Computability of Tilings

Grégory Lafitte[1] and Michael Weiss[2]

[1] Laboratoire d'Informatique Fondamentale de Marseille (LIF), CNRS – Aix-Marseille Université,
39, rue Joliot-Curie, F-13453 Marseille Cedex 13, France
[2] Centre Universitaire d'Informatique, Université de Genève, Battelle bâtiment A,
7 route de Drize, 1227 Carouge, Switzerland

Abstract Wang tiles are unit size squares with colored edges. To know whether a given finite set of Wang tiles can tile the plane while respecting colors on edges is undecidable. Robinson's tiling is an auto-similar tiling in which the computation of a Turing machine can be carried out. By using this construction and by considering a strong notion of simulation between tilings, we prove computability results for tilings. In particular, we prove theorems on tilings that are similar to Kleene's recursion theorems. Then we define and show how to construct reductions between sets of tile sets. We generalize this construction to be able to transform a tile set with a given recursively enumerable property into a tile set with another property. These reductions lead naturally to a Rice-like theorem for tilings.

Introduction

In [17], Wang introduced the study of tilings with colored tiles. A tile is a unit size square with colored edges. Two tiles can be assembled if their common edge has the same color. To tile consists in assembling tiles from a tile set (a finite set of different tiles) on the grid $\mathbb{Z}^2$. The tiles can be repeated as many time as needed, but cannot be turned.

Two questions arose from these definitions. The first one, conjectured true by Wang, was to know whether any tile set that can tile the whole plane can also tile it in a periodic way, *i.e.*, there exists two linearly independant vector u and $v \in \mathbb{Z}^2$ such that for any position $z \in \mathbb{Z}^2$, the tiles at position z, $z+u$ and $z+v$ in the tiling are the same. The second one, known as the *domino problem*, is to know if one can decide whether a given tile set can generate a tiling of the plane.

Both of the questions were answered by Berger in [3]. In his thesis, Berger constructed for any Turing machine M and any input w, a tile set $\tau_{M,w}$ such that this tile set can generate a tiling of the plane if and only if the computation of M stops on the input w. This construction proved the undecidability of the domino problem, and also proved that there exist aperiodic tile sets, *i.e.*, tile set that produces only aperiodic tiling (similarly, a tile set is said to be periodic if it generates at least one periodic tiling). This technical construction was improved later, and simplified constructions of aperiodic tile sets can be found in [16] and [1].

Since the main argument of Berger's proof was to simulate the behavior of a given Turing machine with a tile set, then one of the most important fact concerning tilings is that tilings can constitute a Turing equivalent computation model. This computation model is particularly relevant as a model of computation on the plane.

The study of tilings has made possible the resolution of mathematical logical problems ([1]). Then researchers have been interested in studying the kinds of tilings that one tile set can produce ([16] and more recently [5, 8, 12]). Others have defined tools to quantify the regular structure of a tiling ([6, 2, 13]). Recently, notions of simulation between tilings have been defined to obtain a first approach to computability results on tilings ([12, 14]).

In this paper, we aim at proving computability results for tilings. To reach this goal, we use the construction most used nowadays: Robinson's tiling. In [16], Robinson has built a tile set that generates only auto-similar aperiodic tilings. The construction is based on a hierarchy of squares of ever increasing sizes. In each of these squares, some zone can be used to simulate the behavior of a Turing machine. In [12], notions of simulation and reduction between tilings and tile sets have lead to notions of universality for tilings and completeness for tile sets. Finer notions of simulation have been defined in [14]. These notions rely on Robinson's construction to study the computability of problems related to simulation. In this paper, we make a heavy usage of this construction to prove classical computability results for tilings.

In classical computability (recursion theory) all theorems derive from the enumeration and *s-m-n* theorems. Kleene's recursion (or fixed point) theorem is a direct application of *s-m-n*. With tilings, an *s-m-n* approach would be unnatural because of the particular geometrical nature of computation in this model. Nevertheless, Kleene's theorem is a tool that seems to be more naturally fitted to be transposed on tilings. Our goal in this paper is to show how a computability can be shaped on the geometrical computation model of tilings, and not merely to use classical computability to obtain tools on tilings. In traditional computability, Kleene's theorem states that for any recursive modification of programs M, there exists a program p which is a fixed point for M, *i.e.*, p and $M(p)$ compute the same function. So two Turing machines can be seen as equivalent if they compute the same function. To obtain a Kleene-like theorem for tilings, we need notions of comparison of tile sets: one such notion is the exact simulation. The general idea is to say that a tile set τ exactly simulates a tile set τ' if τ generates a set of rectangles of equal sizes which are isomorphic to the tiles of τ'. From this, we can obtain Kleene-like theorems for tilings.

