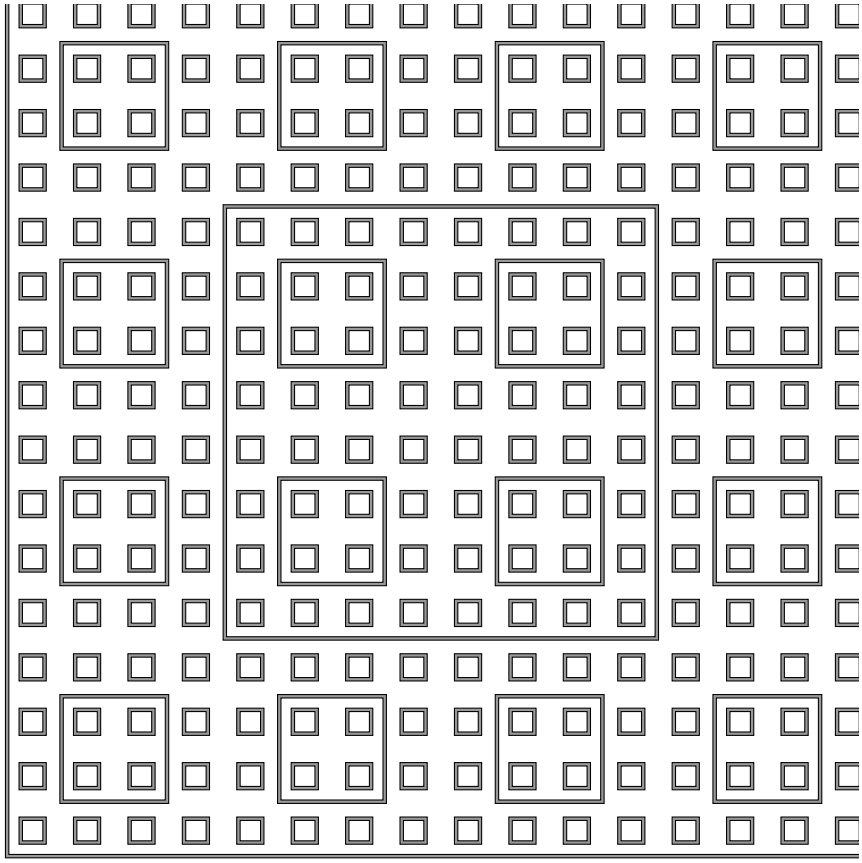


Fig. 17 Squares inside the tilings by τ_s

it is already occupied by other squares. We will therefore first extract in each square the largest "free" subsquare.

Let S be a square, that we see as a subset of positions of the form $C \times D$ for some segments C and D. Let U (for "used") be the set of positions inside S that are occupied by a smallest square.

By a free subsquare of S, we denote a set of horizontal positions $H \subseteq C$ and vertical positions $V \subseteq D$ s.t. $H \times D$ and $C \times V$ do not intersect U.

To obtain the largest free subsquare of S, we just have to cross out every horizontal (resp. vertical) coordinate on which a smallest square lies. We will call these coordinates "obstructed". In our particular case this can be done easily by local rules: The idea is that each square will emit horizontal (resp. vertical) signals on their exterior borders that propagate horizontally (resp. vertically) and that die out when they encounter another border. This is explained nicely in [47], see Figure 18.

To be exact, notice that the parts that are not crossed do not correspond to whole tiles, but only quarters of them. This has no bearing in the following.

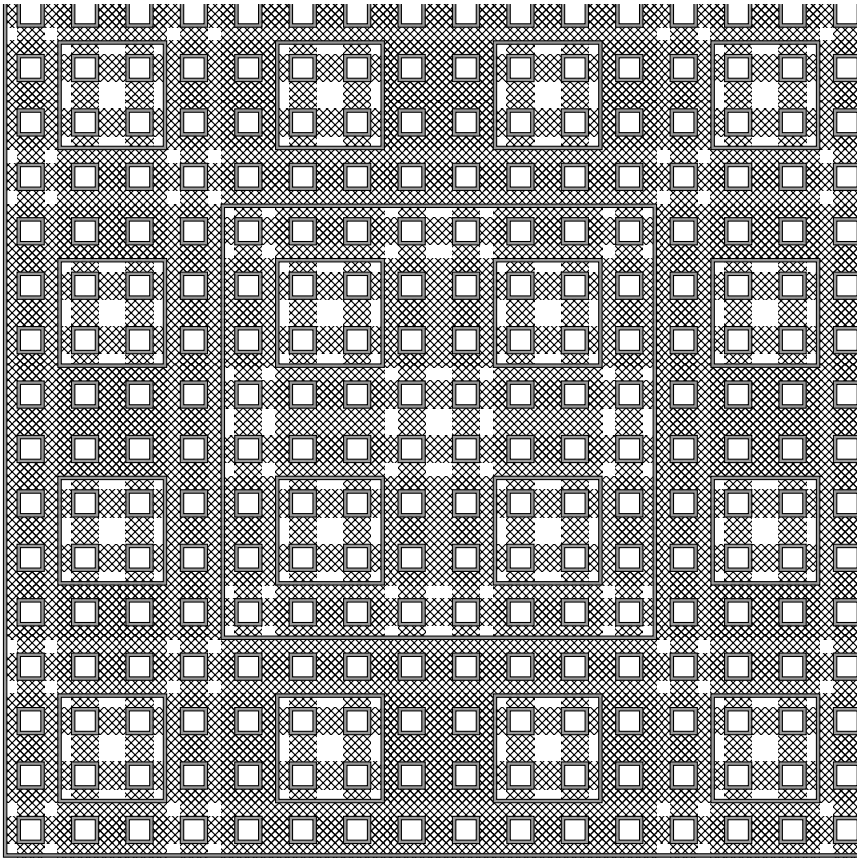


Fig. 18 The squares with the obstructed rows and columns grayed out. Inside a square, the remaining unobstructed space, in white, forms a square whose size is a quarter of its container square.

As can be seen in the Figure, larger squares of the tiling have larger free subsquares formed by joining the unobstructed space. It remains to put the tileset τ inside the free subsquare.

The idea is that the position (i, j) of the supposed tiling by τ will be contained in the position (h_i, v_h) of the square S, where h_k (resp. v_k) is the k-th element of H (resp. V). As no element of $H \times D$ is occupied by a smallest square, we can use cells of $H \times (D \setminus V)$ to propage the vertical constraints of τ inside the grey area of the tiling of Figure 18. We can also force the tile at the bottom-left quarter of the square easily, by enforcing that the only tile that can appear on top of a corner tile of τ_s is the tile t.

Putting everything together we have the desired proof.

Theorem 10. *The Domino Problem is undecidable.*

Proof. Let τ be a tileset and $t \in \tau$. We use the construction above to obtain a new tileset τ_r combining τ and τ_s.

Suppose this new tileset τ_r admits a tiling. This tiling contains arbitary large squares, and therefore arbitrary large free subsquares. But these subsquares form a tiling by τ with the tile t at the corner. Therefore there exist arbitrary large squares tiled by τ with t at the corner.

Conversely, if there exist arbitrary large squares tiled by τ with t at the corner, we can obtain a tiling of the plane by τ_r by putting in each free subsquare of each square a valid tiling.

We have therefore proven that there exists arbitrary large squares tiled by τ with t at the corner iff there exists a tiling by τ_r.

Therefore no algorithm can solve the Domino Problem: If such an algorithm existed, it could be used, via the given transformation, to decide, given a tileset τ and a tile t, whether there exists tilings of arbitrary large squares by τ with t at the corner, which is undecidable by Corollary 2. □

3.4 Bibliographic notes

There exist a lot of different proofs of the Undecidability of the Domino Problem based on building first an aperiodic substitutive tileset and then encoding tilings with a fixed tile t inside the squares it produces. Notice that in theory, this proof doesn't need a substitutive tileset, but only a tileset from which we can extract arbitrary large squares. However all known variants of this proof uses substitutive (or more accurately near substitutive) tilesets.

The first proof of the kind is the proof by Berger [7]. Technically the construction of Berger is not a division into squares which overlap, but a division into infinite vertical strips of arbitrary width which can overlap, but the principle stays the same.

The proof we presented here is essentially from Robinson [47], except we used the tileset of Ollinger [43] as a basis instead of the tileset of Robinson. The tileset used in Robinson's article is very similar to Ollinger; it uses the same tiles as Figure 13 except that the green wires are now *centered*, i.e. the two colors "green on top" and "green on bottom" of the east/west side of the second layer are identified, and the same goes for the colors "green on left" and "green on right" on the north/-south side. Robinson also speaks in his note [46] of a set of 52 tiles, that was later published in an article of Poizat[45]. This set is the same as Figure 13 without the first layer. The fact that these two tilesets of Robinson look very close to Ollinger's tileset and the fact that Robinson himself hints at a set of 104 tiles in his note [46] suggests that the set discovered by Ollinger was already known to Robinson.

There exist a lot of literature explaining and reexplaining Robinson's tileset [28, 48, 2]. The tileset of Ollinger is explained in [43]. Further details that prove the tileset is substitutive and minimal can be found in the thesis of Ballier [5].

The simplifications made by Robinson lead to a smaller tileset but also have a major drawback: the tileset is not intrinsically substitutive, which makes the proof of its aperiodicity (and a complete description of the possible tilings) more difficult. The tileset of Ollinger is not the first intrinsically substitutive tileset that was discovered. In fact, Berger set of 103 tiles is intrinsically substitutive, although this was never proven. This property is also used very often to prove aperiodicity of tilesets in the plane $\mathbb{R}^2$, where it is sometimes called "the unique composition property", see e.g. [3, p. 2].

To finish, remark that the whole construction is based on the fact that an aperiodic substitutive tileset with some specific property exist, but the tileset we use (and similar tilesets used by Berger, Robinson, or others) appears somehow by miracle. Section 6 of this document explains how to prove using tools from computability theory and programming languages that such a tileset exist.

4 The construction of Aanderaa and Lewis

The construction by Aanderaa and Lewis we present here is not well known. As with all early constructions of an aperiodic tilesets, this tileset was built in the context of the undecidability of the $\forall\exists\forall$ fragment of first order logic, and is actually the first published example of an aperiodic *deterministic* tileset, i.e. a tileset s.t. any row determines uniquely all rows in the upper half plane above this row. This also gives a proof of the undecidability of the nilpotency of one-dimensional cellular automata.

The first proof of the idea was published by Aanderaa and Lewis in 1974 [1]; a better version may be found in the book by Lewis [36]. Our exposition here follows somewhat Lewis but the construction is actually quite different, in particular with the introduction of sofic shifts and p-adic numbers.

This construction is quite interesting as it uses only elementary number theory, and is actually building at first a one dimensional object rather than a two-dimensonial tiling.

While not stricly necessary, it is however recommended to have a good understanding of the basics of symbolic dynamics. We refer the reader to the book [38] and to other articles of this book.

4.1 Tiling problems on one-dimensional objects

4.1.1 Symbolic Dynamics

We briefly recall in this section classical notions from symbolic dynamics.

The concept of Wang tiles is not intrinsically two-dimensional, and one can define Wang tiles for tilings of the discrete line $\mathbb{Z}$ rather than the discrete plane $\mathbb{Z}^2$. This can also be generalized obviously to higher dimensions, or even to tilings of finitely generated groups:

Definition 11 (One-dimensional Wang tiles). Let C be a finite set, called the set of colors. A *Wang tile* t over C is a map from $\{E,W\}$ to C. A *tileset* τ is a finite set of Wang tiles.

Definition 12. A *tiling* of $\mathbb{Z}$ by τ assigns to each element of $\mathbb{Z}$ a tile from τ s.t. colors of adjacents tiles agree on their common border.

Formally, a tiling is a map $x : \mathbb{Z} \to \tau$ s.t.

- $\forall i, x(i)(E) = x(i+1)(W)$

We say that τ *tiles* $\mathbb{Z}$ if there exists a tiling of $\mathbb{Z}$ by τ.

Proposition 9. *The Domino Problem is decidable on $\mathbb{Z}$: there exists an algorithm, that on input a tileset τ, decides if τ tiles the discrete line $\mathbb{Z}$.*

The idea is very similar to the construction presented in Figure 5.

