Introduction to Symbolic Dynamics and Aperiodic Tilings

Thomas Fernique

A *dynamical system* is a space X and a map T which acts on it. Given a point $x \in X$, we are interested in its *trajectory* or *orbit*

$$\Omega(x) := \{T^n(x) \mid n \in \mathbb{Z}\}.$$

Some examples of questions arising:

- does x have a *periodic* orbit, *i.e.*, $T^k(x) = x$ for some $k \in \mathbb{N}$?
- does x visit the whole space, *i.e.*, does its trajectory go arbitrarily close to any point $y \in X$?
- does x spend in any subset $A \subset X$ a time proportional to its size, *i.e.*,

$$\lim_{n\infty} \frac{\#\{0 \leq k < n \mid T^k(x) \in A\}}{n} \overset{?}{=} \frac{\mathrm{vol}(A)}{\mathrm{vol}(X)}.$$

A way to study these questions is to split X in finitely many subsets $\{X_a,\ a \in \mathcal{A}\}$, where $\mathcal{A}$ is a finite set called *alphabet*, and to *encode* the trajectory of a point x by a sequence $(x_n)_{n\in\mathbb{Z}} \in \mathcal{A}^{\mathbb{Z}}$, where x_n gives subset the trajectory of x falls in at time n:

$$x_n = a \ \Leftrightarrow\ T^n(x) \in X_a.$$

This is called the *symbolic method.* This is what we focus on here.

A general introduction can be found in [2]. We shall here only briefly recall the main definitions (Section 1) and focus on the *emptyness problem* (Section 2). We then turn to 2-*dimensional symbolic dynamics*, which is strongly related with tiling theory (Section 3) and consider again the emptyness problem, which becomes much harder (Section 4).

1 Shifts

For x and y in $\mathcal{A}^{\mathbb{Z}}$, define

$$d(x,y) := 2^{-\inf\{|k| \ \mid\ x_k \neq y_k\}}.$$

In other words, the farthest from zero x and y agree, the smallest is d.

Proposition 1 *The map d is a distance over X.*

Definition 1 *A* shift *is a subset $X \subset \mathcal{A}^{\mathbb{Z}}$ closed and invariant by translation.*

Example 1 *The set $\mathcal{A}^{\mathbb{Z}}$ of all the trajectories is a shift, called the* full shift.

Example 2 *The set with only the trajectory x such that $x_{2k} = a$ and $x_{2k+1} = b$ is not a shift. It is indeed not invariant by translation: translating x by one position yields the trajectory y such that $y_{2k} = b$ and $y_{2k+1} = a$, which is different from x. We get a shift by adding this trajectory:*

$$X_1 := \{x, y\}.$$

Example 3 *The set of the trajectories over $\{a, b\}$ which contains exactly one b is not a shift. It is indeed not closed: by translating arbitrarily far a given trajectory, we get at the limit the trajectory $a^{\mathbb{Z}}$ (the b "disappears" at infinity) which is not in this set. We get a shift by adding this trajectory:*

$$X_2 := \{x \in \{a, b\}^{\mathbb{Z}} \mid x \text{ contains at most one } b\}.$$

We call *word* over $\mathcal{A}$ a finite sequence of letters of $\mathcal{A}$. For example, $w = abba$ is a word of length 4. Words allow an equivalent definition of shifts:

Proposition 2 *A set $X \subset \mathbb{A}^{\mathbb{Z}}$ is a shift if and only if there exists a set F of words, called* forbidden words*, such that*

$$X = \{x \in \mathbb{A}^{\mathbb{Z}} \mid \text{no word in } F \text{ appears in } x\}.$$

The shift X is said to be *defined* by the forbidden words F.