Beyond Kleene-like theorems, we show how to construct reductions between sets of tile sets. Reductions are fundamental notions in computability theory. Natural notions of reductions between sets of tile sets are also fundamental for tilings. In fact, the idea behind the construction of these reductions lies in Kleene's recursion theorem with parameters: to inject some property in the fixed point being constructed. The reduction constructed is not only interesting for applications but also in itself: it shows how to transform a tile set with a certain property into another tile set with another property. A generalization of this construction leads to another main computability result: Rice's theorem. This theorem states that for any property P on the set of partial recursive functions, if there exist at least one function which satisfies P and one which does not then it is not decidable to know if a given Turing machine computes a function satisfying the property P. Again with the exact simulation, we can state this theorem for tilings as follows: if A is a set of tile sets, then it is not decidable to know whether a given tile set τ exactly simulates a tile set of A. We note that in [4], a first and different approach to a Rice-like theorem for the local constraints has been done, where local constraints are a tiling equivalent model. In this paper, the authors show that it is not decidable to know whether two local constraints can produce the same set of tilings. Our approach is different since we consider the exact simulation as the way to compare tile sets. With the exact simulation, we show how to build reductions between tile sets which lead naturally to a Rice-like theorem.

The main result of this paper is to obtain different Kleene-like theorems using Robinson's construction. We also show that some of these results can be proved with another natural construction introduced in [9] to construct an aperiodic self-similar tiling using Kleene's theorem.

From there, we show how to construct reductions between sets of tile sets and obtain a Rice-like theorem for tilings. The striking aspect of this work holds primarily in the fact that these reductions exist and in the detailed description of their construction.

In Sec. 1, we recall the basic notions of tilings and simulation between tile sets and recall the two main definitions of simulation, the total and the exact ones introduced in [14]. In Sec. 2, we recall the construction of Robinson's tiling and how it can carry out the simulation of a Turing machine. In Sec. 3, we improve this construction to obtain a famous result proved in [10]: the set of periodic tile sets is Σ_1-complete. In Sec. 4, we prove three Kleene-like theorems for tilings. In the last section, we define how to construct reductions between sets of tile sets and prove a Rice-like theorem for tilings.

1 Notions of simulation

We begin with the basic notions of tilings. A tile is an oriented unit size square with colored edges from C, where C is a finite set of colors. A tile set is a finite set of tiles. To tile consists in placing the tiles of a given tile set on the grid $\mathbb{Z}^2$ such that two adjacent tiles share the same color on their common edge. Since a tile set can be described with a finite set of integers, then we can enumerate the tile sets, and τ_i designates the i^{th} tile set.

Let τ be a tile set. A tiling P generated by τ is called a τ-tiling. It is associated to a tiling function f_P where $f_P(x,y)$ gives the tile at position (x,y) in P. When we say that we superimpose the tiles of a tile set τ on the tiles of a tile set τ', we mean that for any tile $t \in \tau$ and any tile $t' \in \tau'$, we build a tile $u = t \times t'$ where the colors of the sides of u are the cartesian product of the colors of the sides of t and t'. Then two tiles $u_1 = t_1 \times t'_1$ and $u_2 = t_2 \times t'_2$ match if and only if t_1 and t_2 match and t'_1 and t'_2 match.

Different notions of reduction have been introduced in [12] and in [14]. We recall some of the notions relative to these reductions and we refer the reader to these papers for detailed explanations and properties.

A pattern is a finite tiling. If it is generated by τ, we call it a τ-pattern. A finite set of rectangular τ-patterns of even size is a τ-pattern set. By analogy with tilings, to tile with a pattern set consists in placing the patterns on a regular subgrid of $\mathbb{Z}^2$ in such a way that the connection between two patterns respects the local constraint of color matching. We call a tiling P generated by a pattern set M, an M-tiling. If M is a set of τ-patterns, then for any M-tiling P, there exists a τ-tiling Q which is a representation of P at the unit tile level.

From this remark we obtain notions of simulation. We say that a pattern tiling P simulates a tiling P' if there exists a function R from the patterns of P to the tiles of P' such that if we replace the patterns of P by their corresponding tiles given by R, then we obtain P'. In such a case, we write $P' \trianglelefteq^R P$ and say that P' reduces to P. If R is not determined, we denote the fact that P' reduces to P by $P' \trianglelefteq P$. The main thing in this reduction is that R is not necessarily a one-to-one function. Different patterns of P can represent the same tile of P'.

This is the least restrictive notion of simulation that we have. We require of a tile set to be able to simulate the behavior of another tile set with patterns. This can be done by any tile set that can produce rectangle patterns whose sides can encode colors. From this simulation, we can define notions of universality for tilings and completeness for tile sets: a tiling P is strongly universal if for any tile set τ, there exists a τ-tiling Q such that $Q \trianglelefteq P$ and a tile set τ is complete if for any tile set τ' and any τ'-tiling Q there exists a τ-tiling P such that $Q \trianglelefteq P$. Therefore, universality is a property of tilings. A tiling is universal if it can simulate the behavior of at least one tiling for any tile set. Completeness is a property of tile sets. A tile set τ is complete if for any tiling P it can generate a tiling having the behavior of P.

In [14], two finer notions have been introduced:

Definition 1. Let τ and τ' be two tile sets. We say that τ *totally simulates* τ' if there exist $a, b \in \mathbb{Z}$ and a reduction R from the $a \times b$ patterns of τ to the tiles of τ' such that the two following conditions are respected:

1. for any τ'-tiling Q, there exists a τ-tiling P such that $Q \trianglelefteq^R P$,
2. for any τ-tiling P, there exists a τ'-tiling Q such that $Q \trianglelefteq^R P$.

We denote it by $\tau' \trianglelefteq_t \tau$ (or $\tau' \trianglelefteq_t^R \tau$ to specify the reduction R).