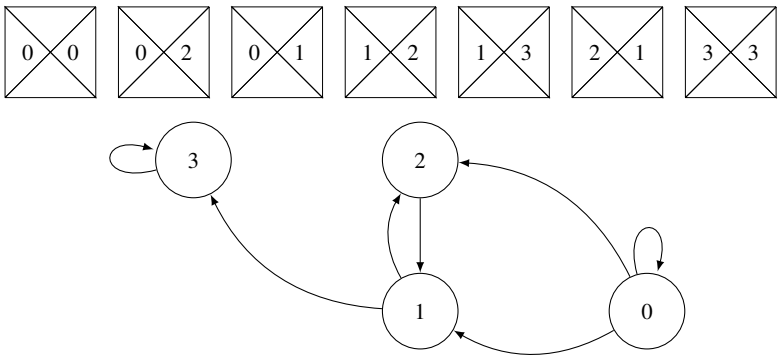


Fig. 19 The tileset τ_2 and its representation as a graph

Proof. Let τ be a one-dimensional tileset. Consider the (multi)-graph whose vertices are the colors C, and there is an edge from u to v if there exists a Wang tile with u on the west side and v on the east side. Then it is obvious that there exists a tiling by τ if and only if the graph G contains a cycle. See Figure 19 for an example. □

In the following, we will be interested in sofic shifts, which correspond to Wang tiles with an additional symbol. We defined them only for one-dimensional objects, although we will also need their two-dimensional variant:

Definition 13 (Decorated one-dimensional Wang tiles). Let C be a finite set, called the set of colors, and A a finite set, called an alphabet. A *decorated Wang tile* t over C is as 2-tuple (t',a) where t' is a Wang tile and $a \in A$ is the decoration. Given a decorated Wang tile $t = (t',a)$, we note $\pi(t) = a$. A *decorated tileset* τ is a finite set of decorated Wang tiles.

Tilings by decorated Wang tiles are defined exactly as tilings by Wang tiles.

Definition 14. If x is a tiling of $\mathbb{Z}$ by a decorated tileset τ, we write $\pi(x)$ the extension of π to words over the alphabet A. It is defined by $\pi(x)_i = \pi(x(i))$.

The sofic shift defined by τ, denoted X_τ is the set of all words $\pi(x)$, for x a tiling by τ.

An example of a decorated tileset τ, a tiling by τ, and the corresponding infinite word are given in Figure 20.

The term "shift" corresponds to the fact that the tilings are invariant by shift:

Definition 15. Let x be a bi-infinite word over an alphabet A. The shift of x, $\sigma(x)$, is the bi-infinite word defined by $\sigma(x)_i = x_{i+1}$

We define $\sigma^n(x)$ for $n \in \mathbb{Z}$ similarly.

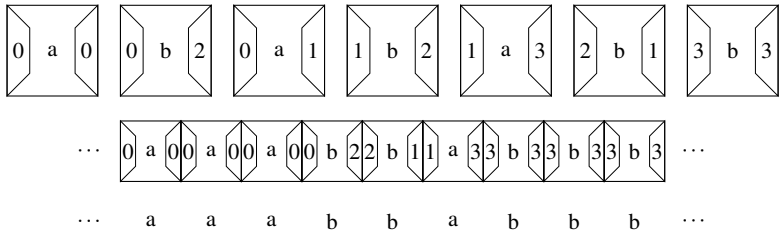


Fig. 20 The tileset τ_3, a tiling x by τ_3, and the corresponding bi-infinite word $\pi(x)$

Definition 16. A set of infinite words $X \subseteq A^{\mathbb{Z}}$ is a subshift if it is closed under shift ($x \in X \iff \sigma(x) \in X$) and topologically closed (for the product topology on $A^{\mathbb{Z}}$)

The subshift *generated* by a set T is the smallest subshift that contains T, i.e. the topological closure of $\{\sigma^n(x), x \in T, n \in \mathbb{Z}\}$

We refer to other articles in this volume for more reference about symbolic dynamics.

Proposition 10. *X_τ is a subshift.*

If we decorate the edges from the graph obtained in Figure 19 with the labels from Figure 20, we obtain a finite automaton such that the projections $\pi(x)$ correspond exactly to the biinfinite paths in this automaton.

In a sense, this means that the behaviour of one-dimensional sets of Wang tiles are entirely understood once we understand the theory of finite automata. In particular:

Proposition 11. *Let L be the set of finite words that can occur in some infinite word in X_τ. Then L is regular.*

Conversely, let L be a regular language. The set of infinite words, all factors of which are in L is a sofic shift.

See [38, Chapter 3] for more details.

4.1.2 Distance shifts

Now that we have the concept of a sofic shift, we can introduce the main construction of Aanderaa and Lewis. The core of the proof is based on the following definition:

Definition 17. Given a sofic shift $X \subseteq (A \times A)^{\mathbb{Z}}$, the *distance shift* X^Δ corresponding to X is the set of all pairs $(x, y) \in A^{\mathbb{Z}} \times A^{\mathbb{Z}}$ s.t. for all $i \in \mathbb{Z}$, $(x, \sigma^i(y)) \in X$.

The definition is of course symmetric in x and y: a pair (x, y) is in the distance shift iff for all $n, m \in \mathbb{Z}$, $(\sigma^n(x), \sigma^m(y)) \in X$.

We call this shift a distance shift, as it can compare patterns in x and in y that are at any distance from each other, just by shifting x or y to put them in the same position.

As an example, let X be the set of all words $(x,y) \in (\{0,1\} \times \{0,1\})^{\mathbb{Z}}$ with the following property: If $x_i = y_i = 1$ for some i, then $x = y$. It is an exercise left to the reader to show that X is a sofic shift.

What is the distance shift defined by X ? To know this, let $(x,y) \in (\{0,1\} \times \{0,1\})^{\mathbb{Z}}$ such that for all $n,m \in \mathbb{N}$, $(\sigma^n(x), \sigma^m(y)) \in X$.

- If $\forall i, x_i = 0$, then any y can do. That is, $(0^{\mathbb{Z}}, y) \in X^{\Delta}$ for all $y \in \{0,1\}^{\mathbb{Z}}$.
- If $\forall i, y_i = 0$, then any x can do. That is, $(x, 0^{\mathbb{Z}}) \in X^{\Delta}$ for all $x \in \{0,1\}^{\mathbb{Z}}$.
- Suppose that x contains only one occurrence of the letter 1, say in position i. Then it is easy to see that y should contain at most one occurrence of 1: shift y so that one of the occurences of 1 in y coincides with the only occurence in x: As $\sigma^n(y)_i = x_i = 1$ we get $\sigma^n(y) = x$.
- Similarly, if y contains only one occurence of the letter 1, then x contains at most one occurrence of the letter 1.
- Otherwise x and y contains at least two occurences of the letter 1. Then it is easy to see that x and y are periodic words. Indeed suppose that $x_i = x_j = 1$ and that $y_k = 1$. Then $\sigma^i(x)_0 = \sigma^k(y)_0 = 1$ and therefore $\sigma^k(y) = \sigma^i(x)$. Similary, $\sigma^j(x) = \sigma^k(y)$ and therefore $\sigma^i(x) = \sigma^j(x)$. With a bit more work, we can show that x and y are (up to shift) of the form $(0^p 1)^{\mathbb{Z}}$ for some p.

This easy example shows that the structure of X^{Δ} can already be quite complicated. In particular, if X is sofic, X^{Δ} might be far from sofic.

The result of Aanderaa and Lewis is the following:

Theorem 11 ([1, 36]). *There is no algorithm that decides, given a sofic shift X whether X^{Δ} is empty.*

The proof of this theorem will be given over the next sections.

4.1.3 Application to the Domino Problem

It is not obvious from the theorem how it can be applied for the undecidability of the Domino Problem. The key idea is the following: Starting from a sofic shift $X \subseteq (A \times A)^{\mathbb{Z}}$, we build a two-dimensional object X_2 s.t. for each configuration $(x,y) \in X$, the n-th row of the corresponding configuration in X_2 is the pair $(x, \sigma^n(y))$. The result is depicted on figure 21. It is easy to see that X_2 is indeed a sofic shift:

Proposition 12. *Let X be a one-dimensional sofic shift over an alphabet $A \times A$. Consider the following two-dimensional shift X_2:*

- *Every cell of the subshift is composed of two layers.*
- *The first layer contains a word $w \in A^{\mathbb{Z}^2}$ that is identical on all rows $w_{i,j} = x_i$ for some $x \in A^{\mathbb{Z}}$.*
- *The second layer contains a word $z \in A^{\mathbb{Z}^2}$ that is shifted on consecutive rows: $z_{i,j+1} = z_{i+1,j}$. In other words $z_{i,j} = y_{i+j}$ for some $y \in A^{\mathbb{Z}}$*

x_0,y_9	x_1,y_{10}	x_2,y_{11}	x_3,y_{12}	x_4,y_{13}	x_5,y_{14}	x_6,y_{15}	x_7,y_{16}	x_8,y_{17}	x_9,y_{18}
x_0,y_8	x_1,y_9	x_2,y_{10}	x_3,y_{11}	x_4,y_{12}	x_5,y_{13}	x_6,y_{14}	x_7,y_{15}	x_8,y_{16}	x_9,y_{17}
x_0,y_7	x_1,y_8	x_2,y_9	x_3,y_{10}	x_4,y_{11}	x_5,y_{12}	x_6,y_{13}	x_7,y_{14}	x_8,y_{15}	x_9,y_{16}
x_0,y_6	x_1,y_7	x_2,y_8	x_3,y_9	x_4,y_{10}	x_5,y_{11}	x_6,y_{12}	x_7,y_{13}	x_8,y_{14}	x_9,y_{15}
x_0,y_5	x_1,y_6	x_2,y_7	x_3,y_8	x_4,y_9	x_5,y_{10}	x_6,y_{11}	x_7,y_{12}	x_8,y_{13}	x_9,y_{14}
x_0,y_4	x_1,y_5	x_2,y_6	x_3,y_7	x_4,y_8	x_5,y_9	x_6,y_{10}	x_7,y_{11}	x_8,y_{12}	x_9,y_{13}
x_0,y_3	x_1,y_4	x_2,y_5	x_3,y_6	x_4,y_7	x_5,y_8	x_6,y_9	x_7,y_{10}	x_8,y_{11}	x_9,y_{12}
x_0,y_2	x_1,y_3	x_2,y_4	x_3,y_5	x_4,y_6	x_5,y_7	x_6,y_8	x_7,y_9	x_8,y_{10}	x_9,y_{11}
x_0,y_1	x_1,y_2	x_2,y_3	x_3,y_4	x_4,y_5	x_5,y_6	x_6,y_7	x_7,y_8	x_8,y_9	x_9,y_{10}
x_0,y_0	x_1,y_1	x_2,y_2	x_3,y_3	x_4,y_4	x_5,y_5	x_6,y_6	x_7,y_7	x_8,y_8	x_9,y_9

Fig. 21 A portion of the shift X_2

- *The word in each row is in X.*

Then X_2 is sofic and each row of X_2 is composed of pairs (x,y) s.t. for all j, $(x,\sigma^j(y)) \in X$.

Furthermore, there exists an algorithm that can obtain a set of decorated Wang tiles for X_2 from a set of decorated Wang tiles for X.

In fact the j-th row of X_2 proves that $(x,\sigma^j(y)) \in X$. Therefore X_2 is empty iff X^Δ is empty.

Corollary 3. *The Domino problem is undecidable.*

Notice that, even taking into account the projection π that deleted the colors, the tilings we obtain using the Aanderaa-Lewis method are quite different from the tilings of the previous section.

In fact, the construction used above can give us a stronger result if done correctly.

Recall that each row should be part of a sofic shift X, which means they correspond to decorations of Wang tiles. We represent in Figure 22 the shift X_2 with the Wang tiles these decorations are part from.

Now it is well known that each sofic shift can be obtained from a *right-resolving* set of Wang tiles, that is a set of Wang tiles where the colors on the east side is entirely determined from the colors on the west side and the label of the tile (This is obtained by the usual process of determinization of a finite automaton).

If this is the case, the tiling we obtain has a deterministic property: the half plane at the right of the line is entirely determined by the line. In fact, the symbols at the black position are entirely determined by the three gray symbols that are immediately above and at its left: x_4 is obtained from the tile on the top, y_8 from the tile at the top left, z_4^4 by the tile at the left, and z_4^5 is uniquely determined knowing the three others.

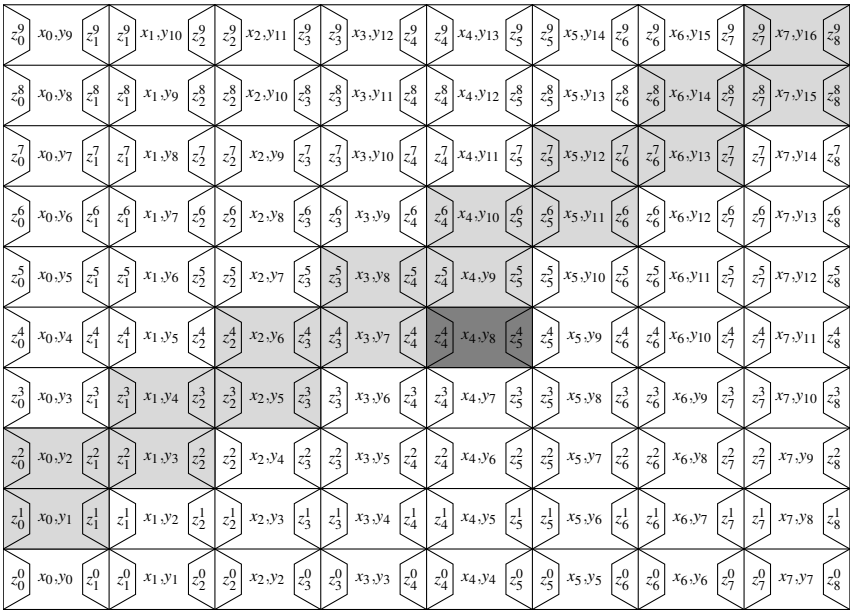


Fig. 22 A tiling from X_2 with the (one-dimensional) Wang tiles drawn

This means that we can convert the picture and the shift X_2 into a north-west deterministic set of Wang tiles. (A set of Wang tiles is north-west deterministic if the colors on the east and south part of the Wang tile are uniquely determined by the two other colors.) It is then easy to see that Theorem 11 implies the following statements:

Theorem 12. *The Domino problem for north-west deterministic set of Wang tiles is undecidable*

Corollary 4. *The nilpotency problem is undecidable for one-dimensional cellular automata.*

The classical proof of this result is attributed to Kari [30] but in fact this result is already mentioned in the original article of Aanderaa and Lewis [1].

