Two-dimensional languages

Dora Giammarresi
Dip. di Matematica Appl. e Informatica
Università Ca' Foscari di Venezia
via Torino, 155 - 30173 Venezia Mestre, Italy.
E-mail: `dora@unive.it`

Antonio Restivo
Dipartimento di Matematica e Applicazioni
Università di Palermo
via Archirafi, 34 - 90123 Palermo, Italy.
E-mail: `restivo@altair.math.unipa.it`

Contents

1 Introduction

The aim of this chapter is to generalize concepts and techniques of formal language theory to two dimensions. Informally, a two-dimensional string is called a *picture* and is defined as a rectangular array of symbols taken from a finite alphabet. A *two-dimensional language* (or *picture language*) is a set of pictures.

A generalization of formal languages to two dimensions is possible in different ways, and several formal models to recognize or generate two-dimensional objects have been proposed in the literature. These approaches were initially motivated by problems arising in the framework of pattern recognition and image processing (cf. [11, 28]), but two-dimensional patterns also appear in studies concerning cellular automata and other models of parallel computing.

In this chapter, we limit ourselves to the study of two-dimensional languages defined (recognized, generated) by finite state devices. Similarly to one-dimensional strings, this corresponds to the "ground" level of the theory: ground level means both the lower level in a hierarchy of languages and also the level on which the study of higher classes can be based. This choice is further motivated by the fact that, for the finite state case we are able to develop a coherent theory that, unifying different approaches, gives rise to a robust notion of recognizable two-dimensional language. The situation looks completely different for the higher levels of the Chomsky hierarchy, where a comparable theory is not yet known.

Before embarking on a formal treatment, we would like to mention first some of our expectations from a sound definition of finite-state recognizability for picture languages. As a primary requirement, we would like that the class of recognizable picture languages includes in some sense the class of recognizable string languages. More precisely, we would like that, when we restrict ourselves to pictures of size $(1, n)$ (or size $(n, 1)$), the definitions of recognizability for pictures and for strings respectively, coincide. Moreover we would like that the definition of recognizability for pictures comes as "natural" generalization of some corresponding definition for strings and that, of course, the new definition inherits as many as possible properties from the corresponding definition for strings.

As for as the notion of recognizability for string languages is concerned, it is well known that the same family can be defined starting from different approaches. More precisely: 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 monadic second-order formulas (Büchi's Theorem) and with the one defined in an algebraic setting (Myhill-Nerode's Theorem). An analogous "robustness" for the notion of recognizability in two dimensions is the most ambitious property we would like to get for picture languages.

Let us shortly indicate the different approaches considered in this chapter. A first natural approach is to define picture languages by means of regular expressions. The following (regular) operations are introduced for set of pictures: row and column concatenations, row and column Kleene closures and boolean operations. A regular expression is then a formula expressing how a specific picture language can be obtained from some elementary languages by regular operations. Different families of languages can be defined, depending on the choice of operations allowed to be used in the expression.

An important role in the theory of picture recognizability is played by the approach in terms of automata. The first generalization of finite-state automata to two dimensions can be attributed to M. Blum and C. Hewitt who in [1] introduced the notion of a four-way automaton moving on a two-dimensional tape as the natural extension of a one-dimensional two-way finite automaton. Unlike the finite automata for strings this corresponding model in two dimensions is not actually a powerful model: in fact some important properties are not satisfied in this model.

After this paper, much work have been done in studying properties of picture languages recognized by finite-state machines and several other models have been designed. A survey on this subject can be found in [21]. An intersting model of two-dimensional tape acceptor is the *two-dimensional on-line tessellation automaton* introduced by K. Inoue and A. Nakamura in [18]. This is defined as an infinite two-dimensional array of identical conventional finite-state automata and it is a special type of cellular automaton. Despite it is not evident that it is a generalization of a one-dimensional model, it can be easily identified to a conventional automaton when restricted to one-row (or one-column) pictures. Moreover, the family of picture languages recognized by this model of automaton satisfy many important properties.

Different systems to generate pictures using grammars have been also explored (cf. [31, 32, 33, 35, 34, 36, 29, 30, 39]). However, in the finite state case, this approach is shown to be less powerful than others.

Another possible generalization is to describe picture languages by logic formulas. Recently, W. Thomas gave a general formalism to describe graphs (and, in particular, pictures) as model theoretical structures and showed as "recognizability" corresponds to the notions of *definability on existential monadic second order logic* (cf. [38]). This is coherent with the string language recognizability theory where Büchi's Theorem holds.

In a recent proposal (cf. [13, 14]) a notion of recognizability of a set of pictures in terms of *tiling systems* is introduced. The underlying idea is to define recognizability by "projection of local properties". Informally, recognition in a tiling system is defined in terms of a finite set of square pictures of side two which correspond somehow to automaton transitions and are called "tiles". In a picture to be recognized (say over the alphabet Σ), each quadruple of positions that form a square is to be covered by a tile (with symbols say in the alphabet Γ) such that a coherent assignment of picture positions to labels in Γ is built up, and such that a projection from Γ to Σ reestablishes the considered picture. Then the tiles can be viewed as local "automaton transitions", and tiling a given picture means to construct a run of the automaton on it. In a more formal way, the tiles define a *local language* over Γ and a projection from Γ to Σ, applied to this local language, produces the language over Σ "recognized" by the tiling system.

A leading part of this chapter is devoted to showing that some of these different approaches are indeed equivalent and give rise to the same notion of finite-state recognizability for picture languages. In a specific way we prove that the following families of two-dimensional languages coincide: languages recognized by on-line tesselation automata, languages expressed by formulas of existential monadic second order logic, languages recognized by finite tiling systems, languages corresponding to regular expressions of a special type. All these results indicate that recognizability of pictures in the sense described here is a robust notion, which at the same time can be defined in terms of machine models, regular expressions, logic formulas and tiling systems.

We further show that some remarkable properties of recognizable string languages (as closure properties, iteration lemma, etc.) can be extended to the two-dimensional case. However, at the same time, we show that the notion of recognizable picture languages do not share some properties that are fundamental in the theory of recognizable string languages. In particular, differently from the case of words, the family of recognizable picture languages is not closed under complementation. Furthermore, while the emptiness problem is decidable for finite automata on words, it is undecidable in the two-dimensional case for recognizable picture languages. These results indicate that two-dimensional languages are considerably more complicated than string languages, for which non-deterministic is not more powerful than deterministic, and for which the emptiness problem is decidable.

A special role is played by picture languages over one-letter alphabet. This leads to the theory of recognizable functions. The chapter ends reporting some ideas and suggestions to extend the arguments here presented beyond finite state recognizability.

2 Preliminaries

We assume that the reader is familiar with the basic terminology and properties of the theory of one-dimensional languages as can be found for example in [17]. We will first introduce some definitions about two-dimensional languages by borrowing and extending notation from the theory of one-dimensional languages. Next, we will give formal definitions of concatenation operations between two-dimensional strings (pictures) and two-dimensional languages. The notations used can be mainly found in [21] or in [14].

Let Σ be a finite alphabet.

Definition 2.1 *A* two-dimensional string *(or a* picture *) over Σ is a two-dimensional rectangular array* *of elements of* Σ. *The set of all two-dimensional strings over* Σ *is denoted by* Σ^{**}. *A* two-dimensional language *over* Σ *is a subset of* Σ^{**}.

Given a picture $p \in \Sigma^{**}$, let $\ell_1(p)$ denote the number of rows of p and $\ell_2(p)$ denote the number of columns of p. The pair $(\ell_1(p), \ell_2(p))$ is called the *size* of the picture p. The *empty picture* is the only picture of size $(0,0)$ and it will be denoted by λ. Pictures of size $(0,n)$ or $(n,0)$ where $n > 0$ are not defined. The set of all pictures over Σ of size (m,n), with $m,n > 0$ will be indicated by $\Sigma^{m\times n}$. Furthermore, if $1 \leq i \leq \ell_1(p)$ and $1 \leq j \leq \ell_2(p)$, $p(i,j)$ or, equivalently, $p_{i,j}$ denotes the symbol in p with coordinates (i,j).

Before continuing with notations and definitions, we give some simple examples of two-dimensional languages. We will return to some of these examples later in the chapter.

Example 2.1 Let $\Sigma = \{a\}$ be a one-letter alphabet. The set of pictures of a's with three columns is a two-dimensional language over Σ. It can be formally described as

$$L = \{p \mid p \in \Sigma^{**} \text{ and } \ell_2(p) = 3\}.$$

Example 2.2 Let $\Sigma = \{a\}$ be a one-letter alphabet and let L be the subset of Σ^{**} that contains all the pictures with a shape of "squares". More formally, language L can be described as

$$L = \{p \mid p \in \Sigma^{**} \text{ and } \ell_1(p) = \ell_2(p)\}.$$

Many other simple languages can be derived from the previous one. For example, we could define "squares of odd side" over Σ. Other "languages of squares" can be defined. For instance, take a two-letter alphabet $\Gamma = \{0, 1\}$, and consider the set of squares in which all the letters in the main diagonal (i.e., in position (i,i)) are 1, whereas the remaining positions carry letter 0. Namely, pictures like the following:

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

Another "language of squares" to which we will refer in the sequel is the subset of Γ^{**} of squares of odd side-length whose "central position" (i.e., the position where the two diagonals meet) carries letter 1. Namely, pictures like the following:

1	1	0	0	1
1	0	0	0	1
1	1	**1**	1	0
1	0	0	0	1
0	0	0	1	1

Example 2.3 Let Σ be a finite alphabet. A language $L \subseteq \Sigma^{**}$ of strings can be thought of as a set of pictures consisting of one row only. More formally, L is defined as:

$$L = \{p \,|\, p \in \Sigma^{**} \text{ and } \ell_1(p) = 1\}$$

Example 2.4 Let $\Sigma = \{0, 1\}$ be an alphabet. We can define the language of pictures over Σ whose first column is equal to the last one. Then, formally L is defined as:

$$L = \{p \,|\, p(i,1) = p(i, \ell_2(p)),\ i = 1, \ldots, \ell_1(p)\}.$$

Another (similar) example is given by the language of pictures over Σ whose first column is equal to some other column. It can be formally defined as:

$$L_1 = \{p \,|\, \exists j,\, 2 \leq j \leq \ell_2(p) : \forall i = 1, \ldots, \ell_1(p)\, p(i,1) = p(i,j)\}.$$

Definition 2.2 *Let p be a picture of size (m,n). A* block *(or a* sub-picture*) of p is a picture p' that is a sub-array of p. That is, if (m',n') is the size of p', then $m' \leq m$ and $n' \leq n$ and there exist integers h,k ($h \leq m-m'$, $k \leq n-n'$) such that $p'(i,j) = p(i+h, j+k)$ for all $0 \leq i \leq m'$ and $0 \leq j \leq n'$.*

In the sequel we will need to describe scanning strategies for pictures. To this end, we identify the boundary of a picture by using special symbols. Namely, for any picture p of size (m,n), we define $\widehat{p}$ as the picture of size $(m+2, n+2)$ obtained by surrounding p with a special *boundary symbol* $\# \notin \Sigma$.

$$\widehat{p} = \begin{array}{|ccccc|}\hline \# & \# & \cdots & \# & \# \\ \# & p_{11} & \cdots & p_{1n} & \# \\ \vdots & & \ddots & & \vdots \\ \# & p_{m1} & \cdots & p_{mn} & \# \\ \# & \# & \cdots & \# & \# \\ \hline \end{array}$$

Other notions we will need will be those of *projection of a picture* and of *projection of a language*. Let Γ and Σ be two finite alphabets and $\pi : \Gamma \to \Sigma$ be a mapping to which we will refer as a *projection*.