If $\tau' \trianglelefteq_t \tau$, then there exists a reduction R such that any τ-tiling can be cut in rectangle patterns of size $a \times b$ such that if one replaces these patterns by their corresponding tiles given by R then one obtains a τ'-tiling. And the set of all τ'-tilings that reduce to a τ-tiling is exactly the set of all τ'-tiling. The total simulation is thus more specific than the simulation introduced in [12]. In this way, τ can be seen as a tile set which *computes* in a same way than τ'.

A tile set τ exactly simulates a tile set τ' if τ totally simulates τ' and if the reduction R between τ and τ' is one-to-one. In the total simulation, different patterns can represent the same tile; in the exact one, any tile is represented by only one pattern. It is this simulation that we use to prove our computability theorems for tilings.

To be able to study these notions of simulation, we now recall the classical Robinson construction and some of its specific aspects that we will use later on.

2 Basic notions of simulation of a tile set

Since Berger's proof of the domino problem, we know that we can simulate a Turing machine with a tiling. To any Turing machine M and any input w, we can associate a tile set which simulates the behavior of the computation of M on w. Nowadays, the most used construction to simulate a Turing machine is based on Robinson's tile set (Fig. 1). In [16], Robinson built an aperiodic tiling. This tiling is based on a hierarchy of squares of ever-increasing sizes (Fig. 1.1) shows this hierarchy for the first three levels. These squares are of sizes $2^n + 1$. The idea is to dedicate spaces (the white spaces in Fig. 1.2) in each square of size $2^{2n} + 1$ to simulate a Turing machine by forcing the lowest southwest tile of any of these squares to have the tile representing the initial state of M on the input w. For more details and explanations of this construction, we refer the reader to [1].

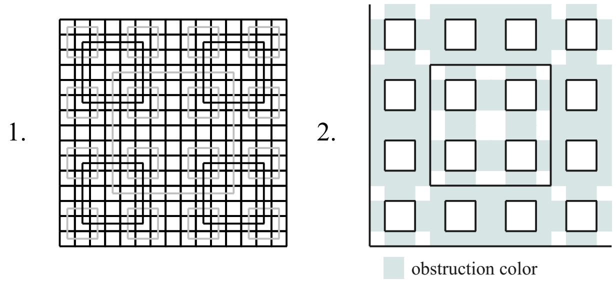

Fig. 1 The hierarchical structure and the obstruction zone in Robinson's tiling

In [12], a tile set is simulated by a Turing machine, in the sense that for any tile set τ, we build a Turing machine M_τ that produces space×time diagrams of same size which are isomorphic to the tiles of τ, where the size of the space×time diagrams are the length and width of the diagrams , *i.e.*, the time and space needed to reach a final state.. This can be done with a Turing machine that takes as input two integers: i, the code of the index of a tile set, and j, the code of a color of τ_i. The Turing machine checks if j is the code of a color of the south side of τ_i. If yes, it computes in a non-deterministic way a tile of τ_i with south color j, as shown in Fig. 2. Then we can simulate this Turing machine in Robinson's tiling and obtain a tile set which simulates totally or exactly, depending on the conditions used, another tile set. For a detailed explanation we refer the reader to [12] and also [14] where constructions of particular tile sets with simulation conditions are built.

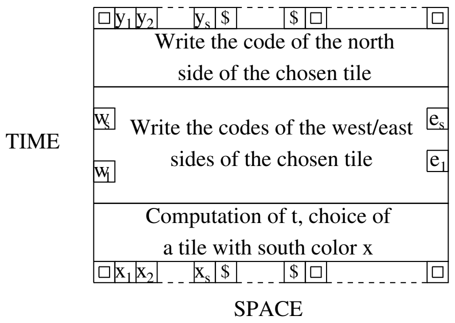
□ y1 y2 ys $ $ □ □
Write the code of the north side of the chosen tile
ws es
Write the codes of the west/east sides of the chosen tile
w1 e1
Computation of t, choice of a tile with south color x
□ x1 x2 xs $ $ □ □
TIME
SPACE

Fig. 2 The space×time diagram of a Turing machine representing the simulation of a tile

3 Periodicity if and only if a Turing machine stops

In this section we use the construction making possible the simulation of a Turing machine in Robinson's tiling in order to obtain a well known result proved in [10]: the undecidability of the periodic tilability of the plane. The explanations that follow are an introduction to the construction that we will use in the following sections to prove computability results for tilings.

Robinson's tiling is a tiling with a hierarchy of squares of ever increasing sizes. The squares of level one are of size 3 and the squares of level i are of size $2^i + 1$. We can see that the squares of level n are based on a regular subgrid of $\mathbb{Z}^2$ where two lines and two columns are separated by $2^n - 1$ tiles. Actually, one can note that these lines and columns are composed of the alternation of two different sequences of $2^n - 1$ tiles separated by corner tiles, one of these sequences representing the side of a square of the n^{th} level. We call this subgrid on which is based the squares of level n, the n^{th} grid. Therefore, the sides of any squares of level n is part of the n^{th} grid.

