Combinatorial remarks on two-dimensional Languages

Francesca de Carli

Combinatorial remarks on Two-Dimensional Languages

Francesca De Carli

Contents

Introduction

The study of *two-dimensional languages* is a research topic which recently acquainted some interest in different fields of mathematics and computer science.
Since 1967, there has been an attempt to extend results and techniques used on string (one-dimensional) languages to two-dimensional case, defining the two-dimensional languages or *picture languages*. Starting from a finite alphabet Σ, we call *pictures*, or *two-dimensional words* over Σ, the elements of $\Sigma^{**} = \bigcup_{n,m}(\Sigma^n)^m$, where we identify an element $p = p_1 \dots p_m$ of $(\Sigma^n)^m$ with the matrix $p = (q_{ij})$ of size $m \times n$. We call *picture language* over Σ any subset $L \subseteq \Sigma^{**}$.
One of the most interesting and simple class of picture languages, in practice the two-dimensional counterpart of string regular languages, is that of *tiling-system recognizable languages*. This thesis is dedicated to the study of some theoretical, algebraic and combinatorial properties of such languages.
There are several reasons to determine an analogy between tiling-system recognizable languages and regular string languages. As a matter of fact, the various known characterizations of tiling system recognizable languages are very similar to the classic characterizations of regular languages: they are the languages recognized by particular finite state automata, the *on-line tesselation* automata; or the languages coinciding with certain regular expressions; or also definable by existential second order formulas.

The characterization that we will use in the thesis is that one proposed by Giammarresi and Restivo in [33]. It is based on the fact that recognizable string languages can be characterized in terms of local string languages and alphabetic projections. The same idea is applied to the two dimensional languages:

1. it is given the definition of *local languages* over an alphabet Σ and it is added the symbol #. A language $L \subseteq \Sigma^{**}$ is said *local* if there exist a

set Θ of pictures over $\Sigma \cup \{\#\}$ of size 2×2 such that L consists exactly of pictures p such that all the sub-blocks of size 2×2 of $\widehat{p}$ (obtained surrounding p with $\#$) are in Θ. In this case we say $L = L(\Theta)$.

2. A language $L \subseteq \Sigma^{**}$ is said *tiling-system recognizable* if there exists a local language L' over Γ and a projection $\pi : \Gamma \longrightarrow \Sigma$ such that $\pi(L') = L$.

The studies have pointed out also some differences between the classes of regular string languages and tiling system recognizable languages. To begin with, there are some closure properties in one-dimensional case, which do not hold in two-dimensional case, for instance the family of regular string languages is closed under complementation instead the family of tiling system recognizable languages, that we denote by REC, has not this property. Subsequently, there have been some works ([2, 16, 47]) dedicated to the study of the class REC $\cup$ co-REC of recognizable languages or such that their complements are recognizable, and the class UREC of picture languages that admits an *unambiguous* tiling system, [2] (that is the languages of pictures which have a unique counter-image in the local language). In those works it was proved a strict inclusion among the classes REC $\cup$ co-REC, REC and UREC, precisely we have: U-REC $\subset$ REC $\subset$ REC $\cup$ co-REC. Moreover, in [47], Matz describes a technique for showing that a picture language is non-recognizable; in this paper, he gives a necessary condition satisfied by tiling recognizable picture languages and he poses the problem of finding a non-recognizable picture language for which his technique for proving the non-recognizability fails.

In this thesis we investigate the possibility of using two-dimensional languages to recognize particular classes of discrete objects, especially polyominoes. The connection could be interesting since *polyominoes* are simple and important discrete structures that appear in several problems related to theoretical computer science and discrete mathematics; moreover the study of polyominoes has proved a fertile topic of research. We recall that in the Cartesian plane $\mathbb{Z} \times \mathbb{Z}$ a *cell* is a unit square; a *polyomino* is a finite connected union of cells. Polyominoes are firstly studied by Golomb, [36] and divulged by Gardner, [29]. They are related to many problems as enumeration ([7]), tiling ([37, 35]) and discrete tomography.
In order to simplify these problems, several classes of polyominoes have been defined, using the notions of convexity and directed growth.

Since polyominoes can be naturally represented by pictures languages, it is natural to ask if the several classes of polyominoes are tiling recognizable or not.
Reinhardt in [54] proves that the picture language which represents the class of the polyominoes is tiling recognizable. Moreover, as we will see later in details, also the classes of convex, column-convex, parallelogram, and directed-convex polyominoes can be recognized by tiling system recognizable languages [24, 25]. Later, in order to study the borderline between recognizability and non recognizability (by a tiling system) in the theory of picture language, other authors ([60]) face the problem by studying the family of L-convex polyominoes and some closed families strictly related both to the tomographical characterization of L-convexity and to the recognizable family of all polyominoes. They proved that the family of L-convex polyominoes satisfies the necessary condition given by Matz for the recognizability and they conjectured that the family of L-convex polyominoes is non-recognizable.

Another field of research in which two-dimensional languages recently took relevance, is that one of *DNA computation*. The DNA computing using self assembling of DNA oligo-nuceotides is a very active domain and the transfer of concepts from theoretical computer science to nano structure is a challenge in order to construct biomolecular machines [23, 61]. The works in this field have shown how information and algorithms can be encoded in biochemical systems and some efficiency problems deserve interest, for example what kinds of shapes and patterns can be assembled using a small number of tiles, or how quickly they can be assembled. Of late, several works pointed out the connection between particular classes of two-dimensional languages and the *DNA computation*. More precisely, it seems possible to construct with DNA some pictures, or better some two dimensional languages.
In [27], the authors show an algorithm to translate a tiling system in a set of Wang tiles (Wang tiles are a set of squares in which each edge of each tile is colored; matching colored edges are aligned to tile the plane). In [61] it is shown how to construct with DNA an assembling of Wang tiles.
In our thesis, we use these results with the aim to show that the discrete objects, which can be recognized by tiling systems recognizable languages, can be constructed with DNA.

Another aspect we have treated in the thesis is the study of the algebraic structure of local languages. These languages form a lattice with the inclusion

relation. To simplify the subject we divided the problem in easier problems: first we studied the algebraic structure limiting the alphabet of the local languages to one symbol, then we passed to the alphabet of two symbols and finally we extended the results to the general case, i.e. to the local languages over an alphabet of n symbols.

Moreover, we have deepened some classic computability problems which have been solved for string languages, but have not been yet considered for the two-dimensional case. As usual in that field, we care about the decidability/indecidability of some problems regarding local languages and tiling system recognizable languages. For instance, we consider the *emptiness problem*. Such a problem asks to establish if a language is empty; this problem is decidable for regular string languages, but it is not for local languages (and consequently for tiling system recognizable languages). In the thesis we show that this problem is even Σ_1^0-*complete*, and we place in the arithmetical hierarchy the other common problems.

Overview of the thesis

Chapter 1 contains preliminaries on two-dimensional languages, we give a brief review of the main results and the different characterizations of tiling system recognizable languages which play the central role in the thesis. In Chapter 2 we describe the algebraic structure of the families of local languages. We show that this structure is a lattice with respect to the inclusion and we investigate the properties of the lattice. In Chapter 3 we deal with computational problems, studying their decidability. Moreover we give the position, in the arithmetical hierarchy, of the classical problems on string languages now turned to two-dimensional languages. In Chapter 4, after some basic definitions concerning polyominoes, we deal with the recognizability of several classes of polyominoes by tiling system recognizable languages. In particular we give the tiling systems for languages representing some classes of convex polyominoes, as the h-convex or parallelogram. Moreover we investigate the recognizability of L-convex polyominoes. Finally, Chapter 5 is devoted to the application of tiling system recognizable languages to DNA computation. We give the idea about the construction with DNA of some classes of polyominoes (i.e. the class of parallelogram polyominoes), get through to the family of REC.

Chapter 1

Two-dimensional languages

This chapter contains the basic definitions and notations about two-dimensional languages. In particular we give the definitions of the most important classes of two-dimensional languages used in our work (that are the *local languages* and the *tiling system recognizable languages*) and their different characterizations (for all the details see [6, 19, 40, 43, 59, 31, 32, 33]).

We recall that for all set X it is defined $X^* = \bigcup_{n\in\omega} X^n$, where ω is the set of natural numbers. Given an alphabet (that is a finite non empty set) Σ, the elements of Σ^* are *words* (or *strings*) over Σ. If $u \in \Sigma^*$ is a word, then we say *length* of u the minimum n such that $u \in \Sigma^n$. On Σ^* it is possible to define the binary operation concatenation, so that Σ^* with the concatenation is a monoid with identity λ, the string of length 0 or *empty* string. We say that any subset $L \subseteq \Sigma^*$ is a *language* over Σ.

In [33] the authors extend these notions to the bidimensional case. We apply the same notations to the rest of the thesis.

Definition 1 Let Σ be a finite alphabet, a *two-dimensional string*, or *picture*, over Σ, is a rectangular array of elements of Σ.

$$p = \begin{array}{|ccc|}\hline p_{11} & \cdots & p_{1n} \\ \cdot & & \\ \cdot & \cdot & \\ \cdot & & \cdot \\ p_{m1} & \cdots & p_{mn} \\ \hline\end{array}.$$

The set of all the pictures over Σ is denoted by Σ^{**}. A *two-dimensional language* over Σ is a subset of Σ^{**}.

Definition 2 Let $p \in \Sigma^{**}$ we denote by $\hat{p}$ the picture obtained by surrounding p with a special symbol $\sharp$. (That is $\hat{p}$ is a word of $(\Sigma \cup \{\sharp\})^{**}$)

$$\hat{p} = \begin{array}{|cccccc|}\hline \sharp & \sharp & . & . & \sharp & \sharp \\ \sharp & p_{11} & . & . & p_{1n} & \sharp \\ \sharp & . & & & & \sharp \\ \sharp & & . & & & \sharp \\ \sharp & & & . & & \sharp \\ \sharp & p_{m1} & . & . & p_{mn} & \sharp \\ \sharp & \sharp & . & . & \sharp & \sharp \\ \hline \end{array} .$$

Definition 3 Let p be a picture with m rows and n columns, we say p has size (m, n). A *block* (or *subpicture*) of p is a submatrix of p, formally it is a picture p' such that if p' has size (m', n'), then $m' \leq m, n' \leq n$ and there exist $h, k \in \mathbf{N}$ such that

$$k \leq n - n',\ h \leq m - m'$$

and

$$\forall\ 0 \leq i \leq m',\ 0 \leq j \leq n'\ \ p'(i,j) = p(i+h, j+k).$$

Let $p \in \Sigma^{**}$ has size (m, n), $h \leq m$ and $k \leq n$,we denote by $B_{h,k}(p)$ the set of all possible blocks of p that have size (h, k).

Example 1 Given $\Sigma = \{0, 1\}$, consider the following picture p over Σ:

$$p = \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline 1 & 1 & 0 \\ \hline \end{array};$$

and

$$\widehat{p} = \begin{array}{|c|c|c|c|c|}\hline \sharp & \sharp & \sharp & \sharp & \sharp \\ \hline \sharp & 0 & 1 & 0 & \sharp \\ \hline \sharp & 0 & 0 & 1 & \sharp \\ \hline \sharp & 1 & 1 & 0 & \sharp \\ \hline \sharp & \sharp & \sharp & \sharp & \sharp \\ \hline \end{array}.$$

We give some examples of subpictures of p:

$$a = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & 1 \\ \hline 1 & 0 \\ \hline \end{array}, \quad b = \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 1 \\ \hline \end{array}, \quad c = \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline \end{array}$$

where $a \in B_{3,2}(p)$, $b \in B_{2,2}(p)$ and $c \in B_{2,3}(p)$. But

$$d = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

is not a subpicture of p.

Now we can introduce two partial operations between pictures.

Definition 4 The *row concatenation* between p and q ($p \ominus q$) is a partial operation, which is defined if p and q have the same number of columns. In this case we have:

$$p \ominus q = \begin{array}{|c|}\hline p_{11} \cdots p_{1n} \\ \begin{array}{c} \cdot \ \cdot \\ \cdot \quad \cdot \\ \cdot \quad \cdot \end{array} \\ p_{m1} \cdots p_{mn} \\ \hline q_{11} \cdots q_{1n'} \\ \begin{array}{c} \cdot \ \cdot \\ \cdot \quad \cdot \\ \cdot \quad \cdot \end{array} \\ q_{m'1} \cdots q_{m'n'} \\ \hline \end{array}.$$

Analogously we can define the *column concatenation* between p and q if they have the same number of rows (and it is denoted by $p \oplus q$).

We can extend the definitions of concatenation above introduced, as in the unidimensional case, to pictures languages.

Definition 5 Let L_1, L_2 be two-dimensional languages over an alphabet Σ. The *row concatenation between L_1 and L_2* (denoted by $L_1 \ominus L_2$) is defined as follows:

$$L_1 \ominus L_2 = \{p \ominus q \mid p \in L_1 \text{ e } q \in L_2\}.$$

Similarly we define the *column concatenation between L_1 and L_2* (denoted by $L_1 \oplus L_2$).

We can also define the transitive closure of the concatenation operations by iterating the operations (the analogous of the Kleene star operation).

Definition 6 Let L be a two-dimensional language. The *row closure of* L (denoted by $L^{*\ominus}$) is defined as:

$$L^{*\ominus} = \bigcup_{i \geq 0} L^{i\ominus}$$

where $L^{0\ominus} = \lambda$, $L^{1\ominus} = L$, $L^{n\ominus} = L \ominus L^{(n-1)\ominus}$.
Analogously we have the *column closure of* L (denoted by $L^{*\textcircled{\scriptsize 1}}$).

1.1 Local Languages

Now we introduce a class of picture languages, called the local languages.

Definition 7 Let Σ be a finite alphabet, a two-dimensional language $L \subseteq \Sigma^{**}$ is *local* if there exists a finite set Θ, of blocks of size $(2,2)$, which we call a set of *tiles*, over $\Sigma \cup \{\sharp\}$, such that

$$L = \{p \in \Gamma^{**} | B_{2,2}(\hat{p}) \subseteq \Theta\}.$$

In this case we write $L = L(\Theta)$ and we say that Θ is a *rapresentation by tiles* of the local language $L = L(\Theta)$.

We give two examples of local languages which will be recalled later in the thesis.

Example 2 Given $\Sigma = \{a, b, c\}$, consider the following set of tiles Θ over Σ:

$$\Theta = \left\{ \begin{array}{cccc} \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \sharp & a \\ \hline \end{array}, & \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline a & a \\ \hline \end{array}, & \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline a & \sharp \\ \hline \end{array}, & \begin{array}{|c|c|}\hline \sharp & a \\ \hline \sharp & b \\ \hline \end{array}, \\ \begin{array}{|c|c|}\hline \sharp & b \\ \hline \sharp & b \\ \hline \end{array}, & \begin{array}{|c|c|}\hline \sharp & b \\ \hline \sharp & c \\ \hline \end{array}, & \begin{array}{|c|c|}\hline \sharp & c \\ \hline \sharp & \sharp \\ \hline \end{array}, & \begin{array}{|c|c|}\hline a & a \\ \hline b & b \\ \hline \end{array}, \\ \begin{array}{|c|c|}\hline a & \sharp \\ \hline b & \sharp \\ \hline \end{array}, & \begin{array}{|c|c|}\hline c & \sharp \\ \hline \sharp & \sharp \\ \hline \end{array}, & \begin{array}{|c|c|}\hline b & \sharp \\ \hline c & \sharp \\ \hline \end{array}, & \begin{array}{|c|c|}\hline b & b \\ \hline c & c \\ \hline \end{array}, \\ \begin{array}{|c|c|}\hline c & c \\ \hline \sharp & \sharp \\ \hline \end{array} & & & \end{array} \right\}.$$

This set Θ is a representation by tiles of the two-dimensional language of pictures with three rows over Σ, where the first row contains only the symbol a, the second the symbol b and the third the symbol c. An example of picture of $L(\Theta)$ is:

$$p = \begin{array}{|c|c|c|c|} \hline \mathrm{a} & \mathrm{a} & \mathrm{a} & \mathrm{a} \\ \hline \mathrm{b} & \mathrm{b} & \mathrm{b} & \mathrm{b} \\ \hline \mathrm{c} & \mathrm{c} & \mathrm{c} & \mathrm{c} \\ \hline \end{array}$$

Example 3 Given $\Sigma = \{0, 1\}$ and

$$\overline{\Theta} = \left\{ \begin{array}{|c|c|} \hline \sharp & \sharp \\ \hline \sharp & 0 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \sharp & \sharp \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \sharp & \sharp \\ \hline 0 & 1 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \sharp & \sharp \\ \hline 0 & \sharp \\ \hline \end{array}, \begin{array}{|c|c|} \hline \sharp & \sharp \\ \hline \sharp & 1 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \sharp & \sharp \\ \hline 1 & \sharp \\ \hline \end{array}, \begin{array}{|c|c|} \hline \sharp & 0 \\ \hline \sharp & 1 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \sharp & 1 \\ \hline \sharp & 0 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \sharp & 1 \\ \hline \sharp & \sharp \\ \hline \end{array}, \begin{array}{|c|c|} \hline \sharp & 0 \\ \hline \sharp & \sharp \\ \hline \end{array}, \begin{array}{|c|c|} \hline 0 & \sharp \\ \hline 1 & \sharp \\ \hline \end{array}, \begin{array}{|c|c|} \hline 1 & \sharp \\ \hline 0 & \sharp \\ \hline \end{array}, \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array}, \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|} \hline 0 & 1 \\ \hline \sharp & \sharp \\ \hline \end{array}, \begin{array}{|c|c|} \hline 0 & \sharp \\ \hline \sharp & \sharp \\ \hline \end{array}, \begin{array}{|c|c|} \hline 1 & 0 \\ \hline \sharp & \sharp \\ \hline \end{array}, \begin{array}{|c|c|} \hline 1 & \sharp \\ \hline \sharp & \sharp \\ \hline \end{array} \right\}.$$

This set of tiles is a representation by tiles of the two-dimensional language of the pictures with 0 and 1 arranged as the black and white of a chessboard. Consider:

$$\begin{array}{|c|c|c|c|} \hline 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 \\ \hline \end{array}.$$

This picture is in $L(\Theta)$.

We recall that in the unidimensional case the *local string languages* are defined in analogous way, through 1×2 *tiles*. Informally, a (string) language is local if it contains strings such that each substring of length 2 is contained in a certain set of 1×2 tiles. The local (string) languages are a subclass

of regular languages as known for the literature and showed in the following lemma (Lemma 1), for which we give a simple direct proof.
Notation. Let us use the following notations:

- by writing $[a$,we mean the set of two tiles

$$\left\{ \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \sharp & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \sharp & a \\ \hline \sharp & \sharp \\ \hline \end{array} \right\};$$

  the meaning of $a]$ is similar;

- by writing $\overline{\underline{ab}}$, we mean the set of two tiles

$$\left\{ \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline a & b \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & b \\ \hline \sharp & \sharp \\ \hline \end{array} \right\}.$$

Lemma 1 *Each string languages is regular.*

Proof. Let $L = L(\Theta)$. Consider the following nondeterministic finite state automaton with three states, q, q_0, q_1 and state transition function δ:

- q is the initial state; add a transition $\delta(q, i) = q_i$ if and only if $[i \subseteq \Theta$;
- q_i is an accepting state if and only if $i] \subseteq \Theta$
- add a transition $\delta(q_i, j) = q_j$ if and only if $\overline{\underline{ij}} \subseteq \Theta$.

It is easy to see that the automaton M accepts the language L. ■

1.2 Tiling System recognizable languages

In this paragraph we introduce a new class of picture languages, the class of tiling system recognizable languages.

Definition 8 Let $p \in \Sigma^{**}$ be a picture with size (m, n), Γ a finite alphabet and $\pi : \Sigma \to \Gamma$ a surjective function. The *projection of* p by π is the picture $p' \in \Gamma^{**}$ such that $p'(i, j) = \pi(p(i, j))$, for all $1 \leq i \leq m, 1 \leq j \leq n$.

Definition 9 Let $L \subseteq \Sigma^{**}$ be a two-dimensional language, Γ a finite alphabet. The *projection of* L by $\pi : \Sigma \to \Gamma$ is the language

$$L' = \{p' \in \Gamma^{**} | p' = \pi(p) \forall p \in L\} \subseteq \Gamma^{**}.$$

Definition 10 A *tiling system* (briefly TS) is a 4-tuple $T = (\Sigma, \Gamma, \Theta, \pi)$, where Σ, Γ are finite alphabets, Θ is a finite set of tiles over $\Gamma \cup \{\sharp\}$ and $\pi : \Gamma \to \Sigma$ a projection.

Definition 11 We say that a tiling system T *recognizes* a language $L \subseteq \Sigma^{**}$ if there exists a projection $\pi : \Gamma \to \Sigma$ and a finite set of tiles Θ over Γ, where $L' = L(\Theta)$, such that $L = \pi(L')$. In this case we write $L = L(T)$. We say $L(\Theta)$ is an *underlying* language for $L(T)$.
A language $L \subseteq \Sigma^{**}$ is *tiling system ricognizable* (briefly TS-ricognizable) if there exists a tiling system $T = (\Sigma, \Gamma, \Theta, \pi)$ such that $L = L(T)$.

Example 4 We give an example of a tiling system recognizable language. Consider the set of tiles $\overline{\Theta}$ of Example 3 and the projection $\pi : \{0, 1\} \to \{a\}$ that maps the symbols $0, 1$ in a ($\pi(0) = \pi(1) = a$). Then the tiling system $T = \langle \{a\}, \{0, 1\}, \overline{\Theta}, \pi \rangle$ recognizes the language of all square pictures over $\{a\}$.

Theorem 1 *Each local language is a tiling system recognizable language.*

Proof. Let $L(\Theta)$ be a local language. That language is a tiling system recognizable language, it suffices to consider $\Sigma = \Gamma$, $L(\Theta)$ as underlying language, the identity as projection: $\pi : \Sigma \to \Sigma$. So we have $T = (\Sigma, \Gamma, L(\Theta), \pi)$ with $L(\Theta) = L(T)$. Thus the thesis follows. ■

Corollary 1 *The family of local languages is contained strictly into the family of tiling system recognizable languages.*

In order to show the validity of the corollary we give an example of a local language that is not a tiling system recognizable language.

Example 5 We recall the local language $L(\Theta)$ showed in Example 2. Let $\Gamma = \{1\}$, $\pi : \Sigma \to \Gamma$ be the projection given by: $\pi(a) = \pi(b) = \pi(c) = 1$; consider $T = (\Sigma, \Gamma, L(\Theta), \pi)$ then we have that the language of only three rows is tiling system recognizable. Observe that $L(T)$ is not local. To prove this assertion, let us proceed by contradiction, and suppose that there exists a set of tiles Θ' such that $L(T) = L(\Theta')$. Then we can obtain the word

$$p = \begin{array}{|c|c|c|c|} \hline 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 \\ \hline \end{array}$$

which belongs to $L(\Theta')$ and the word

$$p' = \begin{array}{|c|c|c|c|}\hline 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 \\ \hline \end{array}$$

which also belongs to $L(\Theta')$ too: $B_{2,2}(p') = B_{2,2}(p) \subseteq \Theta'$. But p' does not belong to the language $L(T)$, which is a contradiction.

Later in the thesis, we will use the following definition and lemma:

Definition 12 Let Θ be a finite set of tiles, we say Θ is *irredundant* if it contains tiles which are used at least one time in a word of the language $L(\Theta)$.

Lemma 2 Let Θ_1 and Θ_2 be two finite set of tiles, if $\Theta_1 \subseteq \Theta_2$ then $L(\Theta_1) \subseteq L(\Theta_2)$. If Θ_1 and Θ_2 are irredundant then also the viceversa is true.

Proof.

($\Rightarrow$) Let $\Theta_1 \subseteq \Theta_2$, $p \in L(\Theta_1)$ then $B_{2,2}(\widehat{p}) \subseteq \Theta_1 \subseteq \Theta_2$ and so $p \in L(\Theta_2)$.

($\Leftarrow$) Let Θ_1, Θ_2 be irredundant sets and $L(\Theta_1) \subseteq L(\Theta_2)$. If $t \in \Theta_1$ then there exists $p \in L(\Theta_1)$, which belongs also to $L(\Theta_2)$ then $t \in \Theta_2$.