Definition 2.3 *Let $p \in \Gamma^{**}$ be a picture. The* projection by mapping π of picture p *is the picture $p' \in \Sigma^{**}$ such that $p'(i,j) = \pi(p(i,j))$, for all $1 \leq i \leq \ell_1(p)$, $1 \leq j \leq \ell_2(p)$.*

When there is no danger of ambiguity, we will use $\pi(p)$ to indicate the projection of picture p by mapping π. The definition of projection of a picture can be extended in a natural way to sets of pictures.

Definition 2.4 *Let $L \subseteq \Gamma^{**}$ be a picture language. The* projection by mapping π of L *is the language $L' = \{p' | p' = \pi(p)\, \forall p \in L\} \subseteq \Sigma^{**}$.*

As in the case of pictures, we will indicate by $\pi(L)$ the projection of language L by mapping π.

We now define some concatenation operations between pictures and two-dimensional languages. Let p and q be two pictures over an alphabet Σ, of size (m,n) and (m',n'), $m,n,m',n' > 0$, respectively:

$$p = \begin{array}{|ccc|}\hline p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{m1} & \cdots & p_{mn} \\ \hline \end{array} \qquad q = \begin{array}{|ccc|}\hline q_{11} & \cdots & q_{1n'} \\ \vdots & \ddots & \vdots \\ q_{m'1} & \cdots & q_{m'n'} \\ \hline \end{array}$$

Definition 2.5 *The* column concatenation *of p and q (denoted by $p \mathbin{\text{⦶}} q$) is a partial operation, defined only if $m = m'$, and it is given by*

$$p \mathbin{\text{⦶}} q = \begin{array}{|ccc|ccc|}\hline p_{11} & \cdots & p_{1n} & q_{11} & \cdots & q_{1n'} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ p_{m1} & \cdots & p_{mn} & q_{m'1} & \cdots & q_{m'n'} \\ \hline \end{array}$$

Similarly, the row concatenation *of p and q (denoted by $p \ominus q$) is a partial operation, defined only if $n = n'$, and it is given by*

$$p \ominus q = \begin{array}{|ccc|}\hline p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{m1} & \cdots & p_{mn} \\ \hline q_{11} & \cdots & q_{1n'} \\ \vdots & \ddots & \vdots \\ q_{m'1} & \cdots & q_{m'n'} \\ \hline \end{array}$$

Moreover, the column and the row concatenation of p and the empty picture λ is always defined and λ is the neutral element for both the operations.

As done in the string language theory, these definitions of pictures concatenation can be extended to define concatenations between set of pictures, i.e., two-dimensional languages.

Definition 2.6 *Let L_1, L_2 be two-dimensional languages over an alphabet Σ, the* column concatenation *of L_1 and L_2 (denoted by $L_1 \obar L_2$) is defined by*

$$L_1 \obar L_2 = \{p \obar q \mid p \in L_1 \, and \, q \in L_2\}.$$

Similarly, the row concatenation *of L_1 and L_2 (denoted by $L_1 \ominus L_2$) is defined by*

$$L_1 \ominus L_2 = \{p \ominus q \mid p \in L_1 \, and \, q \in L_2\}.$$

By iterating the concatenation operations, we can define the corresponding *transitive closures* of columns and rows, which can be viewed as a sort of *"two-dimensional Kleene star"*.

Definition 2.7 *Let L be a picture language. The* column closure *of L (denoted by $L^{*\obar}$) is defined as*

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

where $L^{0\obar} = \lambda$, $L^{1\obar} = L$, $L^{n\obar} = L \obar L^{(n-1)\obar}$.
Similarly, the row closure *of L (denoted by $L^{*\ominus}$) is defined as*

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

where $L^{0\ominus} = \lambda$, $L^{1\ominus} = L$, $L^{n\ominus} = L \ominus L^{(n-1)\ominus}$.

Remark that in the consecutive applications of row and column closure respectively, the two operations commute. Thus, we define $L^{**} = (L^{*\obar})^{*\ominus} = (L^{*\ominus})^{*\obar}$: this notation is coherent with the fact that Σ^{**} denotes the set of all possible pictures over the alphabet Σ.

Using concatenation and closure operations, it is possible to express two-dimensional languages by means of simpler languages. For example, consider the two languages $L, L_1 \subseteq \Sigma^{**}$ defined in Example 2.4. Then $L_1 = L \obar \Sigma^{**}$. Moreover, language L could be obtained as $L = L_0^{*\ominus}$ where L_0 is the language of strings whose first symbol is equal to the last one viewed as a two-dimensional language of pictures of dimension $1 \times n$.

As a further example, consider the language in the Example 2.1 of pictures with three columns. The column closure of this language gives the set of pictures whose number of column is a multiple of three.

Another operation we can define on pictures is the *rotation*. Without loss of generality we define a clockwise rotation.

Definition 2.8 *Let p be a picture. The* (clockwise) rotation *of p, indicated as p^R, is defined by*

$$p^R = \begin{matrix} p_{m1} & \cdots & p_{11} \\ \vdots & \ddots & \vdots \\ p_{mn} & \cdots & p_{1n} \end{matrix}$$

The rotation of a picture can be extended in a natural way to define the rotation of a picture language. The rotation of a language L will be referred to as L^R.

We conclude this section by defining an operator that takes two one-dimensional (string) languages over the same alphabet Σ and produces a two-dimensional language over Σ.

Definition 2.9 *Let Σ be a finite alphabet and let $S_1, S_2 \subseteq \Sigma^*$ be two string languages over Σ. The* row-column combination *of S_1 and S_2 is a two-dimensional language $L = S_1 \oplus S_2 \subseteq \Sigma^{**}$ such that, a picture $p \in \Sigma^{**}$ belongs to L if and only if the strings corresponding to the rows and to the columns of p belong to S_1 and to S_2, respectively.*

As an example of application of the operator $\oplus$, consider again the language L in Example 2.4, which consists of all pictures over $\Sigma = \{0, 1\}$ whose first column is equal to the last column. Denote by $S_0 \subseteq \Sigma^*$ the string language whose first letter is equal to the last one: then $L = S_0 \oplus \Sigma^*$.

3 Regular expressions

The basic operations introduced in Section 2 can be used to obtain, starting from some elementary languages, larger families of picture languages.

Given an alphabet Σ, the empty language $\emptyset$ and every language $\{\boxed{a}\}$, with $a \in \Sigma$, are called *atomic languages* over Σ. Let us denote by $\mathcal{R}$ the following set of operations:

$$\mathcal{R} = \{\ominus, \varobar, *\ominus, *\varobar, \cup, \cap, ^c\}.$$

The elements of $\mathcal{R}$ are called *regular operations*. A language over Σ is regular if it is obtained from atomic languages by finitely many applications of regular operations. A *regular expression* is a formula expressing how a specific language is obtained from atomic languages by regular operations.

Definition 3.1 *A* regular expression *(RE) over an alphabet Σ is defined recursively as follows:*

(i) $\emptyset$ and every letter $a \in \Sigma$ are regular expressions.

(ii) If α and β are regular expressions, then $(\alpha) \cup (\beta)$, $(\alpha) \cap (\beta)$, ${}^c(\alpha)$, $(\alpha) \varobar (\beta)$, $(\alpha) \ominus (\beta)$, $(\alpha)^{*\varobar}$, $(\alpha)^{*\ominus}$ are regular expressions.

Every regular expression over Σ denotes a two-dimensional language over Σ using the standard notation: $\emptyset$ denotes the empty language, a denotes the language $\{\boxed{a}\}$, $(\alpha) \cup (\beta)$ denotes the union of the languages denoted by α and β, $(\alpha) \cap (\beta)$ denotes their intersection, $(\alpha) \varobar (\beta)$ and $(\alpha) \ominus (\beta)$ denote their column and row concatenation respectively, $(\alpha)^{*\varobar}$ and $(\alpha)^{*\ominus}$ denote the column and the row closure respectively of the language denoted by α, while ${}^c(\alpha)$ denotes its complement.

Definition 3.2 *A two-dimensional language $L \subseteq \Sigma^{**}$ is* regular *if it is denoted by a regular expression over Σ.*

The family of regular languages will be denoted by $\mathcal{L}(RE)$.

Example 3.1 Let $\Sigma = \{a, b\}$. The regular expression

$$(((a \ominus b)^{*\ominus}) \varobar ((b \ominus a)^{*\ominus}))^{*\varobar}$$

denotes the language consisting of all "chessboards" with even side-length; that is pictures of the following form:

a	b	a	b	a	b	a	b
b	a	b	a	b	a	b	a
a	b	a	b	a	b	a	b
b	a	b	a	b	a	b	a
a	b	a	b	a	b	a	b
b	a	b	a	b	a	b	a

It is interesting to consider the following subsets of the set $\mathcal{R}$ of regular operations:

$$\mathcal{R}_1 = \{\ominus, \oplus, *\ominus, *\oplus, \cup, \cap\}$$

$$\mathcal{R}_2 = \{\ominus, \oplus, \cup, \cap, ^c\}.$$

The regular expressions that contain only operations from set $\mathcal{R}_1$ are called *complementation–free* regular expressions (CFRE) while the corresponding languages are called *complementation–free regular languages*. The family of these languages is denoted by $\mathcal{L}$(CFRE). For instance, the language in Example 3.1 is in $\mathcal{L}$(CFRE).

Similarly, the regular expressions using only operations from $\mathcal{R}_2$ are called *star–free* regular expressions (SFRE). The corresponding family of languages is the family of *star–free regular languages* and it is denoted by $\mathcal{L}$(SFRE).

Example 3.2 Consider the language L consisting of all pictures over $\Sigma = \{a, b\}$ such that there exists at least a row containing two consecutive occurrences of symbol "a". L is in $\mathcal{L}$(SFRE). In fact, it can be expressed by the following star–free regular expression:

$$^c(\emptyset) \ominus (^c(\emptyset) \oplus (a \oplus a) \oplus ^c(\emptyset)) \ominus ^c(\emptyset).$$

A special role in this theory will be played by expressions that use also the projection operation. We consider in particular the following definition.