The rest of the section is devoted to the proof of Theorem 11. As the proof is quite long and difficult, we will focus on proving that there exists a nonempty distance shift with no periodic points, which implies that there exists a two-dimensional aperiodic tileset. The modification to obtain the undecidability result will be mentioned briefly at the end.

The general idea is to find a set S s.t. if $x, y \in S$, then the subword of x that corresponds to the position where it differs from y (assuming this set of positions is biinfinite) is itself in S. By comparing in particular x with $\sigma^n(x)$, this implies in particular some kind of hierarchical structure in x.

It is easy, with a sofic shift or equivalently with a finite automaton, to isolate the positions where x and y differs. Therefore there is some hope to be able to define such a set S as a distance shift (more precisely, $S \times S$ will be our distance shift).

The set S that will work is the subshift S_p we build below that corresponds to a Toeplitz subshift based on p-adic numbers. There will be an additional difficulty in the proof: if $x, y \in S_p$, then the subword of x that corresponds to the position where it differs from y (assuming this set of positions is biinfinite) is "almost" in S_p. What this "almost" part means is quite technical and will be describe later.

We will first describe this set S_p and how it related to p-adic numbers. We will then describe the hierarchical properties it satisfies. Finally, we will explain why the difference between two words of S_p is almost a word in S_p and how to use this to build our distance shift. This distance shift, when passed through the two-dimensional machinery explained in the few previous pages, will gives us a two-dimensional aperiodic tileset.

We will finish by briefly indicating what remains to be done to obtain the undecidability of the emptiness of a distance shift, and therefore the undecidability of the Domino problem

4.2 The subshift S_p and p-adic numbers

Let p be an integer Let $a_p(n)$ be the first non-zero digit in the expansion of n in base p. That is, $a_p(n) = r$ if $n = q \times p^{k+1} + p^k r$ for some q and $r \neq 0 \mod p$. For example $a_{10}(1664) = 4$, $a_{10}(71500) = 5$ and $a_{10}(71050) = 5$.

Note that $a_p(0)$ is not well defined. We define it as taking all values in the alphabet $\{0, 1, \ldots, p-1\}$ (this makes a_p a multivalued function).

We will be interested in the hierarchical structure of the word u defined by $u_n = a_p(n)$, and of all other words similar to u.

For reference, here are the 80 digits of u around 0 when $p = 3$ with the arbitary choice $a_3(0) = 2$.

22122121122122121122121121122122121122122.121122121121122122121122121121122121121

The main part of the construction of Aanderaa and Lewis is to build the subshift generated by this point, for large values of p. To understand how to do it, we first have to understand this subshift.

For this we need the notion of p-adic integers. p-adic integers are formal sums

$$m = \sum_{k \geq 0} m_k p^k$$

with $m_k \in \{0,\dots,p-1\}$. p-adic integers form a group under addition, where addition is done componentwise with propagation of carry, and even a ring with the multiplication defined the usual way. The integers $\mathbb{Z}$ can be seen as a subset of this group: Nonnegative integers correspond to formal sums where all but finitely many terms are 0. Negative integers correspond to formal sums where all but finitely many terms are equal to $p-1$. The ring of p-adic integers will be denoted $\mathbb{Z}_p$. It has the natural, discrete topology: $d(n,m) = p^{-\min\{i,n_i\neq m_i\}}$ which makes it a compact space, in which $\mathbb{Z}$ is dense.

The function a can be extended to $\mathbb{Z}_p$ in the natural way: For $m \in \mathbb{Z}_p$, let i be the minimal position s.t. $m_i \neq 0$. Then $a_p(m) = m_i$. It is easy to see that a_p is a continuous function[2].

Proposition 13. *Let p be an integer and S_p the subshift generated by the sequence u with $u_i = a_p(i)$. S_p contains exactly all points $i \to a_p(m+i)$ for $m \in \mathbb{Z}_p$.*

Proof. For $m \in \mathbb{Z}_p$, let u^m be the sequence defined by $(u^m)_i = a_p(m+i)$ and let $X = \{u^m, m \in \mathbb{Z}_p\}$. By continuity of a_p, we get that if $m_k \xrightarrow{k\to\infty} m$ then $u^{m_k} \xrightarrow{k\to\infty} u^m$, so X is closed. In particular X is a subshift and thus $S_p \subseteq X$.

For $n \in \mathbb{Z}, u^n = \sigma^n(u)$ and therefore $u^n \in S_p$ as S_p is shift-invariant. If $m \in \mathbb{Z}_p$ then $m = \lim m_k$ for $m_k \in \mathbb{Z}$ and therefore $u^m = \lim u^{m_k} \in S_p$ as S_p is topologically closed. Therefore $X \subseteq S_p$, and thus $X = S_p$. □

Notice that all the points $i \to a_p(m+i)$ are distinct. This gives a *continuous* map:

$$\begin{aligned} f : S_p &\mapsto \mathbb{Z}_p \\ x &\to f(x) \end{aligned}$$

s.t. $f(\sigma(x)) = f(x)+1$. This makes our dynamical system an almost 1-1 extension of the odometer (the dynamical system $(\mathbb{Z}_p, T)$ on the odometer defined by $T(x) = x+1$). The extension is almost 1-1 due to the fact that all points of $\mathbb{Z}_p$ except the integers have only one preimage.

Here are a few useful properties of a_p:

- $a_p(pm) = a_p(m)$ for all $m \in \mathbb{Z}_p$.
- $a_p(m+p) = a_p(m)$ if $m \neq 0 \mod p$
- $a_p(m+p^k) = a_p(m)$ if $m \neq 0 \mod p^k$

Notice that $m = 0 \mod p$ makes sense not only for $m \in \mathbb{Z}$ but also for $m \in \mathbb{Z}_p$.

The following proposition will be useful:

Proposition 14. *Let x be a word s.t. for all k, there exists $n_k \in \mathbb{Z}$ s.t. $x_i = a_p(n_k+i)$ if $n_k + i \neq 0 \mod p^k$. Then there exist $m \in \mathbb{Z}_p$ s.t. $x_i = a_p(m+i)$ for all i. In particular $x \in S_p$.*

[2] This statement is somewhat wrong, as a_p is a multivalued function at 0. An exact statement would be that a_p is upper hemicontinuous: If the sequence y^n converges to y, then $a_p(y)$ contains (but may contain more) all limit points of $a_p(y^n)$). The fact that a_p is multivalued is a burden that is not worth the hassle, and we urge the reader to consider a_p as a normal, single valued function, even though it is not.

Proof. Let $j \in]-p^k/2, p^k/2]$ s.t. $j = -n_k \mod p^k$. Let $w = x_j$ be the character at position j. By replacing n_k by $n_k + p^k r$ for some suitable r we can assume wlog that $j + n_k = p^{k+1}w$. In particular $a_p(j + n_k) = w = x_j$.

Therefore, $x_i = a_p(i + n_k)$ for $i \in]-p^k/2, p^k/2]$.

Let m be a limit point of the sequence $(n_k)_{k\in\mathbb{N}}$. Then by construction $x_i = a_p(m + i)$ for all i. □

4.3 S_p as a Toeplitz subshift

S_p has also the structure of a Toeplitz subshift (as all symbolic almost 1-1 extensions of odometers). Let us recall what the point $u \in S_p$ looks like when $p = 5$:

1234112342123431234412341123411234212343123441234212341123421234312344123431234112342123431234412344

By definition of a_p, $a_p(n + p) = a_p(n)$ if $n \neq 0 \mod p$. So if we delete one symbol every p symbols in u, we obtain a periodic point.

This can be formalized in the following way:

Definition 18. If x is a sequence on the alphabet A, the p-skeleton of x denoted $\text{skel}_p(x)$ is the sequence over the alphabet $A \cup \{\star\}$ defined by :

$$\text{skel}_p(x)_i = \begin{cases} x_i \text{ if } \forall n, x_{i+np} = x_i \\ \star \text{ otherwise} \end{cases}$$

Intuitively, and in what follows, the $\star$ symbol represents a symbol the value of which is unknown. The 5-skeleton of u for $p = 5$ is therefore:

..1234⋆1234⋆1234⋆1234⋆1234⋆1234⋆1234⋆1234⋆1234⋆1234⋆1234⋆1234⋆1234⋆1234⋆1234⋆1234⋆1234⋆1234⋆1234⋆1234..

It is easy to see that any word of S_p actually has the p-skeleton:

$$\ldots 12\ldots p'\star 12\ldots p'\star 12\ldots p'\star 12\ldots p'\star 12\ldots p'\star 12\ldots p'\star 12\ldots p'\star 12\ldots p'\star 12\ldots p'\star 12\ldots p'\star 12\ldots p'\star 12\ldots$$

with $p' = p - 1$. Furthermore, any word of S_p has the following p^2-skeleton (the differences with the p-skeleton have been highlighted)

$$\ldots 12\ldots p'\mathbf{1}12\ldots p'\mathbf{2}1\ldots\ldots\ldots p'\mathbf{p'}12\ldots p'\star 12\ldots p'\mathbf{1}12\ldots p'\mathbf{2}1\ldots\ldots\ldots p'\mathbf{p'}12\ldots p'\star 12\ldots p'\mathbf{1}12\ldots p'\mathbf{2}1\ldots p'\mathbf{p'}12\ldots p'\star 12\ldots$$

As the p-skeleton contains only the periodic part of a word, we will be interested mainly in the complement of the p-skeleton:

Definition 19. If x is a sequence on the alphabet A, the p-coskeleton of x denoted $co\text{skel}_p(x)$ is the sequence over $A \cup \{\_\}$ defined by :

$$co\text{skel}_p(x)_i = \begin{cases} \_ \text{ if } \forall n, x_{i+np} = x_i \\ x_i \text{ otherwise} \end{cases}$$

The *reduced* p-coskeleton is the image of $cos\text{kel}_p(x)_i$ under the morphism h defined by $h(a) = a$ for $a \in A$ and $h(\_) = \varepsilon$

It is a bit painful to define exactly how to apply a morphism on a biinfinite word. Note however that applying an erasing morphism (like h) to a biinfinite word might produce in some cases a finite word, or a word that is finite in one direction and infinite in the other direction.

We go back to the running example:

Its 5-coskeleton is:

and its reduced 5-coskeleton is:

In fact, the following proposition is an easy consequence of the fact that $a_p(pn) = a_p(n)$, and the proof is left to the reader:

Proposition 15. *Let $x \in S_p$. Then the reduced p-coskeleton of x is also in S_p.*

This gives us an easy way to show that a word $x \in S_p$:

Proposition 16. *Recall that $p' = p-1$ and let S be a set of configurations s.t.:*

1. *Every $x \in S$ has the form $(12\dots p'\star)^{\mathbb{Z}}$. That is, there exists $n \in \mathbb{Z}$ s.t. $x_{i+n} = a_p(i)$ if $i \neq 0 \mod p$.*
2. *The word x' defined by $x'_i = x_{pi+n}$ is in S with n as defined in 1.*

Then $S \subseteq S_p$.

Notice that x' is exactly the reduced p-coskeleton of x.

Proof. We first prove by induction on k that if $x \in S$, then there exists some $n \in [0, p^k - 1]$ s.t. $x_{i+n} = a_p(i)$ for $i \neq 0 \mod p^k$. This is true for $k = 1$ by definition. Now suppose the result is true for k and let $x \in S$ and x', n as in the definition. We can suppose wlog that $n \in [0, p-1]$. By induction hypothesis, there exists $n' \in [0, p^k - 1]$ s.t. $x'_{i+n'} = a_p(i)$ for $i \neq 0 \mod p^k$. Let $N = n + pn' \in [0, p^{k+1} - 1]$. Then $x_{i+N} = a_p(i)$ for all $i \neq 0 \mod p^{k+1}$ which ends the induction.

We now conclude by Proposition 14.

□

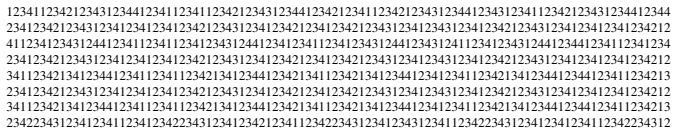


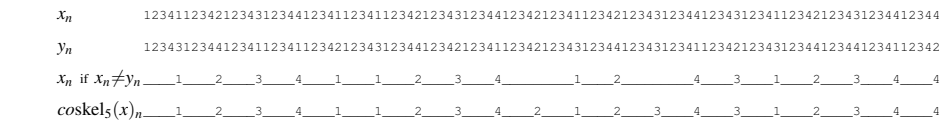
Fig. 23 In the first row, the sequence a_5. In the second row, the values of $a_5(1+n)$ when it differs from $a_5(n)$ (values that are equal are not shown) In the third row, the values of $a_5(3+n)$ when it differs from $a_5(n)$ The other rows correspond to the choices 10, 25, 30 and 50.