■

Now we give an example of an irredundant set of tiles and of a redundant one.

Example 6 Recall the set Θ of the Example 3, Θ is irredundant. Consider the set Θ' obtained adding to Θ the tile:

$$\begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array}$$

this tile does not match with any other tile in Θ, then it will not used in any word of $L(\Theta')$. Then Θ' is redundant. Observe that the two languages $L(\Theta)$ and $L(\Theta')$ coincide.

We recall that in the unidimensional case the family of languages recognized by finite-state automata coincides with the one defined by means of regular expressions (Kleene's theorem), with the one defined in terms of second order formulas (Buchi's theorem) and with the one generated by particular grammars. In the following subsections we show that there is an equivalent situation for two dimensional case.

1.3 Logic Formulas and Local Languages

In this section we propose a different characterization for the classes of two-dimensional languages just introduced; a characterization through logic formulas. For a start we give some basic definitions and we show how to represent a local language by means of logic formulas.

1.3.1 First Order Sentences

Definition 13 Let Σ be a finite alphabet, and p be a a picture of Σ^{**} of size (m, n). We can represent p as follows:

$$\underline{p} = (dom(p), S_1, S_2, (P_a)_{a\in\Sigma})$$

where

- $dom(p) = \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\}$, is the set of the positions where the elements of p lay;
- S_1 and S_2 are the successor relations for the components of the points belonging to $dom(p)$, that is:

$$(i, j)S_1(i + 1, j) \ and \ (i, j)S_2(i, j + 1) \ for \ 1 \leqslant i < m, 1 \leqslant j < n;$$

- $P_a = \{(i, j)|p(i, j) = a\}$, $a \in \Sigma$ is the set of the points of $dom(p)$ which are labeled with a.

We can muddle symbols of predicates with the interpretation relations, so that the *atomic formulas* are of the kind xS_1y, xS_2y, $P_a(x)$, and are naturally interpreted as follows: xS_1y, xS_2y, $x \in P_a$, respectively with S_2, S_2 (the successor relations just introduced) and P_a. The *formulas* are constructed starting from the atomic ones and using the Boolean connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ and the quantifiers $\exists$ and $\forall$. A *sentence* is a formula without free variables.
If $\varphi(x_1, x_2, \ldots, x_n)$ is a formula with at maximum $x_1, x_2, \ldots, x_n$ free variables, $p \in \Sigma^{**}$ is a picture and $q_2, q_2 \ldots, q_n$ are elements of $dom(p)$, then we write

$$(\underline{p}, q_1, \ldots, q_n) \models \varphi(x_1, \ldots, x_n)$$

if p satisfies φ with the natural interpretation above, where x_i is interpreted by q_i. If φ is a sentence, we write $\underline{p} \models \varphi$. The language $L(\varphi)$ *defined by a sentence* φ is the set of all pictures $p \in \Sigma^{**}$ such that $\underline{p} \models \varphi$.

Definition 14 Let L a two-dimensional language, L is *first order definible* if there exists a sentence φ which contains only first order quantifiers such that $L = L(\varphi)$.

Now we can describe particular positions (that is particular elements of $dom(p)$) in a picture by first order formulas,. A position into the higher row, $x = (1, j)$ con $1 \leqslant j \leqslant n$, is described by:

$$\varphi_t(x) = \neg\exists y\ yS_1x.$$

Analogously we can define the positions into the lower row by

$$\varphi_b(x) = \neg\exists y\ xS_1y.$$

$$\varphi_r(x) = \neg\exists y\ xS_2y$$

defines the positions into the rightmost column and

$$\varphi_l(x) = \neg\exists y\ yS_2x$$

the positions into the leftmost column. Now it is clear the meaning of φ_{tr}, φ_{tl}, φ_{br}, φ_{bl}. For example:

$$\varphi_{tr} = \varphi_t \wedge \varphi_r.$$

We give some examples of first order sentences that define local linguages.

Example 7 Let $L(\Theta)$ be the local language given by the set of tiles over the alphabet $\Sigma = \{0, 1\}$:

$$\Theta = \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline 1 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & \# \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & 1 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & 0 \\ \hline \end{array}, \right.$$

$$\left. \begin{array}{|c|c|}\hline \# & \# \\ \hline 0 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & 0 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & \# \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 0 & 1 \\ \hline \end{array}, \right.$$

$$\left. \begin{array}{|c|c|}\hline 0 & 1 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 0 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 0 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 1 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 0 & 0 \\ \hline \end{array}, \right.$$

$$\left. \begin{array}{|c|c|}\hline \# & \# \\ \hline 1 & 1 \\ \hline \end{array} \right\}.$$

The pictures $p \in L(\Theta)$, will be of the kind:

#	#	#	#	#	#	#	#
#	1	1	0	0	1	0	#
#	#	#	#	#	#	#	#

.

It is easy to see that we have exactly the language of only one row

$$L(\Theta) = \{p \text{ of only one row and } p \in \{0,1\}^*\}.$$

The first order sentence $\varphi = \forall x \varphi_t(x)$ defines $L(\Theta)$. By this sentence we say that all the position in each picture that satisfies φ, are that ones of the first row, consequently there is only one row, thus $L(\Theta)$ is:

$$L(\Theta) = L(\varphi) = \{p \in \Sigma^{**} | \underline{p} \models \varphi\}.$$

Analogously we could use φ_b instead of φ_t. Moreover we can use φ_r or φ_l to mean the languages of only one column.

Remark 1 *The language of the square pictures over Σ of only one symbol is not local, moreover we can not describe this language by a first order sentence; see [33].*

1.3.2 Logic Formulas for Local Languages

In this subsection we will determine first order sentences in order to define local languages. The same result is obtained in [32], using a different argumentation. Moreover we will show that it will be sufficient to use only universal quantifiers.
Consider

$$B(u,v,x,y) := \; xS_1u \wedge yS_1v \wedge vS_2u \wedge yS_2x.$$

Intuitively four positions into a picture satisfy $B(u,v,x,y)$ if and only if they constitute a block of size $(2,2)$.
Formally:

Theorem 2 *Let $p \in \Sigma^{**}$, then $(\underline{p}, q_1, q_2, q_3, q_4) \models B(u,v,x,y)$ if and only if (q_1, q_2, q_3, q_4) constitute a tile.*

Proof. $(\underline{p}, q_1, q_2, q_3, q_4)$ satisfies $B(u, v, x, y)$ if and only if

$$q_3 S_1 q_1 \wedge q_4 S_1 q_2 \wedge q_2 S_2 q_1 \wedge q_4 S_2 q_3;$$

this is true if and only if (q_1, q_2, q_3, q_4) constitutes a tile:

q_1	q_2
q_3	q_4

■

Theorem 3 *Let Θ be a finite set of tiles, and t a tile:*

$$t = \begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline \end{array}.$$

Consider the formula:

$$t_{a,b,c,d}(u, v, x, y) := P_a(u) \wedge P_b(v) \wedge P_c(x) \wedge P_d(y),$$

and the sentence:

$$\varphi = \forall u \forall v \forall x \forall y [B(u, v, x, y) \rightarrow \bigvee_{t \in \Theta} t_{a,b,c,d}(u, v, x, y)].$$

Then $L(\Theta) = L(\varphi)$.

Proof. $p \in L(\Theta)$ if and only if all tiles which belong to $\widehat{p}$ are in Θ, that is if and only if all tiles in $\widehat{p}$ satisfy the sentence φ, if and only if $\underline{p} \models \varphi$ this holds if and only if $p \in L(\varphi)$. ■

Example 8 We show the first order sentence which defines the language $L(\Theta)$ over $\Sigma = \{0, 1\}$, of square words which have 1 in the main diagonal and 0 in the other positions.
First of all the set of tiles Θ for the language $L(\Theta)$ is the following:

$$\Theta = \left\{ \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \sharp & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline 0 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline 0 & \sharp \\ \hline \end{array}, \right.$$

$$\begin{array}{|c|c|}\hline \sharp & 1 \\ \hline \sharp & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \sharp & 0 \\ \hline \sharp & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \sharp & 0 \\ \hline \sharp & \sharp \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 0 \\ \hline \sharp & \sharp \\ \hline \end{array},$$

$$\begin{array}{|c|c|}\hline 0 & 1 \\ \hline \sharp & \sharp \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & \sharp \\ \hline \sharp & \sharp \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & \sharp \\ \hline 0 & \sharp \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & \sharp \\ \hline 1 & \sharp \\ \hline \end{array},$$

$$\left. \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 0 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array} \right\}$$

Then the first order sentence for $L(\Theta)$ is:

$$\begin{aligned}\forall u\forall v\forall x\forall y[B(u,v,x,y) \rightarrow (&t_{\sharp,\sharp,\sharp,1}(u,v,x,y) \vee t_{\sharp,\sharp,1,0}(u,v,x,y)\\ &\vee t_{\sharp,\sharp,0,0}(u,v,x,y) \vee t_{\sharp,\sharp,0,\sharp}(u,v,x,y) \vee t_{\sharp,1,\sharp,0}(u,v,x,y)\\ &\vee t_{\sharp,0,\sharp,0}(u,v,x,y) \vee t_{\sharp,0,\sharp,\sharp}(u,v,x,y) \vee t_{0,0,\sharp,\sharp}(u,v,x,y)\\ &\vee t_{0,1,\sharp,\sharp}(u,v,x,y) \vee t_{1,\sharp,\sharp,\sharp}(u,v,x,y) \vee t_{0,\sharp,0,\sharp}(u,v,x,y)\\ &\vee t_{0,\sharp,1,\sharp}(u,v,x,y) \vee t_{1,0,1,0}(u,v,x,y) \vee t_{0,0,1,0}(u,v,x,y)\\ &\vee t_{0,1,0,0}(u,v,x,y) \vee t_{0,0,0,0}(u,v,x,y)]\end{aligned}$$

We found a simply way to define all the local languages: starting from a set of tiles, we found a first order sentence of only four universal quantifiers which allows us to axiomatize the local language generated.

1.3.3 Existential Second Order Formulas

In the previous subsection we observed that is possible identify a picture $p \in \Sigma^{**}$ with the structure

$$\underline{p} = (dom(p), S_1, S_2, (P_a)_{a\in\Sigma}).$$

Moreover the properties of the two-dimensional pictures can be described by first order logic formulas and existential second order formulas using first order variables $x, y, z, x_1, x_2, \ldots$, for the elements of $dom(p)$, that is for positions, and second order variables $X, Y, Z, X_1, X_2, \ldots$, for sets of positions. In other words, variables are interpreted as subsets of $dom(p)$.

Definition 15 A two-dimensional languages L is *second order definable*, if there exists a second order sentence φ, such that $L = L(\varphi)$.

Definition 16 A two-dimensional languages L is *existential second order definable*, if there exists a sentence of the kind:

$$\varphi = \exists X_1 \exists X_2 \ldots \exists X_n \psi(X_1 X_2 \ldots X_n)$$

such that $L = L(\varphi)$, where ψ contains only first order quantifiers.

Example 9 Now we verify that the language of square words over a given alphabet is existential second order definable. It suffices to describe a set of positions which (1) contains the left-upper corner; (2) is closed for diagonal

successor (that is, we can move from the position (i, j) to the position $(i + 1, j + 1)$); and (3) we don't reach the lower border or the righter border, except the right-lower corner. An existential second order which defines the three conditions is:

$$\exists X(\exists x(\varphi_{tl}(x) \wedge X(x)) \wedge \forall x \forall y \forall z((X(x) \wedge xS_1y \wedge yS_2z) \rightarrow X(z)) \wedge \forall x((\varphi_b(x) \vee \varphi_r(x)) \rightarrow (\neg X(x) \vee \varphi_{br}(x)).$$

1.4 Regular Expressions

The basic operations between two-dimensional words that we have defined in the first paragraph can be used to obtain larger family of two-dimensional languages starting from elementary languages. Using those operations we are then able to give a new characterization for local languages and tiling system recognizable languages.

Definition 17 Given an alphabet Σ, we say that the empty language $\emptyset$, and each language $\{\boxed{a}\}$, where $a \in \Sigma$, are *atomic languages* over Σ.

Remark 2 *We denote by R the following set of operations:*

$$R = \{\ominus, \oplus, *\ominus, *\oplus, \cup, \cap, ^C\}.$$

The elements of R are called regular operations.

Definition 18 A language over Σ is *regular* if it is obtained from atomic languages by applying a finite sequences of regular operations.

Informally, a regular expression is a formula which specifies how a certain language is obtained from atomic languages by regular operations.

Definition 19 A *regular expression RE* over an alphabet Σ is recursively defined as:

- $\emptyset$ and all symbols $a \in \Sigma$ are regular expressions;
- if α, β are regular expressions, then

- $(\alpha) \cup (\beta)$
- $(\alpha) \cap (\beta)$
- ${}^{C}(\alpha)$
- $(\alpha) \ominus (\beta)$
- $(\alpha) \oplus (\beta)$
- $(\alpha)^{*\ominus}$
- $(\alpha)^{*\oplus}$

are regular expressions.

Each regular expression over Σ denotes a two-dimensional language. Using standard notations:

- $\emptyset$ denotes the empty language,
- a denotes the language with the only word $\{a\}$,
- $(\alpha) \cup (\beta)$ denotes the union of the languages which are denoted by α and β,
- $(\alpha) \cup (\beta)$ denotes their intersection,
- $(\alpha) \ominus (\beta)$ and $(\alpha) \oplus (\beta)$ denote their row and column concatenation respectively,
- $(\alpha)^{*\ominus}, (\alpha^{*\oplus})$ denote the row and column closure of the language denoted by α respectively,
- finally $(\alpha)^{C}$ denotes its complement.

Definition 20 A two-dimensional language $L \subseteq \Sigma^{**}$ is *regular* if it can be represented by a regular expression over Σ.

Example 10 Consider $\Sigma = \{0, 1\}$, the regular expression:

$$E = (((0 \ominus 1)^{*\ominus}) \oplus ((1 \ominus 0)^{*\ominus}))^{*\oplus}$$

denotes the language of the chess words with an odd number of rows. For example the following word belong sto E:

0	1	0	1	0	1
1	0	1	0	1	0
0	1	0	1	0	1
1	0	1	0	1	0

.

1.5 On-line Tesselation Automata

In literature there have been depicted different kinds of automata to recognize two-dimensional languages [43, 41, 42]. In this section we describe a particular model of cellular automata introduced in [6] which recognizes the tiling system recognizable languages. Informally an *on-line tesselation automata* is a finite sequence of cells, or better, an array where a "wave of transition" passes in diagonal over the array. Each cell changes its state depending on the state of the two neighbor cells, the upper and the leftmost, respectively. Formally:

Definition 21 A two-dimensional non deterministic (deterministic) *on-line tesselation automata* denoted by 2OTA (2DOTA), is defined by $A = (\Sigma, Q, I, F, \delta)$ where:

- Σ is the alphabet in input;
- Q is a finite set of states;
- $I \subseteq Q$ ($I = \{i\} \subseteq Q$) is the set of initial states;
- $F \subseteq Q$ is the set of final states (or accepting states);
- $\delta : Q \times Q \times \Sigma \to 2^Q$ ($\delta : Q \times Q \times \Sigma \to Q$) is the transition function

(see also [33]).

The automata A runs over a picture $p \in \Sigma^{**}$ associating a state (in Q) to each position (i, j) in p. This state is given by the transition function δ and it depends to the other states already associated with the positions $(i-1, j)$ and $(i, j-1)$ and to the symbol $p(i, j)$.
At the zero step, an initial state q_0 is associated with each position in the first row and column of $\widehat{p}$. The computation consists of $l_1(p) + l_2(p) - 1$ steps (where $l_1(p), l_2(p)$ mean the number of rows and column of p respectively). At the following step the automata reads $p(1, 1)$ and the state $\delta(q_0, q_0, p(1, 1))$ is associated with the position $(1, 1)$. At the second step the states are simultaneously associated to the positions $(1, 2)$ e $(2, 1)$, and so on, the automata passes to the next diagonal. At the k step, the states are simultaneously associated with the position (i, j) such that $i + j - 1 = k$.

Definition 22 A 2OTA A *recognizes* a picture $p \in \Sigma^{**}$ if there exists a computation of A over p such that the state associated with the position $(l_1(p), l_2(p))$ is a final state.

We notice that a 2OTA is reduced to a standard automata over strings when we restrict the computation to pictures of one row.
To summarize the results depicted in the previous subsections we have the following theorem of characterization:

Theorem 4 *For a language L the following are equivalent:*

1. *L is tiling system recognizable;*
2. *L can be obtained from atomic languages by applying a finite number of times the regular operations (excluding the complementation) and the projection;*
3. *$L = L(\varphi)$ for an existential second order formula;*
4. *L is recognized by a 2OTA.*

Proof. *See [33, Theorem 8.7].* ∎

1.5.1 Grammars

In literature different systems for generating pictures using grammars have been presented ([46, 48, 49]), with the attempt of generalize the grammars of the unidimensional case for formal string languages. Among the most recently defined grammars, we recall the *tile rewriting grammar* of ([21]). Informally, in tile rewriting grammars, a *rewriting rule* changes a homogeneous rectangular picture into an isometric one, tiled with specified tiles. *Derivation* and *language generation* with tiling rewriting grammar rules are similar to context-free grammars. Anyway there is not a grammar that exactly generalizes the "unidimensional" context free grammars. The model, proposed by Crespi Reghizzi et al., has greater generative capacity than the tiling systems, at yet any models of grammars have the same generative power as tiling systems. Thus, differently to regular languages in one dimension, there is not a characterization for tiling system recognizable languages.

Chapter 2

Local languages and algebraic structures

In this chapter we study the algebraic structure of local languages and tiling system recognizable languages. We observe that it is possible to define, over these two families of languages an order relation (and consequently the two operations meet and join) so that we have a lattice of languages. Our principal purpose is to study the main features of this lattice in particular the distributivity, the modularity etc. The problem appears rather complex, so we start to study the simplest case, i.e. we consider the lattice of the local languages defined over an alphabet of only one symbol (Loc_1). Easily we observe that in this case the lattice is distributive. Then we consider the lattice of local languages defined over an alphabet of two symbols (Loc_2) and we extend the results to the languages over alphabets of more than two symbols. We also consider the characterization of atoms and co-atoms of the lattice and the join-irreducible elements. Finally we restrict the research to the unidimensional case, i.e. we examine the lattice of the horizontal dominoes which is a proper sub-lattice of Loc_2. In this case we give a complete characterization of atoms and co-atoms.

Notation We will use two double brackets $\|p\|$ to denote the set of tiles we can extract from the picture p within brackets (of course, surrounded by $\sharp$). For instance if for $a, b, c, d \in \Sigma$ we write

$$\left\| \begin{array}{cc} a & b \\ c & b \end{array} \right\|$$

then the corresponding set of tiles is

$$\Theta = \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline b & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline b & \# \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & c \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & a \\ \hline \# & c \\ \hline \end{array}, \begin{array}{|c|c|}\hline b & \# \\ \hline b & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline a & b \\ \hline \end{array}, \begin{array}{|c|c|}\hline c & b \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & b \\ \hline c & b \\ \hline \end{array} \right\}.$$

We also often use the notation $L(p) = L(\|p\|)$.

Example 11 *An example of local language is given by the set of "chessboard" pictures (i.e. pictures with alternating* 0*'s and* 1*'s in each row and column) over the alphabet* $\Sigma = \{0, 1\}$*, with* 0 *on each corner, henceforth named* L_s*. Consider the following set of tiles* Θ_s*:*

$$\Theta_s = \left\| \begin{array}{ccc} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{array} \right\|.$$

It is easy to see that $L_s = L(\Theta_s)$*.*

We now define a lattice over the set of local languages on an alphabet of fixed size. Therefore we also need to recall some definitions and theorems from lattice theory. For more details on lattice theory we refer the reader to [1] and [57].

Definition 23 *A* lattice *is a partially ordered set* $\mathcal{L} = \langle L, \leq \rangle$ *such that every pair of elements* $x, y \in P$ *has* greatest lower bound *(or* inf*, or* meet*) denoted by* $x \wedge y$*, and* lowest upper bound *(or* sup*, or* join*) denoted by* $x \vee y$*.*

It is well known that a lattice on a universe L can be equivalently given either by specifying its partial order relation $\leq$, or by specifying its binary operations $\wedge$ and $\vee$. Indeed, for every pair of elements $a, b \in L$, one has

$$a \leq b \Leftrightarrow a = a \wedge b \Leftrightarrow b = a \vee b.$$

It is therefore just a matter of convenience whether one sees a lattice as an ordered structure, or an algebraic structure, or both.

Definition 24 *In a lattice $\mathcal{L} = \langle L, \wedge, \vee \rangle$, an element $b \in L$ is* meet-irreducible *if $b = x \wedge y$ implies $b = x$ or $b = y$ for all $x, y \in L$. Dually, an element $b \in L$ is* join-irreducible *if $b = x \vee y$ implies $b = x$ or $b = y$ for all $x, y \in L$.*

Definition 25 *In a lattice $\mathcal{L} = \langle L, \leq \rangle$, an element $b \in L$ is a* (minimal) cover *of a, if $a < b$ and there is no c such that $a < c < b$. We write $a \prec b$ to denote that b is a cover of a. If $\mathcal{L}$ has least element 0, then an element $a \in L$ is said to be an* atom *if $0 \prec a$. Dually, in a lattice with greatest element 1, a* coatom *is an element a such that $a \prec 1$.*

Definition 26 *Let $\mathcal{L} = \langle L, \wedge, \vee, \leq \rangle$ be a lattice. Then*

1. *$\mathcal{L}$ is* distributive *if for all $a, b, c \in L$, it satisfies:*

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c);$$

2. *$\mathcal{L}$ is* modular *if for all $a, b, c \in L$ it satisfies:*

$$a \leq c \Rightarrow a \vee (b \wedge c) = (a \vee b) \wedge c;$$

3. *$\mathcal{L}$ is* semimodular *if for all $a, b \in L$, it satisfies:*

$$a \wedge b \prec a \Rightarrow b \prec a \vee b.$$

Theorem 5 *Let $\mathcal{L}$ be a lattice. Then*

- *$\mathcal{L}$ is nondistributive if and only if $\mathcal{L}$ has a sublattice isomorphic to $\mathfrak{N}_5$ or $\mathfrak{M}_5$ (see Figure 2.1);*
- *$\mathfrak{L}$ is nonmodular if and only if $\mathcal{L}$ has a sublattice isomorphic to $\mathfrak{N}_5$.*

It is known that a modular lattice is also semimodular, see e.g. [1]. Thus the following strict implications hold:

$$\text{distributive} \;\Rightarrow\; \text{modular} \;\Rightarrow\; \text{semimodular}.$$

Definition 27 *A partially ordered set $\langle P, \leq \rangle$ satisfies the* Jordan-Dedekind condition *if it has only finite chains and, for all pair of elements $x, y \in P$, such that $x \leq y$, all maximal chains between x and y have the same length.*

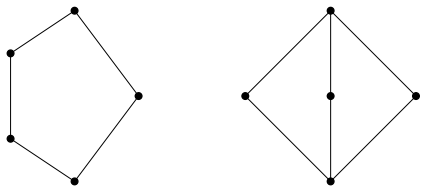


Figure 2.1: The two nondistributive lattices $\mathfrak{N}_5$ and $\mathfrak{M}_5$, rispettivamente.

The following result is almost immediate, see for instance [1, Proposition 2.1].

Theorem 6 *Let $\langle P, \leq \rangle$ be a partially ordered set with only finite chains and with least element 0. P satisfies the Jordan-Dedekind condition if and only if one can define a function (called the* rank function*) $\rho : P \to \mathbb{N}$ (mapping each element $x \in P$ into the* rank *of x), such that*

- *$\rho(0) = 0$;*
- *if y covers x then $\rho(y) = \rho(x) + 1$.*

It is known that a semimodular lattice without infinite chains satisfies the Jordan-Dedekind condition, see for instance [1, Lemma 2.26]. Thus, if a lattice without infinite chains is semimodular then all maximal chains between the same two elements have the same length.