We can tile Robinson's tiling in a sequence of stages: at stage one, we tile the first grid on $\mathbb{Z}^2$. At stage n, we tile the n^{th} grid and modify, if needed, the tiles of the lowest grids with which the n^{th} grid intersects. This can be done without changing the structure of squares made until this stage. We can proceed like that until the end of the process and we will obtain Robinson's tiling. But we can see that after having tiled the n^{th} level, if we choose to add to our tiling a simple grid, *i.e.*, a grid that does not contains square of the Robinson hierarchy, of same size than the n^{th} grid, and translated in such a way that its corner tiles are in the middle of the squares of the n^{th} grid, then we complete the tiling and make it periodic since we have stopped the self-similarity. Fig. 3 shows the black grid which is inserted in the tiling.

We add to Robinson's tile set special tiles that can generate squares of Robinson's tiling marked with a special color. Thus, at a certain level n, we can decide to tile the n^{th} grid either with the tiles of Robinson's tile set or with the special marked tiles. The special colored tiles have the particularity to not allow squares of higher level to intersect it. Therefore, when one has decided to tile a level with these special tiles, then the self-similarity of Robinson's tiling stops. The only way to complete the tiling,

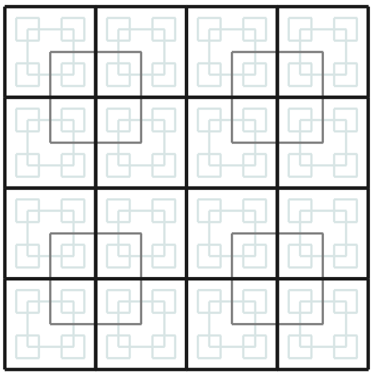

Fig. 3 The blocking color (dark gray) forces the completion of the tiling by adding a regular subgrid (black) that stops the self-similarity of Robinson's tiling (clear gray)

is to do as said in the previous paragraph: we tile a simple grid and, by stopping the self-similarity, we obtain a periodic tiling.

Fig. 3 shows what happens when one decides to tile the squares of level n with the blocking color (here, in black gray). Since no other square of higher level can be added to the tiling, the only way to complete the tiling is to add a simple grid formed of squares of sizes $2^n + 1$ (here, in black).

We now have to add a condition to force to tile with the special colored tiles. Let M be a Turing machine. We build the tile set τ_M which simulates M on the empty input. On the lowest southwest tile of any square of level $2n$, we begin the simulation of M with τ_M with the condition that if a final state is reached before reaching the perimeter of the square, then a special color is sent to the north side of the square that forces the perimeter of the square of level $2n$ - and thus the whole $(2n)^{th}$ grid - to be tiled with the special colored tiles. Then the self-similarity is stopped and the tiling is periodic if and only if M stops on the empty input.

To be more precise, we can compute the exact period of this tiling. If we choose to stop the self-similarity of Robinson's tiling at the level $2n$, then the squares of the hierarchy are of size $2^{2n} + 1$ and at least $2^{2n} - 1$ tiles separate two sides of two squares of level $2n$. Therefore, the smallest period is a square of size 2^{2n+1}. In Fig. 3, the period is represented by a square composed of four blue squares.

In the following sections, we used these different constructions to obtain computability results for tilings.

4 Kleene-like theorems for tilings

The first result we want to obtain is a theorem like Kleene's fixed point theorem but for tilings. Kleene's theorem, in classical computability, states that for any recursive function f, there exists a Turing machine M_e[1] such that the function computed by the

[1] Where M_e denote the e^{th} Turing machine according to an acceptable enumeration of Turing Machines

Turing machine M_e is the same than the one computed by $M_{f(e)}$. We can state it as follows: for any recursive modification of programs f, there exists a program p such that p and its modification $f(p)$ give the same result when computing on the same input. For tilings, we cannot compare functions but we can compare their behavior. We have in the exact simulation the notion of comparison that we need. Therefore, a Kleene-like theorem for tilings can be stated as follows: for any modification f of tile sets, there exists a tile set τ such that τ exactly simulates the modification of τ by f.

Theorem 1. *Given a recursive function f, there exists an e such that τ_e simulates exactly $\tau_{f(e)}$.*

Proof. Let f be a recursive function and M_f a Turing machine which computes f. Let M be the Turing machine that has the following behavior: when the input is the empty word, M computes an integer i. After having computed i, M simulates M_f on the input i. We consider Robinson's tiling where the lowest southwest corner of each square of level n, and thus of size $2^{2n}+1$, of the hierarchy of Robinson's tiling is a tile representing the initial state of M. The simulation of the computation of M is made in this square until it has computed the value $f(i)$. When this value has been computed, a special color is sent to the north board of the square that colors the whole perimeter of this square with this special color (Fig. 4.1). This special color is also a blocking color, *i.e.*, the self-similarity of Robinson's tiling is stopped. Then we send the bits composing $f(i)$ to the south board of the square. This can be done by superimposing the bits of $f(i)$ on the computation tiles.

Therefore, the first line of the square is marked with the bits of $f(i)$ and with the special color, as well as the whole perimeter of the square. When the square is marked with the special color, the computation of a new Turing machine, say N, can begin. N is a Turing machine which takes as inputs an integer x, the index of a tile set, and an integer y, the index of a color of τ_x and computes a tile of the tile set τ_x with south color y, *i.e.*, the space×time diagram of the computation of N on x and y is isomorphic to a tile of τ_x with south color y. In our tiling, we want to simulate a tile of the tile set $\tau_{f(i)}$. Since we already have the bits of $f(i)$ on the first line, we just need to add an integer y, following $f(i)$, which represents the index of a color of the tile set $\tau_{f(i)}$, and then begin the computation of N on $f(i)$ and y (Fig. 4.2).