Definition 3.3 *A language L over an alphabet Σ is a* projection of a complementation–free regular language *if there exists a language L' over an alphabet Γ that is denoted by a regular expression containing operations from $\mathcal{R}_1$ only (i.e. $L' \in \mathcal{L}$(CFRE)) and a projection $\pi : \Gamma \to \Sigma$ such that $L = \pi(L')$.*

The family of languages that are projection of complementation–free regular languages will be denoted by $\mathcal{L}$(PCFRE).

4 Automata

In this section we consider two different kinds of automata that read two-dimensional tapes. The first one, called "four-way automaton", is a sequential device while the other one is a particular cellular automaton called "two-dimensional on-line tesselation automaton".

4.1 Four-way automata

One of the first definition of "recognizable picture language" was proposed by M. Blum and C. Hewitt, who in 1967 introduced a model of finite automaton that reads a two-dimensional tape (cf. [1]). A four-way finite automaton is defined as extension of the two-way automaton that recognizes strings (cf. [17]) by allowing the finite control to move in four directions: *Left, Right, Up, Down.* We give below the formal definition.

Definition 4.1 *A* non-deterministic (deterministic) four-way finite automaton*, referred as 4NFA (4DFA), is a 7-tuple $\mathcal{A} = (\Sigma, Q, \Delta, q_0, q_a, q_r, \delta)$ where:*

- *Σ is the input alphabet;*
- *Q is a finite set of states;*
- *$\Delta = \{R, L, U, D\}$ is the set of "directions";*
- *$q_0 \in Q$ is the "initial" state;*

- $q_a, q_r \in Q$ *are the "accepting" and the "rejecting" state, respectively;*
- $\delta : Q \setminus \{q_a, q_r\} \times \Sigma \longrightarrow 2^{Q\times\Delta}$ *(*$\delta : Q \setminus \{q_a, q_r\} \times \Sigma \longrightarrow Q \times \Delta$ *) is the transition function.*

In what follows, whenever there is no need to specify whether it is deterministic or non-deterministic, we refer to a four-way automaton as 4FA. As the conventional model of automaton on strings, a 4FA can be thought of as a finite control in a state of set Q that reads the input picture. The moves of the finite control depend on the transition function δ: given the actual state and the symbol in the actual position, δ outputs the new state for the control and move it by one position according to the output direction. When the finite control falls in either state q_a or state q_r, the 4FA halts (this is because no transitions are defined in those states). We use the convention that a 4FA reads pictures with borders $\widehat{p}$ and that, when the control moves in a position carring symbol #, it comes back in p immediately at the next move. Equivalently, one could assume that the control is somehow "border sensitive": namely, the control knows when it is close to the border, and never comes out of p.

A 4FA recognizes a picture $p \in \Sigma^{**}$ if, starting from the position $(1, 1)$ in the initial state, it possibly moves around and eventually halts in the accepting state q_a. Notice that, during the recognition process, a 4FA is not required to read all the positions of the input picture; moreover the finite control can come back to a given position as many times as needed. We now give some examples of picture languages recognized by 4FA.

Example 4.1 Let $\Sigma = \{0, 1\}$ be an alphabet and let $L \subseteq \Sigma^{**}$ be the language of pictures whose first column is equal to the last one (see Example 2.4). Then, L is recognized by a 4DFA that operates as follows. It scans a picture p row by row from left to right, proceeding from top to bottom, and by checking at the same time that all positions contain letters in Σ and that the leftmost letter in a row is equal to the rightmost one.

Example 4.2 Let $\Sigma = \{0, 1\}$ be an alphabet and let $L \subseteq \Sigma^{**}$ be the language of squares in Example 2.2. Then L can be recognized by a 4DFA that does the following. It scans a picture p starting from position $(1, 1)$ and moving along the diagonal (i.e., moving one step right/one step down). If it reaches the bottom-right corner then picture p is a square. In this case it does a complete scan of p to check that all positions contain letters of Σ.

Now consider the language $L_1 \subseteq \Sigma^{**}$ consisting of squares of odd side-length with "1" in the central position, that was described in Example 2.2. L_1 can be recognized by a 4NFA that operates as follows. Given a picture p, it starts scanning p from position $(1, 1)$, again moving downward along the diagonal. At some point, with this point being chosen non-deterministically, the automaton "memorizes" the letter in the diagonal and starts moving downward along the other diagonal (i.e., moving one step left/one step down). If it reaches the bottom-left corner, then p is a square of odd side-length whose central position contains the "memorized letter". If the "memorized letter" was 1, then the automaton accepts; otherwise it rejects.

The families of languages recognized by four-way non-deterministic and deterministic automata are denoted by $\mathcal{L}$(4NFA) and $\mathcal{L}$(4DFA), respectively. The main results established on these family of languages are given below (cf. [1] and [23]): they show that, despite 4FA's are a very natural generalization of finite automata to two dimensions, they do not maintain most of the important properties that finite automata have for strings.

Theorem 4.1 *$\mathcal{L}$(4DFA) is strictly included in $\mathcal{L}$(4NFA).*

Proof: We show that the language L_1 over $\Sigma = \{0, 1\}$ consisting of squares of odd side-length whose "central" position is a 1 cannot be recognized by a 4DFA. This proves the theorem, as $L_1 \in \mathcal{L}$(4NFA) (see Example 4.2).

Consider first the behavior of a 4DFA $\mathcal{A}$ when reading a block (sub-picture) b of "0"s and "1"s of size (m, m). $\mathcal{A}$ can enter the block at one of the $4m - 4$ positions of the perimeter of b being at one of the $|Q|$

states. Similarly, A can leave b at one of the $4m-4$ positions in one of the $|Q|$ states. Thus the block b defines a mapping from the $(4m-4)|Q|$ incoming pairs into $(4m-4)|Q|$ outcoming pairs. There are $((4m-4)|Q|)^{(4m-4)|Q|}$ such mappings. On the other hand, there are 2^{m^2} different blocks of size (m,m) on a two-letter alphabet. If we choose m sufficiently large, we have $((4m-4)|Q|)^{(4m-4)|Q|} < 2^{m^2}$ and then there exist two blocks that correspond to the same mapping. Let us call b_1 and b_2 such two blocks.

Now, assume by contradiction that we have a 4DFA $\mathcal{A}$ that can recognize language L_1. Without loss of generality, we assume that $\mathcal{A}$ accepts pictures in L_1 by halting in the lower right corner of the pictures. Let (i,j) be a position where b_1 and b_2 differ (i.e., $b_1(i,j)=1$ and $b_2(i,j)=0$ or vice versa). Then we construct two square pictures p_1 and p_2 of odd side-length as follows. We construct p_1 by taking b_1 and adding rows and columns of 0's to the four sides of b_1 in a way that p_1 is a squares of odd side-length whose central position coincides with position $b_1(i,j)$. Then we construct p_2 by taking p_1 and replacing the sub-picture b_1 with the block b_2. When $\mathcal{A}$ accepts p_1, it starts and ends outside b_1; hence it must accepts also p_2 but the central position of p_2 is a 0. This gives the desired contradiction. □

Theorem 4.2 *$\mathcal{L}$(4DFA) and $\mathcal{L}$(4NFA) are not closed under row and column concatenation and closure operations.*

The complete proof of the above theorem can be found in [23]. As in the proof of Theorem 4.1, the main idea behind the proof is to use combinatorial arguments to show that some particular languages are not in $\mathcal{L}$(4NFA). For example, to prove the non-closure under row concatenation, it is enough to show that the language L_1 described in Example 2.4 cannot be recognized by a 4NFA. Notice that, if we let L be the language described again in Example 2.4, $L \in \mathcal{L}$(4DFA) (as shown in Example 4.1) and $L_1 = L \oplus \Sigma^{**}$.

Regarding boolean operations between languages, the following theorem holds (cf. [21]).

Theorem 4.3 *$\mathcal{L}$(4DFA) and $\mathcal{L}$(4NFA) are closed under boolean union and intersection operations. Moreover $\mathcal{L}$(4DFA) is closed under complement.*

Notice that the question of whether the family $\mathcal{L}$(4NFA) is closed under complement is still open.

4.2 On-line tesselation automata

In Section 4.1 we have considered automata that operate sequentially by moving around on an input tape, reading one symbol at each step. In this section we deal with *cellular automata*, i.e., automata that operate on the entire tape simultaneously. Informally a two-dimensional cellular automaton (2CA for short) is an array of "cells" (identical finite state machines) each of them being in some state at any given time. The cells operate in a sequence of discrete time steps: at each step every cell changes to a (possible) new state depending on the states of its neighbours.

Here, we consider a particular model of 2CA, the *two-dimensional on-line tessellation automata* (2OTA) introduced by K. Inoue and A. Nakamura in [18]. A 2OTA is a restricted type of 2CA in which cells do not make transitions at every time-step: rather a "transition wave" passes once diagonally across the array. Each cell changes its state depending on the two neighbors to the top and to the left, respectively.

The formal definition we give here is slightly different from the one of Inoue et al. and it uses some ideas in [16].

Definition 4.2 *A* non-deterministic (deterministic) two-dimensional on-line tesselation automaton*, referred as 2OTA (2-DOTA), is completely defined by $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ where:*

- *Σ is the input alphabet;*
- *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" (or "accepting") states;*
- *$\delta : Q \times Q \times \Sigma \longrightarrow 2^Q$ ($\delta : Q \times Q \times \Sigma \longrightarrow Q$) is the transition function.*

A *run* of $\mathcal{A}$ on a picture $p \in \Sigma^{**}$ consists of associating a state (from the set Q) to each position (i, j) of p. Such state is given by the transition function δ and depends on the states already associated to positions $(i-1, j)$ and $(i, j-1)$ and on the symbol $p(i, j)$.

#	#	#	#	#	#	#	#
#							#
#							#
#					$p(i-1, j)$		#
#				$p(i, j-1)$	$p(i, j)$		#
#							#
#	#	#	#	#	#	#	#

At time $t = 0$ an initial state q_0 is associated to all the positions of the first row and of the first column of $\widehat{p}$. The computation consists of $\ell_1(p) + \ell_2(p) - 1$ steps. It starts at time $t = 1$ by reading $p(1, 1)$ and associating the state $\delta(q_0, q_0, p(1, 1))$ to position $(1, 1)$. At time $t = 2$, states are simultaneously associated to positions $(1, 2)$ and $(2, 1)$, and so on, to the next diagonals. At time $t = k$, states are simultaneously associated to each position (i, j) such that $i + j - 1 = k$. A 2OTA $\mathcal{A}$ recognizes a picture p if there exists a run of $\mathcal{A}$ on p such that the state associated to position $(\ell_1(p), \ell_2(p))$ is a final state.