2.1 The lattice $\mathcal{L}\mathrm{oc}_n$

Without loss of generality, for every $n \geq 1$, we may fix the alphabet $\Sigma = \{0, 1, ..., n-1\}$ and confine our investigation to the poset $\mathcal{L}\mathrm{oc}_n^* = \langle Loc_n^*, \subseteq \rangle$ of local languages on Σ.

Definition 28 *If Θ is a set of tiles such that for each $t \in \Theta$ there exists a picture $p \in L(\Theta)$ such that $t \in B_{2,2}(\hat{p})$, then we say that Θ is an* irredundant *set of tiles. A set of tiles is* redundant *if it is not irredundant.*

For instance $\| p \|$ is irredundant, for every picture p.
Unless otherwise specified, from this moment on when we are given a set of tiles Θ we will always assume that Θ is irredundant. The problem of recognizing whether a set of tiles is irredundant is in general undecidable, [**?**].

Example 12 *The following set Θ of tiles is redundant, since the last tile in the list can not be combined with any other tile in Θ.*

$$\Theta = \left\{ \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \sharp & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline 0 & \sharp \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & \sharp \\ \hline \sharp & \sharp \\ \hline \end{array}, \begin{array}{|c|c|}\hline \sharp & 1 \\ \hline \sharp & \sharp \\ \hline \end{array}, \begin{array}{|c|c|}\hline \sharp & 0 \\ \hline \sharp & 1 \\ \hline \end{array}, \right.$$
$$\left. \begin{array}{|c|c|}\hline 0 & \sharp \\ \hline 1 & \sharp \\ \hline \end{array}, \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline 0 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 1 \\ \hline \sharp & \sharp \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array} \right\}.$$

However the set $\Theta \setminus \left\{ \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array} \right\}$ *is irredundant.*

Theorem 7 *For every $n \geq 1$, $\mathcal{L}\mathrm{oc}_n^* = \langle Loc_n^*, \subseteq \rangle$ is a finite lattice with operations of meet and join given by, respectively,*

$$L(\Theta_1) \wedge L(\Theta_2) = L(\Theta_1) \cap L(\Theta_2) (= L(\Theta_1 \cap \Theta_2))$$
$$L(\Theta_1) \vee L(\Theta_2) = L(\Theta_1 \cup \Theta_2).$$

(We remark once more that Θ_1 and Θ_2 are assumed to be irredundant.)

Proof. It is a simple calculation to check that

$$L(\Theta_1) \cap L(\Theta_2) = L(\Theta_1 \cap \Theta_2).$$

To show the statement about $\vee$, first notice that $L(\Theta_1 \cup \Theta_2)$ is an upper bound of $L(\Theta_1)$ and $L(\Theta_2)$. On the other hand, suppose that $L(\Theta_1), L(\Theta_2) \subseteq L(\Theta)$, and let $t \in \Theta_1 \cup \Theta_2$. Whether $t \in \Theta_1$ or $t \in \Theta_2$, by irredundancy, we have that t is a 2×2 subtile of a picture v in $L(\Theta_1)$ or in $L(\Theta_2)$; hence $v \in L(\Theta)$, and thus $t \in \Theta$. This shows that $\Theta_1 \cup \Theta_2 \subseteq \Theta$, hence $L(\Theta_1 \cup \Theta_2) \subseteq L(\Theta)$. ■

Remark 3 *If Θ_1, Θ_2 are sets of tiles then $L(\Theta_1) \vee L(\Theta_2) \supseteq L(\Theta_1) \cup L(\Theta_2)$, but equality need not hold, as shown by the following counterexample. Consider the following two sets of tiles:*

$$\Theta_1 = \left\| \begin{array}{cc} 0 & 1 \end{array} \right\|, \qquad \Theta_2 = \left\| \begin{array}{cc} 1 & 0 \end{array} \right\|.$$

It is clear that the picture $p = \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline \end{array}$ *belongs to $L(\Theta_1 \cup \Theta_2)$ but not to $L(\Theta_1) \cup L(\Theta_2)$.*

Remark 4 *In the rest of the paper we will only consider sets of tiles which do not contain the tile* $\boxed{\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \sharp & \sharp \\ \hline \end{array}}$, *hence restricting ourselves to local languages that do not contain the empty picture* $\boxed{\ }$. *We use the symbol* Θ_{Tot} *to denote the set of all possible tiles over* $\{0,1\}$, *minus the tile* $\boxed{\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \sharp & \sharp \\ \hline \end{array}}$. *Notice that* $\Theta_{Tot} = \left\{ \|p\| : p \in \{0,1\} - \{\boxed{\ }\} \right\}$. *Let* $\mathcal{L}\mathrm{oc}_n$ *denote the local languages on* $\Sigma = \{0, \ldots, n-1\}$ *that do not contain the empty picture, i.e.* $\mathcal{L}\mathrm{oc}_n = L(\Theta_{Tot})$. *Clearly* $\mathcal{L}\mathrm{oc}_n = \langle Loc_n, \wedge, \vee, \subseteq \rangle$ *is a sublattice of* $\mathcal{L}\mathrm{oc}_n^*$. *Since we are interested in the lattice theoretic structure of local languages, restriction to* $\mathcal{L}\mathrm{oc}_n$ *amounts to no loss of generality, by the following theorem:*

Theorem 8 $\mathcal{L}\mathrm{oc}_n^* \simeq \mathcal{L}\mathrm{oc}_n \times \mathbf{2}$, *where* $\mathbf{2}$ *is the two-element bounded lattice, and the symbol* $\simeq$ *denotes lattice theoretic isomorphism.*

Finally we point out that, for any $m < n$, the lattice $\mathcal{L}\mathrm{oc}_m$ can be viewed (under inclusion) as an ideal of $\mathcal{L}\mathrm{oc}_n$.

2.1.1 The simplest case: the lattice $\mathcal{L}\mathrm{oc}_1$

We exemplify the notions so far introduced, by facing the simple case of the lattice $\mathcal{L}\mathrm{oc}_1$ relative to the class of local languages restricted to the alphabet Σ of only one symbol, say $\Sigma = \{0\}$.
The local languages we can construct over the alphabet $\{0\}$ are only six, including the empty language: See Figure 2.2. More precisely, the five nonempty languages are:

- $L_0 = L(\,\|0\|\,)$: the language constituted by the only 1×1 picture $\boxed{0}$;
- $L_{r0} = L(\,\left\|\ 0 \quad 0\ \right\|\,)$: the language constituted by the pictures of only one row of 0's;
- $L_{c0} = L(\,\left\|\begin{array}{c} 0 \\ 0 \end{array}\right\|\,)$: the language constituted by the pictures of only one column of 0's;
- $L_{r0} \vee L_{c0}$the language that is the join of the two previous languages, i.e. the language consisting only of rows or columns of 0's;
- 0^{**}: the language constituted by all possible pictures of 0's.

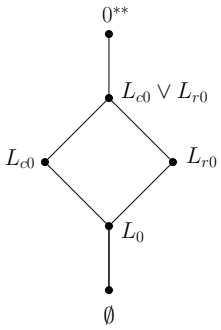


Figure 2.2: The lattice $\mathcal{L}\mathrm{oc}_1$.

We observe that the lattice $\mathcal{L}\mathrm{oc}_1$ is isomorphic to the free distributive lattice with $0, 1$ on two generators.

2.1.2 The general case

We have seen that $\mathcal{L}\mathrm{oc}_1$ is distributive. The situation changes radically, if one considers lattices of local languages on alphabets with at least two symbols.

Theorem 9 *If $n \geq 2$ then $\mathcal{L}\mathrm{oc}_n$ contains the following sublattices:*

(i) a sublattice isomorphic to $\mathfrak{N}_5$;

(ii) a sublattice isomorphic to $\mathfrak{M}_5$.

Hence $\mathcal{L}\mathrm{oc}_n$ is not semimodular,

Proof. The proof is for $\mathcal{L}\mathrm{oc}_2$, and can be extended to $\mathcal{L}\mathrm{oc}_n$ since $\mathcal{L}\mathrm{oc}_2$ is a sublattice of $\mathcal{L}\mathrm{oc}_n$.

(i) Consider the following languages:

$$L_0 = L(\|0\|) = \left\{\boxed{0}\right\}, \quad L_{01} = \left\{\boxed{0}\boxed{1}\right\}, \quad L_{10} = \left\{\boxed{1}\boxed{0}\right\}.$$

It is easy to see that L_{01} and L_{10} are atoms. Obviously L_{01} is a cover of $\emptyset = L_{01} \wedge L_{10}$ but $L_{01} \vee L_{10}$ is not a cover of L_{10}. For example the language $L_0 \vee L_{10}$ is such that $L_{10} \subset L_0 \vee L_{10} \subset L_{01} \vee L_{10}$, where inclusions are strict. See also Figure 2.3 (a).

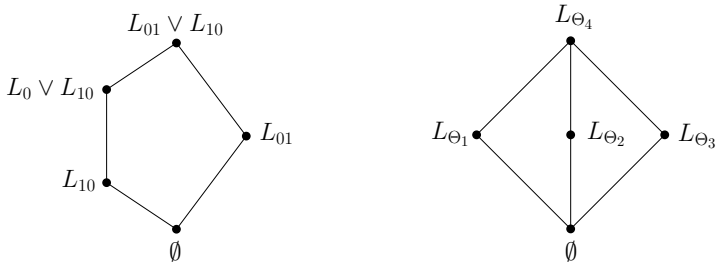


Figure 2.3: (a) Sublattice isomorphic to $\mathfrak{N}_5$; (b) Sublattice isomorphic to $\mathfrak{M}_5$.

(ii) Consider the sets of tiles:

$$\Theta = \Theta_{Tot} \setminus \left\{ \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array} \right\}; \qquad \Theta_1 = \left\| \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{array} \right\|; \qquad \Theta_2 = \left\| \begin{array}{cccc} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{array} \right\|;$$

and the local language $L(\Theta) \wedge (L(\Theta_1) \vee L(\Theta_2)) = L(\Theta_3)$.
Direct inspection shows:

- $L(\Theta_1) \wedge L(\Theta_2) = L(\Theta_1) \wedge L(\Theta_3) = L(\Theta_2) \wedge L(\Theta_3) = \emptyset$.
- $L(\Theta_1) \vee L(\Theta_2) = L(\Theta_1) \vee L(\Theta_3) = L(\Theta_2) \vee L(\Theta_3)$.

Let Θ_4 be such that $L(\Theta_4) = L(\Theta_1) \vee L(\Theta_2)$. Then $\mathcal{L}\mathrm{oc}_2$ contains the sublattice depicted in Figure 2.3 (b). ■

Notice that the embeddings of $\mathfrak{N}_5$ and $\mathfrak{M}_5$ given above, preserve 0 as well.

2.1.3 Meet-irreducible elements and coatoms in $\mathcal{L}\mathrm{oc}_n$

We now consider the set of *meet-irreducible elements* of the lattice $\mathcal{L}\mathrm{oc}_n$. In $\mathcal{L}\mathrm{oc}_1$ all elements except for L_0 are meet-irreducible, as can be easily seen from Figure 2.2. So in the rest of this section we confine ourselves to the case $n \geq 2$.

Notation. Extending the definition given in Remark 4, let

$$\Theta^n_{Tot} = \left\{ \|p\| \, : \, p \in \{0, \ldots, n-1\}^{**} - \{ \square \} \right\}.$$

We will usually write Θ_{Tot} instead of Θ^n_{Tot}, when the alphabet is clearly understood from the context.

We have:

Theorem 10 *Let $n \geq 2$ and $L = L(\Theta) \in \mathcal{L}\mathrm{oc}_n$. The following are equivalent:*

(1) L is a coatom;

(2) $\Theta = \Theta_{Tot} \setminus \{ t \}$, with $t \in \Theta_{Tot}$;

(3) L is meet-irreducible.

Proof. Let us start proving that (1) and (2) are equivalent. If $\Theta = \Theta_{Tot} \setminus \{ t \}$ then clearly Θ is irredundant, and $L(\Theta)$ is a coatom. The converse is trivial.

Clearly, (1) implies (3). So we only have to prove that (3) implies (1). Suppose now that $L = L(\Theta)$ with at least two different tiles $t_1, t_2 \in \Theta_{Tot} \setminus \Theta$. Let us prove the following useful statement:

Claim There are two pictures $u(t_1)$ and $u(t_2)$ such that $t_1 \in \|u(t_1)\|$, $t_2 \in \|u(t_2)\|$, and:

$$\|u(t_1)\| \cap \|u(t_2)\| = \emptyset.$$

Once Claim has been proved we have that:

$$\begin{aligned} L \subset L(\Theta) \vee L(u(t_1)) &= L(\Theta \cup \|u(t_1)\|) \subset L(\Theta_{Tot}), \\ L \subset L(\Theta) \vee L(u(t_2)) &= L(\Theta \cup \|u(t_2)\|) \subset L(\Theta_{Tot}). \end{aligned}$$

Notice that all the inclusions are strict. Moreover, since $\|u(t_1)\|$ and $\|u(t_2)\|$ are disjoint sets, it follows that:

$$(L(\Theta) \vee L(u(t_1)) \wedge (L(\Theta) \vee L(u(t_2))) = L(\Theta),$$

which concludes our proof, since this shows that $L(\Theta)$ is meet-reducible.

The proof of Claim is indeed a mere (long) exercise, based on considering all possible cases of the two tiles t_1 and t_2. Let d_1 (resp. d_2) be the number of occurrences of the symbol $\sharp$ in t_1 (resp. t_2). Without loss of generality we examine the six cases where $d_1 \geq d_2$:

1. $d_1 = 0$, $d_2 = 0$;
2. $d_1 = 2$, $d_2 = 0, 2$;
3. $d_1 = 3$, $d_2 = 0, 2, 3$.

Since the study of each of these cases is quite simple, we only consider the first one of them, which is also the most complex, leaving the others to the reader.

Let us assume that $t_1 = \begin{array}{|c|c|}\hline x & y \\ \hline v & z \\ \hline \end{array}$, and $t_2 = \begin{array}{|c|c|}\hline x' & y' \\ \hline v' & z' \\ \hline \end{array}$ with $x, x', y, y', v, v', z, z' \in \{0, \ldots, n-1\}$. We set

$$u(t_1) = \begin{array}{|c|c|c|c|}\hline \sharp & \sharp & \sharp & \sharp \\ \hline \sharp & x & y & \sharp \\ \hline \sharp & v & z & \sharp \\ \hline \sharp & \sharp & \sharp & \sharp \\ \hline \end{array},$$

while to determine a suitable $u(t_2)$ we must study separately the following cases:

(a) $x \neq x'$, $y \neq y'$, $z \neq z'$, and $v \neq v'$. In this case we easily set:

$$u(t_2) = \begin{array}{|c|c|c|c|}\hline \sharp & \sharp & \sharp & \sharp \\ \hline \sharp & x' & y' & \sharp \\ \hline \sharp & v' & z' & \sharp \\ \hline \sharp & \sharp & \sharp & \sharp \\ \hline \end{array}.$$

(b) $x = x'$, $y \neq y'$, $z \neq z'$, and $v \neq v'$ (and similarly we can treat all the cases where t_1 and t_2 have only one common element). In this case we have that $t_2 = \begin{array}{|c|c|}\hline x & \overline{y} \\ \hline \overline{v} & \overline{z} \\ \hline \end{array}$, where

$$\overline{w} = \begin{cases} 1 & \text{if } w = 0, \\ 0 & \text{if } w \neq 0. \end{cases}$$

Here we can set:

$$u(t_2) = \begin{array}{|c|c|c|c|}\hline \sharp & \sharp & \sharp & \sharp \\ \hline \sharp & \overline{x} & \overline{y} & \sharp \\ \hline \sharp & x & \overline{y} & \sharp \\ \hline \sharp & \overline{v} & \overline{z} & \sharp \\ \hline \sharp & \sharp & \sharp & \sharp \\ \hline \end{array}.$$

The reader can check that $\|u(t_1)\|$ and $\|u(t_2)\|$ have no tiles in common.

(c) $x = x'$, $y = y'$, $z \neq z'$, and $v \neq v'$ (and similarly we can treat all the cases where t_1 and t_2 have an equal row or column), i.e. $t_2 = \begin{array}{|c|c|}\hline x & y \\ \hline \overline{v} & \overline{z} \\ \hline \end{array}$. Setting

$$u(t_2) = \begin{array}{|c|c|c|c|}\hline \sharp & \sharp & \sharp & \sharp \\ \hline \sharp & \overline{x} & \overline{y} & \sharp \\ \hline \sharp & x & y & \sharp \\ \hline \sharp & \overline{v} & \overline{z} & \sharp \\ \hline \sharp & \sharp & \sharp & \sharp \\ \hline \end{array},$$

we can easily see that $\|u(t_1)\|$ and $\|u(t_2)\|$ have no tiles in common.

(d) $x = x'$, $z = z'$, $y \neq y'$, and $v \neq v'$ (and similarly $y = y'$, $v = v'$, $x \neq x'$, and $z \neq z'$), i.e. $t_2 = \begin{array}{|c|c|}\hline x & \overline{y} \\ \hline \overline{v} & z \\ \hline \end{array}$. Setting

$$u(t_2) = \begin{array}{|c|c|c|c|}\hline \sharp & \sharp & \sharp & \sharp \\ \hline \sharp & \overline{x} & \overline{y} & \sharp \\ \hline \sharp & x & \overline{y} & \sharp \\ \hline \sharp & \overline{v} & z & \sharp \\ \hline \sharp & \overline{v} & \overline{z} & \sharp \\ \hline \sharp & \sharp & \sharp & \sharp \\ \hline \end{array},$$

we see that $\|u(t_1)\|$ and $\|u(t_2)\|$ have no tiles in common.

(e) $x = x'$, $y = y'$, $v = v'$, and $z \neq z'$ (and similarly we can treat all the cases where t_1 and t_2 differ in only one element). Here we have $t_2 = \begin{array}{|c|c|}\hline x & y \\ \hline v & \overline{z} \\ \hline \end{array}$. We set:

$$u(t_2) = \begin{array}{|c|c|c|c|c|c|}\hline \sharp & \sharp & \sharp & \sharp & \sharp & \sharp \\ \hline \sharp & \overline{x} & y & \overline{y} & \overline{y} & \sharp \\ \hline \sharp & \overline{x} & x & y & z & \sharp \\ \hline \sharp & \overline{v} & v & \overline{z} & \overline{z} & \sharp \\ \hline \sharp & \sharp & \sharp & \sharp & \sharp & \sharp \\ \hline \end{array}.$$

The picture $u(t_2)$ is made of 20 tiles; it is possible to check that they are all different from the 9 tiles of $u(t_1)$.

■

2.1.4 Join-irreducible elements

We now turn our attention to join-irreducible elements of $\mathcal{L}\mathrm{oc}_n$.
Let v be a picture over $\Sigma = \{0, 1, \ldots, n-1\}$. We can consider the set of tiles $\|v\|$ and the respective local language $L(v)$.

Theorem 11 *A local language $L = L(\Theta) \in \mathcal{L}\mathrm{oc}_n$ is join-irreducible if and only if there exists $v \in \Sigma^{**}$ such that $L = L(v)$ and there are no pictures $u_1, \ldots, u_n$ such that*

- *the $\|u_i\|$ are pairwise different;*
- $\|u_1\| \cup \ldots \cup \|u_n\| = \|v\|$.

Proof. ($\Longrightarrow$) Let L be join-irreducible. Suppose that such a v does not exist, and let $L = L(\Theta)$ for some irredundant Θ. Then for each tile in Θ there is a picture that contains it. Then we can write: $\Theta = \|u_1\| \cup \cdots \cup \|u_n\|$ and we obtain $L(\Theta) = L(u_1) \vee \cdots \vee L(u_n)$. Thus $L(\Theta)$ is not join-irreducible.

The right-to-left implication follows easily from the definition of a join-irreducible element. ■

Let us remark that it is not sufficient to require that $L = L(v)$ to state that L is join-irreducible, as shown in the following example.

Example 13 *(In $\mathcal{L}\mathrm{oc}_2$) Consider the sets of tiles:*

$$\Theta = \left\| \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \right\| \cup \left\| \begin{array}{|c|c|} \hline 0 & 0 \\ \hline 0 & 1 \\ \hline \end{array} \right\| \cup \left\| \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & 0 \\ \hline \end{array} \right\|; \qquad \Theta_1 = \left\| \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \right\|; \qquad \Theta_2 = \left\| \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \right\|.$$

We have: $\Theta = \Theta_1 \cup \Theta_2$ and then $L(\Theta) = L(\Theta_1) \vee L(\Theta_2)$ so $L(\Theta)$ is not join-irreducible. According to Theorem 11, the following pictures witness join-reducibility of $L(\Theta)$:

$$u_1 = \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}; \qquad u_2 = \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline \end{array}.$$

As a neat consequence of Theorem 11 we have that if $L \in \mathcal{L}\mathrm{oc}_n$ is an atom, then there exists $v \in \Sigma^{**}$ such that $L = L(v)$. On the other hand, if L is a coatom then L is not join-irreducible: in fact we can not write a coatom $L(\Theta)$ as $L(v)$, since in Θ there are at least 7 "corner" tiles, whereas in every picture there are exactly 4 "corner" tiles.

2.2 Some undecidable problems

The previous section shows that the meet-irreducible elements (equivalently, the coatoms) of $\mathcal{L}\mathrm{oc}_n$ have an easy characterization, from which one can immediately deduce that the property of being meet-irreducible (or equivalently, a coatom) is decidable.

Corollary 2 *Given n and a set Θ of tiles on $\{0, \ldots, n-1\}$, one can decide whether $L(\Theta)$ is meet-irreducible (equivalently, a coatom).*

Proof. Trivial, by Theorem 10. ∎

Quite surprisingly we show in this section that this is true neither of the join-irreducible elements, nor of the atoms: in fact that the property of being join-irreducible, and the property of being an atom are undecidable.
We deal here with sets of tiles that need not be irredundant. In [24], several undecidable problems concerning local languages are studied. For instance, we recall that the following problems are undecidable:

- the *equality problem*: "Is $L(\Theta) = L(\Theta')$?", where Θ, Θ' are given sets of tiles. (Once again, we remark that here Θ and Θ' are not supposed to be irredundant to begin with, otherwise the problem is decidable, since in the case of irredundant sets of tiles we have that $L(\Theta) = L(\Theta')$ if and only if $\Theta = \Theta'$);
- the *irredundancy problem*: "Is Θ irredundant?" where Θ is a given set of tiles;
- the *infinity problem*: "Is $L(\Theta)$ infinite?", where Θ is a given set of tiles.

We are now going to point out some additional undecidable problems which relate more directly to the lattice theoretic structure of local languages.

Theorem 12 *Suppose that Θ, Θ' are given sets of tiles on a common alphabet, say, of n symbols. The problems of ascertaining, given Θ, Θ' and $m \geq n$ whether:*

1. *$L(\Theta)$ is an atom in $\mathcal{L}\mathrm{oc}_m$,*
2. *$L(\Theta)$ is a cover of $L(\Theta')$ in $\mathcal{L}\mathrm{oc}_m$,*

3. $L(\Theta)$ *is join-irreducible in* $\mathcal{L}\mathrm{oc}_m$,

are undecidable.

Proof. We will show below that for every Turing machine M one can effectively find a set of tiles Θ_M on some alphabet, such that

1. $L(M) \neq \emptyset$ if and only if $L(\Theta_M) \neq \emptyset$;
2. $L(\Theta_M)$ is either empty or a singleton.

This is enough to show the claim. Indeed, consider any atom $L(\Theta_0)$ over the alphabet $\{0\}$. Without loss of generality we may assume that the character 0 does not appear in any tile of Θ_M. Then

1. $L(M) \neq \emptyset$ if and only if $L(\Theta_M)$ is an atom;
2. $L(M) \neq \emptyset$ if and only if $L(\Theta_0 \cup \Theta_M)$ is a cover of $L(\Theta_0)$;
3. $L(M) \neq \emptyset$ if and only if $L(\Theta_0 \cup \Theta_M)$ is join-reducible.