4.4 $S_p \times S_p$ *as a distance shift*

Our goal now is to prove that $S_p \times S_p$ is a distance shift, which means that we can somehow decide if $(x,y) \in S_p \times S_p$ by local rules involving x and all possible shifts of y.

The previous proposition is a first step toward a solution. The problem is that there is no easy way to extract the p^k-coskeleton of a configuration easily. There is however a workaround. It turns out that, if we compare two points $x,y \in S_p$, then the set of positions where they differ looks like the p^k-coskeleton for some k.

Let's look for example at the point x defined by $x_n = a_5(n)$ and the point y defined by $y_n = a_5(n+10)$. The third row shows x when it differs from y.



The only difference between the two last sequences is that some symbols are missing (i.e. replaced by _) in the third sequence. The same is true more generally and is illustrated in Figure 23.

Definition 20. If x and y are two configurations on alphabet A, the difference of x and y, denoted by $\mathrm{diff}(x,y)$ is the configuration on alphabet $A \cup \{\_\}$ defined by

$$\mathrm{diff}(x,y)_i = \begin{cases} \_ & \text{if } x_i = y_i \\ x_i & \text{otherwise} \end{cases}$$

It is of course obvious that most symbols should be the same in x and y. If $x_i = a_p(i)$ and $y_i = a_p(i+p^k)$ then all positions of valuation less than k will be identical in x and y.

Proposition 17. *Let* $x_i = a_p(i)$ *and* $y_i = a_i(i+p^k)$.

Then $\mathrm{diff}(x,y) \subseteq co\mathrm{skel}_{p^k}(x)$*, in the sense that* $\mathrm{diff}(x,y)_i = co\mathrm{skel}_{p^k}(x)$ *whenever* $\mathrm{diff}(x,y)_i \neq \_$.

The above examples seem to suggest that there are only few positions where the reverse does not hold. The following lemma explains, in the case of k of valuation 0, that at most 4 symbols of the 1-skeleton in every block of size p^2 will be replaced by the symbol _ in $\text{diff}(x,y)$.

Lemma 1. *Let $k \neq 0 \mod p$. If $a_p(i+k) = a_p(i)$ then*

- $i = 0 \mod p^2$ *or*
- $i = -k \mod p^2$ *or*
- $i = pk \mod p^2$ *or*
- $i = -kp - k \mod p^2$.

In particular, given any $k \neq 0 \mod p$ there are (at most) four values of $i \mod p^2$ s.t $a_p(i+k) = a_p(i)$ and no three of them are consecutive.

Proof.
- If $i+k \neq 0 \mod p$ and $i \neq 0 \mod p$, then $a_p(i) = i \mod p$ and $a_p(i+k) = i+k \mod p$ and therefore $a_p(i) \neq a_p(i+k)$.
- If $i+k = 0 \mod p$ but $i+k \neq 0 \mod p^2$. Then $a_p(i+k) = (i+k)/p \mod p$ and $a_p(i) = i \mod p$. Therefore we obtain $i = (i+k)/p \mod p$ and $i = -k(p+1) \mod p^2$.
- If $i = 0 \mod p$ but $i \neq 0 \mod p^2$ we get similarly $i+k = i/p \mod p$ and $i = pk \mod p^2$

□

Corollary 5. *Let $m,n \in \mathbb{Z}_p$ s.t $m \neq n$ but m and n coincide exactly on their first r digits, i.e. $m-n = kp^r$ with $k \neq 0$.*

If $a_p(m+i) = a_p(n+i)$ then

- $i \neq -m \mod p^r$ *or*
- $i = -m \mod p^{r+2}$ *or*
- $i = -n \mod p^{r+2}$ *or*
- $i = -n + p(m-n) \mod p^{r+2}$ *or*
- $i = -m - p(m-n) \mod p^{r+2}$.

The proof is an easy generalization of the lemma and is left as an exercise. The corollary states the following: For any x,y, $\text{diff}(x,y)$ looks almost like the p^r-coskeleton of x, except $\text{diff}(x,y)$ might contain at most 4 additional symbols replaced with a symbol _ compared to the coskeleton.

Now we have seen that, if we delete the symbol _ from the p^r-coskeleton of x, we obtain an element of S_p, and in particular an infinite concatenation of words of the form:

$$12\dots p'112\dots p'21\dots p'p'12\dots p'\star$$

where $\star$ denotes any symbol.

As a consequence, if we delete the _ symbol from $\text{diff}(x,y)$, we also obtain infinite concatenations of words of the same form, except that 4 symbols might be missing in each block.

This is formalized in the next proposition.

Proposition 18. *Let W be the collection of blocks of size p^2 of the form*

$$4 \dots p' \; 1 \; 1 \; 2 \dots p' \; 2 \; 1 \dots p' \; p' \; 1 \; 2 \dots p' \star 1 \; 2 \; 3$$

where $\star$ denotes any symbol. Let $\hat{W}$ be the set of subwords of words of W of length at least p^2-4.

Let $x, y \in S_p$ and suppose that x and y differ in more than one position.

Then $h(\mathrm{diff}(x,y)) \in (\hat{W})^{\mathbb{Z}}$ where h is the erasing morphism: $h(\_) = \varepsilon$ and $h(a) = a$ otherwise.

In particular, let $x \neq y \in S_p$. Then the image by h of every factor of $\mathrm{diff}(x,y)$ is a factor of $(\hat{W})^{\mathbb{Z}}$

The words x and y may differ in only one position when $x_i = y_i = a_p(n+i)$ with $n \in \mathbb{Z}$ rather than $n \in \mathbb{Z}_p$.

It would seem more natural to change the words in W so that the factor 123 appears at the beginning rather than at the end. The version we use is essentially cosmetic but greatly simplifies the proof.

What is interesting and nontrivial is that this proposition is a characterization of words in S_p.

Definition 21. We say that a pair of words (x,y) satisfies property $\mathscr{P}$ if every factor of $h(\mathrm{diff}(x,y))$ (resp. $h(\mathrm{diff}(y,x))$) is a factor of $(\hat{W})^{\mathbb{Z}}$.

Notice that "every factor of $h(\mathrm{diff}(x,y))$ is a factor of $(\hat{W})^{\mathbb{Z}}$" is the same as "$h(\mathrm{diff}(x,y)) \in (\hat{W})^{\mathbb{Z}}$" when $h(\mathrm{diff}(x,y))$ is a biinfinite word, however it also makes sense even if $\mathrm{diff}(x,y)$ is a finite word, or is simply infinite.

We are now in position to state the main theorem:

Theorem 13. *Let $p > 6$.*

Let x, y be two words over the alphabet $\{1, \dots, p-1\}$ s.t.

- $x \in W^{\mathbb{Z}}$
- $y \in W^{\mathbb{Z}}$
- *For all k, the words x and $\sigma^k(y)$ have property $\mathscr{P}$*

Then $x \in S_p$ and $y \in S_p$.

The theorem is an easy consequence of the following proposition:

Proposition 19. *Suppose that x and y are composed of blocks of size p of the form*

$$1 \; 2 \dots p' \star$$

and that for all k, the words x and $\sigma^k(y)$ have property $\mathscr{P}$. Then $x \in W^{\mathbb{Z}}$ and $y \in W^{\mathbb{Z}}$

Proof of the Theorem. Let $\mathscr{S}$ be the set of all configurations x s.t. there exists y s.t.:

- $x \in (12 \dots p' \star)^{\mathbb{Z}}$
- $y \in (12 \dots p' \star)^{\mathbb{Z}}$
- For all k, the words x and $\sigma^k(y)$ have property $\mathscr{P}$

Let $x \in \mathscr{S}$ and y that witnesses it. Upto shift, we may suppose that $y_i = x_i = i \mod p$ for $i \neq p$. Let $x'_i = x_{pi}$ and $y'_i = y_{pi}$.

By the previous proposition, $x \in W^{\mathbb{Z}}$ which implies that $x' \in (12\dots p'\star)^{\mathbb{Z}}$, and the same is true for y'.

Furthermore, we can prove that x' and y' satisfy property $\mathscr{P}$. Indeed we have $h(\mathrm{diff}(x', \sigma^k(y'))) = h(\mathrm{diff}(x, \sigma^{pk}(y)))$ as x and y have the same p-skeleton and therefore disagree only on the positions that are inside x' and y'.

Therefore we have proven that $x' \in \mathscr{S}$. We conclude by Proposition 16. □

Proof of the Proposition. The proof of the proposition is quite technical and should be skipped at first read.

The various depictions here suppose that $p = 10$, that is $p - 1 = 9$. The general pictures can be obtained by replacing all occurrences of 9 with "$p-1$" and all occurrences of 8 with "$p-2$".

By taking shifts of x and y we can suppose wlog that $x_i = i \mod p$ for $i \neq 0 \mod p$, and similarly for y. We will prove by comparing x and $\sigma^2(y)$ that x and y almost have the predescribed structure. Considering x and $\sigma^{-2}(y)$ will finish the proof.

First we compare x and $\sigma^2(y)$:

$i \mod p$	4	…	8	9	0	1	2	3
x	4	…	8	9	$\star$	1	2	3
$\sigma^2(y)$	6	…	$\star$	1	2	3	4	5
$z = \mathrm{diff}(x, \sigma^2(y))$	4	…	?	9	?	1	2	3

The $\star$ symbol corresponds to positions in x and $\sigma^2(y)$ where the value is currently unknown. The only symbols in $z = \mathrm{diff}(x, \sigma^2(y))$ that could be equal to the symbol _ are the question mark symbols. By assumption, we know that $h(z) \in (\hat{W})^{\mathbb{Z}}$. This gives us a decomposition of z into blocks: write $z = \dots w_{-1}w_0w_1 \dots$ s.t. $h(w_i) \in \hat{W}$.

It should be obvious that the only possibility is for each w_i to be of size exactly p^2, and to be of the form $45\dots p'\star 45\dots\dots p'\star 123$. Indeed, z contains the factor "34" periodically with period p, and this factor appear at most $p-1$ times inside a word of $\hat{W}$ (which is of length between p^2-4 and p^2), so at least one out of every p occurrences of the factor "34" should correspond to the last letter of a word in $\hat{W}$ followed by the first letter of a word in $\hat{W}$, which proves the result.

We now look at each word w_i independently. We will look at the situation from the point of view of editing distance:

w_i	4	5	…	?	9	?	1	2	…	?	9	?	1	…	…	?	9	?	1	2	…	9	?	1	2	3
t	4	5	…	8	9	1	1	2	…	8	9	2	1	…	…	8	9	9	1	2	…	9	$\star$	1	2	3

The question marks inside w_i represents symbols that are either equal to the symbol of x, or are equal to the symbol _. The word t is a word of W. We know that w_i, when deleting the _ symbols, is equal to a word of $\hat{W}$, i.e. equal to the word t with at most 4 symbols removed. In other words, we should delete at most 4 symbols

from w_i, and that can only be done in the places where the _ symbol appears, and add at most 4 symbols to obtain the word t depicted.

This is only possible if every symbol of w_i which is not a _ symbol has the same value that the symbol at the same position in t.

We now look at the p^2 symbols of x that correspond to w_i. We have just said that any symbol of x that was not changed into a symbol _, that is, every position in x that contains a different value that $\sigma^2(y)$, has the same symbol as t. Now looking back at the table, the only possibility for a $\star$ symbol in x (i.e. a symbol for which we did not know previously the value) to become a _ symbol, if if this symbol was actually a symbol 2.

We have therefore proven that the p^2 symbols of x that correspond to w_i are of the form

$$4\,5\ldots 8\,9\,\underline{1}\,1\,2\ldots 8\,9\,\underline{2}\,1\ldots\ldots 9\,\underline{9}\,1\,2\ldots 9\star 1\,2\,3$$

where at most 4 of the underlined symbols might actually have been replaced by the symbol 2. Thus x is a concatenation of blocks of W where at most four symbols might have been replaced with the symbol 2.

We now start the reasoning again with x and $\sigma^{-2}(y)$, which proves that x is a concatenation of blocks of W, where at most four symbols might have been replaced with the symbol $p-2$.

Both situations cannot happen simultaneously unless no symbols were actually replaced, i.e. $x \in W^{\mathbb{Z}}$. □

Corollary 6. *There exists a sofic shift X s.t. the distance shift X^{Δ} corresponding to X is exactly $S_p \times S_p$. In particular the distance shift has no periodic point and therefore gives rise to an aperiodic 2-dimensional SFT.*

Proof. The sofic shift is the set of all x, y s.t.

- $x \in W^{\mathbb{Z}}$
- $y \in W^{\mathbb{Z}}$
- (x,y) satisfy property $\mathscr{P}$.