We give first some examples of languages recognized by 2OTA.

Example 4.3 Let $\Sigma = \{a\}$ and let $L \subseteq \Sigma^{**}$ the language of all pictures over Σ with an odd number of columns. That is: $L = \{p \,|\, \ell_2(p) \text{ is odd }\}$.

A 2OTA can recognize pictures of L by associating states "1" and "2" to the positions of each odd and even column, respectively. A picture is accepted if positions of the rightmost column contain state "1". More formally, L is recognized by the 2OTA $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ defined as follows:

- $Q = \{0, 1, 2\}$;
- $I = \{0\}$;
- $F = \{1\}$;
- $\delta(0, 0, a) = \delta(0, 2, a) = \delta(1, 0, a) = \delta(1, 2, a) = 1$;
  $\delta(0, 1, a) = \delta(2, 1, a) = 2$.

Example 4.4 Let $\Sigma = \{a\}$ and let $L \subseteq \Sigma^{**}$ the language of all squares over Σ described in Example 2.2 A 2OTA can recognize pictures of L by associating state "1" to positions in the main diagonal and states "2" and "3" to positions above and below such diagonal, respectively. A picture is accepted if the position of the bottom-right corner contains state "1".

More formally, L is recognized by the following 2OTA $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ defined as follows:

- $Q = \{0, 1, 2, 3\}$;
- $I = \{0\}$;
- $F = \{1\}$;
- $\delta(0, 0, a) = \delta(2, 3, a) = 1$;
  $\delta(0, 1, a) = \delta(0, 2, a) = \delta(2, 1, a) = \delta(2, 2, a) = 2$;
  $\delta(1, 0, a) = \delta(3, 0, a) = \delta(1, 3, a) = \delta(3, 3, a) = 3$.

Notice that a 2OTA reduces to a standard automaton on words when we restrict it to operate on one-row pictures only.

The families of two-dimensional languages recognized by a 2OTA and a 2DOTA are denoted by $\mathcal{L}$(2OTA) and $\mathcal{L}$(2DOTA), respectively. We remark that, also in this model, deterministic automata are less powerful than non-deterministic automata. In fact the following theorem holds (see [18] for the proof).

Theorem 4.4 *The family $\mathcal{L}$(2DOTA) is strictly included in $\mathcal{L}$(2OTA).*

We now analyze properties of family $\mathcal{L}$(2OTA)(cf. [18]).

Theorem 4.5 *$\mathcal{L}$(2OTA) is closed under projection.*

Proof: Let $\Gamma$and Σ be two finite alphabets and let $\pi : \Gamma \to \Sigma$ be a projection. Let $L \subseteq \Gamma^{**}$ be a language recognized by a 2OTA $\mathcal{A} = (\Gamma, Q, I, F, \delta)$. Then, it is easy to verify that language $L' = \pi(L)$ is recognized by the 2OTA $\mathcal{A}' = (\Sigma, Q, I, F, \delta')$ where:

$$\delta'(p, q, a) = \bigcup_{b \in \Gamma : \pi(a) = b} \delta(p, q, b).$$

□

Theorem 4.6 *$\mathcal{L}$(2OTA) is closed under row and column concatenation and closure operations.*

To conclude this section, we consider comparisons among the families of two-dimensional languages defined by automata models. The following theorem, whose proof can be found in [18], shows that two-dimensional on-line tesselation automata are more powerful than four-way automata.

Theorem 4.7 *$\mathcal{L}$(4NFA) $\subset$ $\mathcal{L}$(2OTA).*

Regarding the deterministic versions of these automata, the two corresponding families of languages are not related by inclusion relations. In fact in [18] there are examples of picture languages recognized by a DFA and not recognized by any 2DOTA and vice versa.

5 Grammars

Different systems for generating pictures using grammars have been explored in the literature (cf., for example, [31, 32, 33, 35, 34, 36, 29, 30, 39]). We consider here models that consist of two sets of rewriting rules: *horizontal* and *vertical rules*, respectively, that correspond either to a context-free or regular (conventional) grammars.

These models operate by first generating a (horizontal) string σ using the horizontal rules; then generating a rectangular picture from the top row σ by applying in parallel vertical rules. These grammars actually formalize the parallel generation of two-dimensional languages. By parallel generation we mean the simultaneous applications of several production rules.

Since in this chapter we limit ourselves to finite-state models, in this section we consider right-linear grammars only. We give first a formal definition.

Definition 5.1 *A* two-dimensional right-linear grammar (2RLG) *is defined by a 7-tuple* $G = (V_h, V_v, \Sigma_I, \Sigma, S, R_h, R_v)$*, where:*

- V_h *is a finite set of* horizontal variables*;*
- V_v *is a finite set of* vertical variables*;*

- $\Sigma_I \subseteq V_v$ *is a finite set of* intermediates*;*
- Σ *is a finite set of* terminals*;*
- $S \in V_h$ *is a* starting symbol*;*
- R_h *is a finite set of* horizontal rules *of the form* $V \to AV'$ *or* $V \to A$*, where* $V, V' \in V_h$ *and* $A \in \Sigma_I$*;*
- R_v *is a finite set of* vertical rules *of the form* $W \to aW'$ *or* $W \to a$*, where* $W, W' \in V_v$ *and* $a \in \Sigma$*.*

The derivation is performed in two phases. In the first phase, the string grammar $G_h = (V_h, \Sigma_I, S, R_h)$ generates a string language $H(G)$ over the intermediate alphabet Σ_I. The strings in $H(G)$ become the top rows of the pictures to generate. During the second phase, each intermediate symbol is treated as a start symbol: the vertical generation of the columns of the pictures is done in parallel and obtained by applying the rules in R_v. Notice that the rules of R_v must be applied in parallel in order to ensure that the terminating rules (i.e., rules of the form $V_i \to a_i$) are all applied simultaneously in every column. These grammars ensure that the columns can grow only in one direction, namely, downward.

Example 5.1 Let $G = (V_h, V_v, \Sigma_I, \Sigma, S, R_h, R_v)$ be a grammar, where:
$V_h = \{S, T\}$; $V_v = \{A, B, C, D\}$;
$\Sigma_I = \{A, B\}$; $\Sigma = \{0, 1\}$;
$R_h = \{S \to AT; T \to BS; T \to B\}$;
$R_v = \{A \to 1C; C \to 0A; C \to 0; B \to 0D; D \to 1B; D \to 1\}$.
In the first phase, G generates the string language $H(G) = \{AB\}^+$. In the second phase, starting from the string of $H(G)$ considered as top rows of pictures, by application of the vertical rules in R_v, we obtain the picture language L generated by G, which is the set of "chessboard" pictures of even side-length; i.e., pictures of the following form:

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

We denote by $\mathcal{L}$(2RLG) the family of picture languages generated by two-dimensional right-linear grammars.

We now give some notations that will be used in the proof of the theorem below. A string $w \in (\Sigma \cup \{\#\})^*$ is a *standard* string if it is of the form:

$$w = u_1\#u_2\#\ldots\#u_n\#$$

with $u_i \in \Sigma^*$ for $1 \le i \le n$ and $|u_1| = |u_2| = \ldots = |u_n|$. Let $\mathcal{S}$ denotes the set of standard strings.

Given a picture $p \in \Sigma^{**}$ of size (m, n), we denote by $s(p)$ the following string over $\Sigma \cup \{\#\}$:

$$s(p) = p_{11}\ldots p_{m1}\#p_{12}\ldots p_{m2}\#\ldots\quad\ldots\#p_{1n}\ldots p_{mn}\#.$$

Given a picture language $L \subseteq \Sigma^{**}$, we denote by $s(L)$ the string language defined as $s(L) = \{s(p) \mid p \in L\}$.

The following theorem holds.

Theorem 5.1 *$\mathcal{L}$(2RLG) is strictly included in $\mathcal{L}$(4DFA).*

Proof: Let L be a picture language generated by a 2RLG. We have to show that there exists a 4DFA that recognizes L. We first remark that there exists a finite (string) automaton M that reading string of $\mathcal{S}$ only, is able to distinguish those in s(L) from those not in s(L). Indeed, we can associate to the two-dimensional grammar G the right-linear string grammar G_h, generating language $H(G)$ over Σ_I, and, for each element $A \in \Sigma_I$, a right-linear string-grammar G_A, with starting symbol A and rules in R_v, which generates a string language L_A over Σ. Let $r(L)$ be the string language over $\Sigma \cup \{\#\}$ obtained by substituting each symbol A in $H(G)$ with the corresponding language $L_A\#$. By well known results from string language theory, since $H(G)$ and all the L_A's are regular languages, also $r(L)$ is a regular language and then there exists a finite automaton M that recognizes it.

Remark that $s(L) = r(L) \cap \mathcal{S}$. This means that, if M receives as input only standard strings, i.e. strings of $\mathcal{S}$, M is able to distinguish strings belonging to $s(L)$ from strings that are not in $s(L)$.

We are now able to exhibit a 4DFA recognizing L. Such 4DFA works as follows. It scans a picture p starting from position $(1,1)$ and reading downwards the first column of p until it reaches the border in the position position $(m+1,1)$. Then, it moves to the top position of the second column of p, i.e., position $(1,2)$, and reads downwards this column, and so on until it reaches the last position of the last column, i.e., position $(m+1,n)$. In order to recognize L it suffices that the 4DFA, while scanning the picture this way, simulates the behavior of the string automaton M that recognizes the string language $r(L)$. This proves the inclusion $\mathcal{L}(\text{2RLG}) \subseteq \mathcal{L}(\text{4DFA})$.

In order to show that this inclusion is strict, it suffices to consider the language of squares over a one-letter alphabet. This language is recognized by a 4DFA (see Example 4.2), but it cannot be generated by a 2RLG. Indeed, in these grammars the horizontal and the vertical generations are independent and there is no way to control that the number of rows in a picture is equal to the number of columns. □

Several properties of family $\mathcal{L}(\text{2RLG})$ have been studied in [29], [36]. We give here, in particular, the following proposition that will be useful for the comparisons among all the families given later in the chapter. The proof is easy to infer as a consequence of the definition itself.

Proposition 5.1 *The family $\mathcal{L}$(2RLG) is closed under projection.*

The grammars presented in this section were introduced by Siromoney et al. in [32] and are usually called "matrix grammars". In [29], Nivat et al. extended this model introducing the notion of "image grammars". The basic difference between image grammars and matrix grammars is the concept of synchronization defined by a set of tables, where a table represents a set of rules that can be applied simultaneously for rewriting columns.

Another model to generate pictures that has not been considered here is the model of "array grammars" (cf. [35],[31]). An array grammar operates by replacing a subarray α with a subarray β, the same way as a string grammar replaces substrings by other substrings. However, when the two subarrays α and β do not have the same size, it is not clear how α can be replaced by β. To avoid this problem, the array grammars are usually required to be isometric.