Therefore the claim follows by observing that the problem "$L(M) \neq \emptyset$" is undecidable, see for instance [45, Theorem 6.3.1(d)].
It is now left to show how we can build Θ_M starting from M. Let us consider deterministic Turing machines working on an alphabet Σ. Our model of Turing machines follows closely that of [45], with some minor notational variants. Instructions are quadruples $qaXr$, where q, r are states, $a \in \Sigma \cup \{B\}$, and $X \in \Sigma \cup \{B, L, R\}$, with the standard meaning. M accepts a string $u \in \Sigma^*$ with a unique halting state h. The initial state is denoted by q_0. We may assume that $q_0 \neq h$. As in [45] we also assume that the tape used by the machine has a leftmost cell. A configuration is a quadruple (v, q, a, w) which represents that the machine is in state q; a is the content of the cell currently scanned; $v \in (\Sigma \cup \{B\})^*$ is the content of the tape to the left of this cell; $w \in (\Sigma \cup \{B\})^*$ is the content of the tape to the right of this cell, with the understanding that all remaining cells contains B. A *representative* configuration (v, q, a, w) is one in which w does not end with B. We often write below (v, q, a, w) as $vqaw$; if $a = B$, and $w \in \{B\}^*$, then we may simply write vq.
Following [33, Theorem 9.1], a halting computation of a Turing machine M can be coded by a two-dimensional picture in the following way. Suppose that a halting computation consists of a sequence of configurations $c_1, c_2, \ldots, c_n$,

where $c_1 = q_0 u$ and $c_n = vhw$. We can imagine that all c_i's have the same length: if not, let

$$l = \max\{l_i : 1 \leq i \leq n\}$$

where l_i is the length of c_i, and let

$$\hat{c}_i = c_i B^{l-l_i}.$$

Thus the $\hat{c}_i$'s have the same length. Further, let $\overline{\Sigma} = \{\overline{a} : a \in \Sigma\}$ be a copy of the alphabet Σ (with $\Sigma \cap \overline{\Sigma} = \emptyset$), let $\overline{B}$ be a copy of B, and for a word $v \in (\Sigma \cup \{B\})^*$, let $\overline{v}$ be the corresponding copy of u over the alphabet $\overline{\Sigma}$. For every i, if $\hat{c}_i = vqw$ then let $\gamma_i = \overline{v}qw$. The computation can thus be coded by p, where

$$p = \begin{array}{|c|}\hline \gamma_1 \\ \hline \vdots \\ \hline \gamma_n \\ \hline \end{array}.$$

p can be viewed as a picture of the local language given by a suitable set of tiles $\Theta(M)$, whose tiles mirror the instructions of M used to bring the machine from one configuration to the next one. For instance the instruction $paLq$ is coded, among others, by the tiles

$$\begin{array}{|c|c|}\hline \overline{s} & p \\ \hline q & s \\ \hline \end{array}, \quad \begin{array}{|c|c|}\hline p & a \\ \hline s & a \\ \hline \end{array}$$

where $s \in \Sigma \cup \{B\}$. The details of the construction of $\Theta(M)$ can be found in [**?**]. One has:

$$L(M) \neq \emptyset \Leftrightarrow L(\Theta(M)) \neq \emptyset.$$

We now show how, given a Turing machine M, one can effectively construct the desired Turing machine $\tilde{M}$. First of all note that starting from M one can effectively find a Turing machine M' (with, say, initial state q'_0 and halting state h') on the same alphabet, such that $L(M') \subseteq \{e\}$ (where e denotes the empty string) and

$$L(M) \neq \emptyset \Leftrightarrow M' \text{ halts on } e.$$

For this, simply consider a Turing machine M' such that on input u, M' does not halt if $u \neq e$; and M' on e halts if and only if there is a string v such that M halts on v. One would be tempted to take $\Theta_M = \Theta(M')$, but unfortunately although M' can perform at most one halting computation,

nothing guarantees that if M' halts on e then there is only one picture corresponding to this accepting computation. Indeed, suppose for instance that p is a picture that codes a halting computation of M' and the last column of p consists only of symbols $s \in \Sigma \cup \{B\}$. Then due to the presence in $\Theta(M')$ of the tiles

$$\boxed{\begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline \end{array}}, \boxed{\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline a & \sharp \\ \hline \end{array}}, \boxed{\begin{array}{|c|c|}\hline a & \sharp \\ \hline \sharp & \sharp \\ \hline \end{array}}, \boxed{\begin{array}{|c|c|}\hline a & \sharp \\ \hline b & \sharp \\ \hline \end{array}}$$

with $a, b, c, \in \Sigma \cup \{B\}$, one would be able to conclude that $p \cdot_h p' \in L(\Theta(M')$, where p' is any picture of $(\Sigma \cup \{B\})^{**}$ of the same vertical dimension as p, and $p \cdot_h p'$ denotes the picture obtained by horizontally concatenating p and p'. In order to avoid this, we effectively build from M' a machine $\tilde{M}$, with $\tilde{q}_0, \tilde{h}$ respectively as initial state and accepting state, working on an alphabet $\Sigma \cup \{\Omega\}$, where Ω is an extra symbol, such that $L(\tilde{M}) = L(M')$ and with the following additional features:

1. $\tilde{M}$ is not *restarting*, i.e. there are no instruction of the form $qaX\tilde{q}_0$;
2. if $\tilde{M}$ halts on e, then this halting computation consists of a sequence of representative configurations $\tilde{c}_1, \ldots, \tilde{c}_k$ with $\tilde{c}_1 = \tilde{q}_0$ and $\tilde{c}_k = z\tilde{h}$, and the length of $\tilde{c}_k$ is greater than the length of any other $\tilde{c}_i$.

In this case if $p \in L(\Theta(\tilde{M}))$ then the upper left tile of $\hat{p}$ is necessarily $\boxed{\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \sharp & \tilde{q}_0 \\ \hline \end{array}}$, and since $\tilde{M}$ is nonrestarting there is in in $L(\Theta(\tilde{M}))$ no additional horizontal concatenation $p' \cdot_h p$, nor is there any vertical concatenation $p' \cdot_v p$ (meaning that p' has the same horizontal dimension as p and is vertically concatenated on top of p.) Similarly the lower right tile is necessarily $\boxed{\begin{array}{|c|c|}\hline \tilde{h} & \sharp \\ \hline \sharp & \sharp \\ \hline \end{array}}$: again, since the last configuration is of the form $w\tilde{h}$ and is greater than any other $\tilde{c}_i$, no horizontal concatenation, nor any vertical concatenation can produce additional pictures in $L(\Theta(\tilde{M}))$.

The machine $\tilde{M}$ has the same states as M', plus an additional state q_R and a new accepting state $\tilde{h}$. The instructions are as follows: replace every instruction $qaBr$ of M', $a \neq B$, with the instruction $qa\Omega r$; replace every instruction $qBXr$ of M' with $q\Omega Xr$; add the instructions $h'\Omega\Omega q_R$, $q_R a R q_R$ (with $a \neq B$), $q_R BB\tilde{h}$; keep all other instructions of M'. (The idea is that $\tilde{M}$ on e outputs the same string as M', but with Ω replacing all intermediate occurrences of B in the output string, and then moves to the rightmost blank cell.)

It is now easy to see that $\Theta_M = \Theta(\tilde{M})$ is a set of tiles with the desired properties. ■

Even if we have provided in Section 2.1.4 a characterization of the join-irreducible elements, it is not surprising that the property of being join-irreducible is undecidable. The point is that the given characterization does not allow us to decide whether or not a given set of tiles determines a join-irreducible element, in other words the property, for a given set of tiles Θ, of being of the form $L(\Theta) = L(v)$ for some picture v, is itself undecidable.

2.3 The lattice $\mathcal{L}\mathrm{oc}_2$

Let us study in more detail $\mathcal{L}\mathrm{oc}_2$. The observations made in this section can easily be extended, *mutatis mutandis*, to $\mathcal{L}\mathrm{oc}_n$.
From an algorithmic point of view, the lattices $\mathcal{L}\mathrm{oc}_n$ are difficult objects to deal with. Indeed, for every $\Theta \in \Theta^n_{Tot}$, let r_Θ be the least number such that there is a picture in $L(\Theta)$, if any, of dimensions $v, h \leq r_\Theta$; otherwise let $r_\Theta = 0$, and define $f(n) = \max_{\Theta \in \Theta^n_{Tot}} r_\Theta$.
We recall from [24]:

Lemma 3 *The function f is not computable. In fact f is not dominated by any computable function, i.e. for every computable function g there exist infinitely many n such that $f(n) > g(n)$.*

This implies that that the cardinality of $\mathcal{L}\mathrm{oc}_n$ increases extremely fast, which makes a computer-based investigation of these lattices even for low n very problematic. This motivates our direct (not computer-based) investigation of $\mathcal{L}\mathrm{oc}_2$, which is carried out in this section.

2.3.1 Chains

In this section we show that in $\mathcal{L}\mathrm{oc}_2$ there exist two maximal chains c_1 and c_2 of different lengths.

The chain c_1. We build c_1 according to the following procedure: we start up with $L := L(\Theta_{Tot})$; at each step of the execution of the algorithm, given $L = L(\Theta)$ we look for $\Theta' \subset \Theta$ (obtained from Θ by deleting one or two tiles) so that L is a cover of $L(\Theta')$, and we assign $L := L(\Theta')$.
In detail: Consider the set Θ_{Tot}:

1. One by one we erase the corner tiles containing the symbol 1 , starting e.g. from the tile $\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \sharp & 1 \\ \hline \end{array}$, and proceeding in any order. We obtain in this way the languages (each one a cover of the following one) L_1, L_2, L_3, L_4: Notice that L_4 contains all the pictures with 0 in each corner.

2. Next we erase one by one in succession the tiles

$$\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline 1 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 1 \\ \hline \sharp & \sharp \\ \hline \end{array}, \begin{array}{|c|c|}\hline \sharp & 1 \\ \hline \sharp & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & \sharp \\ \hline 1 & \sharp \\ \hline \end{array}$$

   obtaining in succession, languages L_5, L_6, L_7, L_8: Notice that L_8 contains all the pictures such that no consecutive occurrences of 1 can be found in the border.

3. Next we start erasing two tiles at a time. First we erase the border tiles

$$\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline 0 & 1 \\ \hline \end{array}$$

   obtaining the language L_9. Observe that if we simply remove the first one, then the other one is not used in any picture, hence the set is redundant. We continue by erasing, one by one, the remaining pairs of border tiles containing both 0 and 1, obtaining the languages L_{10}, L_{11}, L_{12}. Notice that L_{12} contains pictures consisting of only the symbol 0 in the borders, thus L_{12} is "isomorphic" to $L_{\Theta_{Tot}}$.

4. Next we erase one by one in succession the tiles:

$$\begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 0 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 1 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & 0 \\ \hline \end{array},$$

   obtaining the languages $L_{13}, L_{14}, L_{15}, L_{16}, L_{17}, L_{18}$.

5. We continue by erasing first the tile $\begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & 1 \\ \hline \end{array}$, and then, one by one, the two sets of two tiles

$$\left\{ \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 0 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 1 \\ \hline \end{array} \right\}, \left\{ \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 0 & 1 \\ \hline \end{array} \right\},$$

obtaining the languages L_{19}, L_{20}, L_{21}: notice that L_{21} consists exactly of pictures of only 0's, together with the picture

$$p = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}\,.$$

6. We eliminate p by erasing the set of tiles

$$\left\{ \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 0 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & 0 \\ \hline \end{array} \right\}$$

obtaining the language L_{22} consisting of pictures of only 0's.

7. We erase $\begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array}$, obtaining $L_{23} = L_{c0} \vee L_{r0}$.

8. We erase the set of two vertical border tiles containing two 0's, obtaining the language $L_{24} = L_{r0}$.

9. We erase the set of two horizontal border tiles containing two 0's, obtaining the language

$$L_{25} = \left\{ \begin{array}{|c|}\hline 0 \\ \hline \end{array} \right\}.$$

10. Finally, let $L_{26} = \emptyset$.

Together with $L(\Theta_{Tot})$ the previous languages constitute a maximal chain of length 27.

The chain c_2. Recall the atom $L(\Theta_3)$ of Example 14 and recall that Θ_{Tot} has 39 elements. The set Θ_3 has 17 tiles, so there exists a maximal chain containing $L(\Theta_3)$ that has at most $39 - 17 + 1 + 1 = 24 \neq 27$ elements.

2.4 The lattice $\mathcal{L}\mathrm{oc}_2^h$

A bidimensional language is called a *string language* if its elements are of size $1 \times n$, i.e. they consist of only one row. In this section we describe the lattice of local string languages over the alphabet $\Sigma = \{0, 1\}$. To simplify matters, we directly regard pictures of string languages as strings.

In general, given an alphabet Γ let Loc^h_Γ be the family of local string languages over Γ (minus $\boxed{\ }$). Let us now specialize to the alphabet $\{0,1\}$. The class of local string languages over this alphabet will be denoted by Loc^h_2.
The following results are immediate.

Lemma 4 *$\mathcal{L}\mathrm{oc}^h_2 = \langle Loc^h_2, \wedge, \vee, \subseteq\rangle$, where $\wedge$ and $\vee$ are given by restriction, is a sublattice, in fact an ideal, of $\mathcal{L}\mathrm{oc}_2$.*

Lemma 5 *The atoms of $\mathcal{L}\mathrm{oc}^h_2$ are exactly the following 4 languages:*

- *the languages: $L_0 = \{\boxed{0}\}$ and $L_1 = \{\boxed{1}\}$*
- *the languages: $L_{01} = \{\boxed{0\,|\,1}\}$ and $L_{10} = \{\boxed{1\,|\,0}\}$.*

Remark 5 *The coatoms are obtained in the following way: let Θ^h_{Tot} be the set of all possible horizontal tiles minus the tile $\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \sharp & \sharp \\ \hline \end{array}$, i.e. the set of tiles that can be used to form string pictures. Notice that by erasing from Θ^h_{Tot} only one tile we obtain a redundant set of tiles; on the other hand, we do get an irredundant set of tiles if we erase an upper border tile and the correspondent lower border tile (i.e. a pair of the form*

$$\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline i & j \\ \hline \end{array}\,,\ \begin{array}{|c|c|}\hline i & j \\ \hline \sharp & \sharp \\ \hline \end{array}$$

where $i, j \in \{0,1\}$, or a pair of the form

$$\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \sharp & i \\ \hline \end{array}\,,\ \begin{array}{|c|c|}\hline \sharp & i \\ \hline \sharp & \sharp \\ \hline \end{array}$$

with $i \in \{0,1\}$, or the corresponding pair of right corner border tiles).
It is easy to see that the languages which we obtain by erasing such a pair are all the coatoms.

Notation. Henceforth, for elements of $\mathcal{L}\mathrm{oc}^h_2$ let us employ the notation that we use for the proof of Lemma 1 and the following

- by writing $\backslash t_1, \ldots, t_k$ we mean the language provided by the set of tiles obtained by deleting from Θ^h_{Tot} the tiles $t_1, \ldots, t_k$;

Using the example given in Remark 3 we are able to state the following

Theorem 13 *The lattice $\mathcal{L}\mathrm{oc}_2^h$ is not semimodular.*

Notwithstanding the fact that $\mathcal{L}oc_2^h$ is not semimodular, we have that the Jordan- Dedekind condition holds, i.e. all the chains of $\mathcal{L}oc_2^h$ have the same length. To see this, we describe below, by direct inspection, the rank function for $\mathcal{L}\mathrm{oc}_2^h$. (Except when the language is an atom, we describe each language below by indicating a set of tiles originating the language: in fact we will write $t_1, \ldots, t_k$ for $\{t_1, \ldots, t_k\}$; for languages of rank > 2 we use the notation $\backslash t_1, \ldots, t_k$ introduced earlier.)

1. rank 0: $\emptyset$;
2. rank 1 (4 elements): the four atoms already described $\{\boxed{1 \mid 0}\}$, $\{\boxed{1}\}$, $\{\boxed{0}\}$, $\{\boxed{0 \mid 1;}\}$;
3. rank 2 (15 elements): $[1, \overline{11}, \overline{10}, 0]$; $[1, \overline{10}, 0], 1]$; $[1, \overline{10}, \overline{01}, 0]$; $[1, \overline{10}, \overline{01}, 1]$; $[1, \overline{10}, \overline{00}, 0]$; $[1, \overline{11}, 1]$; $[0, \overline{01}, \overline{11}, 1]$; $[0, \overline{01}, 0], 1]$; $[0, \overline{00}, 0]$; $[0, \overline{01}, \overline{10}, 0]$; $[0, \overline{00}, \overline{01}, 1]$; $[0, [1, 0], 1]$; $[0, [1, \overline{10}, 0]$; $[1, [0, \overline{01}, 1]$.

   Notice that $[1, \overline{10}, 0], 1] = \{\boxed{1}\} \vee \{\boxed{1 \mid 0}\}$; $[0, \overline{01}, 0], 1] = \{\boxed{0}\} \vee \{\boxed{0 \mid 1}\}$; $[0, [1, 0], 1] = \{\boxed{0}\} \vee \{\boxed{1}\}$; $[0, [1, \overline{10}, 0] = \{\boxed{0}\} \vee \{\boxed{1 \mid 0}\}$; $[1, [0, \overline{01}, 1] = \{\boxed{1}\} \vee \{\boxed{0 \mid 1}\}$.
4. rank 3 (26 tiles):$\backslash[0, \overline{00}, \overline{01}$, $\backslash[0, \overline{00}, \overline{11}$, $\backslash[0, \overline{00}, 0]$, $\backslash[0, \overline{00}, 1]$, $\backslash[0, \overline{01}, \overline{11}$, $\backslash[0, \overline{01}, 1]$, $\backslash[0, \overline{11}, 0]$, $\backslash[0, \overline{11}, 1]$, $\backslash[1, \overline{00}, \overline{10}$, $\backslash[1, \overline{00}, \overline{11}$, $\backslash[1, \overline{00}, 0]$, $\backslash[1, \overline{00}, 1]$, $\backslash[1, \overline{10}, \overline{11}$, $\backslash[1, \overline{10}, 0]$, $\backslash[1, \overline{11}, 0]$, $\backslash[1, \overline{11}, 1]$, $\backslash\overline{00}, \overline{01}, \overline{10}$, $\backslash\overline{00}, \overline{01}, \overline{11}$, $\backslash\overline{00}, \overline{01}, 1]$, $\backslash\overline{00}, \overline{10}, \overline{11}$, $\backslash\overline{00}, \overline{10}, 0]$, $\backslash\overline{00}, \overline{11}, 0]$, $\backslash\overline{00}, \overline{11}, 1]$, $\backslash\overline{01}, \overline{10}, \overline{11}$, $\backslash\overline{01}, \overline{11}, 1]$, $\backslash\overline{10}, \overline{11}, 0]$;
5. rank 4 (22 elements): $\backslash[0, \overline{00}$, $\backslash[0, \overline{01}$, $\backslash[0, \overline{11}$, $\backslash[0, 0]$, $\backslash[0, 1]$, $\backslash[1, \overline{00}$, $\backslash[1, \overline{10}$, $\backslash[1, \overline{11}$, $\backslash[1, 0]$, $\backslash[1, 1]$, $\backslash\overline{00}, \overline{11}$, $\backslash\overline{00}, \overline{10}$, $\backslash\overline{00}, \overline{11}$, $\backslash\overline{00}, 0]$, $\backslash\overline{00}, 1]$, $\backslash\overline{01}, \overline{10}$, $\backslash\overline{01}, \overline{11}$, $\backslash\overline{01}, 1]$, $\backslash\overline{10}, \overline{11}$, $\backslash\overline{10}, 0]$, $\backslash\overline{11}, 0]$, $\backslash\overline{11}, 1]$.

   Notice that $\backslash\overline{00}, \overline{11} = \{\boxed{0 \mid 1}\} \vee \{\boxed{1 \mid 0}\}$.
6. rank 5 (8 elements): these are the coatoms (see Remark 5) $\backslash[0$, $\backslash[1$, $\backslash\overline{00}$, $\backslash\overline{01}$, $\backslash\overline{10}$, $\backslash\overline{11}$, $\backslash 0]$, $\backslash 1]$;
7. rank 6: $L(\Theta_{Tot}^h)$.

In total, $\mathcal{L}\mathrm{oc}_2^h$ has 77 elements. By the description of the rank function given above we have:

Corollary 3 *The rank function* $rank : \mathcal{L}\mathrm{oc}_2^h \to \mathbf{N}$ *satisfies:*

$$rank(L(\Theta)) = \begin{cases} \frac{number\ of\ tiles\ of\ \Theta}{2} - 2 & if\ \overline{01} \subseteq \Theta\ or\ \overline{10} \subseteq \Theta \\ \frac{number\ of\ tiles\ of\ \Theta}{2} - 1 & otherwise\ and\ \Theta \neq \emptyset \\ 0 & if\ \Theta = \emptyset. \end{cases}$$

(Once again, we assume that each Θ *is irredundant.)*

2.5 Further Works and Open Problems

We have left open many questions about the lattices $\mathcal{L}\mathrm{oc}_n$, and in particular about $\mathcal{L}\mathrm{oc}_2$. In this section we list some of the problems which we believe of particular interest. For simplicity, we mostly state our problems for $\mathcal{L}\mathrm{oc}_2$. On the other hand it is felt that a solution in the case $n = 2$ should easily extend to the general case of $n \geq 2$.

Atoms. Despite the undecidability result stated in Theorem 12, 1., we can give several examples of atoms in $\mathcal{L}\mathrm{oc}_2$.

Example 14 *The reader can easily verify that the following languages are atoms:*

1. *the singleton languages* L_0, L_{01}, L_{10} *introduced in the proof of Theorem 9, and the singleton language* $L_1 = \left\{\boxed{1}\right\}$*;*
2. *the languages* $L(\Theta_1)$*,* $L(\Theta_2)$*,* $L(\Theta_3)$*, where*

$$\Theta_1 = \left\| \begin{array}{cccc} 0 & 0 & 1 \\ 1 & 0 & 0 \end{array} \right\|, \qquad \Theta_2 = \left\| \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{array} \right\|, \qquad \Theta_3 = \left\| \begin{array}{cccc} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{array} \right\|.$$

All the atoms of the example, except for $L(\Theta_3)$, are finite, and moreover made up only of one picture. The atom $L(\Theta_3)$ is instead infinite.
These simple considerations leave open many questions.

Problem 1 *Is every finite atom of* $\mathcal{L}\mathrm{oc}_2$ *a singleton?*

Problem 2 *Give a useful characterization of the atoms of* $\mathcal{L}\mathrm{oc}_2$*.*

Complements. Other interesting questions concern complements. We recall that in a lattice with maximum element 1 and minimum element 0, we say that b is a *complement* of a if $a \vee b = 1$ and $a \wedge b = 0$.
For example, let L_s be the language of all chessboard pictures with 0 in each corner, introduced in Example 11. The local language represented by the set of tiles below is a complement of L_s.

$$\Theta = \left\| \begin{array}{cccccc} 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{array} \right\|$$

It is sufficient to add the corner tile $\begin{array}{|c|c|}\hline 0 & \sharp \\ \hline \sharp & \sharp \\ \hline \end{array}$ to Θ to obtain a different complement of L_s.
One can easily prove that if $L = L(\Theta)$, and Θ contains $n \leq 7$ corner tiles, then L has (at least) a complement. Using such a condition we can state that all the local languages of the form $L = L(v)$ have complements; hence all join-irreducible elements have complements.
Actually, all elements of $\mathcal{L}\mathrm{oc}_2$ we happened to consider have more than one complement. We have not as yet found examples of elements without complements or with exactly one complement.

Problem 3 *If L is an element of $\mathcal{L}oc_2$, then does there exist in $\mathcal{L}oc_2$ a complement of L? If it exists, when is it unique?*

On the other hand, in $\mathcal{L}\mathrm{oc}_2^h$ not all elements have complements, for instance the coatom $L_a = L(\backslash\overline{00})$ does not have any complement.

The lattice of symmetric elements. Let $L(\Theta)$ be a local language, and define the *transpose language* $L(\Theta)^T$ of $L(\Theta)$ as $L(\Theta)^T = L(\Theta^T)$, where

$$t = \begin{array}{|c|c|}\hline x & y \\ \hline z & t \\ \hline \end{array}, \quad t^T = \begin{array}{|c|c|}\hline x & z \\ \hline y & t \\ \hline \end{array}$$

and $\Theta^T = \{t^T : t \in \Theta\}$.