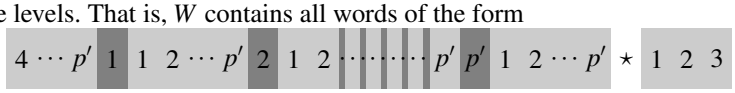
The only nontrivial part of this statement might be that the set of all (x,y) that satisfy property $\mathscr{P}$ is a sofic shift.

Indeed, let $\mathscr{W}$ be the language of $\hat{W}^{\mathbb{Z}}$, i.e. the set of all factors of words in $\hat{W}^{\mathbb{Z}}$. $\mathscr{W}$ is easily seen to be a regular language as $\hat{W}$ is finite. Now let $\mathscr{L}$ be the set of all words (u,v) over the alphabet $(A \times A)$ s.t. $h(\mathrm{diff}(u,v)) \in \mathscr{W}$. Then $\mathscr{L}$ is a regular language as the inverse image by a morphism of a regular language. The set of (x,y) all factors of which are in $\mathscr{L}$ is therefore a sofic shift. □

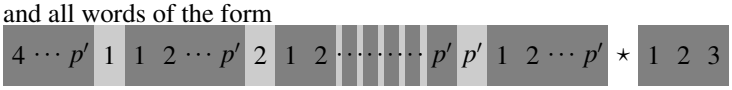
4.5 Undecidability for distance shifts

In this section we will briefly describe what should be changed to prove that there is no algorithm that decide if a distance shift is empty.

The subshift X we build in the previous corollary codes the distance shift $S_p \times S_p$. Now suppose that the letters are colored with two colors, gray and light gray. We do the same construction, but taking for W all words where color alternates between the levels. That is, W contains all words of the form



and all words of the form



and we take as a sofic subshift the set X of all pairs (x,y) s.t.:

- x and y are of the form $\left(1\ 2 \cdots p' \star\right)^{\mathbb{Z}}$
- (x,y) satisfy property $\mathscr{P}$ (for this new set W)

The reader should be convinced that the previous reasoning go through and that the distance shift defined by X is exactly the same as $S_p \times S_p$ except that symbols are colored depending on their valuation (remember that the valuation of an integer n , or a p-adic number n , is k if p^k is the greatest integer that divides n). More precisely if $x_i = a_p(i+n)$ for $n \in \mathbb{Z}_p$, then x_i is light gray if $i+n$ is of valuation k for k even, and dark gray otherwise.

We can go a bit further. Let B be a finite alphabet. Let $P \subseteq B \times B$. We now look at words over the alphabet $\{1,2,\dots,p-1\} \times B$. Symbols in $\{1,2,\dots,p-1\}$ and symbols and B will be written in two different layers to simplify the exposition.

Consider the following set of words W:

$$\begin{array}{l} 4 \dots\ p'\ 1\ 1\ 2 \dots\ p'\ 2\ 1 \dots\ p'\ p'\ 1\ 2 \dots\ p' \star 1\ 2\ 3 \\ a \dots\ a\ b\ a\ a \dots\ a\ b\ a \dots\ a\ \ b\ a\ a \dots\ a \star a\ a\ a \end{array}$$

for all pairs $(a,b) \in P$.

And we take as a sofic subshift the set X of all pairs (x,y) s.t.:

- x and y are of the form

$$\begin{array}{c} 1\ \ 2\ \dots\ p' \star\ 1\ \ 2\ \dots\ p' \star\ 1\ \ 2\ \dots\ p' \star \dots \\ a_0\ a_0 \dots\ a_0 \star a_0\ a_0 \dots\ a_0 \star a_0\ a_0 \dots\ a_0 \star \dots \end{array}$$

- (x,y) satisfy property $\mathscr{P}$ (for this new set W)

Now the distance shift defined by this new set X as the following form: Its elements are pairs $(x,y) \in S_p \times S_p$. Furthermore, if $x_i = a_p(i+n)$ then:

- x_i has the symbol a_0 if $i+n$ is of valuation 0
- All positions j s.t. $j+n$ is of valuation $k \in \mathbb{N}$ contain the same symbol a_k
- For all k, $(a_k, a_{k+1}) \in P$

Now suppose that B are one-dimensional Wang tiles and that P codes the adjacency relation: $(t,t') \in P$ if t' can be placed on the right of t.

Then it is is easy to see that this distance shift codes somehow a tiling of $\mathbb{N}$ by this set of Wang tiles.

It seems a strange idea to code a tiling in this complicated way, but we have gained something here: we were able to fix the tile that should be in position 0 (namely the symbol a_0). This is fundamental: As we saw earlier, this is precisely the difficulty in proving the undecidability of the domino problem: find a way to fix the symbol at position $(0,0)$.

To obtain the undecidability of the domino problem, we therefore have to find a way to do the same construction but for two-dimensional Wang tiles.

The idea is to construct a distance shift that code $S_p \times S_q$ for two different (relatively prime) values p and q. Using the same idea as explained before, we will be able to guarantee that all positions of valuation k for p and k' for q have the same symbol $a_{k,k'}$, that the symbol $a_{0,0}$ is the predescribed symbol, and that the pairs $(a_{k,k'}, a_{k+1,k'})$ and $(a_{k,k'}, a_{k,k'+1})$ satisfy the horizontal and vertical rules.

Bibliographic notes

Our presentation differs greatly from the previous presentations by Aanderaa-Lewis [1] and Lewis [36] In fact, the concept of p-adic numbers, of subshifts and sofic shifts are not present in the original articles. Aanderaa and Lewis introduce in particular the notion of perfect tilings, normal tilings and admissible tilings; The admissible tilings they define in a awkward way are just the elements of the subshift generated by the normal tilings.

There are important differences between the construction of Aanderaa-Lewis and the construction we gave here. In fact, the concept of distance shift that they use are called *sampling systems*, and are *apriori* weaker than distance shifts. As a consequence, they are not able to obtain a construction for something as simple as $S_p \times S_p$.

The construction they obtain correspond more or less to $T_p \times T_p$ where T_p is the subshift generated by the word u defined by $u_n = b_p(n)$ where $b_p(n)$ are the *last two* nonzero coefficients in the expansion of n in base p, i.e. $b_{10}(13370) = 37$ and $b_{10}(13070) = 07$. The subshift they build is actually bigger than $T_p \times T_p$: Each cell n contains in theory the last two nonzero coefficients in the expansion of n in base p, but the second-to-last coefficient (the "3" in $b_{10}(13370) = 37$) can actually differ from his normal value, but in a controlled way.

5 The construction of Kari

We now present the construction by Kari [31] for which the technique is very different from the ones of Berger and Anderaa-Lewis.

We will first present a construction of an aperiodic tileset that was initially designed to have a very small number of tiles, and then we will explain what to change to obtain the undecidability of the Domino Problem. The presentation here is inspired by the two groundbreaking articles of Kari [31, 32], see also [16].

While the Anderaa-Lewis construction can be explained through p-adic numbers, the core of Kari's construction is to represent numbers in a bi-infinite way so that each row of the tileset represents a different number. For the small aperiodic tileset, this will be enforced by ensuring that a row on top of another one can only be obtained by multiplying the previous number by a factor 2 or a factor $\frac{2}{3}$. The fact that it is not possible to obtain 1 by any multiplication of 2's and $\frac{2}{3}$'s will imply aperiodicity.

Let us first talk about the bi-infinite representations of numbers that will be used.

5.1 Balanced representations of reals

Definition 22. Let α be a positive real number, $a \in \{\lfloor\alpha\rfloor, (\lfloor\alpha\rfloor+1)\}^{\mathbb{Z}}$ is a balanced representation of α iff for any subword $a_{[i,i+n-1]}$ we have:

$$n\alpha - 1 \leq \sum_{k=i}^{i+n-1} a_k \leq n\alpha + 1$$

In other words, a balanced representation of a real number is a sequence composed only of its two nearest integers where the average over finite subwords of length n is within $\frac{1}{n}$ of α. This means in particular that if we take longer and longer subwords, their average converges to α.

Such a balanced representation for a finite word is easy to construct by means of Beatty sequences [6]. Given a real number α we denote $\mathscr{B}(\alpha)$ its Beatty sequence, with $\mathscr{B}(\alpha)_k = \lfloor k\alpha \rfloor$. Informally, the Beatty sequence of a number α corresponds to how many horizontal lines of the $\mathbb{Z}^2$ grid the line $y = \alpha x$ has crossed at coordinate x, see Figure 24.

Now if we take the sequence of first differences of the Beatty sequence of α

$$b(\alpha)_k = \mathscr{B}(\alpha)_{k+1} - \mathscr{B}(\alpha)_k,$$

it is a symbolic sequence on alphabet $\{\lfloor\alpha\rfloor, \lfloor\alpha+1\rfloor\}$. And it is a balanced representation of α as for any i:

$$\sum_{k=i}^{i+n-1} b(\alpha)_k = \lfloor (i+n)\alpha \rfloor - \lfloor i\alpha \rfloor$$

and

$$n\alpha - 1 \leq \lfloor (i+n)\alpha \rfloor - \lfloor i\alpha \rfloor < n\alpha + 1.$$

So the sequence $b(\alpha)$ is a balanced representation of α it is also sometimes called the β-sequence for α, see [42], or a rotation sequence [18]. It is interesting to note that when α is irrational this sequence corresponds to the sturmian word for the cutting line $y = \alpha x$.

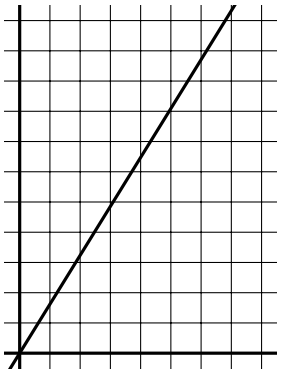


Fig. 24 The Beatty sequence and balanced representation of $\phi = \frac{1+\sqrt{5}}{2}$. The ones and the twos of the balanced representation correspond respectively to cutting a vertical and a horizontal integer line.

5.2 Multiplying balanced representations

Given a positive rational number $q = \frac{n}{m}$, it turns out that by multiplying a balanced representation of $(a_i)_{i\in\mathbb{Z}}$ of some real number α by q one still obtains a balanced representation. We however have to define what we mean by multiplying a balanced representation.

To multiply a by q we will multiply each digit individually while using a carry:

$$c_i + qa_i = b_i + c_{i+1}$$

where c_i and c_{i+1} are respectively the carry inherited from the previous multiplication and the carry sent to the next. With such a definition, when multiplying some subword $a_{[i,i+n-1]}$ by q, one obtains a word $b_{[i,i+n-1]}$ with

$$c_i + q\sum_{k=i}^{i+n-1} a_k = \sum_{k=i}^{i+n-1} b_k + c_{i+n}.$$

See Figure 25 for an example of multiplication of a balanced representation.

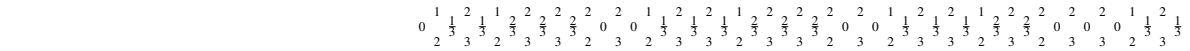


Fig. 25 An example of multiplication of a balanced representation. On the bottom row, a balanced representation of $\phi+1$, on the top a balanced representation of $\frac{2}{3}(\phi+1)$. The carries are shown in the middle.

Lemma 2. *If a is a balanced representation of some α then there exists a balanced representation b of $q\alpha$ that can be obtained by multiplying a by q. Furthermore, the set of carries being used is finite and only depends on q.*

Proof. Take b any balanced representation of $q\alpha$. Since

$$n\alpha - 1 \leq \sum_{k=i}^{i+n} a_i \leq n\alpha + 1$$

and

$$nq\alpha - 1 \leq \sum_{k=i}^{i+n} b_i \leq nq\alpha + 1$$

we have

$$-(1+q) \leq \sum_{k=i}^{i+n} qa_i - b_i \leq 1+q$$

Thus, it suffices to take carries that are in $[-(1+q),(1+q)]$, and because the a_i's and b_i's are integers and $q=\frac{n}{m}$ is a rational, the carries are the multiples of $\frac{1}{m}$ in this interval. □

We now know that this set of carries is finite, which would be sufficient to construct a tileset as will be seen in the next section. However Kari noticed that it is possible to restrict oneself to a smaller set of carries by looking directly at how the Beatty sequences for α and $q\alpha$ are related.