If y is not a south color of a tile of $\tau_{f(i)}$, then the computation enters an error state, and the tiling cannot be completed. Therefore, the tiling process keeps going on if and only if we have chosen a valid color y. Then N computes the simulation of a tile with south color y. Thus, there exists a level $2n$ such that any square of this level carries out the computation of a tile of $\tau_{f(i)}$.

The last thing that has to be done, to guarantee that two neighboring squares of level $2n$ carry out the simulation of two tiles that match, is to send the codes of the colors on the sides of the squares of level $2n$ outside the square. This guarantees that the zone between two neighboring squares contains the code of a common color.

Those squares of level $2n$ are the biggest of the tiling, since the self-similarity has been stopped. Two squares, carrying out the simulation of the same tile, are composed exactly of the same tiles. There exists only one way for a square to carry out the simulation of a given tile. Therefore, the reduction is an isomorphism and the tile set

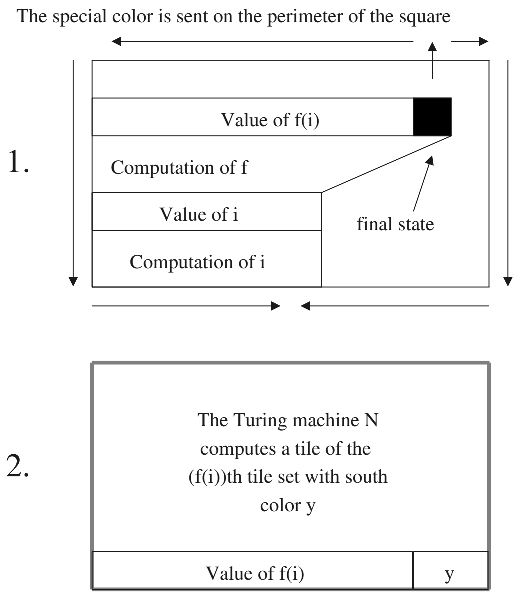
The special color is sent on the perimeter of the square
1.
Value of f(i)
Computation of f
Value of i
final state
Computation of i
2.
The Turing machine N computes a tile of the (f(i))th tile set with south color y
Value of f(i)
y

Fig. 4 The computation of M and N in a square of computation of Robinson's tiling

can simulate any tiling generated by $\tau_{f(i)}$ and does not generate a tiling that does not simulate a $\tau_{f(i)}$-tiling. Therefore, the simulation is exact.

We would like for our tile set to have access to its own index to be able to simulate itself but modified by f. This is not an all natural fact, since each time that we add tiles to our tile set to try to encode the code of the tile set, we change the code of the tile set. To prove this, we need Kleene's theorem with parameters which states that for any recursive function g with two parameters, there exists a recursive function n such that for any index of Turing machine e, $M_{n(e)}$ and $M_{g(n(e),e)}$ compute the same function. We consider a recursive function g which takes as inputs a tile set that generates Robinson's tiling, or a Turing machine able to simulate this tile set, and a Turing machine M, and outputs the code $g(\text{Robinson's tile set}, M)$ of a Turing machine which has the following behavior: it computes the index of the tile set which is the simulation of M in Robinson's tiling. By Kleene's theorem, there exists a function n such that $M_{n(M)} = M_{g(n(M),M)}$. Here, $n(\text{Robinson's tile set})$ is our fixed point and represents a Robinson tiling which has access to its own code. This proves that when we simulate a Turing machine in a tiling, we can always suppose that we can do it by having access to the code of this tile set written somewhere in the tilings that it generates.

Therefore, we can suppose that there exists M which gives the index i of its own tile set and thus, the tile set simulated exactly itself modified by f. This proves that this tile set τ_i exactly simulates $\tau_{f(i)}$. □

We now show another version of Kleene's theorem on tilings: Kleene's theorem with parameters. This theorem in a classical computability setting is of great usefulness, as shown at the end of the previous proof. This theorem states that for any recursive function f, there exists a recursive function n such that $M_{n(y)} = M_{f(n(y),y)}$. For tilings, we expect to obtain a similar result.

Theorem 2. *For any recursive function f with two parameters, there exists a recursive function n such that for any tile set τ_i, $\tau_{n(i)}$ exactly simulates $\tau_{f(n(i),i)}$.*

Proof. Let f be a recursive function which takes as input two indexes i, j of tile sets and transforms them in a tile set $\tau_{f(i,j)}$.

Let M_f be the Turing machine with the following behavior: it takes as input two integers x and y and computes $f(x,y)$.

As we did before, to obtain the inputs x and y we can use two Turing machines M^x and M^y which compute, from the empty input, respectively x and y. Let τ_M be the tile set that simulates the Turing machine M which has the following behavior: it simulates M^x and M^y from the empty input and then simulates M_f on x and y to obtain $f(x,y)$.