Example 4 *The fullshift is defined by $F = \emptyset$.*

Example 5 *The shift X_1 is defined by $F = \{aa, bb\}$.*

Example 6 *The shift X_2 is defined by $F = \{bb, bab, baab, \ldots\} = \{ba^k b,\ k \geq 0\}$.*

Definition 2 *A* shift of finite type *(in short, SFT) is a shift defined by a finite set of forbidden factors.*

Example 7 *The fullshift is an SFT.*

Example 8 *The shift X_1 is an SFT.*

Example 9 *The shift X_2 is not an SFT. The fact that we previously defined it by infinitely many forbidden factors is not sufficient: we shall show that there is no way to define it by finitely many forbidden factors. We proceed by contradiction. Assume that there exists a finite set F of forbidden words which define X_2, with k being the length of the largest one. Consider a trajectory x with exactly two b's (hence not in X_2) at distance k from each other:*

$$x := \cdots aaaba^k baaa \cdots \notin X_2.$$

A word of length k which appears in x has at most one b because the two b's are too far apart. Such a word cannot be forbidden since it appears in the trajectory

$$y = \cdots aaabaaa \cdots \in X_2.$$

Hence x has no forbidden word and should be in X_2. This is a contradiction.

One can show that any SFT can be defined as the set of infinite paths in a *finite state automaton*. But there are finite state automata whose set of infinite paths are not an SFT. This yields the notion of *sofic shifts*, which can be proven to be exactly the set of infinite paths in a finite state automaton.

Definition 3 *A shift X over $\mathcal{A}$ is* sofic *if it is the image of an SFT Y over $\mathcal{B}$ by a* factor map *$\phi : \mathcal{B} \to \mathcal{A}$.*

In particular, any SFT is a sofic shift (take $\mathcal{B} = \mathcal{A}$ and ϕ the identity). The converse is false as we shall see on examples.

Example 10 *The shift X_2 is sofic (but it is not an SFT as already seen). To prove this, let Y_2 be the SFT over $\{a_1, a_2, b\}$ defined by the forbidden words*

$$F := \{ba_1, a_2a_1, a_2b, a_1a_2\}.$$

These forbidden word ensure that, the trajectory of Y_2 are exactly those where there is either no b, or exactly one b with only a_1 before and only a_2 after. Indeed, the first forbidden word ensures that after a b we have a_2 and the two second ones that after this a_2 there are only a_2's. Similarly, the third forbidden word ensures that before a b we have a_1 and the first and last ones that before this a_1 there are only a_1. Now, let ϕ be the factor map defined by $\phi(a_1) = \phi(a_2) = a$ and $\phi(b) = b$. We have $\phi(Y_2) = X_2$.

Example 11 *In a trajectory, we call* run *a maximal sequence of consecutive identical letters. Then the following set is a sofic shift but not an SFT:*

$$X_3 := \{x \in \{a, b\}^{\mathbb{Z}} \mid \textit{any run of } a \textit{ has even length}\}.$$

It is a shift since it can be defined by the forbidden words

$$F = \{ba^{2k+1}b \mid k \geq 0\}.$$

The proof that it is not an SFT looks like the proof that X_2 is not an SFT. Assume indeed that there exists a finite set F of forbidden words which define X_3, with k being the length of the largest one. Any trajectory whose runs of a have length at least k cannot have forbidden words and must thus be in X_3. This yields a contradiction since such trajectories can have runs of odd length. Now, to prove that it is sofic, let Y_3 be the SFT over $\{a_1, a_2, b\}$ defined by

$$F_3 := \{ba_2, a_1b, a_1a_1, a_2a_2\}.$$

These forbidden word ensure that the trajectory of Y_3 are exactly those where the letters between two b's form a word $a_1a_2\cdots a_1a_2$, which has even length. Indeed, the first forbidden word ensures that this word begins with a_1, the second one that it ends with a_2, and the two last ones that it alternates a_1 and a_2. We have $\phi(Y_3)=X_3$, where ϕ is the same factor map as in the previous example.

Example 12 *The following set is a sofic shift but not an SFT:*

$$X_4 := \{x\in\{a,b\}^{\mathbb{Z}} \mid \textit{any run of } a \textit{ has odd length}\}.$$

The proof goes the same way as in the previous example, with instead of Y_3 an SFT Y_4 defined by the forbidden words

$$F_4 := \{ba_1, a_1b, a_1a_1, a_2a_2\},$$

where the letters between two b's of a trajectory form a word $a_2a_1a_2\cdots a_1a_2$, which has odd length.