We say that a local language $L(\Theta)$ is *symmetric* if $L(\Theta) = L(\Theta)^T$. It is easy to see that a local language $L(\Theta)$ is symmetric if and only if $\Theta = \Theta^T$.
For instance, the local language of the square pictures with 1 in one diagonal and 0 on all the other entries is a symmetric language.
The set $\mathcal{S}_n$ of the symmetric local languages on $\{0, \ldots, n-1\}$, is a sublattice of $\mathcal{L}oc_n$. Moreover it is easy to see that for all $L \in \mathcal{L}oc_n \setminus \mathcal{S}_n$ there exists a language $\widetilde{L}$ such that $L \vee \widetilde{L} \in \mathcal{S}_n$ and $L \wedge \widetilde{L} \in \mathcal{S}_n$. (Just take $\widetilde{L} = L^T$).
We feel that a careful investigation of $\mathcal{S}_n$ would give us also information on the lattice $\mathcal{L}oc_n$. We limit ourselves here to the following observation.
In $\mathcal{S}_n$ a language L is a coatom if and only if there exists some tile $t \in \Theta_{Tot}$ such that

$$L = \begin{cases} L(\Theta_{Tot} \setminus \{t\}) & \text{if } t = t^T, \\ L(\Theta \setminus \{t, t^T\} & \text{otherwise.} \end{cases}$$

For example the languages: $L(\Theta_{Tot} \setminus t_1)$, where $t_1 = \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & 1 \\ \hline \end{array}$ and $L(\Theta_{Tot} \setminus \{t_2, t_3\})$, where $t_2 = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 0 & 0 \\ \hline \end{array}$, $t_3 = \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 0 \\ \hline \end{array}$, are coatoms.

Problem 4 *Study the sublattice $\mathcal{S}_n$.*

Chapter 3

Two-dimensional languages and computability

We already said that one of the object of our study is to find the analogies or differences between string and picture languages, and when it is possible to generalize the results known for the unidimensional case to the twodimensional one. Now we analyze the situation in computability theory. In literature it is well known the arithmetical complexity of several problems concerning with string languages. Let us recall for example one of the most famous, the *emptiness problem*; such a problem asks when a language is empty and it is decidable for regular languages. In this chapter, the aim is to formulate the main problems, already acquainted in the unidimensional case, for the local and tiling system recognizable languages, and to classify their complexity in the arithmetical hierarchy.
For the terminology and notions we refer to [56].

We recall some basic definitions.

Definition 29 Given two sets A, B we say that A *is m-reducible to* B (notation: $A \leq_m B$) if $A = f^{-1}[B]$ for some computable function f. A Σ_n^0*-set* A is a set for which there exists a computable relation $R(x, y_1, \ldots, y_n)$ such that

$$A = \{x | \exists y_1 \forall y_2 \ldots R(x, y_1, \ldots, y_n)\}$$

i.e. membership in A can be described by a computable relation prefixed by n alternating quantifiers starting with an existential quantifier. We also write $A \in \Sigma_n^0$ to denote that A is a Σ_n^0-set.

Equivalently, one could give the following inductive definition: $A \in \Sigma_0^0$ if and only if A is computable; and for $n \geq 1$ $A \in \Sigma_n^0$ if and only if A is computably enumerable in some $B \in \Sigma_{n-1}^0$.
A Π_n^0-*set* is just a set A whose complement is a Σ_n^0-set.

Definition 30 A set A is said to be Σ_n^0-*complete* (Π_n^0-*complete*) if $A \in \Sigma_n^0$ ($A \in \Pi_n^0$) and for every $B \in \Sigma_n^0$ ($B \in \Pi_n^0$) one has $B \leq_m A$.

The previous notions and terminology naturally extend to sets that can be encoded by natural numbers, this making possible to talk, in general, about Σ_n^0-complete (Π_n^0-complete) problems, and so on.
Giammarresi and Restivo, [33], show how to code computations of Turing machines into local languages. For later reference we briefly review some of the details of this coding.
We will consider here deterministic Turing machines on alphabet Ξ, for which acceptance is defined only via a fixed terminal state q_f.

Definition 31 A *configuration* is a sequence of the form uqv, where q is a state, $u, v \in (\Xi \cup \{B\})^*$ (B being the usual symbol denoting that the content of the tape cell is empty) are the strings of minimal length such that u contains all occurrences of the symbols different from B present on the tape to the left of the head of M, and v contains all occurrences of the symbols different from B present on the tape to the right of the head of M, including the head.

Definition 32 Given a Turing machine M, a successful *computation* of M on a word w is a sequence $c = c_1, \ldots, c_k$ of instantaneous configurations such that $c_1 = q_0 w$ (q_0 being the initial state), c_k is a terminal configuration, i.e. of the form $uq_f v$, and for all $1 \leq i < k$, c_{i+1} follows legally from c_i according to the instructions of M.

While performing a successful computation, M uses a finite portion of tape consisting of, say, n cells, so that each configuration c_i may be viewed as a substring of a string $\gamma_i \in B^* c_i B^*$ of length $n+1$, and in fact identified with γ_i. For reasons that will become clear later, let us encode a configuration $c = uqv$ by the string uqv' where v' is the image of v under the morphism induced by a fixed bijection of the alphabet $\Xi \cup \{B\}$ onto a copy $\Xi' \cup \{B'\}$. The computation may thus be viewed as a picture of dimension $(k, n+1)$ on the alphabet $\Xi \cup \{B\} \cup Q_M \cup \Xi' \cup \{B'\}$, where Q_M is the set of states of M.

It is rather straightforward to construct a set of tiles Θ_M on this alphabet such that M accepts a string w if and only if there is a picture $p \in L(\Theta_M)$ which represents a computation, as described earlier. The crucial point is in choosing the tiles that contain the state symbols so as to faithfully represent the instructions of M. Thus for instance an instruction $pabLq$ (where $a, b \in \Xi \cup \{B\}$, $p, q \in Q_M$, and L instructs to move "left") is represented by the tiles $\begin{array}{|c|c|}\hline s & p \\ \hline q & s' \\ \hline \end{array}$, $\begin{array}{|c|c|}\hline p & a' \\ \hline s' & b' \\ \hline \end{array}$ for every $s \in \Xi \cup \{B\}$. One then needs tiles to insure that the first row of a picture must represent an initial configuration, and the last row must represent a terminal configuration. The trick of considering uqv' instead of just uqv is motivated by the fact that in this way we can easily limit the language of Θ_M to pictures in which each row has only one occurrence of a state symbol (see [33] for more details).
The *Emptyness Problem* for tiling recognizable languages asks for an effective procedure to decide, given a tiling system T, whether the corresponding language $L(T)$ is empty or not, in other words if the set $\text{NEmpty}_{TS} = \{T : L(T) \neq \emptyset\}$ is decidable. The corresponding problem for local languages is given by $\text{NEmpty}_{LOC} = \{\Theta : L(\Theta) \neq \emptyset\}$. Using the above coding $M \mapsto \Theta_M$, Giammarresi and Restivo, [33], show that NEmpty_{TS}, NEmpty_{LOC} are undecidable. In fact:

Theorem 14 NEmpty_{TS}, NEmpty_{LOC} *are* Σ_1^0*-complete.*

Proof. Since $\text{NEmpty}_{LOC} \leq_m \text{NEmpty}_{TS}$ (where the symbol $\leq_m$ denotes many-one reducibility), it is enough to show that NEmpty_{TS} is Σ_1^0, i.e. recursively enumerable, and NEmpty_{LOC} is Σ_1^0-hard. The former claim is trivial. The latter claim follows from observing that for every Turing machine M,

$$L(M) \neq \emptyset \Leftrightarrow L(\Theta_M) \neq \emptyset,$$

and $\{M : L(M) \neq \emptyset\}$ is Σ_1^0-hard, as is well known. ■

An easy comment on the above proof is in order. For every $n \geq 1$, let $\mathcal{T}_n$ denote the family of all sets of tiles on the alphabet $\{0, 1, \ldots, n\}$. We can assume that each set of tiles Θ lies in some $\mathcal{T}_n$. For every $\Theta \in \mathcal{T}_n$, let r_Θ be the least number such that there is a picture in $L(\Theta)$, if any, of dimensions $v, h \leq r_\Theta$; otherwise let $r_\Theta = 0$. Finally let $f(n) = \max_{\Theta \in \mathcal{T}_n} r_\Theta$.

Corollary 4 *The function f is not computable. In fact f is not dominated by any computable function, i.e. for every computable function g there exist infinitely many n such that $f(n) > g(n)$.*

Proof. Immediate, as otherwise, one could decide NEmpty_{LOC} by the following procedure (which works at least for cofinitely many n): given Θ find the least n such that $\Theta \in \mathcal{T}_n$ and search for a picture $p \in L(\Theta)$ of dimensions h, v such that $h, v \leq g(n)$. The *Equality Problem* asks for an effective procedure to decide, given tiling systems T_1, T_2, whether the corresponding languages $L(T_1)$ and $L(T_2)$ are equal, in other words if the set $\mathrm{Eq}_{TS} = \{(T_1, T_2) : L(T_1) = L(T_2)\}$ is decidable. The Equality Problem for local languages is defined accordingly, and asks if the set $\mathrm{Eq}_{LOC} = \{(\Theta_1, \Theta_2) : L(\Theta_1) = L(\Theta_2)\}$ is decidable. ■

Theorem 15 Eq_{TS} *and* Eq_{LOC} *are* Π_1^0*-complete.*

Proof. As

$$(T_1, T_2) \in \mathrm{Eq}_{TS} \Leftrightarrow (\forall p)[p \in L(T_1) \Leftrightarrow p \in L(T_2)]$$

and since each $L(T)$ is decidable, it follows that Eq_{TS} is Π_1^0. On the other hand let Θ^0 be any set of tiles with empty local language. Then, in the notation of the proof of Theorem 14,

$$L(M) = \emptyset \Leftrightarrow (\Theta^0, \Theta_M) \in \mathrm{Eq}_{LOC},$$

showing that $\{M : L(M) = \emptyset\} \leq_m \mathrm{Eq}_{LOC}$, and thus Eq_{LOC} is Π_1^0-hard since so is the former set. The claims thus follow, since $\mathrm{Eq}_{LOC} \leq_m \mathrm{Eq}_{TS}$. ■

The *Infinity Problem* asks whether the set $\mathrm{Inf}_{TS} = \{T : L(T) \text{ infinite}\}$ is decidable. Again we will classify the complexity of this set, via a proof which in fact shows how to reduce $\mathrm{Inf} = \{M : L(M) \text{ infinite}\}$, the Infinity Problem for Turing machines (a well known Π_2^0-complete problem, see e.g. [56, p. 66]) to the problem $\mathrm{Inf}_{LOC} = \{\Theta : L(\Theta) \text{ infinite}\}$. We begin with observing that Inf_{TS} is Π_2^0, as

$$T \in \mathrm{Inf}_{TS} \Leftrightarrow (\forall n)(\exists p)[p \in L(T) \text{ and } v_p + h_p > n].$$

Unfortunately the coding $M \mapsto \Theta_M$ used in the previous proofs does not suffice to show that $\mathrm{Inf} \leq_m \mathrm{Inf}_{LOC}$, as it is easy to see that if $p \in L(\Theta_M)$ and p codes a computation then all pictures p' obtained expanding p to the left or to the right with columns of symbols B (or B') lie in $L(\Theta_M)$ as well. To classify the complexity of the Infinity Problem, we thus turn to Linear Bounded Automata. We recall that a Linear Bounded Automaton (LBA) is

a Turing machine which never exceeds the portion of tape that is delimited by the initial configuration. Thus if M is a deterministic LBA, then for each word $w \in L(M)$ we may assume that to the computation of M on w corresponds exactly a word in $L(\Theta_M)$ (with Θ_M containing tiles forcing every picture p to have its leftmost column filled with A, and its rightmost column filled with Ω, i. e. the tiles

$$\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \sharp & A \\ \hline \end{array}, \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \Omega & \sharp \\ \hline \end{array}, \begin{array}{|c|c|}\hline \sharp & A \\ \hline \sharp & A \\ \hline \end{array}, \begin{array}{|c|c|}\hline \Omega & \sharp \\ \hline \Omega & \sharp \\ \hline \end{array}, \begin{array}{|c|c|}\hline \sharp & A \\ \hline \sharp & \sharp \\ \hline \end{array}, \begin{array}{|c|c|}\hline \Omega & \sharp \\ \hline \sharp & \sharp \\ \hline \end{array}$$

where A and Ω are extra symbols that delimit the portion of tape used by the machine during its computation).
For the proof of the following theorems and corollary we refer to [**?**].

Theorem 16 Inf_{TS} *and* Inf_{LOC} *are* Π_2^0*-complete.*

Let now $\mathrm{Inf}^v_{LOC} = \{\Theta : (\forall w)(\exists p)[p \in L(\Theta) \text{ and } v_p > w]\}$. (The reader should also guess what Inf^v_{TS}, Inf^h_{LOC}, Inf^h_{TS} are). Then

Corollary 5 Inf^v_{LOC}, Inf^v_{TS}, Inf^h_{LOC}, Inf^h_{TS} *are* Π_2^0*-complete.*

There are however interesting problems connected with infinite languages that are decidable. For instance, if we define:

$$\mathrm{Inf}^{fin,v}_{LOC} = \{(\Theta, n) : L(\Theta) \text{ has infinitely many pictures } p \text{ with } h_p \leq n\}$$

and $\mathrm{Inf}^{h,fin}_{LOC}$ as its "horizontal" version, then we have the following:

Theorem 17 *The set* $\mathrm{Inf}^{fin,v}_{LOC}, \mathrm{Inf}^{h,fin}_{LOC}$*, and the corresponding tiling systems versions, are decidable.*

The following observation breaks the symmetry so far noticed between local languages and tiling recognizable languages in relation to decidability/undecidability questions. It is shown in [33] that $\mathrm{All}_{TS} = \{T : L(T) = \Sigma^{**}\}$ (where Σ is the alphabet of T) is undecidable, in fact Π_1^0-complete, being $\{M : L(M) \neq \emptyset\} \leq_m \mathrm{All}_{TS}$. As $L(\Theta) = \Sigma^{**}$ if and only if the set of tiles Θ contains all tiles (except the useless ones, in which the symbol # is misplaced) one easily sees that the set $\mathrm{All}_{LOC} = \{\Theta : L(\Theta) = \Sigma^{**}\}$ is decidable. A related consideration is the following. Given any set of tiles Θ,

there exists a minimal set of tiles $\Theta^m \subseteq \Theta$ such that $L(\Theta^m) = L(\Theta)$, where $\Theta^m = \{t \in \Theta : (\exists p \in L(\Theta))[t \in B_{2,2}(\hat{p})]\}$. Although it is trivial to find Θ^m if $L(\Theta) = \Sigma^{**}$, there is no effective procedure that allows to find Θ^m, given input Θ, even if one requires of such a procedure to work only for those sets Θ for which $L(\Theta) = \emptyset$, as $L(M) = \emptyset$ if and only if $L(\Theta^m_M) = \emptyset$.

Chapter 4

Tiling recognizability of various classes of polyominoes

As already disclosed in the Introduction, a research topic which recently acquired some relevance is that one of tiling recognizability of polyominoes. We notice that in fact, a polyomino can be naturally represented by a picture of a two-dimensional language. In addition, Reinhardt in [54], proved that the picture language that represents the class of the polyominoes is tiling recognizable. We limit the problem of recognizability to several classes of convex polyominoes, so our aim is to depict the recognizable languages for these classes of polyominoes. First of all we look for tiling system recognizable languages for the classes of convex polyominoes, Ferrer diagrams polyominoes, parallelogram polyominoes, directed polyominoes and column-convex polyominoes, moreover we provide a set of tiles for each languages and prove that convexity constrains can be formulated by means of local properties of the boundary of the polyomino. Then we pay attention to the class of L-convex polyominoes, which, differently from the other classes of convex polyominoes, are not defined by a "local" property on the boundary. In his Ph.D. thesis, [60], R.Vaglica conjectured that this class of polyominoes is not tiling system recognizable, we are not able to determine a tiling system for this class, but we prove that L-convex polyominoes can be recognized by union of tiling systems recognizable languages.

4.1 Polyominoes

In the plane $\mathbb{Z}\times\mathbb{Z}$ a *cell* is a unit square, and a *polyomino* is a finite connected union of cells having no cut point. Polyominoes are defined up to translations. A *column* (*row*) of a polyomino is the intersection between the polyomino and an infinite strip of cells whose centers lie on a vertical (horizontal) line.

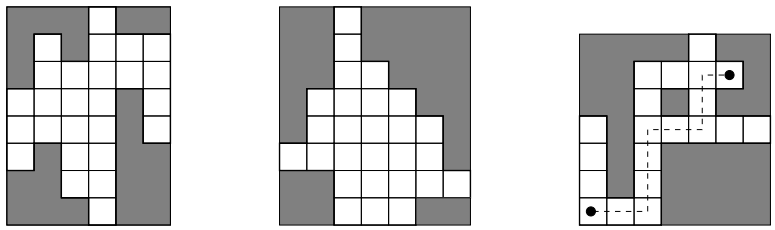


Figure 4.1: (a) column-convex polyomino; (b) a convex polyomino; (c) a directed (not convex) polyomino.

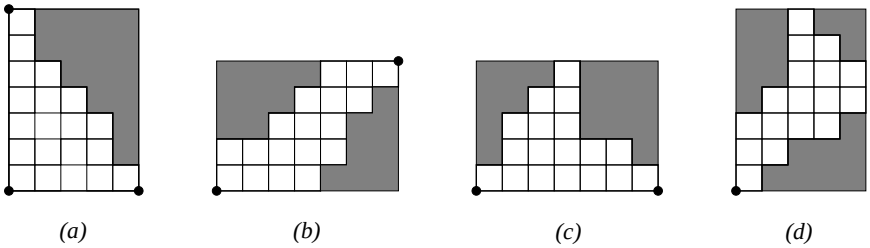
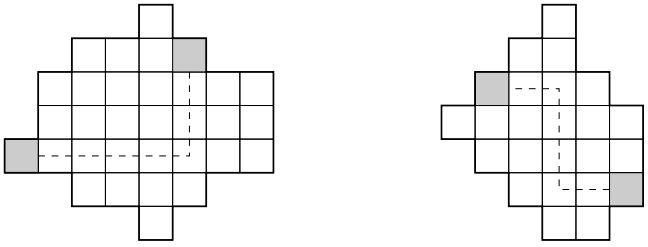


Figure 4.2: (a) A Ferrers diagram; (c) A parallelogram polyomino; (c) A stack polyomino; (d) A directed-convex polyomino.

In order to simplify many problems which are still open on the class of polyominoes, several subclasses were defined by combining two notions: the geometrical notion of *convexity*, and the notion of *directed growth*, which comes from statistical physics. A polyomino is said to be *column-convex* [*row-convex*] when its intersection with any vertical [horizontal] line is convex (Fig. 5.2 (a)). A polyomino is *convex* if it is both column and row convex (Fig. 5.2 (b)). A polyomino P is said to be *directed* when every cell of P can be reached from a distinguished cell (usually the leftmost at the lowest ordinate), by a path which is contained in P and only uses north and east

unit steps (Fig.5.2 (c)). Figure (Fig.4.2 (d)) depicts a polyomino that is both directed and convex. Moreover we can define three types of directed and convex polyominoes, i.e. the *Ferrers diagrams* (Fig.4.2 (a)), the *parallelogram polyominoes* (Fig.4.2 (b)), and the *stack polyominoes* (Fig.4.2 (c)). As Figure 4.2 shows, each of these three subsets can be characterized, in the set of convex polyominoes, by the fact that two or three vertices of the minimal bounding rectangle of the polyomino must also belong to the polyomino itself.

In a polyomino a *path* is a self-avoiding sequence of unitary steps of four types: north $N = (0, 1)$, south $S = (0, -1)$, east $E = (1, 0)$, and west $W = (-1, 0)$. We say that a path is *monotone* if it is made with steps of only two types. The authors of [14] observed that convex polyominoes have the property that every pair of cells is connected by a monotone path entirely contained in the polyomino. In this way each convex polyomino is characterized by a parameter k that represents the minimal number of changes of direction in these paths. More precisely, a convex polyomino is called k-convex if, for every pair of its cells, there is at least a monotone path with at most k changes of direction that connects them. When the value of k is 1 we have the so called L-convex polyominoes, where this terminology is motivated by the L-shape of the path that connects any two cells (see Figure 4.3).



(a) *(b)*

Figure 4.3: (a) an L-convex polyomino, and a monotone path with a single change of direction joining two of its cells; (b) a convex but not L-convex polyomino: the two highlighted cells cannot be connected by a path with only one change of direction.

This class of polyominoes has been successively considered by several points of view: in [15] it is shown that L-convex polyominoes are a well-ordering according to the sub-picture order, in [12] the authors have investigated some

tomographical aspects of this family, and have discovered that L-convex polyominoes are uniquely determined by their horizontal and vertical projections. Finally, in [13] it is proved that the number f_n of L-convex polyominoes having semi-perimeter equal to $n+2$ satisfies the recurrence relation:

$$f_n = 4f_{n-1} - 2f_{n-2}, \qquad n \geq 3, \tag{4.1}$$

with $f_0 = 1$, $f_1 = 2$, $f_2 = 7$. Successively [11] the authors have studied the problem of enumerating L-convex polyominoes by the area, and provided a coding of L-convex polyominoes in terms of words of a regular language.

4.2 Polyominoes and tiling systems

As the reader can easily argue, the class of all the two-dimensional words representing a polyomino is not a local language; indeed to form all the possible shapes of polyominoes we need that the set of tiles coincides with all the $(2,2)$ pictures of $\{0,1\}^{**}$. But such set of tiles trivially allows also pictures that do not represent polyominoes.
On the contrary we have the tiling recognizability of the family of polyominoes, as Reinhardt proves in [54]. This result however does not tell us anything about the tiling recognizability of several classes of convex polyominoes.
Let us consider the following two-dimensional languages on the alphabet $\{0,1\}$: $\mathcal{C}$ (resp. $\mathcal{F}$, $\mathcal{S}$, $\mathcal{P}$, $\mathcal{D}$, $\mathcal{V}$) is the class of pictures that represent convex polyominoes (resp. Ferrer diagrams, stack polyominoes, parallelogram polyominoes, directed-convex polyominoes, column-convex polyominoes). We will first prove that $\mathcal{C}$ is a tiling recognizable language, and, as a consequence, that $\mathcal{F}$ is a local language and that $\mathcal{S}$, $\mathcal{P}$, $\mathcal{D}$, and $\mathcal{V}$ are tiling recognizable languages.
Let P be a convex polyomino, $R(P)$ be its minimal bounding rectangle; we start by observing that four disjoint (possibly empty) sets of unit cells in $R(P)\backslash P$ are easily individuated, each of them located at one of the four vertices of $R(P)$. Let us call these sets A, B, C, and D (see Fig. 4.4 (a)). An easy check reveals that

Proposition 1 *P is convex iff for each cell (i,j) of $R(P)$ it holds*

- if $(i,j) \in A$ then both $(i-1,j) \in A$ and $(i,j-1) \in A$;

- if $(i,j) \in B$ then both $(i-1,j) \in B$ and $(i,j+1) \in B$;

- if $(i,j) \in C$ then both $(i+1,j) \in C$ and $(i,j-1) \in C$;

- if $(i,j) \in D$ then both $(i+1,j) \in D$ and $(i,j+1) \in D$

To each convex polyomino we associate a picture obtained by representing with a 1 every cell belonging to the polyomino, and with the symbol a (resp. b, c, d) every each cell in A (resp. B, C, D), as depicted in Fig. 4.4 (b). Let $\mathcal{L}_C$ be the language of these rectangles over the alphabet $\{1, a, b, c, d\}$.