We simulate the behavior of these Turing machines with τ_M in Robinson's tiling. To do that, the lowest southwest tile of any square of size $2^{2n}+1$ contains the tile representing the initial state of M: then, the tiling τ_M generates two integers x and y and computes $f(x,y)$. We send to the southeast line of the square, the bits of $f(x,y)$, to have a plain access to this code. They represent the index of the tile set we want to simulate. As we did before, the final state of M sends a special color to the north side of the square that forces the perimeter of the square to be colored with this special color. This special color triggers the computation of a new Turing machine, say N, that simulates the tiles of the tile set $f(x,y)$. If the square is big enough to carry out the computation of the tiles of the tile set of index $f(x,y)$, then a blocking color is sent to the north side of the square of computation which forces the whole perimeter of the square to be colored with this blocking color and stops the self-similarity of Robinson's tiling. As we have seen in the previous proof, stopping the self-similarity allows the simulation to be exact.

Therefore, we have a tile set τ^{M^x,M^y}, depending on M^x and M^y, which simulates exactly the tile set $\tau_{f(x,y)}$. For any tile set τ_i, and any Turing machine M^i which computes i when given the empty input, by using Kleene's theorem with parameters, we have seen that we can find a Turing machine $\mathscr{M}^x$ such that $\mathscr{M}^x$ outputs the index of the tile set $\tau^{\mathscr{M}^x,M^i}$, *i.e.*, the tile set that has the following behavior: it simulates $\mathscr{M}^x$ on the empty input, which gives the code of the tile set, say k; then it simulates M^i which outputs i and computes $f(k,i)$. Finally, it simulates the tile set with index $f(k,i)$. Let n be the recursive function that transforms the index i into the index of the tile set $\tau^{\mathscr{M}^x,M^i}$, *i.e.*, k. Therefore, $n(i)$ is a fixed point. Indeed, $\tau_{n(i)} = \tau^{\mathscr{M}^x,M^i}$ exactly simulates the tile set $\tau_{f(\mathscr{M}^x(\varepsilon),M^i(\varepsilon))} = \tau_{f(n(i),i)}$. □

The two previous theorems can be proved without using Robinson's construction. To do that, we can use the construction introduced in the paper [9]. In this paper, the authors use Kleene's recursion theorem to build an aperiodic tiling. The idea is to cut $\mathbb{Z}^2$ with rectangular equal patterns, where each tile of the rectangle knows its position in this rectangle. This can be done by using a special tile for any position of these rectangles. Then one superimposes on each rectangle the computation of a Turing

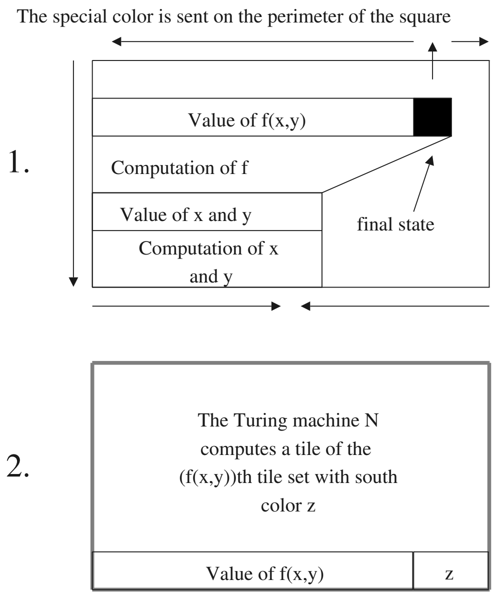
The special color is sent on the perimeter of the square
1.
Value of f(x,y)
Computation of f
Value of x and y
Computation of x and y
final state
2.
The Turing machine N computes a tile of the (f(x,y))th tile set with south color z
Value of f(x,y)
z

Fig. 5 The computation of M and N in a square of computation of Robinson's tiling

machine simulating a tile of a tile set. One can modify this tile set, say τ, in such a way that each rectangle simulates a tile of τ. By using Kleene's recursion theorem, one obtains a tile set that simulates itself and thus, cannot be periodic.

We can also use this construction to prove our theorem. Since we can know the time needed to compute x, y and $f(x,y)$ then we can apply the same argument and simulate M_x, M_y, the computation of $f(x,y)$ and the simulation of the tiles of $f(x,y)$ in a determined rectangle. The conclusion is the same than in the previous proof. We just have to simulate, as before, the tile set which simulates itself modified by f. Therefore, the recursive function n, that takes as input the code i of a tile set, and outputs the code $n(i)$ of a tile set which computes: $M^{n(i)}$, M^i, and the tiles of the tile set with index $f(M^{n(i)}(\varepsilon),M^i(\varepsilon)) = f(n(i),i)$, is a fixed point and $\tau_{n(i)}$ exactly simulates the tiles of the tile set $\tau_{f(n(i),i)}$.

Another version of Kleene's theorem that we prove is the doubled-fixed point theorem: if f and g are two recursive functions of two variables, then there exist a and b such that: $M_a = M_{f(a,b)}$ and $M_b = M_{g(a,b)}$. In the context of tilings, we obtain the following theorem:

Corollary 1. *Let f and g be two recursive functions of two variables. Then there exist two indexes k and j of tile sets such that τ_k exactly simulates $f(k,j)$ and τ_j exactly simulates $g(k,j)$.*

Proof. We use the two Kleene-like theorems we have just introduced. Since f is a recursive function with two variables, then, by theorem 2, there exists a recursive function n such that for any index i of a tile set, $n(i)$ exactly simulates the tile set with index $f(n(i),i)$. Now, by theorem 1, there exists a tile set of index j which exactly simulates the tile set $g(n(j),j)$. Then set $k = n(j)$. □

In the next section we show how we can reduce properties between tilings to study their computability, and obtain a Rice-like theorem for tilings and simulation.