Example 13 *The following set is a shift but it is not sofic:*

$$X_5 := \{x\in\{a,b\}^{\mathbb{Z}} \mid \textit{the runs of } a \textit{ have all the same length}\}.$$

It is indeed a shift since it can be defined by the following forbidden words, which enforce two consecutive runs of a to have the same length

$$F_5 = \{ba^pb^qa^rb \mid p,q,r\geq 1,\ p\neq r\}.$$

We prove that it is not sofic by contradiction. Assume indeed that there is an SFT over an alphabet $\mathcal{B}$, say Y_5, and a factor map ϕ such that $X_5=\phi(Y_5)$. Consider a finite set of forbidden words which define Y_5, with k being the length of the largest one. Fix $q\geq 1$ and define

$$x := \cdots bbba^qba^qbbb\cdots$$

where the b between the two runs of a is in position 0. It is a trajectory of X_5 since its two runs of a have the same length. It must thus be the image by ϕ of some trajectory $y\in Y_5$. Since there is only finitely many different words of length k over the (finite) alphabet $\mathcal{A}$, if we choose q large enough[1], the word $y_1\cdots y_q$ contains two times the same word of length k, say in positions $i<j$. Define a new trajectory y' by inserting in y at position j the word $y_i\cdots y_{j-1}$:

$$y' = \cdots y_0y_1\cdots y_i\cdots y_{j-1}|y_i\cdots y_{j-1}|y_j\cdots y_qy_{q+1}\cdots$$

The way y' is constructed ensures that any word of length k which appears in y' between position 1 (the letter y_1) and $q+j-i$ (the letter y_q) appears in y between position 1 and q (the case $k<j-i$ is maybe easier to see than the case $k\geq j-i$). Such a word of length k is thus not a forbidden words, whence $y'\in Y_5$. Moreover,

$$\phi(y') = \cdots bbba^qba^{q+j-i}bbb\cdots$$

which is not in X_5. This is a contradiction with $X_5=\phi(Y_5)$.

The proof can also be carried on an automaton that describes X_5: it is then just an application of the *pumping lemma.*

[1] Choosing $q=\mathrm{Card}(\mathcal{A})^k+k$ is sufficient.

2 Empty or not empty?

We shall here focus on a basic question on SFT's: given X an SFT defined by a finite set of forbidden words, can we decide whether X is empty or not?

Example 14 *Let $X \subset \{a,b\}^{\mathbb{Z}}$ be the SFT defined by*

$$F = \{bb, aaa, babab, baabaab\}.$$

It is not empty since it contains the periodic trajectory $(aabab)^{\mathbb{Z}}$.

In the above example, we showed that an SFT is not empty by explicitly giving a trajectory, namely a periodic one. This is general:

Proposition 3 *Any non-empty SFT contains a periodic trajectory.*

Proof. Let X be an SFT defined by a finite set of forbidden factors, with k being the length of the largest one. If X is not empty, let $x \in X$. Since there are finitely many different words of length k in x, one of them must repeat at least twice, say in position $i < j$. Then, any factor of length k of the following trajectory appears in x:

$$x' := (x_i x_{i+1} \cdots x_{j-1})^{\mathbb{Z}}.$$

This yields a periodic trajectory in X. □

Conversely, the following proposition shows that if the SFT is empty, we do not have to search a periodic trajectory for ever:

Proposition 4 *If an SFT is empty, then there exists $k \geq 0$ such that that any word of length k contains a forbidden word.*

Proof. Assume, conversely, that for each k, there exists a word u_k of length k which does not contain any forbidden word. Let us show that X is not empty. Up to deleting its last letter, one can assume that each u_k has odd length. We build with these words an infinite stack, with u_k being on top of u_{k+1} and the central letter of u_k being in position 0. Since the alphabet the u_k's are defined on is finite, at least one of the infinitely many letters in the column at position 0 of this stack, say x_0, appears infinitely many times. We remove from the stack all the words u_k that do not have x_0 in position 0. We get a new stack which is still infinite. By iterating the process on the columns $\{1, -1, 2, -2, \ldots\}$, we get after $2n+1$ steps a sequence x_i such that all the words of our infinite stack have the letter x_i is position i, for $-n \leq i \leq n$. Let x be the trajectory defined by the x_i's. Since any finite word which appears in x appears in a large enough u_k, x cannot have any forbidden word. Thus $x \in X$: the SFT is not empty. □

The two previous propositions yield an algorithm to decide in finite time whether an SFT defined by forbidden words is empty or not (Algorithm 1).