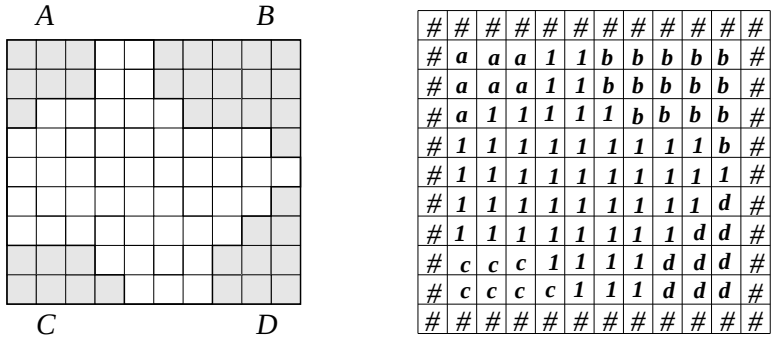


Figure 4.4: (a) A convex polyomino P individuates four disjoint sets of cells in $R(P)\backslash P$; (b) The representation of P as a word of $\mathcal{L}_C$.

Let us now consider the following sets of tiles:

$$\theta_R = \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 1 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & 1 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & \# \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 1 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 1 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & 1 \\ \hline \# & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & \# \\ \hline 1 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & 1 \\ \hline \end{array} \right\},$$

$$\theta_A = \left\{ \begin{array}{|c|c|}\hline a & a \\ \hline a & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline a & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & a \\ \hline \# & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline a & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & a \\ \hline \# & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & a \\ \hline a & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & 1 \\ \hline a & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & a \\ \hline 1 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & 1 \\ \hline 1 & 1 \\ \hline \end{array} \right\},$$

$$\theta_B = \left\{ \begin{array}{|c|c|}\hline b & b \\ \hline b & b \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline b & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline b & b \\ \hline \end{array}, \begin{array}{|c|c|}\hline b & \# \\ \hline b & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 1 & b \\ \hline \end{array}, \right.$$
$$\left. \begin{array}{|c|c|}\hline b & \# \\ \hline 1 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline b & b \\ \hline 1 & b \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & b \\ \hline 1 & b \\ \hline \end{array}, \begin{array}{|c|c|}\hline b & b \\ \hline 1 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & b \\ \hline 1 & 1 \\ \hline \end{array} \right\},$$

$$\theta_C = \left\{ \begin{array}{|c|c|}\hline c & c \\ \hline c & c \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & c \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline c & c \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & c \\ \hline \# & c \\ \hline \end{array}, \begin{array}{|c|c|}\hline c & 1 \\ \hline \# & \# \\ \hline \end{array}, \right.$$
$$\left. \begin{array}{|c|c|}\hline \# & 1 \\ \hline \# & c \\ \hline \end{array}, \begin{array}{|c|c|}\hline c & 1 \\ \hline c & c \\ \hline \end{array}, \begin{array}{|c|c|}\hline c & 1 \\ \hline c & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 1 \\ \hline c & c \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 1 \\ \hline c & 1 \\ \hline \end{array} \right\},$$

$$\theta_D = \left\{ \begin{array}{|c|c|}\hline d & d \\ \hline d & d \\ \hline \end{array}, \begin{array}{|c|c|}\hline d & \# \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline d & \# \\ \hline d & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline d & d \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & d \\ \hline \# & \# \\ \hline \end{array}, \right.$$
$$\left. \begin{array}{|c|c|}\hline 1 & \# \\ \hline d & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & d \\ \hline d & d \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & d \\ \hline 1 & d \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 1 \\ \hline d & d \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & d \\ \hline \end{array} \right\}.$$

It is easy to prove that the sets θ_A, θ_B, θ_C, and θ_D realize the conditions of Proposition 1 with respect to the cells of A, B, C, and D, respectively, and so, together with θ_R which characterizes the internal part of P, they allow the following:

Theorem 18 $\mathcal{L}_C$ is a local language over the alphabet $\Sigma_C = \{1, a, b, c, d\}$, and $\mathcal{L}_C = L\left(\theta_R \cup \theta_A \cup \theta_B \cup \theta_C \cup \theta_D\right)$.

Proof.

- ($\subseteq$) Immediate.
- ($\supseteq$) let $P \in \theta_R \cup \theta_A \cup \theta_B \cup \theta_C \cup \theta_D$. We prove that, for each cell $(i, j) \in R(P)$, if $(i, j) \in A$, then both $(i-1, j) \in A$ and $(i, j-1) \in A$. Let us proceed by contradiction assuming that there exists a cell $(i_0, j_0) \in R(P)$ such that $(i_0, j_0) \notin A$ and $(i_0 - 1, j_0) \in A$ (if we assume that $(i_0, j_0 - 1) \in A$ a similar reasoning holds). Since $P \in L(\theta_A ...)$, then there exist a tile

$$uu \in \theta_R \cup \theta_A \cup \theta_B \cup \theta_C \cup \theta_D$$

such that $X \neq a$, and $X, Y, Z \in \{1, a, b, c\}$, a contradiction.

A similar reasoning can be used to prove that, for each cell $(i,j) \in R(P)$,

- if $(i,j) \in B$, then both $(i-1,j) \in B$ and $(i,j+1) \in B$;

- if $(i,j) \in C$, then both $(i+1,j) \in C$ and $(i,j-1) \in C$;

- if $(i,j) \in D$, then both $(i+1,j) \in D$ and $(i,j+1) \in D$,

and, using Proposition 1, the thesis. ■

Now, defining the projection $\pi_C : \Sigma_C \to \{0,1\}$, such that $\pi_C(a) = \pi_C(b) = \pi_C(c) = \pi_C(d) = 0$, $\pi_C(1) = 1$, we have that $\pi_C(\mathcal{L}_C) = \mathcal{C}$. Therefore $\mathcal{C}$ is tiling recognizable.
In a similar way we can prove the other statements.
Let $\mathcal{L}_F = L(\theta_R \cup \theta_B)$; since easily we have that $\mathcal{L}_F = \mathcal{F}$, then $\mathcal{F}$ is a local language. Furthermore, let us consider the following local languages:

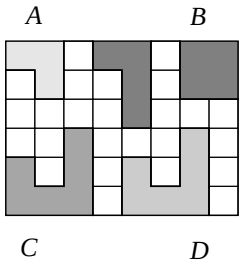
- $\mathcal{L}_S = L(\theta_R \cup \theta_A \cup \theta_B)$, over $\Sigma_S = \{1,a,b\}$;
- $\mathcal{L}_P = L(\theta_R \cup \theta_A \cup \theta_D)$, over $\Sigma_P = \{1,a,d\}$;
- $\mathcal{L}_D = L(\theta_R \cup \theta_A \cup \theta_B \cup \theta_D)$, over $\Sigma_D = \{1,a,b,d\}$,

and the projections:

- $\pi_S : \Sigma_S \to \{0,1\}$, such that $\pi_S(a) = \pi_S(b) = 0$, $\pi_S(1) = 1$;
- $\pi_P : \Sigma_P \to \{0,1\}$, such that $\pi_P(a) = \pi_P(d) = 0$, $\pi_P(1) = 1$;
- $\pi_D : \Sigma_D \to \{0,1\}$, such that $\pi_D(a) = \pi_D(b) = \pi_D(d) = 0$, $\pi_D(1) = 1$,

we finally have that:

- $\pi_S(\mathcal{L}_S) = \mathcal{S}$, thus $\mathcal{S}$ is tiling recognizable;
- $\pi_P(\mathcal{L}_P) = \mathcal{P}$, thus $\mathcal{P}$ is tiling recognizable;
- $\pi_D(\mathcal{L}_D) = \mathcal{D}$, thus $\mathcal{D}$ is tiling recognizable.



(a)

#	#	#	#	#	#	#	#	#	#
#	*a*	*a*	*1*	*b*	*b*	*1*	*b*	*b*	#
#	*1*	*a*	*1*	*1*	*b*	*1*	*b*	*b*	#
#	*1*	*1*	*1*	*1*	*b*	*1*	*1*	*1*	#
#	*1*	*1*	*c*	*1*	*1*	*1*	*d*	*1*	#
#	*c*	*1*	*c*	*1*	*d*	*1*	*d*	*1*	#
#	*c*	*c*	*c*	*1*	*d*	*d*	*d*	*1*	#
#	#	#	#	#	#	#	#	#	#

(b)

Figure 4.5: (a) A column-convex polyomino P individuates four disjoint sets of cells in $R(P)\backslash P$; (b) The representation of P as a word of $\mathcal{L}_V$.

The proof that $\mathcal{V}$ is tiling recognizable resembles the previous proofs. Let P be a column-convex polyomino, and $R(P)$ its minimal bounding rectangle; two disjoint (possibly empty) sets of unit cells in $R(P)\backslash P$ can now be easily individuated: one comprehends the cells above P, and the other comprehends the cells below P. Each of these two zones is further divided into two sets: the leftmost set of the upper [resp. lower] zone is still indicated by A [resp. C], and its remaining part by B [resp. D], as shown in Fig. 4.5 (a). Let us now consider the language $\mathcal{L}_V$ of rectangles over the alphabet $\{1, a, b, c, d\}$ obtained representing each convex polyomino as follows: a cell belonging to the polyomino is coded by 1, each cell in A (resp. B, C, D) is coded by a (resp. b, c, d), as depicted in Fig. 4.5 (b).

Proposition 2 $\mathcal{L}_V$ is a local language, i.e. $\mathcal{L}_V = L\left(\theta_R \cup \theta'_A \cup \theta'_B \cup \theta'_C \cup \theta'_D\right)$, where

$$\theta'_A = \theta_A \cup \left\{ \begin{array}{|c|c|}\hline a & a \\ \hline 1 & a \\ \hline \end{array} \;,\; \begin{array}{|c|c|}\hline 1 & a \\ \hline 1 & a \\ \hline \end{array} \;,\; \begin{array}{|c|c|}\hline 1 & a \\ \hline 1 & 1 \\ \hline \end{array} \right\},$$

$$\theta'_B = \theta_B \cup \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline b & 1 \\ \hline \end{array} \;,\; \begin{array}{|c|c|}\hline b & b \\ \hline b & 1 \\ \hline \end{array} \;,\; \begin{array}{|c|c|}\hline b & 1 \\ \hline b & 1 \\ \hline \end{array} \;,\; \begin{array}{|c|c|}\hline b & 1 \\ \hline 1 & 1 \\ \hline \end{array} \right\},$$

$$\theta'_C = \theta_C \cup \left\{ \begin{array}{|c|c|}\hline 1 & c \\ \hline c & c \\ \hline \end{array} \;,\; \begin{array}{|c|c|}\hline 1 & c \\ \hline 1 & c \\ \hline \end{array} \;,\; \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & c \\ \hline \end{array} \right\},$$

$$\theta'_D = \theta_D \cup \left\{ \begin{array}{|c|c|}\hline d & 1 \\ \hline \# & \# \\ \hline \end{array} \;,\; \begin{array}{|c|c|}\hline d & 1 \\ \hline d & d \\ \hline \end{array} \;,\; \begin{array}{|c|c|}\hline d & 1 \\ \hline d & 1 \\ \hline \end{array} \;,\; \begin{array}{|c|c|}\hline 1 & d \\ \hline 1 & 1 \\ \hline \end{array} \right\}.$$

Finally, we have that $\pi_V(\mathcal{L}_V) = \mathcal{V}$, where π_V is a projection from $\{1, a, b, c, d\}$

to $\{0,1\}$ defined as: $\pi_V(a) = \pi_V(b) = \pi_V(b) = \pi_V(b) = 0$, $\pi_V(1) = 1$. Thus $\mathcal{V}$ is tiling recognizable.

4.3 Construction of L-convex polyominoes using tiling system

Let us focus our attention on the two dimensional language $\mathcal{L}_{Conv}$ on the alphabet $\{0,1\}$, which represents the class of L-convex polyominoes. After recalling their characterization in terms of maximal rectangles given in [14], we proceed in studying the tiling recognizability of $\mathcal{L}_{Conv}$.
By abuse of notation, for any two polyominoes P and P' we will write $P \subseteq P'$ to mean that P is geometrically included in P'. A *rectangle*, that we denote by $[x, y]$, with $x, y \in \mathbb{N} \setminus \{0\}$, is a rectangular polyomino with x columns and y rows. We say $[x, y]$ to be *maximal* in P if

$$\forall\, [x', y'], \qquad [x, y] \subseteq [x', y'] \subseteq P \;\Rightarrow\; [x, y] = [x', y'].$$

Two rectangles $[x, y]$ and $[x'y']$ have *crossing intersection* if their intersection is a rectangle whose basis is the smallest of the two bases and whose height is the smallest of the two heights (see Fig. 4.6), i.e.

$$[x, y] \cap [x', y'] = [min(x, x'), min(y, y')]\,.$$

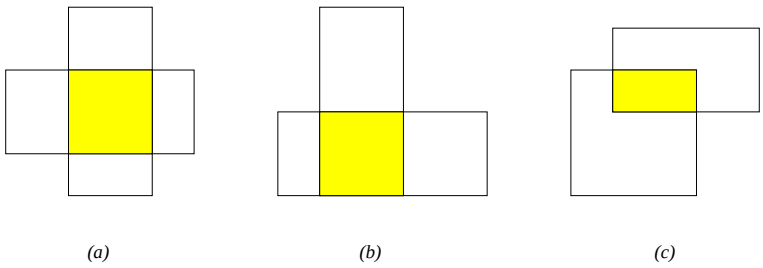
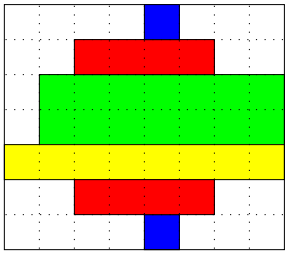


Figure 4.6: The two couples of rectangles in (a) and (b) have crossing intersection, while the couple in (c) does not.

The following theorem gives an useful characterization of L-convex polyominoes in terms of maximal rectangles [14].

Theorem 19 *A convex polyomino P is L-convex iff every pair of its maximal rectangles has crossing intersection (see Fig. 4.7).*



a	*a*	*a*	*a*	(0,3)	*b*	*b*	*b*
a	*a*	(−1,2)	(−1,2)	(0,2)	(1,2)	*b*	*b*
a	(−2,1)	(−1,1)	(−1,1)	(0,1)	(1,1)	(2,1)	(2,1)
a	(−2,1)	(−1,1)	(−1,1)	(0,1)	(1,1)	(2,1)	(2,1)
(−3,0)	(−2,0)	(−1,0)	(−1,0)	(0,0)	(1,0)	(2,0)	(2,0)
c	*c*	(−1,−1)	(−1,−1)	(0,−1)	(1,−1)	*d*	*d*
c	*c*	*c*	*c*	(0,−2)	*d*	*d*	*d*

Figure 4.7: The L-convex polyomino in Fig. 4.3 (a), and its coding by means of a picture in Γ^4; the zone O has been highlighted. The polyomino is the union of four maximal rectangles having crossing intersection.

Let $\mathcal{L}^k_{Conv}$ be the class of (pictures representing) L-convex polyominoes having at most k maximal rectangles. Clearly it holds $\bigcup_{k\geq 1} \mathcal{L}^k_{Conv} = \mathcal{L}_{Conv}$. Our aim is to prove that, for each $k \geq 2$, $\mathcal{L}^k_{Conv}$ is tiling recognizable (when $k = 1$ the assumption trivially holds).
Let us enrich the alphabet used for convex polyominoes and define

$$\Gamma_k = \{(x, y) \mid x, y \in \mathbb{Z},\ |x|, |y| < k\} \cup \{a, b, c, d\},$$

in order to represent each cell of a L-convex polyomino by means of a couple of integers in Γ_k, and each cell in A (resp. B, C, and D), by means of the symbol a (resp. b, c, and d), (get the idea from Fig. 4.7).
For each $0 \leq x_1, y_1, i, j \leq k-1$, and $-k+1 \leq x_2, y_2 \leq 0$, let us consider the following (redundant) sets of tiles:

$$\theta^k_A = \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline a & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & a \\ \hline \# & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline a & (0, y_1+1) \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & a \\ \hline \# & (x_2-1, 0) \\ \hline \end{array}, \right.$$
$$\left. \begin{array}{|c|c|}\hline (x_2-i, y_1+j) & (x_2, y_1+j) \\ \hline (x_2-i, y_1) & (x_2, y_1) \\ \hline \end{array} \right\},$$

$$\theta_B^k = \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline b & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline b & b \\ \hline \end{array}, \begin{array}{|c|c|}\hline b & \# \\ \hline b & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline (0, y_1+1) & b \\ \hline \end{array}, \begin{array}{|c|c|}\hline b & \# \\ \hline (x_1+1, 0) & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline (x_1, y_1+j) & (x_1+i, y_1+j) \\ \hline (x_1, y_1) & (x_1+i, y_1) \\ \hline \end{array} \right\},$$

$$\theta_B^k = \left\{ \begin{array}{|c|c|}\hline \# & c \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline c & c \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & c \\ \hline \# & c \\ \hline \end{array}, \begin{array}{|c|c|}\hline c & (0, y_2-1) \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & (x_2-1, 0) \\ \hline \# & c \\ \hline \end{array}, \begin{array}{|c|c|}\hline (x_2-i, y_2) & (x_2, y_2) \\ \hline (x_2-i, y_2-j) & (x_2, y_2-j) \\ \hline \end{array} \right\},$$

$$\theta_B^k = \left\{ \begin{array}{|c|c|}\hline d & \# \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline d & d \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline d & \# \\ \hline d & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline (0, y_2-1) & d \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline (x_1+1, 0) & \# \\ \hline d & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline (x_1, y_2) & (x_1+i, y_2) \\ \hline (x_1, y_2-j) & (x_1+i, y_2-j) \\ \hline \end{array} \right\},$$

$$\theta_R^k = \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & (x_2, y_1) \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline (x_2-i, y_1) & (x_2, y_1) \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline (x_1, y_1) & (x_1+i, y_1) \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline (x_1, y_1) & \# \\ \hline \end{array}, \right.$$

$$\begin{array}{|c|c|}\hline \# & (x_2, y_2) \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline (x_2-i, y_2) & (x_2, y_2) \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline (x_1, y_2) & (x_1+i, y_2) \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline (x_1, y_2) & \# \\ \hline \# & \# \\ \hline \end{array},$$

$$\begin{array}{|c|c|}\hline \# & (x_2, y_1+j) \\ \hline \# & (x_2, y_1) \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & (x_2, y_2) \\ \hline \# & (x_2, y_2-j) \\ \hline \end{array}, \begin{array}{|c|c|}\hline (x_1, y_1+j) & \# \\ \hline (x_1, y_1) & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline (x_1, y_2) & \# \\ \hline (x_1, y_2-j) & \# \\ \hline \end{array},$$

$$\begin{array}{|c|c|}\hline (x_2-i, y_1+j) & (x_2, y_1+j) \\ \hline (x_2-i, y_1) & (x_2, y_1) \\ \hline \end{array}, \begin{array}{|c|c|}\hline (x_1, y_1+j) & (x_1+i, y_1+j) \\ \hline (x_1, y_1) & (x_1+i, y_1) \\ \hline \end{array},$$

$$\left. \begin{array}{|c|c|}\hline (x_2-i, y_2) & (x_2, y_2) \\ \hline (x_2-i, y_2-j) & (x_2, y_2-j) \\ \hline \end{array}, \begin{array}{|c|c|}\hline (x_1, y_2) & (x_1+i, y_2) \\ \hline (x_1, y_2-j) & (x_1+i, y_2-j) \\ \hline \end{array} \right\},$$

with the following constraints:

- if the element (x, y) belongs to a tile in θ_A^k (resp. θ_B^k, θ_C^k, and θ_D^k), with $|x| + |y| \geq k$, then (x, y) is replaced by a (resp. b, c, and d);

- each tile in θ_A^k (resp. θ_B^k, θ_C^k, and θ_D^k) contains at least one element a (resp. b, c, and d);

- the elements (x, y) of the tiles in θ_R^k satisfy $|x| + |y| < k$.

Remark 6 *Let $\varphi : \Gamma_k \to \Gamma_C$ be the projection such that $\varphi((x, y)) = 1$, $\varphi(a) = a$, $\varphi(b) = b$, $\varphi(c) = c$, and $\varphi(d) = d$. It holds that*

$$\varphi(\theta_A^k) = \theta_A, \ \varphi(\theta_B^k) = \theta_B, \ \varphi(\theta_C^k) = \theta_C, \ \varphi(\theta_D^k) = \theta_D, \quad \text{and} \quad \varphi(\theta_R^k) = \theta_R,$$

where the sets θ_A, θ_B, θ_C, θ_D, and θ_R have been considered in the previous section.

By definition of tiling system, it follows that each element of

$$\mathcal{L}_{Conv}^k = L\left(\theta_A^k \cup \theta_B^k \cup \theta_C^k \cup \theta_D^k \cup \theta_R^k\right)$$

can be mapped into a convex polyomino, i.e. $\mathcal{L}_{Conv}^k \subset \mathcal{L}_C$, for each $k \geq 1$.

Remark 7 *Let $1 \leq h \leq k$, and let $\gamma : \Gamma_k \to \Gamma_{k-h}$ be the projection such that $\gamma(a) = a$, $\gamma(b) = b$, $\gamma(c) = c$, $\gamma(d) = d$, and*

$$\gamma\left(\,(x, y)\,\right) = \begin{cases} (x, y), & \text{if } |x| + |y| < k - h; \\ a, & \text{if } -x + y \geq k - h; \\ b, & \text{if } x + y \geq k - h; \\ c, & \text{if } -x - y \geq k - h; \\ d, & \text{if } x - y \geq k - h. \end{cases}$$

It holds that

$$\varphi(\theta_A^k) = \theta_A^{k-h}, \varphi(\theta_B^k) = \theta_B^{k-h}, \varphi(\theta_C^k) = \theta_C^{k-h}, \varphi(\theta_D^k) = \theta_D^{k-h}, \ \text{and} \ \varphi(\theta_R^k) = \theta_R^{k-h}.$$

As an immediate consequence of Remark 7, it holds $\mathcal{L}_{Conv}^{k-h} \subset \mathcal{L}_{Conv}^k$. In practice the projection φ works on each set θ_A^k (resp. B, C, D) leaving unaltered the tiles which are also tiles of θ_X^{k-h}, and setting all the others equal to a (resp. b, c, d).

The following example will clarify the construction of the sets of tiles:

Example 15 Let us set $k = 2$. We explicitly describe the sets θ^2_A, we partially list θ^2_R (allowing redundances), and we leave θ^2_B, θ^2_C, and θ^2_D as a simple exercise:

$$\theta^2_A = \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline a & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & a \\ \hline \# & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline a & (0,1) \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline a & (-1,0) \\ \hline \end{array}, \right.$$
$$\left. \begin{array}{|c|c|}\hline a & (0,1) \\ \hline (-1,0) & (0,0) \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & (0,1) \\ \hline a & (0,1) \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & a \\ \hline (-1,0) & (-1,0) \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & a \\ \hline a & a \\ \hline \end{array} \right\},$$

$$\theta^2_R = \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & (-1,0) \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & (0,0) \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline (-1,0) & (-1,0) \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline (-1,0) & (0,0) \\ \hline \end{array}, \right.$$
$$\begin{array}{|c|c|}\hline \# & \# \\ \hline (0,0) & (0,0) \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline (0,0) & (1,0) \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline (1,0) & (1,0) \\ \hline \end{array}, \ldots,$$
$$\begin{array}{|c|c|}\hline (0,0) & (0,0) \\ \hline (0,0) & (0,0) \\ \hline \end{array}, \begin{array}{|c|c|}\hline (0,1) & (0,1) \\ \hline (0,0) & (0,0) \\ \hline \end{array}, \begin{array}{|c|c|}\hline (-1,0) & (0,0) \\ \hline (-1,0) & (0,0) \\ \hline \end{array}, \begin{array}{|c|c|}\hline (0,1) & (0,1) \\ \hline (0,1) & (0,1) \\ \hline \end{array},$$
$$\begin{array}{|c|c|}\hline (-1,0) & (-1,0) \\ \hline (-1,0) & (-1,0) \\ \hline \end{array}, \begin{array}{|c|c|}\hline (0,0) & (0,0) \\ \hline (0,0) & (0,0) \\ \hline \end{array}, \begin{array}{|c|c|}\hline (1,0) & (1,0) \\ \hline (0,0) & (0,0) \\ \hline \end{array}, \begin{array}{|c|c|}\hline (0,0) & (0,1) \\ \hline (0,0) & (0,1) \\ \hline \end{array},$$
$$\left. \begin{array}{|c|c|}\hline (0,1) & (0,1) \\ \hline (0,1) & (0,1) \\ \hline \end{array}, \begin{array}{|c|c|}\hline (1,0) & (1,0) \\ \hline (1,0) & (1,0) \\ \hline \end{array}, \begin{array}{|c|c|}\hline (0,0) & (0,0) \\ \hline (0,0) & (0,0) \\ \hline \end{array}, \ldots \right\},$$

We have finally collected all the tools for proving the main result of this section:

Theorem 20 *For each $k \geq 2$, the language $\mathcal{L}^k_{Conv}$ is tiling recognizable.*

Proof.
Let us define the tiling system $\mathcal{T} = \big(\{0,1\}, \Gamma_k, \theta^k, \pi\big)$, with

$$\theta^k = \theta^k_A \cup \theta^k_B \cup \theta^k_C \cup \theta^k_D \cup \theta^k_R,$$

and $\pi : \Gamma_k \to \{0,1\}$, such that $\pi(a) = \pi(b) = \pi(c) = \pi(d) = 0$, and, for each couple $(x,y) \in \Gamma_k$, $\pi((x,y)) = 1$.
The thesis is achieved after proving:

a) if P is a L-convex containing at most k maximal rectangles, then there exists an element in $L(\theta^k)$ which represents it;

b) each element of $L(\theta^k)$ represents a L-convex polyomino;

c) each element of $L(\theta^k)$ represents a L-convex polyomino having k maximal rectangles at most.

a) By Remark 7, we assume the polyomino P to contain k maximal rectangles, and we describe how to represent it by means of a picture on Γ^k. Let $r_0 < r_1 < \cdots < r_{k-1}$ be the maximal rectangles of P ordered according to the length of the basis.