5 Reductions of properties and Rice-like theorem for tilings

The construction used in the previous section can be modified to obtain other computability results for tilings. This construction can be slightly adapted to obtain the simulation of a certain tile set if a condition is fulfilled. Thereby, we are able to study the computability of different properties on tilings.

We consider the set $A_P = \{\, i \,|\, \tau_i$ has the property $P\,\}$, where P is a property on the tilings generated by τ_i. One example can be the set A_{per}, the set of tile sets that generates a periodic tiling. We prove the following theorem, that has first been proved in [10]:

Theorem 3. *$A_{per} \equiv K_0$, where K_0 is the set of pairs $\langle i,w \rangle$ such that the Turing machine M_i stops on the input w, and thus is Σ_1-complete.*

Proof. In Sec. 3, we have shown that $K_0 \leq_1 A_{per}$. It suffices to show that A_{per} is in Σ_1. The property "τ *is periodic*" can be defined as follows: there exists an n such that τ generates a pattern of size n which is a periodic pattern. Thus, A_{per} can be defined with an $\exists$ arithmetical property. □

To prove the previous theorem, we have reduced the halting problem to the problem to know whether a tile set generates periodicity, by forcing a tile set to have a property if a Turing machine halts on a given input. This kind of argument can be generalized to tile sets to obtain reduction between sets of tile sets. We have the following definition:

Definition 2. Let A and B be two sets of tile sets. A reduces to B (noted $A \leq B$) if there exists a recursive function f such that $i \in A \Leftrightarrow f(i) \in B$.

We show a first kind of reduction between sets of tile sets by proving that the set of periodic tile sets reduces to non-recursive tile sets, *i.e.*, tile sets that produces only non recursive tilings of the plane.

Theorem 4. *Let B_{nr} be the set of non recursive tile sets, i.e., tile sets that produce only tilings of the plane which cannot be defined by a recursive function. Then $A_{per} \leq B_{nr}$ and thus, B_{nr} is not a recursive set.*

Proof. Let τ be a periodic tile set and ρ be a non-recursive tile set. Since [11] and [15], we know that such tile sets exist. Let M be the Turing machine that enumerates the rectangle patterns generated by τ and which stops if and only if τ generates a periodic pattern. As we did before, we simulate M in Robinson's tiling and we block the self-similarity of Robinson's tiling if a final state is reached. Thus, if a period exists, then there exists a level of squares which is all tiled with the blocking color.

We want for our property of generating only non-recursive tilings to appear if and only if τ generates a periodic tiling. We have shown that we can simulate a tile set τ with another tile set, by inserting in Robinson's tiling the simulation of a Turing machine that has the particularity to produce space×time diagrams which are isomorphic to the tiles of τ. Therefore, if a square is marked with the blocking color, it allows the beginning of the computation of a new Turing machine, say N, which has the particularity to produce space×time diagrams which are isomorphic to the tiles of ρ. Without loss of generality, we can consider that N takes always less time and space than M to reach a final state, and thus, if a square can carry out the computation of M, it can also carry out the one of N. Let τ' be this tile set. A simulation of a tile of ρ by τ' is made in a square if and only if the computation of M stops in this square. By adding the condition that the color of the sides of the squares of level n are sent outside the square to force the matching with the neighboring squares, then we obtain the simulation of a ρ-tiling. This tiling cannot be recursive, since it would imply that the tiling it simulates is recursive too. If τ does not generate a periodic tiling, then the squares never carry out the simulation of tiles of ρ and thus, the tile set τ' can generate recursive tilings.

By construction, we have that the self-similarity is stopped and the simulation of the tiles of ρ is made if and only if τ is periodic. As seen before, τ' exactly simulates ρ and thus τ' cannot be recursive. Therefore, τ is periodic if and only if τ' is not recursive.

The reduction that associates to any tile set τ, the tile set τ' shows that B_{nr} is not a recursive set. □

In the previous proof, we have reduced the property of being periodic to the property of being non recursive. This construction can be generalized to obtain other reductions. The main argument of the proof is that, as for Kleene's theorem with parameters, we can inject in a tiling the computation of a program who checks if a property is satisfied in order to obtain a tiling with another property if the previous one is satisfied. The property that we want to verify can be any property P such that it is recursively enumerable to know whether a tile set satisfies it or not. Therefore, we can reduce tile sets satisfying a recursively enumerable property to tile sets with another property. Such recursively enumerable property can be, for example: τ does not tile the plane, τ simulates exactly ρ (where ρ is fixed), τ generates patterns using all its tiles.... Then, if the property is satisfied, we can trigger the start of an exact simulation of a tile set satisfying another property.

By generalizing this kind of construction, we can obtain a Rice-like theorem for exact simulation of sets of tile sets. The only thing we need, is to have a set of tile sets such that if a tile set τ satisfies the property, then any tile set simulating exactly τ has the property too. We define formally this property:

Definition 3. Let A be a set of tile sets. A is an *exact index set* if for any index $i \in A$ of a tile set, if a tile set τ_j exactly simulates τ_i then $j \in A$.