Algorithm 1: isEmptySFT

```
Data: A finite set $F$ of forbidden words which define an SFT $X$
Result: $X$ is empty or not
$k$ ←length of the largest word in $F$;
hope ← false;
while hope do
    hope ← false;
    for $u$ in words of length $k$ do
        if $uu$ has no word in $F$ then
            return "X is not empty";
        if $u$ has no word in $F$ then
            hope ← true;
return "X is empty";
```

3 2-dimensional shifts

A *configuration* is an element of $\mathcal{A}^{\mathbb{Z}^2}$. The distance between two configurations is defined by

$$d(x, y) := 2^{-\inf\{\max(|i|,|j|) \mid x_{ij} \neq y_{ij}\}}.$$

A *pattern* is a finite set of letters indexed by $\mathbb{Z}^2$ (their positions). A *shift* is a subset of $\mathcal{A}^{\mathbb{Z}^2}$ which avoid a set of forbidden patterns. An *SFT* is a shift defined by finitely many forbidden patterns. A shift over $\mathcal{A}$ is *sofic* if it is the image of an SFT over $\mathcal{B}$ by a map from $\mathcal{B} \to \mathcal{A}$.

Example 15 *The SFT over $\{a, b\}$ where a and b alternate as black and white cells on a checkerboard is an SFT which can be defined by two forbidden pattern:*

$$F = \left\{ aa, \begin{matrix} b \\ b \end{matrix} \right\}$$

Example 16 *We generalize Example 11. Consider the set X_6 of configurations over $\{a, b\}$ such that any finite connected component of a's has even size, where two letters are connected if they are horizontal or vertical neighboor. This is a shift: it can be defined by all the forbidden patterns made of an odd size connected component of a's surrounded by a layer of b's (that is, there is a letter b in each cell connected to an a of the connected component). Let us show that it is sofic.*

Consider an even size connected component (Fig. 1, first picture, with the a's being depicted as white tiles and the b's as dark tiles). Take a spanning tree of the graph whose vertices are the a's and the edges connect neighboor a's. Consider an edge of this spanning tree (Fig. 1, second picture, encircled edge). Removing this edge would split the tree into two subtrees which must have either *both odd size or both even size (to sum up to an even size). We indeed remove this edge only if both subtrees have even size (Fig. 1, third picture). We proceed the same way for each edge (Fig. 1, last picture).*

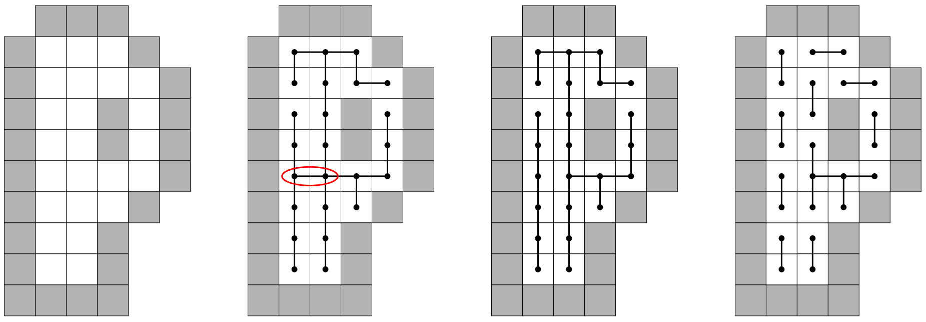

Figure 1: How to transform an odd size connected component into a tiling.

This process transforms the connected component of a's into a tiling by two types of tiles (up to rotation): a white square with one half-edge and one with three half-edges. Indeed, removing a vertex v of the spanning tree would split it into at most 4 subtrees, with the number of subtrees of odd size being odd in order to sum up, together with v, to an even number (the even size subtrees do not change the parity). There is thus 1 or 3 such subtrees, and they are exactly those which remain connected to v once the process completed. Note that the tile with three half-edges can be necessary, for example for a connected component made of three a's in line and a fourth one adjacent to the middle one.

Conversely, consider a finite connected component of a's surrounded by a layer of b's and assume that the connected component can be tiled by the two previous white tiles. If there are p tiles with a half-edge and q ones with three half-edges, then the total number of edges is $(p+q)/2$, which must be an integer. Hence the size $p+q$ of the connected component must be even.