Lemma 3. *The balanced sequence $b(q\alpha)$ can be obtained by multiplying $b(\alpha)$ using carries belonging only to*

$$S = \left\{ -\frac{n-1}{m}, -\frac{n-2}{m}, \ldots, \frac{m-1}{m} \right\}$$

Proof. Since

$$q\lfloor\alpha\rfloor - 1 < \lfloor q\alpha\rfloor < q(\lfloor\alpha\rfloor + 1)$$

we have that

$$-q < q\lfloor\alpha\rfloor - \lfloor q\alpha\rfloor < 1$$

which means in particular that for any α:

$$(q\lfloor\alpha\rfloor - \lfloor q\alpha\rfloor) \in \left\{-\frac{n-1}{m}, -\frac{n-2}{m}, \ldots, \frac{m-1}{m}\right\}$$

Now suppose we have incarry $c_i \in S$ and $c_i = q\mathscr{B}(\alpha)_{i-1} - \mathscr{B}(q\alpha)_{i-1}$:

$$\begin{aligned} c_i + qb(\alpha)_i - b(q\alpha)_i &= q\mathscr{B}(\alpha)_{i-1} - \mathscr{B}(q\alpha)_{i-1} \\ &\quad + q\mathscr{B}(\alpha)_i - q\mathscr{B}(\alpha)_{i-1} \\ &\quad - \mathscr{B}(q\alpha)_i + \mathscr{B}(q\alpha)_{i-1} \\ &= (q\mathscr{B}(\alpha)_i - \mathscr{B}(q\alpha)_i) \\ &= (q\lfloor i\alpha\rfloor - \lfloor qi\alpha\rfloor) \in S \end{aligned}$$

It is thus possible to obtain $b(q\alpha)$ by multiplying $b(\alpha)$ by q using only carries from S. □

So we now know that multiplying balanced representations by some positive rational can yield balanced representations. Let us now see in the next subsection how this translates to aperiodic tilings.

5.3 An aperiodic tileset

The idea now is to use the fact that our multiplication uses only a finite number of rational carries (see Lemma 2) to implement multiplication by using non-deterministic transducers.

For a multiplication by $q = \frac{n}{m}$, the states will correspond to the possible carries:

$$S = \left\{-\frac{n-1}{m}, -\frac{n-2}{m}, \ldots, \frac{m-1}{m}\right\}$$

The idea is to take balanced representations of numbers that belong only to the interval $[1,3]$, avoiding any possible representation of 0. In particular, numbers in $[1,1.5]$ will be multiplied by 2 and numbers in $[1.5,3]$ by $\frac{2}{3}$. The alphabet on which the transducers will act will hence be $\Sigma = \{1,2,3\}$. The transition table will contain all transitions

$$s \xrightarrow{a|b} s'$$

verifying $aq + s = b + s'$.

We can now construct a tileset corresponding to our transducer: for each transition we get one tile where the bottom and top represent the input and the output respectively and the left and right edges represent the inward carry and the outward carry respectively.

So in order to apply multiplications by 2 and $\frac{2}{3}$ we need two different transducers and corresponding sets of tiles. We can assume their set of states is disjoint, up to renaming the states. Limiting the output of the transducer multiplying by 2 to $\{2,3\}$ de facto limits the input interval to $[1,1.5]$.

Figure 26 shows the tiles for 2 and Figure 28 show the tiles for $\frac{2}{3}$.

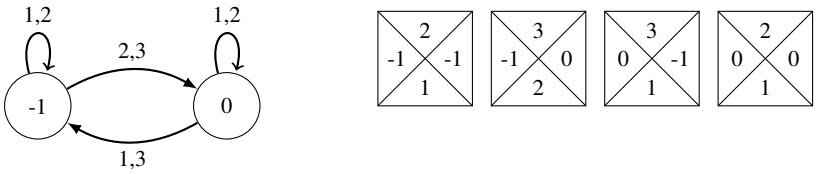


Fig. 26 On the left the transducer corresponding to multiplication by 2 and on the right the corresponding tiles. Notice how the input must belong to $\{1,2\}$ and the outputs to $\{2,3\}$, which translates the fact that the input belongs to $[1,2]$ and the output to $[2,3]$.

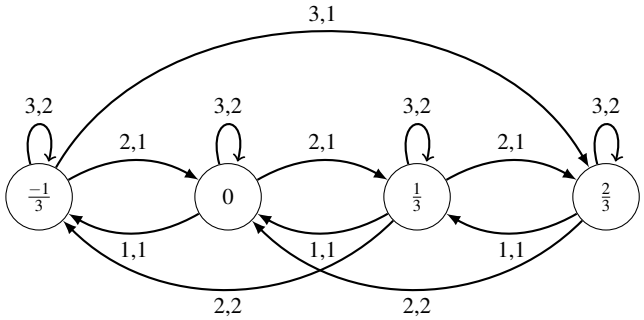


Fig. 27 The transducer corresponding to multiplication by $\frac{2}{3}$ for inputs in $[1,2]$ and outputs in $[1,3]$.

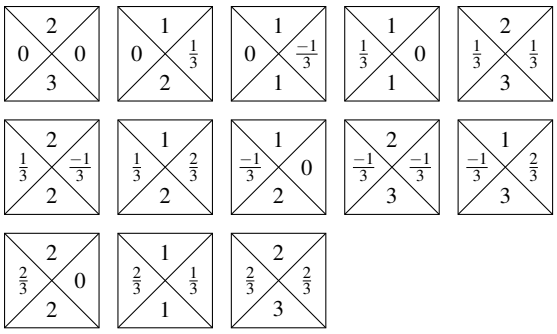


Fig. 28 The set of tiles for the multiplication by $\frac{2}{3}$. We assume the 0 of this part of the tileset is different from the one of the multiplication by 2.

Lemma 4. *The tileset T tiles the plane.*

Proof. Take some irrational real α in $[1,3]$, one can tile $\mathbb{Z}\times\mathbb{N}$, by applying iteratively a multiplication by 2 to $b(\alpha)$ when $\alpha\in[1,1.5]$ and a multiplication by $\frac{2}{3}$ when $\alpha\in[1.5,3]$: the result always stays in $[1,3]$. By compactness the tileset then tiles the whole plane. □

We now know that we can tile the plane aperiodically however, lines of a valid tiling do not necessarily correspond to a balanced representation, we therefore still have to prove that the tileset always tiles aperiodically.

Lemma 5. *The tileset T tiles only aperiodically.*

Proof. Suppose T tiles periodically, then there exists a tiling $c:\mathbb{Z}^2\to T$ periodic in both directions. Let a and b be its horizontal and vertical periods respectively and let x_i be the finite sum of the values represented on top of $c(1,i)$ to $c(a,i)$. Since all the values are positive, x_i cannot be zero. And since the carry on the left is equal to the carry on the right of the period, $x_i = q_i x_{(i-1)}$ with q_i equal to 2 or $\frac{2}{3}$.

Therefore, $x_0 = q_n\cdots q_1 x_0$ which is imposible since no nonempty product of 2's and $\frac{2}{3}$'s can equal 1. □

See Figure 29 for an example of tiling by this tileset. The tileset obtained here is not Kari's tileset but a tileset built using the same techniques. Kari's original tileset used only numbers in the interval $[\frac{1}{2},2]$ but still used multiplications by 2 and $\frac{2}{3}$ and used a technical trick to avoid 0.

5.4 Undecidability

What is interesting about this construction of aperiodic tilesets is that it allows a different encoding of computation: we will use piecewise affine maps in order to encode computations. While in the usual encoding of Turing machines in tilesets, the head and the current state are explicitly encoded in some tile, using piecewise affine functions will allow to encode the state implicitly in two real numbers representing the full instantaneous description of the Turing machine at some timestep.

5.4.1 Computing with piecewise affine maps

Let us first explain how Turing machine computations may be encoded by piecewise affine maps. Let M be a Turing machine working on alphabet $\Gamma=\{0,1\}$ and with state set $Q=\{0,\dots,n\}$.

Let $(c,h,q)\in\Gamma^{\mathbb{Z}}\times\mathbb{Z}\times Q$ be an ID of said Turing machine. We will encode this ID inside two real numbers (l,r) so that the fractional part of l and r will encode the left and right side of the infinite tape respectively and $\lfloor l\rfloor$ will encode the symbol currently under the head and $\lfloor r\rfloor$ the current state of the machine. This encoding will

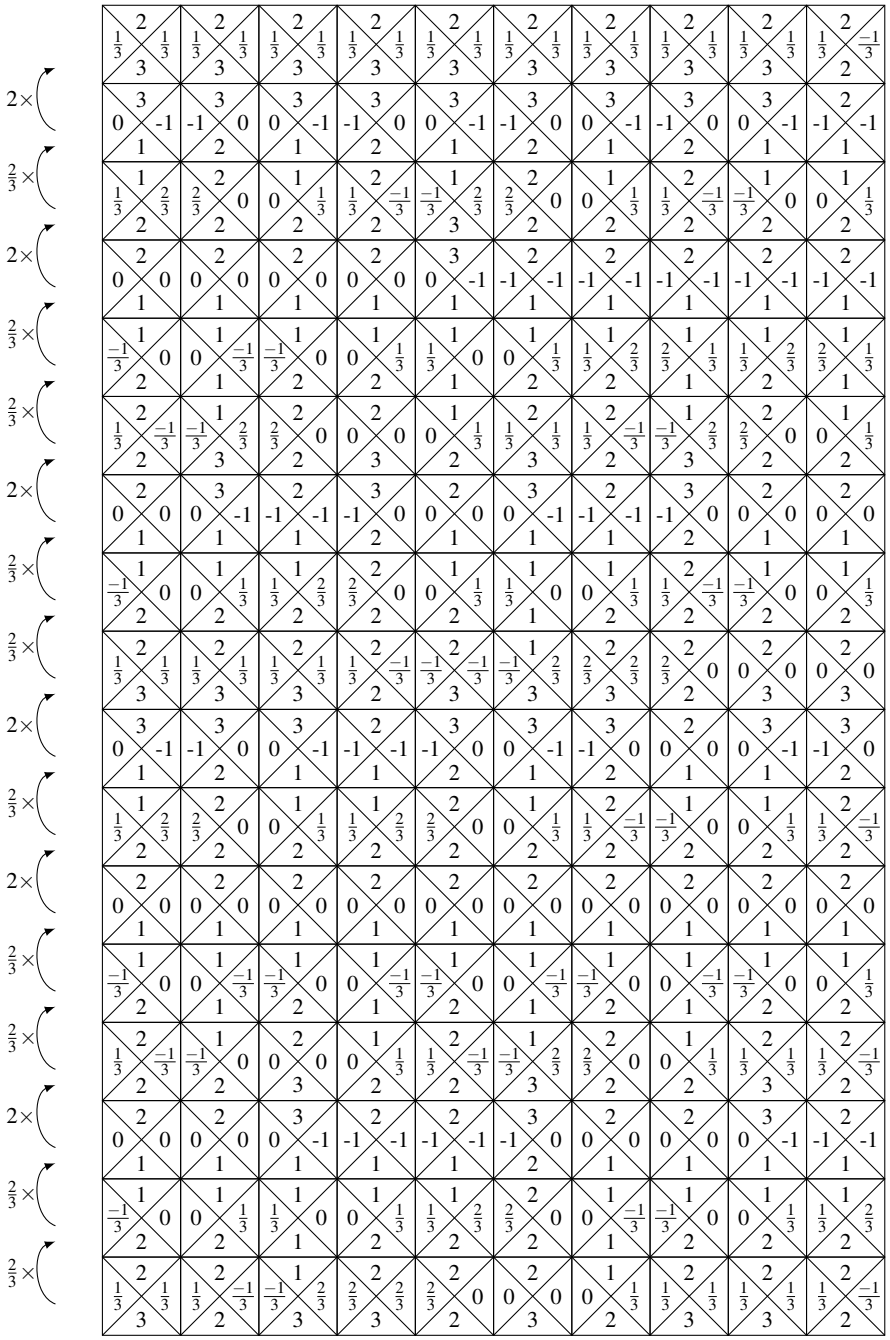


Fig. 29 An example of tiling, the bottom row starts from the balanced representation of $\phi + 1$, and the transducers are then applied according to which interval the resulting multiplied integer belongs to.

be done in a way reminiscent of Cantor middle thirds: if the symbol currently under the head is 1, then $l \in [2,3]$ and if it is 0 then $l \in [0,1]$.

Knowing the state of the machine is thus equivalent to determining to which $[k,k+1[$ interval r belongs to, moving to a new state corresponds to adding/substracting some integer to r and moving the head corresponds to shifting (multiplying) the tape and writing the new symbol (adding a constant).

The transition function hence corresponds to applying a rational piecewise affine map depending only on $\lfloor r \rfloor$ and $\lfloor l \rfloor$. This map will not be defined on $[1,2]$ reflecting the fact that $[1,2]$ does not code for any valid symbol of the tape.

Let's see an example, with the Turing machine in state q and the following tape:

$$\ldots 0000101001001.1.0101010101010101010\ldots$$

(l,r) can be as follows:

$$l = 2.2002002020000\ldots$$
$$r = q.0202020202020202020\ldots$$

If the machine goes from state q to state q', writes s and shifts the tape to the right, it is equivalent to do the following operations on l and r:

$$l := (l - \lfloor l \rfloor) \cdot 10$$
$$r := ((r - \lfloor r \rfloor) + 2 \cdot s)/10 + q'$$

which leads to the following new values of l and r:

$$l = 2.002002020000\ldots$$
$$r = q'.(2s)0202020202020202020\ldots$$

Thus our Turing machine has become a set of affine maps whose application depend only on which rational intervals r and l belong to. Let us now formalize this.

Definition 23. (l,r) is a valid coding of ID (c,h,q) iff

- for any $i \in \mathbb{N}$, $10(10^i \cdot l - \lfloor 10^i \cdot l \rfloor)$ is in $[0,1]$ when $c_{h-i} = 0$ and is in $[2,3]$ when $c_{h-i} = 1$,
- for any $i \in \mathbb{N}^*$, $10(10^i \cdot r - \lfloor 10^i \cdot r \rfloor)$ is in $[0,1]$ when $c_{h+i} = 0$ and is in $[2,3]$ when $c_{h+i} = 1$,
- and $\lfloor r \rfloor \in Q$.