To each cell of $r_h \cap r_{h+1} \cap \cdots \cap r_{h'}$, we associate the element $(x, y) \in \Gamma^k$, such that $|x| = h$ and $|y| = k - 1 - h'$. The signs of x and y are determined by the position of the cell with respect to the central zone $O = r_0 \cap r_1 \cap \cdots \cap r_{k-1}$, i.e. the sign of x (resp. y) is positive if and only if the cell is on the right of (resp. above) O.

Finally, to each cell in the zone A (resp. B, C, and D) we associate the symbol a (resp. b, c, and d). Figure 4.7 shows the correspondence between a polyomino having four maximal rectangles and a picture on Γ^4.

The check that each picture representing a L-convex polyomino with k maximal rectangles belongs to $L(\theta^k)$ is immediate.

b) Remark 6 assures that each set of cells P represented by a pictures of $L(\theta_k)$ is a convex polyomino. To prove the L-convexity of P we choose any two of its cells and we show the existence of a path connecting them having at most one change of direction. Let p be the picture on Γ_k representing P; the definition of θ^k allows us to achieve the L-convexity by simply proving that, for any two couples (x, y) and (x', y') in p, if $|x| + |y'| \geq k$ then $|x'| + |y| < k$. The constraints $|x| + |y| < k$, and $|x'| + |y'| < k$ directly lead to the thesis.

c) For any fixed k, we prove that no picture on Γ^k represents a L-convex polyomino having more than k maximal rectangles.

This result is achieved in two steps:

i) we show that the value x (resp. y) of all the elements (x, y) in each row (resp. column) of a picture on Γ^k is constant, while the sequence of the values of the y (resp. x) weakly increases;

ii) we use *i*) to show that if two elements (x_1, y_1) or (x_2, y_2) are such that $|x_1| = |x_2|$ or $|y_1| = |y_2|$, then they belong to the same maximal rectangles.

The claim *i*) directly follows from the definition of the tiles of θ^k, i.e. for each element (x, y), its left and right (resp. north and south) neighbors in each tile share the same first (resp. second) component, while the second (resp. first) component weakly increases from left to right (resp. from south to north). We call *index* of a row (resp. column) the number x (resp. y) which is common to all its elements.

The claim *ii*) needs a little bit more attention: by definition, each time two consecutive maximal rectangles r and r' intersect, there exist at least a 2×2 square having one cell in $R(P) \setminus P$, one in $r \setminus r'$, one in $r' \setminus r$, and the fourth one in $r \cap r'$. By looking at the tiles of θ^k, this situation is represented by one among the tiles (see the left part of Fig.4.8):

a	$(x, y+j)$
$(x-i, y)$	(x, y)

,

$(x, y+j)$	b
(x, y)	$(x+i, y)$

,

$(x-i, y)$	(x, y)
c	$(x, y-j)$

,

(x, y)	$(x+i, y)$
$(x, y-j)$	d

,

with $i, j > 0$. So, each time two maximal rectangles intersect both the row and the column indexes change. Let us consider two elements (x_1, y_1) and (x_2, y_2) such that $|x_1| = |x_2|$ (if $|y_1| = |y_2|$ a similar reasoning holds).

By *i*), there exist four north and south extremal elements (x_1, y_1^{up}) (x_1, y_1^{low}), (x_2, y_2^{up}), and (x_2, y_2^{low}), such that the first two elements lie in the same column of (x_1, y_1), and the last two elements lie in the same column of (x_2, y_2) (see the right part of Fig.4.8).

Let us proceed by contradiction and assume that $y_1^{up} \neq y_2^{up}$ (resp. $y_1^{down} \neq y_2^{down}$). By *i*), the rows where (x_1, y_1^{up}) and (x_2, y_2^{up}) (resp. (x_1, y_1^{down}) and (x_2, y_2^{down})) lie are different. Let us assume w.l.g. that

a	*a*	*a*	*a*	(0,3)	*b*	*b*	*b*
a	*a*	(−1,2)	(−1,2)	(0,2)	(1,2)	*b*	*b*
a	(−2,1)	(−1,1)	(−1,1)	(0,1)	(1,1)	(2,1)	(2,1)
a	(−2,1)	(−1,1)	(−1,1)	(0,1)	(1,1)	(2,1)	(2,1)
(−3,0)	(−2,0)	(−1,0)	(−1,0)	(0,0)	(1,0)	(2,0)	(2,0)
c	*c*	(−1,−1)	(−1,−1)	(0,−1)	(1,−1)	*d*	*d*
c	*c*	*c*	*c*	(0,−2)	*d*	*d*	*d*

a	*a*	*a*	*a*	(0,3)	*b*	*b*	*b*
a	*a*	**(−1,2)**	(−1,2)	(0,2)	**(1,2)**	*b*	*b*
a	(−2,1)	(−1,1)	(−1,1)	(0,1)	**(1,1)**	(2,1)	(2,1)
a	(−2,1)	(−1,1)	(−1,1)	(0,1)	(1,1)	(2,1)	(2,1)
(−3,0)	(−2,0)	**(−1,0)**	(−1,0)	(0,0)	(1,0)	(2,0)	(2,0)
c	*c*	(−1,−1)	(−1,−1)	(0,−1)	(1,−1)	*d*	*d*
c	*c*	*c*	*c*	(0,−2)	*d*	*d*	*d*

Figure 4.8: A graphical explanation of the proof of part c), Theorem 20 on the picture of Figure 4.7; on the right part the elements (x_1, y_1) and (x_2, y_2) are in red, while (x_1, y_1^{up}), (x_2, y_2^{up}) are in blue.

$y_1^{up} < y_2^{up}$. Each element in the intersection of the columns having index x_1 and the rows having index y_2^{up} has value (x_1, y_2^{up}) (by definition of θ^k, since $|x_1| + |y_2^{up}| < k$), and one between the values a and b (since (x_1, y_1^{up}) is a north extremal point), and this is absurd.

So the only possibility is that $y_1^{up} = y_2^{up}$ and $y_1^{down} = y_2^{down}$, and the claim ii) holds.

From ii), it follows that the symbols of Γ^k are not sufficient to represent a L-convex polyomino having more than k maximal rectangles, and the proof of c) is complete.

■

We would like to point out that the Theorem 20 does not imply that the class $\mathcal{L}_{Conv}$ is tiling recognizable unless we admit that the tiling system can have an infinite alphabet $\Gamma = \bigcup_{k \geq 1} \Gamma^k$. Therefore the problem of establishing if the class of L-convex polyominoes is tiling recognizable is not solved yet. (we can recall, to this end, the conjecture of Vaglica [60] that this class is not recognizable.)
However, as we observed in the introduction, the statement of Theorem 20 is rather interesting since, while L-convexity is a global property of the polyomino, we are able to represent it by means of a set of local properties (i.e. the tiling system). A further effort should be made to check if other classes of polyominoes considered in literature, in particular those not defined by means of convexity constrains, can be represented by means of pictures of a tiling system.

Chapter 5

Two-dimensional languages and DNA-computation

The results that we have exposed in the previous chapter can be applied to bioinformatics, through the use of labeled Wang tiles, which derive from Wang tiles, see Figure 5.1. Wang tiles were introduced in [62, 63] and later studied in [22] in relation to problems concerning the tiling of the infinite Euclidean plane. Labeled Wang tiles were introduced in [27] in matter of recognizability of picture languages. Very recently Wang tiles are used for image generation (i.e. [18]) and DNA computing, as in [23, 61] where the authors, using labeled Wang tiles, show how information and algorithms can be encoded in biochemical systems. Essentially, E. Winfree constructs DNA "pictures", where the pictures are legal aggregations of labeled Wang tiles. Moreover in [27], S. Varricchio et al. wrote an algorithm to transform tiling systems into labeled Wang tiles. Our aim is to combine the works of Varricchio and Winfree and then to present a method to construct various classes of convex polyominoes using DNA Wang tiles. In particular, we focus on the classes of directed polyominoes and parallelogram polyominoes. Using this approach polyominoes can be viewed as a brick for investigating different machines that send a planar signal and compute on the plane using self assembling of DNA oligo-nuceotides.

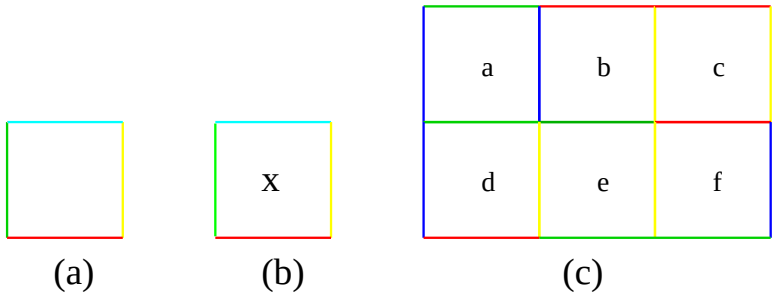


Figure 5.1: *a*) A Wang Tiles; *b*) A Labeled Wang Tiles; *c*) A small portion of the plane with a valid tiling.

5.1 Algorithm to transform Tiling Systems into labeled Wang Tiles

Now we recall some results that link Wang tiles and tiling systems. First of all we recall that a *Wang Tile* is a square in which each edge is assigned a color (or symbol) and a *labeled Wang tile* is a Wang tile in which the interior is assigned a symbol.

We represent a labeled Wang tile by: $\begin{array}{|ccc|}\hline & \alpha & \\ \beta & \boxed{x} & \gamma \\ & \delta & \\ \hline\end{array}$ where $\alpha, \beta, \gamma, \delta$ are symbols representing colors for the edges, and x is the symbol for the label.
Given a set of Wang tiles, a valid tiling requires all shared edges between tiles to have matching colors.
Let $T = \langle \Sigma, \Gamma, \Theta, \pi \rangle$ be a tiling system, we consider five subsets of tiles over the set of tiles Θ: Θ_N the tiles of the northern border, Θ_E the tiles of the eastern border, Θ_S the tiles of the southern border, Θ_W the tiles of the western border, Θ_C the corner tiles and Θ_I the set of the remaining tiles (the tiles of the interior).
Before giving the algorithm we observe the following facts:

1. If we are given a picture, obviously we want to represent it exactly how it appears both with the tiling system and the labeled Wang tiles, then its dimension (i.e. the number of its columns and rows) must be the same in both the representations. While this can be a trivial observation, it will be useful in the following.

2. The notation that we will use for the labeled Wang tiles can deceive because we will use more than one symbol to mean only one. But we will specify when it will happen.

3. We must take care to the projection π. This projection maps the alphabet Σ (i.e. the alphabet of $L(\Theta)$) in Γ (i.e. the alphabet of $L(T)$) in such a way that a word $p \in L(\Theta)$ is mapped into $p' \in L(T)$; more precisely the symbol in position (i, j) in p' is the image by π of the symbol in the position (i, j) in p. Thus we insert a label in the labeled Wang tile, at position (i, j), which is the image through π of the symbol in the position (i, j) in p (that is the symbol at position (i, j) in p').

Algorithm to transform tiling systems to labeled Wang tiles

The algorithm performs in the following three steps.

- First we consider Θ_N, these tiles are of the kind $\begin{array}{|c|c|}\hline \# & \# \\ \hline a & b \\ \hline \end{array}$; for each of them we construct the labeled Wang tile $\begin{array}{|c|}\hline \overset{B_N}{\overset{\#}{a}\ \boxed{\pi(b)}\ \overset{\#}{b}} \\ \underset{ab}{} \\ \hline \end{array}$.

  With B_N we mean that we are on the northern border, with $\overset{\#}{a}$ we mean one symbol, and the same holds for $\overset{\#}{b}$ and for ab (as specified in the observations above).

- Concerning the set of tiles Θ_W, it contains tiles of the kind $\begin{array}{|c|c|}\hline \# & a \\ \hline \# & b \\ \hline \end{array}$.

  We replace them by $\begin{array}{|c|}\hline \overset{\#a}{} \\ B_W\ \boxed{\pi(b)}\ \overset{a}{b} \\ \underset{\#b}{} \\ \hline \end{array}$.

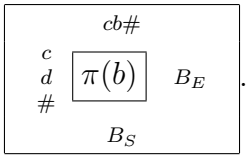
- For the corner tile west-north $\begin{array}{|c|c|}\hline \# & \# \\ \hline \# & a \\ \hline \end{array} \in \Theta_I$ we have the labeled Wang tile $\begin{array}{|c|}\hline \overset{B_N}{} \\ B_W\ \boxed{\pi(a)}\ \overset{\#}{a} \\ \underset{\#a}{} \\ \hline \end{array}$.

We can notice that the label of the labeled Wang tiles is given by the projection of the symbol which is placed on the rightmost position of the label of the south edge, and on the lowest position of the label of the east edge of the labeled Wang tile.

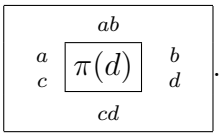
As we say in the previous observations we can not repeat the easy mechanism that we used to translate $\Theta_N, \Theta_W, \Theta_I$ also to translate the remaining subsets of tiles.

In fact, if we do this we will construct a picture with one row and one column more than the corresponding two-dimensional picture of the tiling system. To solve this trouble we use a trick to contract the two southern rows into a unique row and the two eastern columns into one column.

- For the subset Θ_S we must translate a pair of tiles of the tiling system in one labeled Wang tile, precisely the first component of the pair is a tile of Θ_S and the second one a tile that matches with the row on its north, in order to contract the two southern rows. So, for each tile of Θ_S we translate the pair $\left(\begin{array}{|c|c|}\hline a & b \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline c & d \\ \hline a & b \\ \hline \end{array} \right)$ with the labeled Wang tile $\begin{array}{ccc} & cd & \\ c & \boxed{\pi(b)} & d \\ a & & b \\ \# & B_S & \# \end{array}$.
- For the tiles of the eastern border we repeat an analogous of the previous reasoning in order to contract the two eastern columns: the generic pair of tiles $\left(\begin{array}{|c|c|}\hline a & \# \\ \hline b & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline c & a \\ \hline d & b \\ \hline \end{array} \right)$ is translated with the labeled Wang tile $\begin{array}{ccc} & ca\# & \\ c & \boxed{\pi(b)} & B_E \\ d & & \\ & db\# & \end{array}$.
- For the N-E corner tile the pair $\left(\begin{array}{|c|c|}\hline \# & \# \\ \hline a & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline b & a \\ \hline \end{array} \right)$ is translated into the labeled Wang tile $\begin{array}{ccc} & B_N & \\ \# & \boxed{\pi(a)} & B_E \\ b & & \\ & ba\# & \end{array}$.
- For the S-W corner tile the pair $\left(\begin{array}{|c|c|}\hline \# & a \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & b \\ \hline \# & a \\ \hline \end{array} \right)$ is translated into the labeled Wang tile $\begin{array}{ccc} & \#b & \\ B_W & \boxed{\pi(a)} & b \\ & & a \\ & B_S & \# \end{array}$.
- For the E-S corner tile, the tiles $\begin{array}{|c|c|}\hline a & \# \\ \hline \# & \# \\ \hline \end{array}$, $\begin{array}{|c|c|}\hline c & b \\ \hline d & a \\ \hline \end{array}$, $\begin{array}{|c|c|}\hline d & a \\ \hline \# & \# \\ \hline \end{array}$, $\begin{array}{|c|c|}\hline b & \# \\ \hline a & \# \\ \hline \end{array}$

are translated into the labeled Wang tile 

.

- In Θ_I we have the tiles of the kind $\begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline \end{array}$ that we translate with the labeled Wang tile 

.

Performing the translation we have obtained all the labeled Wang tiles necessary to represent the local language $L(\Theta)$ recognized by the tiling system T. Finally, we must take care of the projection π. We recall that π maps the alphabet Σ, that is the alphabet of $L(\Theta)$ and of the label of the labeled Wang tiles, in Γ. Then we must replace the labels of the labeled Wang tiles with the respective images through π we have completed the translation.

The reader can find the proof of the correctedness and validity of the algorithm in [27]. In the next section we make a slight improvement to the previously defined procedure, by adding a class W_{lambda}, in order to control the transformation in labeled Wang tiles of the empty polyomino and of polyominoes of sizes $1 \times n$ and $m \times 1$.

5.2 Convex polyominoes constructed on labeled Wang tiles

We already proved that many classes of convex polyominoes can be encoded as pictures of tiling recognizable two-dimensional languages. In particular we showed the coding for the class of convex polyominoes.

We are now able to encode T_c using the algorithm presented in the previous section,

and then give the set of labeled Wang tiles that represents the language $\mathcal{L}_c$.

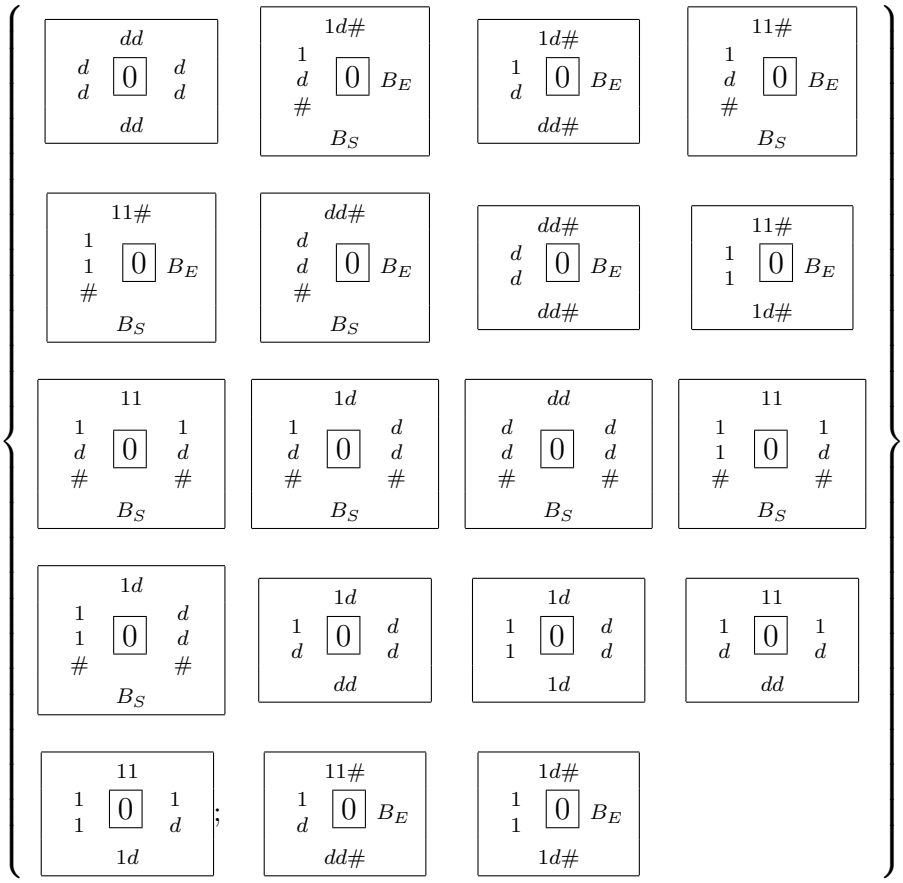
$$W_\lambda = \left\{ \begin{array}{cccc} \begin{array}{ccc} & B_N & \\ B_W & \square & B_E \\ & B_S & \end{array} & \begin{array}{ccc} & B_N & \\ B_W & \boxed{1} & B_E \\ & \#1\# & \end{array} & \begin{array}{ccc} & \#1\# & \\ B_W & \boxed{1} & B_E \\ & \#1\# & \end{array} & \begin{array}{ccc} & \#1\# & \\ B_W & \boxed{1} & B_E \\ & B_S & \end{array} \\ \begin{array}{ccc} & B_N & \\ B_W & \boxed{1} & \begin{array}{c}\#\\1\\\#\end{array} \\ & B_S & \end{array} & \begin{array}{ccc} & B_N & \\ \begin{array}{c}\#\\1\\\#\end{array} & \boxed{1} & \begin{array}{c}\#\\1\\\#\end{array} \\ & B_S & \end{array} & \begin{array}{ccc} & B_N & \\ \begin{array}{c}\#\\1\\\#\end{array} & \boxed{1} & B_E \\ & B_S & \end{array} & \end{array} \right\},$$

$$W_R = \left\{ \begin{array}{cccc} \begin{array}{ccc} & B_N & \\ B_W & \boxed{1} & B_E \\ & B_S & \end{array} & \begin{array}{ccc} & B_N & \\ B_W & \boxed{1} & \begin{array}{c}\#\\1\end{array} \\ & \#1 & \end{array} & \begin{array}{ccc} & B_N & \\ \begin{array}{c}\#\\1\end{array} & \boxed{1} & B_E \\ & 11\# & \end{array} & \begin{array}{ccc} & B_N & \\ \begin{array}{c}\#\\1\end{array} & \boxed{1} & \begin{array}{c}\#\\1\end{array} \\ & 11 & \end{array} \\ \begin{array}{ccc} & \#1 & \\ B_W & \boxed{1} & \begin{array}{c}1\\1\end{array} \\ & \#1 & \end{array} & \begin{array}{ccc} & \#1 & \\ B_w & \boxed{1} & \begin{array}{c}1\\1\\\#\end{array} \\ & B_S & \end{array} & \begin{array}{ccc} & 11 & \\ \begin{array}{c}1\\1\\\#\end{array} & \boxed{1} & \begin{array}{c}1\\1\\\#\end{array} \\ & B_S & \end{array} & \begin{array}{ccc} & 11\# & \\ \begin{array}{c}1\\1\\\#\end{array} & \boxed{1} & B_E \\ & B_S & \end{array} \\ \begin{array}{ccc} & 11\# & \\ \begin{array}{c}1\\1\end{array} & \boxed{1} & B_E \\ & 11\# & \end{array} & \begin{array}{ccc} & 11 & \\ \begin{array}{c}1\\1\end{array} & \boxed{1} & \begin{array}{c}1\\1\end{array} \\ & 11 & \end{array} & & \end{array} \right\},$$