In conclusion, if ϕ maps the two white tiles to a and the dark one to b, then X_6 is exactly the image under ϕ of the set of tilings of the plane by these tiles. This set of tilings can be easily seen as an SFT by using one letter for each tile (in each orientation) and by forbidding two letters to be neighboor if the half-edges on the corresponding tiles do not match. This proves that X_6 is sofic.

The shift defined in the same way except that connected components of a's must have odd size is also sofic, but the proof turns out to be much more difficult.

Let us focus again on the basic question: given a finite set of forbidden patterns, can we decide whether the 2-dim. SFT they define is empty or not? The proof of Proposition 4 extends to this 2-dim. case without difficulty, but the proof of Proposition 3 is very specific to the 1-dim. case. Actually, Proposition 3 does not extend at all to the 2-dim. case:

Theorem 1 *There exists a non-empty SFT without any periodic configurations.*

The first such SFT has been defined by Berger in 1964. A somehow simplified version appears in [3] (see also [1] for a short proof of only this result). The tiles and a (partial view of a) tiling are depicted on Fig. 2.

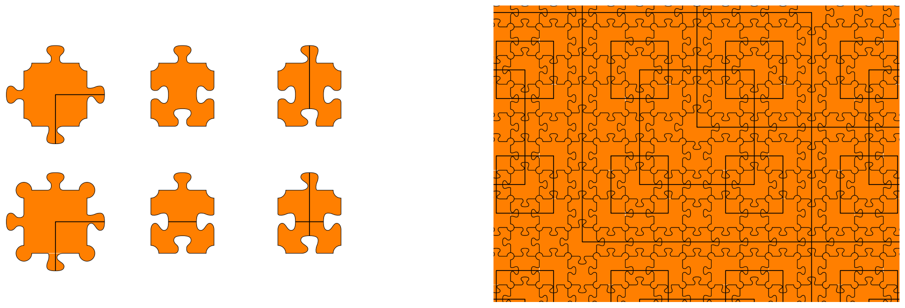

Figure 2: The Robinson tiles (left) and a partial view of a tiling by them.

This means that Algorithm 1 cannot be easily extended to an algorithm which decides whether a 2-dim. SFT is empty or not. As we shall see, such an algorithm cannot exist. The reader who wants to convince himself that the question is hard can try to decide whether the sets of *Wang tiles*[2] on Fig. 3, 4 and 5 can tile the plane or not (the answer is given without a proof).

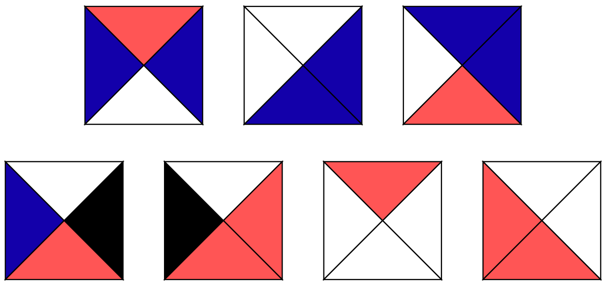

Figure 3: Wang tiles which tile periodically the plane.

[2]Wang tiles are unit square with colored sides which can be translated but not rotated. Each tile can be used as many time as needed to form a tiling of the plane where two adjacent tiles always have the same color on their common side (if such a tiling does exist).

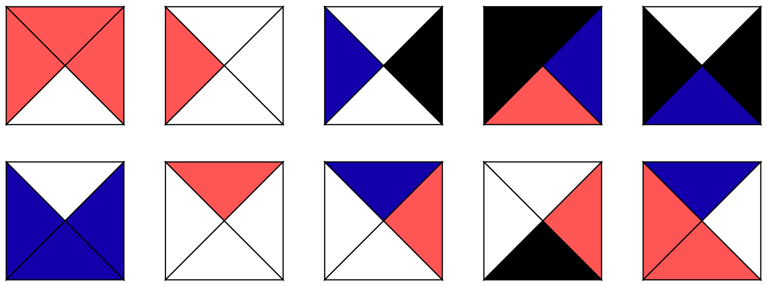

Figure 4: Wang tiles which do not tile the whole plane.