Theorem 14. *Let M be a Turing machine and f a piecewise affine map constructed as stated before:*

1. *If c is an ID of M, and (l,r) a valid coding of c, then $f(l,r)$ is a valid coding of c' the ID obtained after one more step of M.*
2. *If (l,r) is such that $f^n(l,r)$ is defined for all $n \in \mathbb{N}$, then there exists an ID c and a valid coding (l',r') of c such that $f^n(l',r')$ is defined for all $n \in \mathbb{N}$.*

Proof. 1. is straightforward. For 2., let (l,r) be such reals, suppose (l,r) is not a valid coding, and let $l = \sum_{i=0}^{\infty} l_i 10^{-i}$ and $r = \sum_{i=0}^{\infty} r_i 10^{-i}$, since (l,r) is not valid, this means that some l_i and r_i are not zeroes or twos, but since $f^n(l,r)$ is always defined, this also means that these never appear as l_0, so the computation to which they correspond is a computation that never reaches them. It then suffices to replace them by zeroes to obtain a valid coding, c then corresponds to this valid coding. See [12] for a more precise proof.

□

Using two disjoint sets like $[0,1]$ and $[2,3]$ allows us to avoid some decoding problems that would necessarily happen if the tape was directly coded with its representation in binary[3].

5.4.2 Undecidability of the Domino Problem

The piecewise affine map that we have constructed can easily be transformed into a set of tiles: it has rational coefficients, adding or substracting a rational number can be done in a similar way to what was done for multiplication and the fact that we are now considering two reals and not only one does not constitute an obstacle, the transducers can as easily be constructed.

The outputs of any transducer coding for a transition will belong either to $\{0,1\}$ or to $\{2,3\}$, thus ensuring that numbers written on one row either belong to $[0,1]$ or to $[2,3]$ and that invalid codings can never appear as a result of applying piecewise affine map.

In order to obtain the undecidability of the domino problem, we will use the immortality problem:

Theorem 8 (Hooper [25]). *There is no algorithm that, given a Turing machine M, decides if M halts on all configurations.*

This is still true if we restrict ourselves to Turing machines with a working alphabet of size 2*.*

By not having any representation for any halting state in the tileset corresponding to the piecewise affine map, any tiling by this tileset will represent an infinite computation of our Turing machine. And thus the tileset will only be able to tile a half plane if the Turing machine is immortal, by compactness, if the tileset tiles the half plane, it tiles the whole plane.

[3] The representations $0.100\ldots$ and $0.011\ldots$ code the same real but should not encode the same tape

6 The Substitutive Method 2/2

The construction of an aperiodic tileset, and the undecidability result, that we describe in this section comes from the work of Durand, Romashchenko and Shen [14] and is inspired by the work of Gács in the 70's.

We saw in section 3 the concept of an intrinsically substitutive tileset, and we gave an example of such a tileset. However, we gave no indication how this tileset was obtained, or how to prove in general that such a tileset should exist. The core of the construction of Durand, Romashchenko and Shen is to use recursion theory to prove this.

We will first redefine more generally the notion of substitutive tiling, and of a simulation

Definition 24 ([13, 43, 5, 14]). Let σ, τ be two tilesets.

We say that σ simulates τ with zoom factor N if there exists a one-to-one function $\phi : \tau \to \sigma^{N\times N}$ s.t.

- For any finite pattern w, $\phi(w)$ is a valid tiling for σ precisely when w is a valid tiling for τ.
- For any tiling of the plane x by σ, there exist w s.t. $\phi(w) = x$ upto shift. More precisely there exists a *unique* pair $(i', j') \in [0, N-1] \times [0, N-1]$ and a unique w s.t. $x_{i+i', j+j'} = \phi(w)_{i,j}$.

The uniqueness property means that there is only one way to shift x to obtain an image of ϕ.

Theorem 15 ([14]). *There exists a tileset that simulates itself.*

We will explain in this section how this theorem is proven, and then how to apply it to obtain the undecidabilty of the domino problem.

The idea is that, given a tileset τ, it is easy to build a tileset σ that simulates τ with a given zoom factor N.

We now somehow want to have $\sigma = \tau$ in the previous construction.

How to do this is similar to the fixed point theorem of recursion theorem, which allows in particular to prove that functions that call themselves are (partial) computable.

We will first illustrate this fixed point theorem in a particular example.

Suppose we want to compute recursively the factorial function, in a language where recursion is not allowed. Here is what we would write in Python:

```
def f(n):
     if n == 0:
         return 1
     else:
         return n*f(n-1)
```

But in our fictive language, recursion is not allowed, so we cannot write call f.

One solution is to have f call another function g, and then somehow set g to be equal to f. A first version would be:

```
def f(n):
     if n == 0:
         return 1
     else:
         return n*g(n-1)
g = f
```

In this example, g is a global variable that was somehow defined before the definition of f and then changed value afterwards. This is bad programming practice, so we prefer to give it as an argument to f:

```
def f(g,n):
     if n == 0:
         return 1
     else:
         return n*g(g,n-1)
```

we can then call f with input (f,n) to obtain the result. This version would however make type theorists jump out of their skin[4], as it not easy to find what type is f.

We will then proceed differently. Suppose that our programming language has a way to convert the code of a program (which is just a string of characters) into a function. Let call this procedure eval. Then we can write:

```
def f(x,n):
     if n == 0:
         return 1
     else:
         return n*eval(x)(x,n-1)
```

Now f is just a function that takes an integer and a string and outputs an integer. Now if we feed f with n and its own code, then we obtain the factorial function. [5]

The tileset we will construct mimics exactly this idea.

6.1 The fixed point theorem of computability theory

The generalization of what preceeds is called Kleene's fixed point theorem and is one of the fundamental theorems of computability theory. Usually in computability theory functions, strings, integers are all represented by finite binary sequences that may be seen as integers, and thus all theorems are on computable functions on integers.

[4] It however works in python.

[5] Up to a few syntactic changes, the reasoning we took actually works in python, as shown by the following two lines:

```
x = "lambda n,x: 1 if n == 0 else n*eval(x)(n-1,x)"
f = lambda n: eval(x)(n,x)
```

Theorem 16 (Kleene's fixed point). *If $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is a computable function, then there exists a computable function g such that $f(g,n) = g(n)$.*

Proof. Let U be a universal program: given $p, n_1, \ldots, n_k$ as arguments, U computes p with arguments $n_1, \ldots, n_k$, that is to say:

$$U(p, n_1, \ldots, n_k) = p(n_1, \ldots, n_k).$$

This is essentially what the `eval` function from before does.

Consider $s(p) = U(p,p)$, the function that applies p to its own code and denote $h(i,n) = f(s(i),n)$. Now set $g(n) = h(h,n)$ we have:

$$g(n) = h(h,n) = f(s(h),n) = f(h(h),n) = f(g,n)$$

□

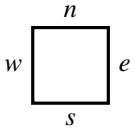
One way to interpret this theorem is that there exists a function g that "mimics" an infinite application of f on itself: $g(n) = f(g,n) = f(f(g,n)) = f(f(f(\ldots),n),n)$.

The universal Turing machine of the proof induces some overhead to the time $t(n)$ that the program p would take by itself on an input of size n. Using several tapes, one can design a universal TM taking an overhead at most polynomial in $t(n)$, see for instance [22, 4].

The aim is to construct a tileset using Kleene's fixed point theorem to obtain a self simulating tileset.

6.2 Simulating a tileset with a Turing machine

A tile can be summed up by the colors $c = \langle n, w, s, e \rangle$ of its four borders:



And thus a tileset can be seen as a computable function $g(c)$ that accepts when c represents a tile belonging to the tileset and rejects otherwise:

Definition 25. A computable function *g codes a tileset τ with k colors* if:

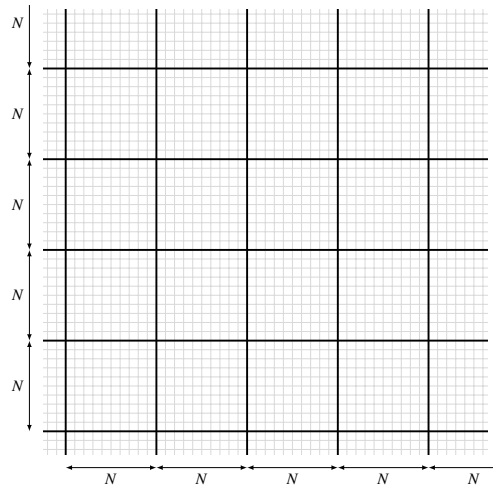
- g interprets its argument as a quadruplet of colors: g rejects if its input is not of size $4 \log k$.
- g accepts only when c represents a tile of τ.

Now if g is a function defining a tileset τ, we are going to design a computable function $f(N,k,g,c)$ that verifies if the tile c is in a tileset τ' simulating τ with zoom

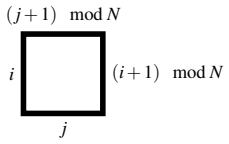
factor N. That is to say, f builds the tileset τ' from g and then checks that its input c belongs to it.

The tileset constructed by f will divide the plane into *macrotiles*, virtual tiles formed by an $N \times N$ square of smaller tiles. Each macrotile then runs the function g on input $C = \langle n, w, s, e \rangle$, the 4 colors coded in the border of the macrotile. The program g is only allowed to accept, so that only valid macrotiles may be formed. Let us now see the details.

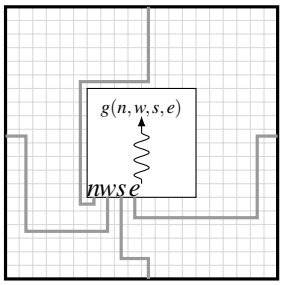
The tileset constructed by f first starts by separating the grid into $N \times N$ squares:



This can easily be done by having tiles that count modulo N, the borders of the macrotile appearing on the transition between 0 and $N-1$:



The center of each macrotile then contains a zone on each side that represents the color of the side and wires that route them towards a computation zone which applies g to c:



We make the computation zone of size $N/2 \times N/2$ inside each macrotile.

If N is big enough, this computation zone has enough time and space to run g:

Theorem 17. *If g codes τ with k colors, there exists N such that for any $N' > N$, $f(N',k,g,c)$ codes a tileset simulating τ with zoom factor N'.*

Proof. Let t be an upper bound of the time taken by g on any accepting input of size $4\log k$, when g is executed on a universal machine. We now take $N = 2t$.

Let $N' > N$. By our choice of N, the computation zone inside each macrotile defined by $f(N',k,g,c)$ is big enough for the computation of g to fit.

In this case, each macrotile codes colors n,s,e,w on its border s.t. $g(\langle n,s,e,w\rangle)$ accepts, i.e. the macrotile codes a tile of τ.

Therefore the whole tileset τ' is simulating τ with zoom factor N'. □

6.3 A fixedpoint based tileset

Wishful thinking would lead us to apply the fixedpoint theorem to f however this is not possible as is for several reasons:

- First, g does not take the same arguments as f. As defined before, g supposes that N and k are externally defined.
- N and k would depend on the size of f to be defined.

The solution to this is straightforward and is to make k and N become arguments of g. The only modification needed in the previous construction for is to hardcode N and k as inputs of g inside the macrotiles that are constructed by f. As the number of tiles generated by f is $O(N^2)$, this means that taking $k = N^2$ colors is sufficient to code them all. Furthermore, there is always room in the macrotiles to carry N^2 colors, as this only takes $2\log N$ bits.

So we now have a function $f(g,N,c)$ which takes a function $g(N,c)$ as an argument.

It now suffices to apply Kleene's fixed point theorem to f. We obtain a new program ρ such that $f(\rho,N,c) = \rho(N,c)$.

Theorem 18. *There exists N such that for any $n > N$, $\rho(N,c)$ codes a tileset τ simulating itself with zoom factor N.*

Proof. As f is polynomial in $\log N$, $|g|$ and $|c|$, this means that ρ's runtime is polynomial in $\log N$. So taking $N \gg poly(\log N)$ suffices to have enough runtime inside the macrotiles.

ρ defines a tileset τ which produces macrotiles of size $N \times N$ that themselves check if they belong to τ by running ρ in their computation zone. Thus ρ simulates itself with zoom factor N. □

We can now show that τ indeed tiles the plane by inductively defining macrotiles of level k, this will also give us a picture of what tiling by τ look like:

- The macrotiles of level 0 are the macrotiles formed by $N \times N$ squares of tiles of τ.
- The macrotiles formed $N \times N$ squares of macrotiles of level k are macrotiles of level $k+1$, see figure 30.

τ does indeed tile the plane since for any k one can tile an $N^{2^k} \times N^{2^k}$ square just by taking a macrotile of level k, which implies that there exists a tiling of the plane, by compactness.