6 Logic formulas

In a logic formalism, pictures are considered as model theoretic structures. We refer here to [15]. The notations are borrowed from [38], where general graphs are considered in the framework of relational structures. The result is actually a "translation" of the logic formalism to describe words to a formalism to describe labeled grids.

A picture p of size (m,n) over Σ can be viewed as a vertex-labeled "grid graph" with $m \cdot n$ vertices and then it can be represented as a relational structure. More formally we give the following definition.

Definition 6.1 *Let p be a picture of size (m,n). Then p can be represented by the signature $\underline{p} = (dom(p), S_1, S_2, (P_a)_{a\in\Sigma})$, where:*

- *dom*$(p) = \{1, 2, \ldots, \ell_1(p)\} \times \{1, 2, \ldots, \ell_2(p)\}$*;*

- S_1 *and* S_2 *are the* successor relations *for the two components of points of dom*(p)*, that is:* $(i, j)\ S_1\ (i+1, j)$ *and* $(i, j)\ S_2\ (i, j+1)$ *for* $1 \leq i \leq m$, $1 \leq j \leq n$*;*

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

Properties of pictures can be described by first-order and monadic second-order formulas, using first-order variables $x, y, z, x_1, x_2, \ldots$ for points of *dom*(p), i.e., positions, and monadic second-order variables $X, Y, Z, X_1, X_2, \ldots$ for sets of positions.

Atomic formulas are of the form xS_1y, xS_2y, $X(x)$, and $P_a(x)$: they are interpreted in a natural way by $(x, y) \in S_i$, $x \in X$, $x \in P_a$, respectively. *Formulas* are built up from atomic formulas by means of the Boolean connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ and the quantifiers $\exists, \forall$, applicable to first-order as well as to second-order variables. A formula without free variables is called a *sentence*.

If $\varphi(X_1, ..., X_n)$ is a formula with at most $X_1, \ldots, X_n$ occurring free in φ, p is a picture, and $Q_1, \ldots, Q_n$ are subsets of *dom*(p), we write

$$(\underline{p}, Q_1, \ldots, Q_n) \models \varphi(X_1, \ldots, X_n)$$

if p satisfies φ under the above mentioned interpretation, where Q_i is taken as interpretation of X_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 6.2 *Let* L *be a picture language.*

- L *is* Monadic Second-Order definable *if there is a monadic second-order sentence* φ *with* $L = L(\varphi)$*.*

- L *is* First-Order definable *if there is a sentence* φ *containing only first-order quantifiers (i.e., ranging over position variables only) such that* $L = L(\varphi)$*.*

- L *is* Existential Monadic Second-Order definable *if there is a sentence of the form* $\varphi = \exists X_1 \ldots \exists X_n \psi(X_1, \ldots, X_n)$*, where* ψ *contains only first-order quantifiers, such that* $L = L(\varphi)$*.*

The families of two-dimensional languages that are Monadic Second-Order, First-Order and Existential Monadic Second-Order definable will be denoted by $\mathcal{L}$(MSO), $\mathcal{L}$(FO) and $\mathcal{L}$(EMSO), respectively.

In [15] some examples of formulas defining properties of two-dimensional languages are given. Let us mention only some properties of positions and pictures which are easily described by first-order formulas. An upper border position x of a picture, i.e., a position $x = (1, j)$ for some j, is described by

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

where "t" stands for "top". Similarly, the other borders (left, right and bottom) can be described by corresponding formulas $\varphi_l(x)$, $\varphi_r(x)$, $\varphi_b(x)$. The four corner positions (top-left, top-right, bottom-left, bottom-right) are defined by appropriate conjunctions of these formulas and they will be indicated here by $\varphi_{tl}(x), \varphi_{tr}(x), \varphi_{bl}(x), \varphi_{br}(x)$, respectively.

Example 6.1 Consider the language L described in Example 2.3 of all the words, considered as pictures of size $(1, n)$, over an alphabet Σ. L is definable by the first-order sentence $\forall x \varphi_t(x)$.

Example 6.2 Let L be the language of squares over a one-letter alphabet $\Sigma = \{a\}$ described in Example 2.2. Then L can be described by an existential monadic second order formula that postulates a set of positions which (i) contains the left upper corner, (ii) is "closed under diagonal successors" (i.e., passing from (i,j) to $(i+1,j+1)$), (iii) does not hit the bottom or right border, excepting the bottom right corner. An existential monadic second-order sentence expressing this is the following:

$$\begin{array}{ll}\exists X & (\exists x(\varphi_{tl}(x) \wedge X(x)) \\ & \wedge \forall x \forall y \forall z\ (X(x) \wedge x S_1 y \wedge y S_2 z) \rightarrow X(z)) \\ & \wedge \forall x((\varphi_b(x) \vee \varphi_r(x)) \rightarrow (\neg X(x) \vee \varphi_{br}(x)))\end{array}$$

where $\varphi_{tl}(x), \varphi_b(x), \varphi_r(x)$ and $\varphi_{br}(x)$ are first-order formulas expressing the properties that x is a "top-left", "bottom", "right", "bottom-right" position, respectively.

Moreover it can be shown that the set of squares cannot be described by a first-order sentence.

7 Tiling Systems

In this section, we consider a notion of finite state recognizability for picture languages introduced recently in [13]. This notion takes as starting point a well known characterization of recognizable string languages in terms of local languages and projections. Namely, any recognizable (by means of finite automata) string language can be obtained as projection of a local string language over a larger alphabet (cf. Theorem 6.1 in [7]). Such notions can be extended in a natural way to the two dimensional case: more precisely, we define local picture languages by means of a set of square arrays of side-length two, here called "tiles", that represent the only allowed blocks of that size in the pictures of the language. Then we say that a two-dimensional language is "tiling recognizable" if it can be obtained as a projection of a local picture language.

We remark that this approach is very close to that one proposed by W. Thomas in the more general framework of graphs (cf. [38]).

7.1 Local two-dimensional languages

Given a picture p of size (m,n), let $h \leq m, k \leq n$: we denote by $B_{h,k}(p)$ the set of all blocks (or sub-pictures) of p of size (h,k). We call *tile* a square picture of size $(2,2)$.

Definition 7.1 *Let Γ be a finite alphabet. A two-dimensional language $L \subseteq \Gamma^{**}$ is* local *if there exists a finite set Θ of* tiles *over the alphabet $\Gamma \cup \{\#\}$ such that $L = \{p \in \Gamma^{**} | B_{2,2}(\widehat{p}) \subseteq \Theta\}$.*

Therefore Θ represents the set of *allowed blocks* for pictures belonging to the local language L. Given a language L, we can consider the set Θ as the set of all possible blocks of size $(2,2)$ of pictures that belong to L (when considered with the frame of # symbols). The language L is local if, given such a set Θ, we can exactly retrieve the language L. We will assume implicitely, that the empty picture λ belongs to L if and only if Θ contains the tile with four # symbols. We call the set Θ a *representation by tiles* for the local language L and write $L = L(\Theta)$.

The family of local picture languages will be denoted by LOC. We now give an example of a local two-dimensional language.

Example 7.1 Let $\Gamma = \{0,1\}$ be an alphabet and let Θ be the following set of tiles over Γ.

$$\Theta = \left\{ \begin{array}{l}
\begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & 1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 0 \\ \hline\end{array}, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 0 & 0 \\ \hline\end{array}, \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 0 \\ \hline\end{array}, \begin{array}{|c|c|}\hline \# & 1 \\ \hline \# & 0 \\ \hline\end{array}, \begin{array}{|c|c|}\hline \# & 0 \\ \hline \# & 0 \\ \hline\end{array}, \\[2ex]
\begin{array}{|c|c|}\hline 0 & 0 \\ \hline \# & \# \\ \hline\end{array}, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline \# & \# \\ \hline\end{array}, \begin{array}{|c|c|}\hline 0 & \# \\ \hline 1 & \# \\ \hline\end{array}, \begin{array}{|c|c|}\hline 0 & \# \\ \hline 0 & \# \\ \hline\end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 0 & 0 \\ \hline\end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 1 & 0 \\ \hline\end{array}, \\[2ex]
\begin{array}{|c|c|}\hline \# & \# \\ \hline \# & 1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 0 & \# \\ \hline\end{array}, \begin{array}{|c|c|}\hline \# & 0 \\ \hline \# & \# \\ \hline\end{array}, \begin{array}{|c|c|}\hline 1 & \# \\ \hline \# & \# \\ \hline\end{array}
\end{array} \right\}$$

The language $L = L(\Theta)$ is the language of squares pictures in which all main diagonal positions carry symbol 1, whereas the remaining positions carry symbol "0": this is the two-dimensional language described in Example 2.2.

Notice that the language of squares over a one-letter alphabet is not a local language because there is no "local strategy" to compare the number of rows and columns using only one symbol.

Another way to understand the notion of local two-dimensional language is to reason in terms of a computational procedure to recognize a picture: a window of size 2×2 is moved around the picture and a record is made of the blocks of size $(2, 2)$ observed through the window, regardless of the order and the number of occurrences of these blocks. A picture is "accepted" if the set of the recorded blocks is included in the given set Θ of tiles.

The definition of the family LOC extends the classic notion of a local string language to two dimensions. We recall, (cf. [7]), that a local string language is defined by means of a set of strings of length two (that contains all possible allowed factors of length two for the strings of the language) and a pair of sets of letters (that correspond to the possible beginnings and endings for the string in the language).

As a final remark, we notice that, analogously to the one-dimensional theory, we can generalize this notion of local language by taking blocks of different size. Let h, k be two positive integers: a two-dimensional language L is (h, k)-*local* when we can test whether a picture x belongs to L by checking only set $B_{h,k}(x)$ (i.e., set containing its blocks of size (h, k)). Using this more general definition, the family LOC corresponds to the family of $(2, 2)$-*local* two-dimensional languages. We will return to this generalization later.

7.2 Tiling recognizable languages

We now define two-dimensional languages using the notion of local languages introduced above and the notion of projection of a language (see Section 2). Combining these two notions, yields the definition of *tiling system*.

Definition 7.2 *A* tiling system *(TS) is 4-tuple* $\mathcal{T} = (\Sigma, \Gamma, \Theta, \pi)$, *where* Σ *and* Γ *are two finite alphabets,* Θ *is finite set of tiles over the alphabet* $\Gamma \cup \{\#\}$ *and* $\pi : \Gamma \to \Sigma$ *is a* projection.

The tiling system $\mathcal{T}$ defines ("recognizes") a language L over the alphabet Σ as follows: $L = \pi(L')$ where $L' = L(\Theta)$ is the local language over Γ corresponding to the set of tiles Θ. We write $L = L(\mathcal{T})$ and we say that L is the language recognized by $\mathcal{T}$. We will refer to the local language $L' \subseteq \Gamma^{**}$ as the *underlying local language* for L, while we will call Γ the *local alphabet*.