Rice's theorem for Turing machines states that to know whether a Turing machine accepts a language which is in a set A of recursively enumerable languages is not decidable except if A is trivial (empty or if it contains all enumerable languages). We can compare Turing machines by the functions they accept. For tile sets, we do not have a notion of function to compare them. Therefore, if we want a Rice-like theorem for tile sets, the set of tile sets has to be an exact index set and contains the tile sets which "compute" in a same way.

Theorem 5. *Let A be an exact index set. Then the set A is recursive if and only if A is trivial, i.e., $A \neq \mathbb{N}$ and $A \neq \emptyset$.*

Proof. Let A be an exact index set. Since A is not trivial, thus there exist at least one index $i \in A$ and one index $j \notin A$. We first suppose that Robinson's tile set is not in A.

We will reduce L_{per} to L_A as we did in the previous proof. For that, we just have to build from a tile set τ_k, a tile set $\tau_{f(k)}$ such that $\tau_{f(k)}$ simulates τ_i - whose index is in A - if τ_k is periodic, and does not simulate it if τ_k is not periodic. Therefore, this tile set is in A since A is an exact index set.

If τ_k is not periodic, then the only tile set that $\tau_{f(k)}$ exactly simulates is Robinson's tile set.

Therefore, $\tau_k \in L_{per} \Leftrightarrow \tau_{f(k)} \in L_A$.

If Robinson's tile set is in A, then we just have to consider $\overline{L_A}$ instead of $L(A)$. □

To have a better intuitive understanding of this theorem, we can state it as follows: let P be a property on the tilings generated by a tile set satisfying the following statement: if τ satisfies P, then any τ', that exactly simulates τ, satisfies P. Then to know whether a given tile set satisfies P or not is undecidable except if any or no tile set satisfies P.

Acknowledgements

We warmly thank Bruno Durand, Andrei Romashchenko and Alexander Shen for giving us a preprint of their paper [9] and for the discussions on their exciting use of Kleene's fixed point theorem for constructing an aperiodic tile set.

References

1. ALLAUZEN (C.) and DURAND (B.), *The Classical Decision Problem*, appendix A: "Tiling problems", p. 407–420. Springer, 1996.
2. BALLIER (A.), DURAND (B.) and JEANDEL (E.), « Structural Aspects of Tilings », to appear in *Proceedings of the Symposium on Theoretical Aspects of Computer Science*, 2008.
3. BERGER (R.), « The undecidability of the domino problem », *Memoirs of the American Mathematical Society*, vol. **66**, 1966, p. 1–72.
4. CERVELLE (J.) and DURAND (B.), « Tilings: recursivity and regularity », *Theoretical Computer Science*, vol. **310**, n^{o} 1-3, 2004, p. 469–477.
5. CULIK II (K.) and KARI (J.), « On aperiodic sets of Wang tiles », in *Foundations of Computer Science: Potential - Theory - Cognition*, p. 153–162, 1997.

6. DURAND (B.), « Tilings and quasiperiodicity », *Theoretical Computer Science*, vol. **221**, n^o 1-2, 1999, p. 61–75.
7. DURAND (B.), « De la logique aux pavages », *Theoretical Computer Science*, vol. **281**, n^o 1-2, 2002, p. 311–324.
8. DURAND (B.), LEVIN (L. A.) and SHEN (A.), « Complex tilings », in *Proceedings of the Symposium on Theory of Computing*, p. 732–739, 2001.
9. DURAND (B.), ROMASHCHENKO (A.) and SHEN (A.), « Fixed point and aperiodic tilings », preprint.
10. GUREVICH (Y.) and KORIAKOV (I.), « A remark on Berger's paper on the domino problem », in *Siberian Journal of Mathematics*, **13**:459–463, 1972. (In Russian).
11. HANF (W. P.), « Non-recursive tilings of the plane. I », *Journal of Symbolic Logic*, vol. **39**, n^o 2, 1974, p. 283–285.
12. LAFITTE (G.) and WEISS (M.), « Universal Tilings », in *Proceedings of the Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science n^o **4393**, p. 367–380, 2007.
13. LAFITTE (G.) and WEISS (M.), « A topological study of tilings », to appear in *Proceedings of the conference on Theory and Aspects of Models of Computation* , TAMC'08, 2008.
14. LAFITTE (G.) and WEISS (M.), « Simulation between tilings », submitted to Computability in Europe, CIE'08, 2008.
15. MYERS (D.), « Non-recursive tilings of the plane. II », *Journal of Symbolic Logic*, vol. **39**, n^o 2, 1974, p. 286–294.
16. ROBINSON (R.), « Undecidability and nonperiodicity for tilings of the plane », *Inventiones Mathematicae*, vol. **12**, 1971, p. 177–209.
17. WANG (H.), « Proving theorems by pattern recognition II », *Bell System Technical Journal*, vol. **40**, 1961, p. 1–41.
18. WANG (H.), « Dominoes and the $\forall\exists\forall$-case of the decision problem », in *Proceedings of the Symposium on Mathematical Theory of Automata*, p. 23–55, 1962.