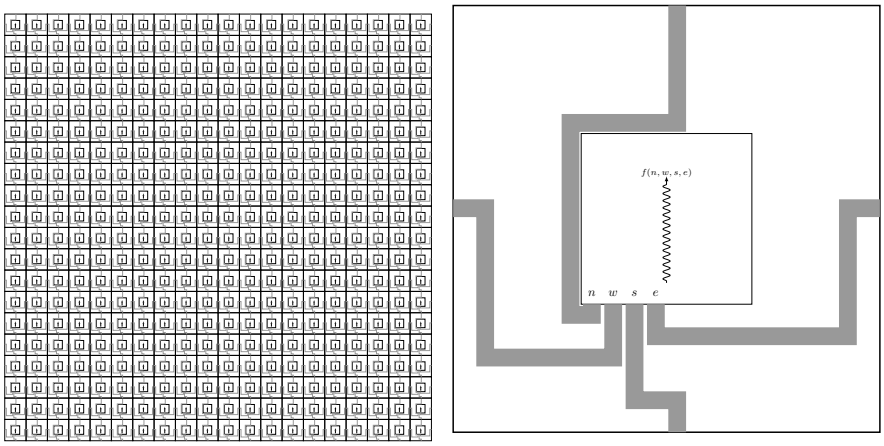


Fig. 30 An $N \times N$ square of macrotiles of level k forms a macrotile of level $k+1$.

One may also show that τ is aperiodic:

Lemma 6. *A tileset that simulates itself is aperiodic.*

Proof. Let T be a tileset that simulates itself with zoom factor N and suppose it is periodic. Since any tiling is uniquely subdivided in $N \times N$ zones, the period must be divisible by N. However, since the tileset simulates itself, any tiling can also be uniquely subdivided in $N^2 \times N^2$ zones, so N^2 must also divide the period. By the same argument this must be true for any N^{2^k}, which is impossible. □

6.4 Undecidability

The tileset τ we constructed in the previous subsection, while aperiodic, does not prove the undecidability of the Domino Problem. In order to do this, given a Turing machine M we will construct τ_M in a similar fashion, but we will embed computations of M into the macrotiles which will prevent the tileability in case M halts.

We cannot use exactly the construction from before, as a zoom factor of N for each level of macrotiles will mean that the macrotiles of different levels have the

same computation space and thus all computations would be bounded by the same N. The goal being that macrotiles of level k simultaneously generate macrotiles of level $k+1$ and simulate a computation of M for more and more time steps.

Thus, if M halts, some macrotile of level k would uncover it and prevent the formation of macrotiles of level $k+1$ and the tileset to tile the plane.

To achieve this, instead of making f hardcode N as an input to g, f hardcodes $2N$. Thus the computation zone is doubled with each simulation. So now a macrotile of level $k+1$ divides the plane in $2^kN \times 2^kN$ macrotiles of level k. And the computation zone inside the macrotiles of level $k+1$ now is $2^kN/2$.

The program run by a macrotile can be summed up by the following pseudocode:

```
def f(g,N,c):
  constuct T_k: a tileset dividing the plane in N×N
                macrotiles which contain a
                computation of g(2N,c').
  if c ∉ T_k:
     reject
  else:
     launch N/2 timesteps of the simulation of M
     if the simulation halts reject
```

Instead of a constant zoom factor between the different levels, each level k now has zoom factor $N_k = 2^kN$ which gives more and more computational room. So each level of macrotiles now ensures more simulation steps of M and thus if M halts, it will happen at some finite level which will prevent the tileability.

6.5 Bibliographic notes

The fixed point tileset construction originally appeared in [15]. Its versatility allowed Durand, Romashchenko and Shen to prove new and original results on tilings as well as reprove ancient results [14]. For instance, among the results that can be proven with almost no modifications of the construction is a result by Hanf and Myers [21, 41] stating that there exists tilesets producing only non computable tilings of the plane. Other results that can be proved using variations of this technique include:

- Substitutive tilesets defined by a rectangular substitution are sofic (Mozes [40]).
- d-dimensional effective subshifts are subactions of $d+1$ dimensional sofic subshifts, a result originally discovered only with $d+2$-dimensional sofic shifts by Hochman [23].
- A characterization of the entropy of tilings by right recursively enumerable numbers (Hochman and Meyerovitch [24]).
- That there exists tilesets robust to errors.
- A characterization of the set of non-expansive directions of tilings [54].
- The subshift whose configurations are constituted only of squares whose sizes are chosen from a co-recursively enumerable set [53].

References

1. Aanderaa, S.O., Lewis, H.R.: Linear Sampling and the ∀∃∀ Case of the Decision Problem. Journal of Symbolic Logic **39**(3), 519–548 (1974). See also [36]
2. Allauzen, C., Durand, B.: Tiling Problems. In: The Classical Decision Problem, Perspectives in Mathematical Logic, chap. A, pp. 407–420. Springer (2001)
3. Ammann, R., Grunbaum, B., Shephard, G.: Aperiodic tiles. Discrete and Computational Geometry **8**(1), 1–25 (1992)
4. Arora, S., Barak, B.: Computational Complexity: A Modern Approach, 1st edn. Cambridge University Press, New York, NY, USA (2009)
5. Ballier, A.: Propriétés structurelles, combinatoires et logiques des pavages. Ph.D. thesis, Aix-Marseille Université (2009)
6. Beatty, S.: Problem 3173. American Mathematical Monthly **33**, 159 (1926)
7. Berger, R.: The Undecidability of the Domino Problem. Ph.D. thesis, Harvard University (1964)
8. Berger, R.: The Undecidability of the Domino Problem. No. 66 in Memoirs of the American Mathematical Society. The American Mathematical Society (1966)
9. Büchi, J.R.: Turing-Machines and the Entscheidungsproblem. Math. Annalen **148**(3), 201–213 (1962). DOI 10.1007/BF01470748
10. Chazottes, J.R., Gambaudo, J.M., Gautero, F.: Tilings of the plane and Thurston semi-norm. Geometriae Dedicata **173**(1), 129–142 (2014). DOI 10.1007/s10711-013-9932-4
11. Church, A.: An unsolvable problem of elementary number theory. American Journal of Mathematics **58**(2), 345–363 (1936)
12. Collins, P., van Schuppen, J.H.: Observability of Hybrid Systems and Turing Machines. In: 43rd IEEE conference on Decision and Control, pp. 7–12 (2004). DOI 0.1109/CDC.2004.1428598
13. Durand, B., Levin, L.A., Shen, A.: Local rules and global order, or aperiodic tilings. Mathematical Intelligencer **27**(1), 64–68 (2004)
14. Durand, B., Romashchenko, A., Shen, A.: Fixed-point tile sets and their applications. Journal of Computer and System Sciences **78**(3), 731–764 (2012). DOI 10.1016/j.jcss.2011.11.001
15. Durand, B., Shen, A., Romashchenko, A.: Fixed Point and Aperiodic Tilings. Tech. Rep. TR08-030, ECCC (2008)
16. Eigen, S., Navarro, J., Prasad, V.S.: An aperiodic tiling using a dynamical system and Beatty sequences. In: Recent Progress in Dynamics, *MSRI Publications*, vol. 54. Cambridge University Press (2007)
17. Fernique, T., Ollinger, N.: Combinatorial Substitutions and Sofic Tilings . In: JournÃ©es Automates Cellulaires (JAC), TUCS, pp. 100–110 (2010)
18. Fogg, N.P.: Substitutions in Dynamics, Arithmetics and Combinatorics, chap. Sturmian Sequences. Lecture Notes in Mathematics. Springer (2002)
19. Gähler, F., Julien, A., Savinien, J.: Combinatorics and Topology of the Robinson tiling. Comptes Rendus de l'Académie des Sciences de Paris, Série I - Mathématiques pp. 627–631 (2012). DOI 10.1016/j.crma.2012.06.007
20. Goodman-Strauss, C.: Matching Rules and Substitution Tilings. Annals of Mathematics **147**(1), 181–223 (1998)
21. Hanf, W.: Non Recursive Tilings of the Plane I. Journal of Symbolic Logic **39**(2), 283–285 (1974)
22. Hennie, F.C., Stearns, R.E.: Two-tape simulation of multitape turing machines. J. ACM **13**(4), 533–546 (1966). DOI 10.1145/321356.321362. URL http://doi.acm.org/10.1145/321356.321362
23. Hochman, M.: On the dynamics and recursive properties of multidimensional symbolic systems. Inventiones Mathematicae **176**(1), 2009 (2009)
24. Hochman, M., Meyerovitch, T.: A characterization of the entropies of multidimensional shifts of finite type. Annals of Mathematics **171**(3), 2011–2038 (2010). DOI 10.4007/annals.2010.171.2011

25. Hooper, P.K.: The Undecidability of the Turing Machine Immortality Problem. Journal of Symbolic Logic **31**(2), 219–234 (1966)
26. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979)
27. Jeandel, E., Rao, M.: An aperiodic set of 11 Wang tiles. Preprint
28. Johnson, A., Madden, K.: Putting the Pieces Together: Understanding Robinson's Nonperiodic Tilings. The College Mathematics Journal **28**(3), 172–181 (1997)
29. Kahr, A., Moore, E.F., Wang, H.: Entscheidungsproblem reduced to the $\forall\exists\forall$ case. Proceedings of the National Academy of Sciences of the United States of America **48**(3), 365–377 (1962)
30. Kari, J.: The Nilpotency Problem of One-Dimensional Cellular Automata. SIAM Journal on Computing **21**(3), 571–586 (1992)
31. Kari, J.: A small aperiodic set of Wang tiles. Discrete Mathematics **160**, 259–264 (1996)
32. Kari, J.: The Tiling Problem Revisited. In: Machines, Computations, and Universality (MCU), no. 4664 in Lecture Notes in Computer Science, pp. 72–79 (2007)
33. Kari, J., Ollinger, N.: Periodicity and Immortality in Reversible Computing. In: MFCS 2008, LNCS 5162, pp. 419–430 (2008)
34. Kleene, S.: Two Papers on the Predicate Calculus., chap. Finite Axiomatizability of Theories in the Predicate Calculus Using Additional Predicate Symbols. No. 10 in Memoirs of the American Mathematical Society. American Mathematical Society (1952)
35. Levin, L.A.: Forbidden Information. Journal of the ACM **60**(2), 9:1–9 (2013). DOI 10.1145/2450142.2450145
36. Lewis, H.R.: Unsolvable Classes of Quantificational Formulas. Addison-Wesley (1979)
37. Lewis, H.R., Papadimitriou, C.H.: Elements of the Theory of Computation. Prentice-Hall (1998)
38. Lind, D.A., Marcus, B.: An Introduction to Symbolic Dynamics and Coding. Cambridge University Press, New York, NY, USA (1995)
39. Minsky, M.L.: Computation: Finite and Infinite Machines. Prentice-Hall (1967)
40. Mozes, S.: Tilings, substitutions systems and dynamical systems generated by them. J. d'Analyse Math. **53**, 139–186 (1989)
41. Myers, D.: Non Recursive Tilings of the Plane II. Journal of Symbolic Logic **39**(2), 286–294 (1974)
42. Nillsen, R., Tognetti, K., Winley, G.: Bernoulli (beta) and integer part sequences. Australian Mathematical Society (1999)
43. Ollinger, N.: Two-by-Two Substitution Systems and the Undecidability of the Domino Problem. In: CiE 2008, no. 5028 in Lecture Notes in Computer Science, pp. 476–485 (2008)
44. Papadimitriou, C.H.: Computational Complexity. Addison Wesley (1995)
45. Poizat, B.: Une théorie finiement axiomatisable et superstable. Groupe d'études de théories stables **3**, 1–9 (1980)
46. Robinson, R.M.: Seven polygons which permit only nonperiodic tilings of the plane. Notices of the American Mathematical Society **14**, 835 (1967)
47. Robinson, R.M.: Undecidability and Nonperiodicity for Tilings of the Plane. Inventiones Mathematicae **12**(3), 177–209 (1971). DOI 10.1007/BF01418780
48. Salon, O.: Quelles tuiles! (Pavages apériodiques du plan et automates bidimensionnels). Journal de Théorie des Nombres de Bordeaux **1**(1), 1–26 (1989)
49. Simpson, S.G.: Medvedev degrees of two-dimensional subshifts of finite type. Ergodic Theory and Dynamical Systems **34**, 679–688 (2014). DOI 10.1017/etds.2012.152
50. Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. Proceedings of the London Mathematical Society **s2-42**(1), 230–265 (1937)
51. Wang, H.: Proving theorems by Pattern Recognition II. Bell Systems technical journal **40**, 1–41 (1961)
52. Wang, H.: Dominoes and the $\forall\exists\forall$ case of the decision problem. Mathematical Theory of Automata pp. 23–55 (1963)
53. Westrick, L.B.: Seas of squares with sizes from a Π^0_1 set. Israel Journal of Mathematics **222**(1), 431–462 (2017). DOI 10.1007/s11856-017-1596-6

54. Zinoviadis, C.: Hierarchy and expansiveness in 2d subshifts of finite type. In: Language and Automata Theory and Applications - 9th International Conference, LATA 2015, Nice, France, March 2-6, 2015, Proceedings, pp. 365–377 (2015)