We say that a language $L \subseteq \Sigma^{**}$ is *recognizable by tiling systems* (or *tiling recognizable*) if there exists a tiling system $\mathcal{T} = (\Sigma, \Gamma, \Theta, \pi)$ such that $L = L(\mathcal{T})$. We denote by $\mathcal{L}(TS)$ the family of all two-dimensional languages recognizable by tiling systems. In other words $L \in \mathcal{L}(TS)$ if it is a projection of some local language.

We show first an example.

Example 7.2 Let $\Sigma = \{a\}$ be a one-letter alphabet and let L be the language of squares over Σ, that is $L = \{x \mid \ell_1(x) = \ell_2(x)\} \subseteq \Sigma^{**}$. Language L is recognizable by a TS. In fact we can take as underlying local language L', the one in Example 7.1 (i.e., the language of squares over the alphabet $\Gamma = \{0, 1\}$ with 1's in the main diagonal and 0's in the other positions) and apply the projection $\pi : \Gamma \to \Sigma$ such that $\pi(0) = \pi(1) = a$. It is easy to see that $L = \pi(L')$.

A similar argument can be used to prove that the set of squares over any alphabet is recognizable by a TS. For example, if $\Sigma = \{a, b\}$, we take a local language over the alphabet $\Gamma = \{a_0, b_0, a_1, b_1\}$ such that in the main diagonal positions there can be only a_1 and b_1 while the remaining positions can contain only a_0 and b_0. Then we use a projection $\pi : \Gamma \to \Sigma$ that "restores" each a_i to a and each b_i to b for $i = 0, 1$.

Remark: A tiling system $(\Sigma, \Gamma, \Theta, \pi)$ for a picture language is in some sense a generalization to two dimensions of an automaton that recognizes a string language. Indeed, in the one-dimensional case, the 4-tuple $(\Sigma, \Gamma, \Theta, \pi)$ corresponds exactly to the automaton. The alphabet Γ is in a one-to-one correspondence with the edges of the state-graph of the automaton. The set Θ (that in one dimension is a set of strings of length two on the alphabet $\Gamma \cup \{\#\}$) describes the edges adjacencies and thus provides a description of the state-graph. The mapping $\pi : \Gamma \to \Sigma$ gives the labeling of the edges in the state-graph. Then, the set of words of the underlying local language defined by set Θ corresponds to all accepting paths in the state-graph and its projection by π gives the language recognized by the automaton (cf. [7]). □

As consequence of the above remark, we have that when rectangles degenerate in strings, the definition of recognizability by tiling systems coincides with the classical definition for strings. Equivalently, given a recognizable (by means of finite automata) string language L, if we consider L as a two-dimensional language whose pictures have only one row, then L is also recognizable by (two-dimensional) tiling systems.

Remark: Finally, we notice that this definition of recognizability in terms of local languages and projections is implicitly non-deterministic. This can be easily understood if we refer to the above remark and look at the one-dimensional case: if no particular constraints are given for the set Θ, the 4-tuple $(\Sigma, \Gamma, \Theta, \pi)$ corresponds in general to a non-deterministic automaton. □

We now give some examples of two-dimensional languages recognizable by tiling systems to emphasize the power of this definition of recognizability in describing properties of pictures.

Example 7.3 Let Σ be a finite alphabet and let $p \in \Sigma^{**}$ be a picture over Σ. The language $L = \{p\}$ that contains the single picture p belongs to $\mathcal{L}(\mathrm{TS})$.

The easier way to define a tiling system $(\Sigma, \Gamma, \Theta, \pi)$ for L is the following. Let (m, n) be the size of p (i.e., $\ell_1(p) = m$ and $\ell_2(p) = n$). We take a local alphabet of $m \cdot n$ symbols $\Gamma = \{a_{i,j} \mid i = 1, \ldots, m\ j = 1, \ldots, n\}$. Then, we consider the picture $p' \in \Gamma^{**}$ of the same size as p such that $p'(i,j) = a_{i,j}$, $\forall i = 1, \ldots, m$, and $j = 1, \ldots, n$. The language $L' = \{p'\}$ is local and the corresponding set Θ contains all the $(m+1) \times (n+1)$ different blocks of size (2,2) of $\widehat{p'}$.

$$\Theta = \left\{ \begin{array}{l} \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & a_{11} \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & a_{1n} \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline a_{n1} & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline a_{mn} & \# \\ \hline \# & \# \\ \hline \end{array}, \\ \\ \begin{array}{|c|c|}\hline a_{mj} & \# \\ \hline a_{mj+1} & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & a_{1j} \\ \hline \# & a_{1j+1} \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline a_{i1} & a_{i+11} \\ \hline \end{array}, \quad \Big| \begin{array}{l} i = 1, \ldots, m-1 \\ j = 1, \ldots, n-1 \end{array} \\ \\ \begin{array}{|c|c|}\hline a_{in} & a_{i+1n} \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline a_{ij} & a_{i+1j} \\ \hline a_{ij+1} & a_{i+1j+1} \\ \hline \end{array} \end{array} \right\}$$

The projection $\pi : \Gamma \to \Sigma$ is defined as $\pi(a_{i,j}) = p(i,j)$.

Example 7.4 Let $\Sigma = \{a\}$ be a one-letter alphabet and let L be the languages of all pictures over Σ with 3 columns described in Example 2.1. Language L is in $\mathcal{L}(\mathrm{TS})$: it is not difficult to verify that we can take a local language over a three letters alphabet with a set of allowed tiles Θ containing all the blocks of size $(2,2)$ of the following picture.

#	#	#	#	#
#	1	2	3	#
#	1	2	3	#
#	1	2	3	#
#	1	2	3	#
#	1	2	3	#
#	1	2	3	#
#	#	#	#	#

Obviously, the projection π is such that $\pi(1) = \pi(2) = \pi(3) = a$.

Similarly, one can show that the language $L_c = \{x \mid \ell_2(x) = c\} \subseteq \Sigma^{**}$ where c is any positive integer constant, belongs to $\mathcal{L}$(TS). (It suffices to take a local alphabet of size c and define an underlying local language whose rectangles have a different letter for each column). Moreover, it is not difficult to prove that the corresponding languages over any alphabet Γ are recognizable by using a local alphabet of size $c \cdot |\Gamma|$ (see Example 7.2).

7.3 Closure properties

In this section we examine some properties of the family $\mathcal{L}$(TS) concerning closure under different kinds of operations. All these results can be also found in [14].

We start with a simple theorem whose proof derives directly from the definition of tiling systems.

Theorem 7.1 *The family $\mathcal{L}$(TS) is closed under projection.*

Proof: Let Σ_1, Σ_2 be two finite alphabets and let $\varphi : \Sigma_1 \to \Sigma_2$ be a projection. We have to prove that, if $L_1 \subseteq \Sigma_1^{**}$ is recognizable by tiling systems then $L_2 = \varphi(L_1)$ is recognizable by tiling systems, too.

Let $\mathcal{T}_1 = (\Sigma_1, \Gamma, \Theta, \pi_1)$ be a tiling system for L_1: that is $L_1 = L(\mathcal{T}_1)$. Then $L_1 = \pi_1(L')$, where L' is the (underlying) local language represented by Θ. It easy to see that L' is an underlying local language also for L_2; in fact $L_2 = \pi_2(L')$ where $\pi_2 = \varphi \cdot \pi_1 : \Gamma \to \Sigma_2$. Hence $\mathcal{T}_2 = (\Sigma_2, \Gamma, \Theta, \pi_2)$ is a tiling system for L_2. $\square$

We now consider concatenation operations (see Section 2 for the formal definitions).

Theorem 7.2 *The family $\mathcal{L}$(TS) is closed under row and column concatenation operations.*

Proof: Let L_1 and L_2 be picture languages over an alphabet Σ and let $L = L_1 \varobar L_2$ be the language corresponding to the column concatenation of L_1 and L_2. By definition of column concatenation, a picture $p \in L$ is composed by a pair of pictures $p_1 \in L_1$ and $p_2 \in L_2$ with the same number of rows such that the rightmost column of p_1 is glued to the leftmost column of p_2.

Let $(\Sigma, \Gamma_1, \Theta_1, \pi_1)$ and $(\Sigma, \Gamma_2, \Theta_2, \pi_2)$ be two tiling systems for L_1 and L_2, respectively. Without loss of generality we assume that the local alphabets Γ_1 and Γ_2 are disjoint. We can define a tiling system $(\Sigma, \Gamma, \Theta, \pi)$ for L as follows. We take $\Gamma = \Gamma_1 \cup \Gamma_2$.

Note that set Θ has to contain all the elements from set Θ_1 except those corresponding to the right borders and all elements from set Θ_2 except those corresponding to the left borders. Moreover, we should add some "middle tiles" corresponding to the two columns where the gluing is done. Such tiles contain pieces of the right border of pictures in L_1 in the left side and pieces of the left border of pictures in L_2 in the right side. More formally, we first define the following three sets of tiles.

- $\Theta_1' = \left\{ \begin{array}{|c|c|} \hline a_1 & b_1 \\ \hline c_1 & d_1 \\ \hline \end{array} \mid \begin{array}{|c|c|} \hline a_1 & b_1 \\ \hline c_1 & d_1 \\ \hline \end{array} \in \Theta_1 \text{ and } b_1, d_1 \neq \# \right\}$

- $\Theta_2' = \left\{ \begin{array}{|c|c|} \hline a_2 & b_2 \\ \hline c_2 & d_2 \\ \hline \end{array} \mid \begin{array}{|c|c|} \hline a_2 & b_2 \\ \hline c_2 & d_2 \\ \hline \end{array} \in \Theta_2 \text{ and } a_2, c_2 \neq \# \right\}$

- $\Theta_{12} = \left\{ \begin{array}{|c|c|} \hline a_1 & a_2 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & \# \\ \hline b_1 & b_2 \\ \hline \end{array}, \begin{array}{|c|c|} \hline c_1 & c_2 \\ \hline d_1 & d_2 \\ \hline \end{array} \mid \begin{array}{l} \begin{array}{|c|c|} \hline a_1 & \# \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & \# \\ \hline b_1 & \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline c_1 & \# \\ \hline d_1 & \# \\ \hline \end{array} \in \Theta_1, \\ \\ \begin{array}{|c|c|} \hline \# & a_2 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & \# \\ \hline \# & b_2 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & c_2 \\ \hline \# & d_2 \\ \hline \end{array} \in \Theta_2 \end{array} \right\}$

Then, we take: $\Theta = \Theta_1' \cup \Theta_2' \cup \Theta_{12}$.