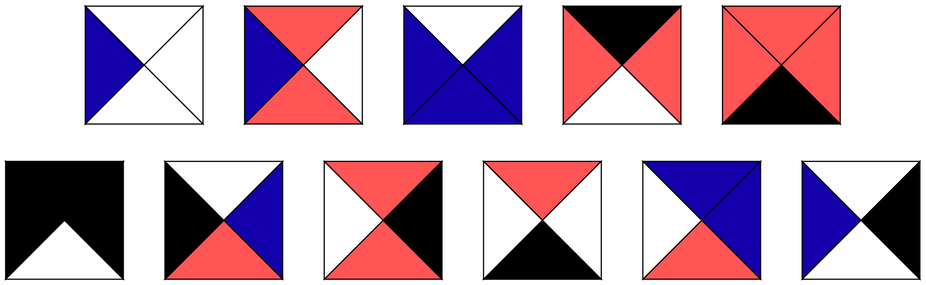

Figure 5: Wang tiles which tile the plane but never periodically.

4 Indecidability

Running a big computation on a computer, it is worth knowing if it will eventually halt or if a bug in the program will make it last forever…This is the so-called *halting problem*, proven to be *undecidable* by Turing in 1937:

Theorem 2 *There is no algorithm deciding whether a program eventually halts.*

Indeed, if such a program `halt` would exist, then it would give a wrong answer on the following program:

```
prog P():
  if halt(P())
    loop forever
  else
    return "goodbye"
```

Programs are here assumed to have no entry parameter (this is what we need further), but the result still holds with entry parameters.

We will need to formalize the notion of program. We rely on *Turing machine*. A Turing machine is a device which can read or write the cells of an infinite tape. It has a finite number of states, a head positioned on the cell of the tape it can read in or write on, and it is defined by transition rules of the type

$$(x, q_j) \to (y, \pm 1, q_k),$$

which means that if the machine reads x in cell i being in state q_j, then it writes y on cell i, moves its head to cell $i \pm 1$ and enters state q_k. It has also a state denoted by $\square$ such that the computation halts if the machine enters this state.

Example 17 *The following Turing machine adds* 1 *to the number written in binary on the tape (least significant bit in the cell it is positioned on), then halts:*

$$\begin{array}{rcl} (1, q) & \to & (0, +1, q), \\ (0, q) & \to & (1, +1, \square). \end{array}$$

Example 18 *What does the following Turing machine on a tape filled with* 0*?*

$$\begin{array}{rcl} (0, q_0) & \to & (1, +1, q_1), \\ (0, q_1) & \to & (1, -1, q_1), \\ (0, q_2) & \to & (1, -1, q_2), \\ (1, q_0) & \to & (1, +1, \square), \\ (1, q_1) & \to & (0, +1, q_2), \\ (1, q_2) & \to & (1, -1, q_0). \end{array}$$

Now, we want to simulate the computation of a Turing machine by a tiling. The idea is that tiles will be squares corresponding to the cells of the tape, with the k-th row of the tiling corresponding to the state of the tape after k steps of computation. We assume that tiles cannot be rotated, and that two neighboor tiles have the same "color" on the side they are adjacent. We also need some tiles to start the computation. Fig. 6–8 illustrates this.

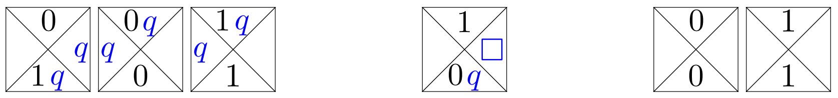

Figure 6: Tiles to simulate the Turing machine of Example 17. The three first tiles simulate the first transition, the next one the second transition, and the two last ones just keep from time k to $k + 1$.

Now, given a Turing machine, consider the corresponding tiles and place somewhere on the plane a tile which starts the computation. If the computation simulated by these tiles eventually halts, then the tiling can not be further extended since no tile match with the state $\square$. On the contrary, if the computation lasts forever, then one gets a tiling of the upper half plane, and there is

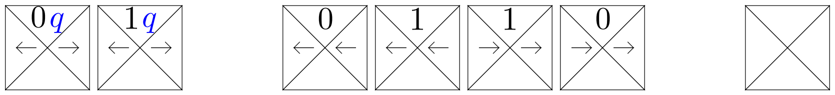

Figure 7: Tiles to initialize the computation of the Turing machine of Example 17. The two first tiles start the computation in state q with 0 or 1 in the current celle. The four next ones allow to fill freely the initial tape with 0 and 1, but with no other head (that is, no parallel computation). The last tile is for the half plane below the initial tape.