$$W_A = \left\{ \begin{array}{cccc} \begin{array}{ccc} & aa & \\ \begin{array}{c}a\\a\end{array} & \boxed{0} & \begin{array}{c}a\\a\end{array} \\ & aa & \end{array} & \begin{array}{ccc} & B_N & \\ B_W & \boxed{0} & \begin{array}{c}\#\\a\end{array} \\ & \#a & \end{array} & \begin{array}{ccc} & B_N & \\ \begin{array}{c}\#\\a\end{array} & \boxed{0} & \begin{array}{c}\#\\a\end{array} \\ & aa & \end{array} & \begin{array}{ccc} & \#a & \\ B_W & \boxed{0} & \begin{array}{c}a\\a\end{array} \\ & \#a & \end{array} \\ \begin{array}{ccc} & B_N & \\ \begin{array}{c}\#\\a\end{array} & \boxed{0} & \begin{array}{c}\#\\1\end{array} \\ & a1 & \end{array} & \begin{array}{ccc} & \#a & \\ B_W & \boxed{1} & \begin{array}{c}a\\1\end{array} \\ & \#1 & \end{array} & \begin{array}{ccc} & aa & \\ \begin{array}{c}a\\a\end{array} & \boxed{0} & \begin{array}{c}a\\1\end{array} \\ & a1 & \end{array} & \begin{array}{ccc} & a1 & \\ \begin{array}{c}a\\a\end{array} & \boxed{1} & \begin{array}{c}1\\1\end{array} \\ & a1 & \end{array} \\ \begin{array}{ccc} & aa & \\ \begin{array}{c}a\\1\end{array} & \boxed{1} & \begin{array}{c}a\\1\end{array} \\ & 11 & \end{array} & \begin{array}{ccc} & a1 & \\ \begin{array}{c}a\\1\end{array} & \boxed{1} & \begin{array}{c}1\\1\end{array} \\ & 11 & \end{array} & & \end{array} \right\},$$

$$W_B = \left\{ \begin{array}{cccc} \begin{array}{c} bb \\ b\ \boxed{0}\ b \\ b\ \ \ b \\ bb \end{array} & \begin{array}{c} B_N \\ \#\ \boxed{0}\ B_E \\ b \\ bb\# \end{array} & \begin{array}{c} B_N \\ \#\ \boxed{0}\ B_E \\ 1 \\ 1b\# \end{array} & \begin{array}{c} 1b\# \\ 1\ \boxed{0}\ B_E \\ 1 \\ 1b\# \end{array} \\ \begin{array}{c} bb\# \\ b\ \boxed{0}\ B_E \\ 1 \\ 1b\# \end{array} & \begin{array}{c} B_N \\ \#\ \boxed{0}\ \# \\ 1 \\ 1b \end{array} & \begin{array}{c} bb\# \\ b\ \boxed{1}\ B_E \\ 1 \\ 11\# \end{array} & \begin{array}{c} 1b\# \\ 1\ \boxed{1}\ B_E \\ 1 \\ 11\# \end{array} \\ \begin{array}{c} bb \\ b\ \boxed{0}\ b \\ 1\ \ \ b \\ 1b \end{array} & \begin{array}{c} 1b \\ 1\ \boxed{0}\ b \\ 1 \\ 1b \end{array} & \begin{array}{c} bb \\ b\ \boxed{0}\ b \\ 1\ \ \ 1 \\ 11 \end{array} & \begin{array}{c} bb\# \\ b\ \boxed{1}\ B_E \\ b \\ bb\# \end{array} \\ \begin{array}{c} B_N \\ \#\ \boxed{0}\ \# \\ b \\ bb \end{array} & \begin{array}{c} 1b \\ 1\ \boxed{1}\ b \\ 1 \\ 11 \end{array} & & \end{array} \right\},$$

$$W_C = \left\{ \begin{array}{cccc} \begin{array}{c} cc \\ c\ \boxed{0}\ c \\ c\ \ \ c \\ cc \end{array} & \begin{array}{c} \#c \\ B_W\ \boxed{0}\ c \\ \ \ \ c \\ \ \ \ \# \\ B_S \end{array} & \begin{array}{c} c1 \\ c\ \boxed{0}\ 1 \\ c\ \ \ c \\ \#\ \ \ \# \\ B_S \end{array} & \begin{array}{c} 11 \\ 1\ \boxed{0}\ 1 \\ c\ \ \ c \\ \#\ \ \ \# \\ B_S \end{array} \\ \begin{array}{c} \#c \\ B_W\ \boxed{0}\ c \\ \ \ \ c \\ \#c \end{array} & \begin{array}{c} 11 \\ 1\ \boxed{1}\ 1 \\ c\ \ \ 1 \\ \#\ \ \ \# \\ B_S \end{array} & \begin{array}{c} c1 \\ c\ \boxed{1}\ 1 \\ c\ \ \ 1 \\ \#\ \ \ \# \\ B_S \end{array} & \begin{array}{c} \#1 \\ B_W\ \boxed{0}\ 1 \\ \ \ \ c \\ \ \ \ \# \\ B_S \end{array} \\ \begin{array}{c} \#1 \\ B_W\ \boxed{0}\ 1 \\ \ \ \ c \\ \#c \end{array} & \begin{array}{c} c1 \\ c\ \boxed{0}\ 1 \\ c\ \ \ c \\ cc \end{array} & \begin{array}{c} c1 \\ c\ \boxed{1}\ 1 \\ c\ \ \ 1 \\ c1 \end{array} & \begin{array}{c} 11 \\ 1\ \boxed{0}\ 1 \\ c\ \ \ c \\ cc \end{array} \\ \begin{array}{c} 11 \\ 1\ \boxed{1}\ 1 \\ c\ \ \ 1 \\ c1 \end{array} & \begin{array}{c} cc \\ c\ \boxed{0}\ c \\ c\ \ \ c \\ \#\ \ \ \# \\ B_S \end{array} & & \end{array} \right\},$$

$W_D =$ 

To summarize, convex polyominoes are generated on the plane using the set of labeled Wang tiles $W_{Conv} = W_\lambda \cup W_R \cup W_A \cup W_B \cup W_C \cup W_D$ where:

i. W_λ generates rectangles of dimensions 0, $1 \times n$ and $m \times 1$;

ii. W_R generates the other rectangles;

iii. W_A generates the upper right side of the exterior of the polyomino (side A in the picture Fig. 4.4), W_B the upper left side, W_C the lower right side and W_D the lower left side.

5.3 Example of Parallelogram polyomino on labeled Wang tiles

A refinement of the algorithm we have proposed is the encoding of parallelogram polyominoes using labeled Wang tiles. The utility of such encoding is for instance to simulate a planar signal. So, let us denote by $\mathcal{L}_P$ the recognizable two-dimensional language that represents parallelogram polyominoes. Then –applying again our algorithm– we can translate $\mathcal{L}_P$ into a set W_P of labeled Wang tiles. More explicitly this set of Wang tiles is given by $W_P = W_\lambda \cup W_R \cup W_A \cup W_C$, where W_λ, W_R, W_A, and W_C have been defined in the previous section.

Just to give an example, we show a parallelogram polyomino, the two-dimensional word coming from $\mathcal{L}_P$, its projection by π (where $\pi(a) = \pi(c) = 0, \pi(1) = 1$) and its encoding by means of labeled Wang tiles.

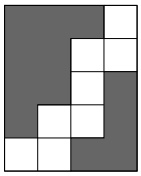


Figure 5.2: A directed (parallelogram) polyomino

#	#	#	#	#	#
#	a	a	a	1	#
#	a	a	1	1	#
#	a	a	1	c	#
#	a	1	1	c	#
#	1	1	c	c	#
#	#	#	#	#	#

$\overrightarrow{\pi}$

#	#	#	#	#	#
#	0	0	0	1	#
#	0	0	1	1	#
#	0	0	1	0	#
#	0	1	1	0	#
#	1	1	0	0	#
#	#	#	#	#	#

Figure 5.3: The two dimensional word that represents the polyomino in Figure 5.2, and its projection by π.

Similarly, using labeled Wang tiles, we can construct the following families of convex polyominoes:

- directed convex polyominoes, using the set $W_{DC} = W_\lambda \cup W_R \cup W_A \cup W_B \cup W_C$;

		B_N				B_N				B_N				B_N	
B_W		$\boxed{0}$	$\begin{matrix}\# \\ 0\end{matrix}$		$\begin{matrix}\# \\ 0\end{matrix}$	$\boxed{0}$	$\begin{matrix}\# \\ 0\end{matrix}$		$\begin{matrix}\# \\ 0\end{matrix}$	$\boxed{0}$	$\begin{matrix}\# \\ 0\end{matrix}$		$\begin{matrix}\# \\ 0\end{matrix}$	$\boxed{1}$	B_E
		$\#0$				00				00				$01\#$	
		$\#0$				00				00				$01\#$	
B_W		$\boxed{0}$	$\begin{matrix}0 \\ 0\end{matrix}$		$\begin{matrix}0 \\ 0\end{matrix}$	$\boxed{0}$	$\begin{matrix}0 \\ 0\end{matrix}$		$\begin{matrix}0 \\ 0\end{matrix}$	$\boxed{1}$	$\begin{matrix}0 \\ 1\end{matrix}$		$\begin{matrix}0 \\ 1\end{matrix}$	$\boxed{1}$	B_E
		$\#0$				00				01				$11\#$	
		$\#0$				00				01				$11\#$	
B_W		$\boxed{0}$	$\begin{matrix}0 \\ 0\end{matrix}$		$\begin{matrix}0 \\ 0\end{matrix}$	$\boxed{0}$	$\begin{matrix}0 \\ 0\end{matrix}$		$\begin{matrix}0 \\ 0\end{matrix}$	$\boxed{1}$	$\begin{matrix}1 \\ 1\end{matrix}$		$\begin{matrix}1 \\ 1\end{matrix}$	$\boxed{0}$	$\begin{matrix}\# \\ 0\end{matrix}$
		$\#0$				00				01				$10\#$	
		$\#0$				00				01				$10\#$	
B_W		$\boxed{0}$	$\begin{matrix}0 \\ 0\end{matrix}$		$\begin{matrix}0 \\ 0\end{matrix}$	$\boxed{1}$	$\begin{matrix}0 \\ 1\end{matrix}$		$\begin{matrix}0 \\ 1\end{matrix}$	$\boxed{1}$	$\begin{matrix}1 \\ 1\end{matrix}$		$\begin{matrix}1 \\ 1\end{matrix}$	$\boxed{0}$	B_E
		$\#0$				01				11				$10\#$	
		$\#0$				01				11				$10\#$	
B_W		$\boxed{1}$	$\begin{matrix}0 \\ 1 \\ \#\end{matrix}$		$\begin{matrix}0 \\ 1 \\ \#\end{matrix}$	$\boxed{1}$	$\begin{matrix}1 \\ 1 \\ \#\end{matrix}$		$\begin{matrix}1 \\ 1 \\ \#\end{matrix}$	$\boxed{0}$	$\begin{matrix}1 \\ 0 \\ \#\end{matrix}$		$\begin{matrix}1 \\ 0 \\ \#\end{matrix}$	$\boxed{0}$	B_E
		B_S				B_S				B_S				B_S	

Figure 5.4: The encoding by labeled Wang tiles of the polyomino in Figure 5.2.

- Ferrers diagrams, using the set $W_F = W_\lambda \cup W_R \cup W_B$
- stack polyominoes the set $W_S = W_\lambda \cup W_R \cup W_A \cup W_B$.

5.4 Form labeled Wang tiles to DNA Wang tiles

Using the construction of Barish, Rothemund and Winfree [3], we have a method to transform the set of labeled Wang tiles into the set of DNA Wang tiles. In fact, we are able to construct a nano structure carrying a bit (0 or 1) according to the exterior of the polyomino (labeled by 0) or the interior of the polyomino (labeled by 1). Following the construction, we could construct the set of DNA Wang tiles associated with convex polyominoes, parallelogram polyominoes, directed-convex polyominoes, stack polyominoes or Ferres diagrams.
The next step consists in the real construction of the nano structures in solution. In particular, it will be interesting to investigate the typical shape of a convex polyomino constructed using DNA tiles according to the temperature or the concentration in solution of different tiles.

To end this study, an interesting further problem is to construct a strand in order to control the perimeter of the polyomino generated by DNA Wang tiles. In each step of the construction, each picture is surrounded by the symbol #. In the last set of tiles, this symbol appears in the Wang tiles of the border. Nevertheless, we also use the symbols B_N, B_E, B_S and B_W for coding the border when we pass to DNA tiles.
Actually, it is sufficient to use 7 symbols (the 4 symbols B_N, B_E, B_S, B_W, plus 3 others) to construct a DNA strand to impose the size of the polyomino. The goal of this last construction is to force the size of the polyomino. This DNA strand begins with B_S, then in the corner we have B_{WS}, then m times B_W, then B_{WN}, then n times B_N, then B_{NE} and at the end B_E. This strand imposes that the constructed polyomino has perimeter equal to $2m + 2n$.

Of course, such a construction gives only by a theoretical point of view a convex polyomino with given perimeter, since in the real situation there can be errors in the self-assembling. In some future work it would be interesting to study in solution the average number of errors concerning the perimeter of the polyomino generated using DNA tiles.

B_{WN}	B_N	B_N	B_N	B_N	B_{NE}
B_W					B_E
B_W					
B_W					
B_W					
B_W					
B_{WS}	B_S				

Figure 5.5: Strand that controls the perimeter of the convex polyomino.

Bibliography

[1] M. Aigner, *Combinatorial Theory.* Springer-Verlag, Berlin, Heidelberg, New York, 1979

[2] M. Anselmo, D. Giammarresi, M. Madonia, A. Restivo *Unambiguous Recognizable two-dimensional languages. RAIRO - Inf. Teor. Appl.*, Vol.40, 277–293 (2006).

[3] R. Barish, P. Rothemund and E. Winfree, Two computational primitives for algorithmic self-assembly: copying and counting, *Nano letters*, Vol. 5, 2586–2592 (2005).

[4] D. Beauquier, M. Nivat, On translating one polyomino to tile the plane, *Disc. Comput. Geom.* 6 (1991) 575–592.

[5] D. Beauquier, M. Nivat, Tiling the plane with one tile, *Proc. of the 6th Annual Symposium on Computational geometry (SGC'90)* (Berkeley, CA, 1990), ACM press (1990) 128–138.

[6] M. Blum, C. Hewitt, Automata on a two-dimensional tape, *IEEE Symposium on Switching and Automata Theory* (1967) 155–160.

[7] M. Bousquet-Mélou, A method for the enumeration of various classes of column-convex polygons, *Disc. Math.* 154 (1996) 1–25.

[8] M. Bousquet-Mélou, Marko Petkovsek, Walks confined in a quadrant are not always D-finite, *Theoret. Comput. Sci.* 307 (2003) 257-276.

[9] M. Bousquet-Mélou, A. Rechnitzer, The site-perimeter of bargraphs, *Advances in Applied Mathematics*, 31 (2003) 86-112.

[10] J.R. Büchi, Weak second-order arithmetic and finite automata, *Z. Math, Logic Grundlagen Math.*, Vol. 6 (1960) 66–92.

[11] G. Castiglione, A. Frosini, E. Munarini, A. Restivo, S. Rinaldi, Enumeration of L-convex polyominoes II. Bijection and area, *Proceedings of Formal Power Series and Algebraic Combinatorics*, 20-26 June 2005, Taormina (Italy) # 49.

[12] G. Castiglione, A. Frosini, A. Restivo, S. Rinaldi, A tomographical characterization of L-convex polyominoes, *Proceedings of Discrete Geometry for Computer Imagery 12th International Conference, DGCI 2005*, Eds. E. Andres, G. Damiand, P. Lienhard, Lecture Notes in Computer Science, Poitiers 115-125.

[13] G. Castiglione, A. Frosini, A. Restivo, S. Rinaldi, Enumeration of L-convex polyominoes by rows and columns, *Theor. Comp. Sci.*, 347, 336-352 (2005).

[14] G. Castiglione, A. Restivo, Reconstruction of L-convex Polyominoes, *Electronic Notes in Disc. Math.* Vol. 12, Elsevier Science (2003).

[15] G. Castiglione, A. Restivo, Ordering and Convex Polyominoes, *Machines, Computations, and Universality, 4th International Conference, MCU 2004*, M.Margenstern Eds., Saint Petersburg, Russia, Lecture Notes in Computer Science 3354 Springer (2005).

[16] J. Cervelle, Langages de Figures *Rapport de stage* Ecole Normale Supriure de Lyon, Dept de Mathmatiques et Informatique (1997).

[17] S. Chaiken, D. J. Kleitman, M. Saks and J. Shearer, Covering regions by rectangles, *SIAM J. Discr. and Alg. Meth.* 2 (1981) 394–410.

[18] M.F. Choen, J. Shade, S. Hiller, O. Deussen, Wang Tiles for image and texture generation, *ACM Transaction on Graphics* (2003).

[19] N. Chomsky and M. P. Schützenberger, The algebraic theory of context-free languages, *In P. Braffort and D. Hirschberg (ed.), Computer Programming and Formal Systems*, (1963) 118-161 North-Holland, Amsterdam.

[20] J. H. Conway, J. C. Lagarias, Tiling with polyominoes and combinatorial group theory, *J. Comb. Th. A*, 53 (1990) 183–208.

[21] S. Crespi Reghizzi, M. Pradella Tile rewriting grammars, *Proc. of the Seventh Internat. Conf. on Developments in Language Theory (DLT 2003),* Lecture Notes in Computer Science, Vol. 2710, Szeged, Hungary, July 2003, Springer, Berlin, pp. 206217.

[22] K. Culik II, An aperiodic set of 13 wang tiles, *Discrete Mathematics* 160 (1996) 245–251.

[23] M.Daley, L.Kari, DNA computing: Models and implementations, *Comments on Theoretical Biology*, vol.7, No.3, 2002, 177–198.

[24] F. De Carli, A. Frosini, S. Rinaldi, A. Sorbi Some remarks on tiling recognizable languages, *Pure Mathematics and Applications*, Vol. 16, 69-80 (2005).

[25] F. De Carli, A. Frosini, S. Rinaldi, L. Vuillon On the Tiling System Recognizability of Various Classes of Convex Polyominoes, *Annals of Combinatorics*, to appear.

[26] M. Delest, X. Viennot, Algebraic languages and polyominoes enumeration, *Theor. Comp. Sci.*, 34 (1984) 169–206.

[27] L. De Prophetis, S. Varricchio, Recognizability of rectangular pictures by Wang systems, *Journal of Automata, Languages and Combinatorics*, Vol. 2, 269 – 288 (1998).

[28] P. Flajolet, Analytic models and ambiguity of context-free languages, *Theoret. Comput. Sci.* 49 (1987) 283–309.

[29] M. Gardner, Mathematical games, *Scientific American,* (1958) Sept. 182-192, Nov. 136-142.

[30] P. G. de Gennes, Scaling concepts in polymers physics, *Cornell University Press*, (1979).

[31] D. Giammarresi, A. Restivo, Recognizable picture languages, *Proc. First International Colloquium on Parallel Image Processing*, vol.6, No. 2-3 (1992) 241–256.

[32] D. Giammarresi, A. Restivo, S. Seibert, W. Thomas, Monadic second order logic over rectangular pictures and recognizability by tiling systems, *Information and Computation*, vol.125, No. 1 (1996) 32–45.

[33] D. Giammarresi, A. Restivo, Two-dimensional languages, *Handbook of Formal Languages*, vol.3, Springer-Verlag Berlin, (A. Salomaa and G. Rozenberg Eds.) (1997) 215–267.

[34] D. Giammarresi, A. Restivo, Matrix based complexity functions and recognizable picture languages, *Logic and Automata, TLG2, (J. Flum, E. Graedel, T. Wilke Editors)* Amsterdam University Press.

[35] D. Girault-Beauquier, M. Nivat, Tiling the plane with one tile, *Proceedings of the sixth annual symposium on Computational geometry*, Berkley, California, United States June 07 - 09, 1990.

[36] S. W. Golomb, Checker boards and polyominoes, *Amer. Math. Monthly*, vol. 61, n. 10 (1954) 675-682.

[37] S. W. Golomb, Polyominoes: Puzzles, Patterns, Problems, and Packings, *Princeton Academic Press*, 1996.

[38] A. J. Guttmann, Indicators of solvability for lattice models, *Discrete Mathematics*, 217 (2000) 167–189.

[39] J.M. Hammersley, Percolation processes II: the connective constant, *Proc. Cambridge. Philos. Soc.*, 53 (1957) 642–645.

[40] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages and Computation, *Addison-Wesley*, Reading, MA (1979).

[41] K. Inoue, A. Nakamura, Some properties of two-dimensional on-line tessellation acceptors, *Information Sciences*, Vol.13, (1977) 95–121.

[42] K. Inoue, A. Nakamura, Non closure properties of two-dimensional online tessellation acceptors, *Transaction of IECE of Japan*, Vol.6, (1977) 475–476.

[43] K. Inoue, A. Takanami, A Survey of two-dimensional automata theory, *Proc. 5th Int. Meeting of Young Computer Scientists*, Lecture Notes in Computer Science 654, Springer-Verlag, Berlin (J.Dasson, J. Kelemen Eds.) (1993).

[44] I. Jensen, A parallel algorithm for the enumeration of self-avoiding polygons on the square lattice, *Journal of Physis* A, Vol. 36 5731-5745 (2003).

[45] H.R. Lewis, C.H. Papadimitriou, *Elements of the THeory of Computation*, Prentice Hall, Inc., Englewoods Cliffs, New Jersey, 1981

[46] O. Matz, Regular expressions and context-free grammars for picture languages, *Proc. of the 14th Annu. Symp. on Theoretical Aspects of Computer Science,* Lecture Notes in Computer Science, Vol. 1200, Lbeck, Germany, 27 February1 March 1997, Springer, Berlin, pp. 283–294.

[47] O. Matz, On piecewise testable, starfree, and recognizable picture languages, *Maurice Nivat, editor, Foundations of Software Science and Computation Struc- tures*, Lecture Notes in Comp. Sci. 1378 203-210. Springer (1998).

[48] M. Nivat, A. Saoudi, V.R. Dare Parallel generation of finite images, *International Journal of Pattern Recognition and Artificial Intelligence*, Vol.3, No.3-4 (1989) 279–294.

[49] M. Nivat, A. Saoudi, V.R. Dare Parallel generation of infinite images, *International Journal of Computer Math*, Vol.35, (1990) 25–42.

[50] V. Privman, N. M. Svrakic, Difference equations in statistical mechanics. I. Cluster statistics models, *J. Stat. Phys.*, 51 (1988) 1091–1110.

[51] A. Rechnitzer, Haruspicy and anisotropic generating functions, *Advances in Applied Mathematics*, 30 (2003) 228-257.

[52] A. Rechnitzer, Haruspicy 2: The self-avoiding polygon generating function is not D-finite, to appear in *J. Comb. Th., Series A*.

[53] D. H. Redelmeier, Counting polyominoes: yet another attack, *Disc. Math.*, 36 (1981) 191–203.

[54] K. Reinhardt, On some recognizable picture-languages, *Proc. of the 23th MFCS*, Springer-Verlag, No. 1450 in LNCS (L. Brim, editor) (1998) 760–770.

[55] K. Reinhardt, The #a = #b Pictures are Recognizable, *Proc. of the 18th STACS*, Springer-Verlag, Dresden, No. 2010 in LNCS (L. Brim, editor) (2001) 527–538.

[56] R. I. Soare, Recursively Enumerable Sets and Degrees, *Perspectives in Mathematical Logic, Omega Series,* Springer Verlag, Heidelberg, 1987

[57] R. P. Stanley, *Enumerative Combinatorics,* Vol.2, Cambridge University Press, Cambridge (1999).

[58] H. N. V. Temperley, Combinatorial problems suggested by the statistical mechanics of domains and of rubber-like molecules, *Phys. review* 2 103 (1956) 1–16.

[59] W. Thomas, On logics, Tilings, and Automata, *In Proc. 18th Int. Colloquium on Automata, Languages and Programming* Lecture Notes in Computer Sciences no.510, Springer-Verlag, Berlin (1991) 441–453.

[60] R. Vaglica, Reconstruction and Recognizability of Polyominoes *Ph.D. Thesis*

[61] E. Winfree, Algorithmic Self-Assembly of DNA: Theoretical Motivations and 2D Assembly Experiments, *Journal of Biomolecular Structure & Dynamics*, ISSN 0739-1102 Conversation 11, Issue #2 (2000) 0-940030-81-0 Proceedings of the Eleventh Conversation, University at Albany, SUNY, June 15-19, 1999.

[62] H. Wang, Proving theorems by pattern recognition II, *Bell SystemsTechnical Journal*, 40 (1961) 1–42.

[63] H. Wang, Games, logic, and computers, *Scientific American* November (1965), 98–106.