The projection $\pi : \Gamma \to \Sigma$ is defined in a way that its restrictions to alphabets Γ_1 and Γ_2 coincide with projections π_1 and π_2, respectively. In formulas, we have:

$$\forall a \in \Gamma, \qquad \pi(a) = \begin{cases} \pi_1(a) & \text{if } a \in \Gamma_1 \\ \pi_2(a) & \text{if } a \in \Gamma_2 \end{cases}$$

In a similar way, we can obtain a tiling system for the row concatenation $L = L_1 \ominus L_2$. By definition of row concatenation, a picture $p \in L$ consists of a pair of pictures $p_1 \in L_1$ and $p_2 \in L_2$ with the same number of columns such that the row at the bottom of p_1 is glued to the row at the top of p_2. Thus, in order to define a tiling system for L starting from the tiling systems for L_1 and L_2, we can proceed as before. The only difference is that, this time, set Θ should contain all the elements from set Θ_1 except for those corresponding to the bottom borders and all elements from set Θ_2 except for those corresponding to the top borders and plus some "middle tiles" corresponding to the two rows where the gluing is done. $\square$

Theorem 7.3 *The family $\mathcal{L}(TS)$ is closed under row and column closure operations.*

Proof: Let L be a picture language over an alphabet Σ that is recognizable by tiling systems. By definition, the column closure $L^{*\Phi}$ of L is given by successions of column concatenation operations between pictures in L. Then we can find a tiling system $(\Sigma, \Gamma, \Theta, \pi)$ for $L^{*\Phi}$ using the same technique as in the proof of the previous theorem.

We consider two different tiling systems for L, say $(\Sigma, \Gamma_1, \Theta_1, \pi_1)$ and $(\Sigma, \Gamma_2, \Theta_2, \pi_2)$, such that the local alphabets Γ_1 and Γ_2 are disjoint. We define $\Gamma = \Gamma_1 \cup \Gamma_2$ and build a set Θ_{12} as in the proof of the previous theorem. Then, the set of tiles Θ is defined as $\Theta = \Theta_1 \cup \Theta_{12}$. The projection $\pi : \Gamma \to \Sigma$ is defined again as in the proof of the previous theorem.

The same idea can be used to define a tiling system for the picture language $L^{*\ominus}$ corresponding to the row closure of L. $\square$

We now consider the Boolean operations.

Theorem 7.4 *The family $\mathcal{L}(TS)$ is closed under union and intersection.*

Proof: Let L_1 and L_2 be two picture languages over an alphabet Σ and let $(\Sigma, \Gamma_1, \Theta_1, \pi_1)$ and $(\Sigma, \Gamma_2, \Theta_2, \pi_2)$ be two tiling systems to recognize L_1 and L_2, respectively. Once again, we assume that the local alphabets Γ_1 and Γ_2 are disjoint. A tiling system $(\Sigma, \Gamma, \Theta, \pi)$ for the "union language" $L = L_1 \cup L_2$ is quite easy to construct. We take as local alphabet $\Gamma = \Gamma_1 \cup \Gamma_2$ and define the projection π so that its restrictions to alphabets Γ_1 and Γ_2 coincide with π_1 and π_2, respectively (see proof of Theorem 7.2). The set of tiles Θ is the union of sets Θ_1 and Θ_2.

To construct a tiling system $(\Sigma, \Gamma, \Theta, \pi)$ for the "intersection language" $L = L_1 \cap L_2$ we need to find an underlying local language for L whose pictures "belong", in some sense, to both the underlying local languages for L_1 and L_2. This can be accomplished by taking as local alphabet Γ a particular subset of $\Gamma_1 \times \Gamma_2$ such that

$$(a_1, a_2) \in \Gamma \Leftrightarrow \pi_1(a_1) = \pi_2(a_2).$$

Then, a set of tiles Θ that represents the intersection of sets Θ_1 and Θ_2 can be defined as follows.

$$\Theta = \left\{ \begin{array}{|c|c|}\hline (a_1, a_2) & (b_1, b_2) \\ \hline (c_1, c_2) & (d_1, d_2) \\ \hline \end{array} \;\middle|\; \begin{array}{|c|c|}\hline a_1 & b_1 \\ \hline c_1 & d_1 \\ \hline \end{array} \in \Theta_1 \text{ and } \begin{array}{|c|c|}\hline a_2 & b_2 \\ \hline c_2 & d_2 \\ \hline \end{array} \in \Theta_2 \right\}.$$

Finally, the projection $\pi : \Gamma \to \Sigma$ is well defined as

$$\pi((a_1, a_2)) = \pi_1(a_1) = \pi_2(a_2), \;\; \forall (a_1, a_2) \in \Gamma_1 \times \Gamma_2.$$

$\square$

Remark that the closure under union together with Example 7.3 imply that all finite languages belong to $\mathcal{L}(\mathrm{TS})$.

In a different set-up, K. Inoue and I. Takanami (cf. [22], [18], [19]) proved that $\mathcal{L}(\mathrm{TS})$ is not closed under Boolean complementation. We give here a shorter proof of this fact, that refers directly to family $\mathcal{L}(\mathrm{TS})$ and uses a combinatorial argument

Theorem 7.5 *The family $\mathcal{L}(TS)$ is* not *closed under complement.*

Proof: Let $\Sigma = \{a, b\}$ an alphabet and let

$$L = \{p \in \Sigma^{**} \,|\, p = s \ominus s \text{ where } s \text{ is a square}\}.$$

That is, language L contains pictures of size $(2n, n)$ for every $n \in \mathbb{N}$ such that the top and the bottom square halves are identical. The proof of the theorem is given by showing that $L \notin \mathcal{L}(\mathrm{TS})$ while ${}^cL \in \mathcal{L}(TS)$.

We first prove that $L \notin \mathcal{L}(\mathrm{TS})$. Suppose, by contradiction, that $L \in \mathcal{L}(\mathrm{TS})$. Then L is a projection of a local language L' over an alphabet Γ. A counting argument will show that this leads to a contradiction. Let σ and γ be the sizes of the alphabets Σ and Γ respectively. Without loss of generality, we assume that $\sigma \leq \gamma$. Fix an integer n and let

$$L_n = \{p \in \Sigma^{**} \,|\, p = s \ominus s \text{ where } s \text{ is a square of size } n\}.$$

The number of pictures in L_n is σ^{n^2}. Let L'_n be the set of pictures in L' (over Γ) whose projections are in L_n. Notice that by choice of γ there are at most γ^{2n} possibilities for the n-th and $(n+1)$-th rows in the pictures of L'_n, since the number of stripe-pictures of size $(2, n)$ on the alphabet Γ is γ^{2n}.

For n sufficiently large it will be that $\sigma^{n^2} \geq \gamma^{2n}$. Therefore, for n sufficiently large, there will be two different pictures $p = s_p \ominus s_p$, $q = s_q \ominus s_q \in L_n$ (with $s_p \neq s_q$) such that the corresponding $p' = s'_p \ominus s''_p$, $q' = s'_q \ominus s''_q \in L'_n$ have the same nth and $(n+1)$th rows. This implies that, by definition of local language, pictures $v' = s'_p \ominus s''_q$ and $w' = s'_q \ominus s''_p$ belong to L'_n, too. Therefore pictures $\pi(v') = s_p \ominus s_q$ and $\pi(w') = s_q \ominus s_p$ belong to L_n. This is the required contradiction.

We now prove that ${}^cL \in \mathcal{L}(\mathrm{TS})$. We decompose ${}^cL = L_1 \cup L_2$, where:
$L_1 = \{p \in \Sigma^{**} |\, \ell_1(p) \neq 2\ell_2(p)\}$
$L_2 = \{p \in \Sigma^{**} |\, \ell_1(p) = 2\ell_2(p) \text{ and top and bottom halves are different}\}$.
It is quite easy to show that L_1 is recognizable, using a tiling system which, within a rectangle, builds up a line declining stepwise two squares by one, starting at the top left corner and missing the bottom right corner. On the other hand, L_2 can be written as:

$$L_2 = L_3 \cap (\Sigma^{**} \ominus (L_4 \cap (\Sigma^{**} \oplus L_5 \oplus \Sigma^{**})) \ominus \Sigma^{**})$$

where:

$L_3 = \{p \in \Sigma^{**} \,|\, \ell_1(p) = 2\ell_2(p)\}$
$L_4 = \{p \in \Sigma^{**} \,|\, \ell_1(p) = \ell_2(p) + 1\}$
$L_5 = \{p \in \Sigma^{**} \,|\, \ell_2(p) = 1 \text{ and } p(1,1) \neq p(1, \ell_1(p))\}$.

Language L_3 and L_4 can be recognized by techniques similar to the one for L_1 above described (using the opposite of the last condition for e.g. L_3). To see that L_5 is recognizable it suffices to observe that it is actually a recognizable string language. This shows that language $L_2 \in \mathcal{L}(\mathrm{TS})$ and consequently that the whole ${}^cL$ is recognizable. □

Remark: In Remark 7.2 we noticed that the definition of family $\mathcal{L}(\mathrm{TS})$ is implicitly non-deterministic. Then, as consequence of non-closure under complement of family $\mathcal{L}(\mathrm{TS})$, we infer that it is not possible to eliminate the non-determinism from this model without loosing in power of recognition (as long as deterministic versions allow complementation). □

We conclude this section by considering the rotations (see Section 2 for the formal definition). The following theorem holds.

Theorem 7.6 *The family $\mathcal{L}(TS)$ is closed under rotation.*

Proof: Let $L \subseteq \Sigma^{**}$ and let $\mathcal{T} = (\Sigma, \Gamma, \Theta, \pi)$ be a tiling system to recognize L. It is not difficult to verify that the rotation of L can be recognized by tiling system $\mathcal{T}^R = (\Sigma, \Gamma, \Theta^R, \pi)$ where Θ^R is the rotation of set Θ. □

7.4 Domino Systems

We defined "local languages" as languages given by a finite set of authorized tiles of size $(2, 2)$. The use of blocks of size $(2, 2)$ implies that, in a computational procedure to recognize a given picture, the horizontal and vertical controls are done at the same time. Then it is natural to ask what happens when the two scannings are done separately and in particular what this can imply when we apply projections afterwords.

In [25] the so-called *hv-local* picture languages are defined, where the square tiles of side 2 are replaced by "dominoes" that correspond to two kinds of tiles: *horizontal dominoes* of size $(1, 2)$ and *vertical dominoes* of size $(2, 1)$. Notice that, from a computational point of view, this corresponds to the fact that the horizontal and the vertical scanning of an input picture can be done separately. We now give some more formal definitions.

Recall that, given a picture p of size (m, n), and $h \leq m, k \leq n$, we denote by $B_{h,k}(p)$ the set of all blocks (sub-pictures) of p of size (h, k). We call *domino* a picture whose size is either $(1, 2)$ or $(2, 1)$.

Let Γ be a finite alphabet.