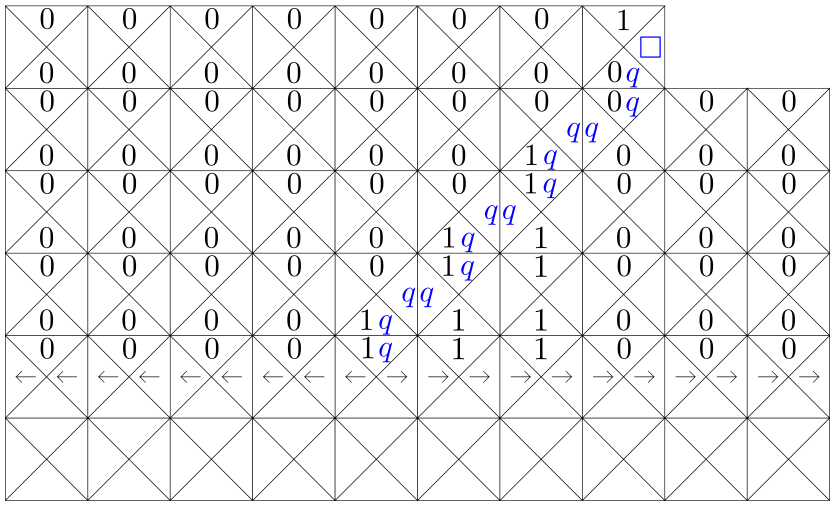

Figure 8: Example of a computation by the above tile (partial view of the tiling). A tile initializes the computation in state q on the second row (from bottom). The computations runs for 3 steps (upwards) and then enters the halting state $\Box$ (top row): the tiling can not be further extended.

thus a tiling of the whole plane via the argument used in the proof of Proposition 4 (if one can tile arbitrarily large squares, then one can tile the whole plane). Hence, any algorithm to decide whether a tile set tiles the plane *with a tile starting the computation* would yield an algorithm to decide whether a Turing machine halts or not.

However, with the above tiles, one can easily tile the plane without using the tile which starts a computation: we could indeed use only tiles that transfer a letter. We thus need a new idea to enforce the tile which starts a computation to appear somewhere. Actually, the computation cannot start in only one place, because it would yields arbitrarily large squares in the tilings without any computation, and again the same argument as in the proof of Proposition 4 would yield a tiling of the whole plane without any computation. The same computation must thus start everywhere! A first idea is to use a periodic tiling to start periodically the computation, but this mean that the whole computation must fit into a period of the tiling, that is, it has both bounded space and bounded time to work. In particular, if the simulated Turing machine needs more time or space to halt, it will not be detected. We thus need a tile set which tile the plane but not periodically.

A solution is to combine the tiles which compute with the Robinson tiles, following [3] (recall Fig. 2). This is done so that any square of the Robinson tiling[3] starts on its lower edge a computation that cannot go outside the square itself. Since the size of these squares is unbounded, the Turing machine eventually halts if and only if a tile with the halting state shall necessarily occur in a large enough square. Since no other tile match this tile, this means that the tiling cannot be completed on the whole plane. Hence, any algorithm to decide whether a tile set tiles the plane or not would yield an algorithm to decide whether a Turing machine halts or not. We skiped here many details (for example, the computation in a large square shall not be mixed with the computations of the smaller squares contained in this large square), but these are the main ideas which allow to prove

Theorem 3 *There is no algorithm deciding whether a 2-dim. SFT is empty.*

References

[1] Thomas Fernique, *Yet another proof of the aperiodicity of Robinson tiles*, arxiv:1711.03401 (2010).

[2] Douglas Lind, Brian Marcus, *An Introduction to symbolic dynamics and coding*, Cambridge University Press, 1995.

[3] Raphael Robinson, *Undecidability and nonperiodicity for tilings of the plane*, Invent. Math. **12** (1971), pp. 177-209.

[3]Actually on one level over two of the hierarchy of squares drawn on any Robinson tiling.