Definition 7.3 *A two-dimensional language $L \subseteq \Gamma^{**}$ is* hv-local *if there exists a finite set Δ of* dominoes *over the alphabet $\Gamma \cup \{\#\}$ such that language $L = \{p \in \Gamma^{**} | (B_{1,2}(\widehat{p}) \cup B_{2,1}(\widehat{p})) \subseteq \Delta\}$.*

The set Δ will be called a *representation by dominoes* for the hv-local language L and we will write $L = L(\Delta)$. It will be implicitly assumed that the empty picture λ belongs to an hv-local language if both the dominoes of sizes $(1, 2)$ and $(2, 1)$ with two $\#$ symbols belong to Δ.

The family of hv-local picture languages is strictly contained in the family LOC of local picture language as it is proved in the following proposition.

Proposition 7.1 *If $L \subseteq \Gamma^{**}$ is an* hv-local *two-dimensional language then L is a local language.*

Proof: Let $L \subseteq \Gamma^{**}$ be an *hv-local* picture language. Then $L = L(\Delta)$ where Δ is a finite set of dominoes. We will construct a finite set of tiles Θ and show that $L = L(\Theta)$. The set of tiles Θ will be defined in a way that all the sub-blocks of sizes $(1, 2)$ and $(2, 1)$ of each tile in Θ should belong to the set of dominoes Δ. More formally, we define Θ as follows.

$$\Theta = \left\{ \theta \in (\Gamma \cup \{\#\})^{2\times 2} \,|\, \theta \neq \begin{array}{|c|c|} \hline \# & \# \\ \hline \# & \# \\ \hline \end{array}, (B_{1,2}(\theta) \cup B_{2,1}(\theta)) \subseteq \Delta \right\}$$

Let $L' = L(\Theta)$. We now show that $L' = L$. Let $p \in L'$: then, by definition, $B_{2,2}(\widehat{p}) \in \Theta$. This implies that $B_{1,2}(\widehat{p}) \subseteq B_{1,2}(B_{2,2}(\widehat{p})) \subseteq B_{1,2}(\Theta) \subseteq \Delta$. Similarly $B_{2,1}(\widehat{p}) \subseteq \Delta$. Hence $p \in L$.

Conversely, let $p \in L$ and $q \in B_{2,2}(\widehat{p})$. Then $B_{1,2}(q) \subseteq B_{1,2}(\widehat{p}) \subseteq \Delta$ and $B_{2,1}(q) \subseteq B_{2,1}(\widehat{p}) \subseteq \Delta$. Therefore $q \in \Theta$ and $p \in L'$. □

We remark that the converse of Proposition 7.1 is not true, that is there are languages that are local but not hv-local. This can be easily understood by considering the local language over $\Sigma = \{0, 1\}$ of squares of 0's with the main diagonal of 1's described in Example 7.1.

The relevance of family of hv-local languages lies in the fact that it can be used to define a special kind of tiling system called *domino system*.

Definition 7.4 *A domino system (DS) is a 4-tuple $\mathcal{D} = (\Sigma, \Gamma, \Delta, \pi)$, where Σ and Γ are two finite alphabets, Δ is a finite set of dominoes over the alphabet $\Gamma \cup \{\#\}$ and $\pi : \Gamma \to \Sigma$ is a projection from Γ to Σ.*

The domino system $\mathcal{D}$ "recognizes" a language L over the alphabet Σ defined as $L = \pi(L')$ where $L' = L(\Delta)$ is the hv-local language over Γ corresponding to the set of dominoes Δ.

We denote by $\mathcal{L}$(DS) the family of two-dimensional languages recognized by domino systems. The following theorem holds.

Theorem 7.7 *$\mathcal{L}$(TS)=$\mathcal{L}$(DS)*

Proof: The inclusion $\mathcal{L}$(DS)$\subseteq$ $\mathcal{L}$(TS) is an immediate consequence of Proposition 7.1. The inverse inclusion is based on the following lemma. The proof we give here is shorter than the original one in [25]. $\square$

Lemma 7.1 *Let L be a local language over an alphabet Σ. Then there exists an hv-local language L' over an alphabet Γ and a mapping $\pi : \Gamma \to \Sigma$ such that $L = \pi(L')$.*

Proof: Let $L = L(\Theta)$ where Θ is a finite set of tiles over $\Sigma \cup \{\#\}$. Recall that, by definition, Θ contains all allowed sub-pictures of size $(2, 2)$ of pictures in L. The idea of the proof is to show that we can express the property of "beeing an allowed sub-picture of size $(2, 2)$ of picture in L" by means of dominoes over a larger alphabet Γ. This is accomplished by choosing Γ as the set Θ itself and defining the set Δ of dominoes by forcing some conditions on the tiles to stay in the same domino. More formally, we do the following.

Let $\Gamma = \Theta \subseteq (\Sigma \cup \{\#\})^{2\times 2}$ and let

$$\Delta = \left\{ \boxed{\begin{array}{cc|cc} a_1 & a_2 & b_1 & b_2 \\ a_3 & a_4 & b_3 & b_4 \end{array}}, \boxed{\begin{array}{cc} c_1 & c_2 \\ c_3 & c_4 \\ \hline d_1 & d_2 \\ d_3 & d_4 \end{array}} \;\middle|\; \begin{array}{c} \boxed{\begin{array}{cc} a_1 & a_2 \\ a_3 & a_4 \end{array}}, \boxed{\begin{array}{cc} b_1 & b_2 \\ b_3 & b_4 \end{array}}, \boxed{\begin{array}{cc} c_1 & c_2 \\ c_3 & c_4 \end{array}}, \boxed{\begin{array}{cc} d_1 & d_2 \\ d_3 & d_4 \end{array}} \in \Gamma \\ \text{and } a_2 = b_1, a_4 = b_3,\, c_3 = d_1, c_4 = d_2 \end{array} \right\}$$

Language $L' = L(\Delta)$ is an hv-local language over Γ. Then we define a mapping π between the two alphabets as follows.

$$\begin{array}{rccc} \pi : & \Gamma & \longrightarrow & \Sigma \\ & \boxed{\begin{array}{cc} a_1 & a_2 \\ a_3 & a_4 \end{array}} & \longrightarrow & a_1 \end{array}$$

To complete the proof, we have to show that $\pi(L') = L$. Before proving it formally, we give an example to clarify how a picture $p \in L$ and a picture $p' \in L'$ such that $\pi(p') = p$ are related. Suppose that picture $\widehat{p}$ is the one below:

$$\widehat{p} = \begin{array}{|c|c|c|c|c|} \hline \# & \# & \# & \# & \# \\ \hline \# & p_{11} & p_{12} & p_{13} & \# \\ \hline \# & p_{21} & p_{22} & p_{23} & \# \\ \hline \# & \# & \# & \# & \# \\ \hline \end{array}$$

Then the corresponding picture $\widehat{p'}$ will be the following.

$$\widehat{p'} = \begin{array}{|cc|cc|cc|cc|cc|} \hline \# & \# & \# & \# & \# & \# & \# & \# & \# & \# \\ \# & p_{11} & p_{11} & p_{12} & p_{12} & p_{13} & p_{13} & \# & \# & \# \\ \hline \# & p_{11} & p_{11} & p_{12} & p_{12} & p_{13} & p_{13} & \# & \# & \# \\ \# & p_{21} & p_{21} & p_{22} & p_{22} & p_{23} & p_{23} & \# & \# & \# \\ \hline \# & p_{21} & p_{21} & p_{22} & p_{22} & p_{23} & p_{23} & \# & \# & \# \\ \# & \# & \# & \# & \# & \# & \# & \# & \# & \# \\ \hline \# & \# & \# & \# & \# & \# & \# & \# & \# & \# \\ \# & \# & \# & \# & \# & \# & \# & \# & \# & \# \\ \hline \end{array}$$

Note that in the definition of $\widehat{p'}$ we have used several different "border symbols". More precisely, we have assumed without loss of generality that border symbols for Γ are all tiles containing symbol $\#$ at the top-left position.

Now we prove that $L = \pi(L')$. Let $p \in L$, p of size (m, n). We consider a picture p' over Γ defined as follows.

$$p'(i,j) = \begin{array}{|cc|}\hline p(i,j) & p(i,j{+}1) \\ p(i{+}1,j) & p(i{+}1,j{+}1) \\ \hline\end{array} \quad \text{for } i = 1, ..., m-1 \quad , j = 1, ..., n-1 \ ;$$

$$p'(i,n) = \begin{array}{|cc|}\hline p(i,n) & \# \\ p(i{+}1,n) & \# \\ \hline\end{array} \quad \text{for } i = 1, ..., m-1 \ ;$$

$$p'(m,j) = \begin{array}{|cc|}\hline p(m,j) & p(m,j{+}1) \\ \# & \# \\ \hline\end{array} \quad \text{for } j = 1, ..., n-1 \ ;$$

$$p'(m,n) = \begin{array}{|cc|}\hline p(m,n) & \# \\ \# & \# \\ \hline\end{array} .$$

It is easy to verify (by definitions of L' and π) that $p' \in L'$ and $\pi(p') = p$.

Conversely, let $p' \in L'$ and let $q \in B_{2,2}(\widehat{p'})$ be a sub-picture of $\widehat{p'}$ of size $(2,2)$. To prove that $\pi(p') \in L$ it suffices to show that $\pi(q) \in \Theta$. Without loss of generality, (because of definition of L'), suppose the block q is the following.

$$q = \begin{array}{|cc|cc|}\hline a_1 & b_1 & b_1 & b_2 \\ c_1 & d_1 & d_1 & d_2 \\ \hline c_1 & d_1 & d_1 & d_2 \\ c_3 & c_4 & d_3 & d_4 \\ \hline\end{array}$$

where all the symbols $a_1, b_1, b_2, c_1, c_2, c_3, d_1, d_2, d_3, d_4 \in \Sigma \cup \{\#\}$ and the four quadruples $\begin{array}{|cc|}\hline a_1 & b_1 \\ c_1 & d_1 \\ \hline\end{array}$, $\begin{array}{|cc|}\hline b_1 & b_2 \\ d_1 & d_2 \\ \hline\end{array}$, $\begin{array}{|cc|}\hline c_1 & d_1 \\ c_3 & c_4 \\ \hline\end{array}$, $\begin{array}{|cc|}\hline d_1 & d_2 \\ d_3 & d_4 \\ \hline\end{array} \in \Theta$.

Then $\pi(q) = \begin{array}{|cc|}\hline a_1 & b_1 \\ c_1 & d_1 \\ \hline\end{array} \in \Theta$. $\square$

Before concluding we give, as example, an application of Theorem 7.7.

Example 7.5 Consider language L of squares over the one-letter alphabet $\Sigma = \{a\}$. In Example 7.2 we saw that $L \in \mathcal{L}(\text{TS})$. In order to show that $L \in \mathcal{L}(\text{DS})$ it suffices to verify that it can be obtained as projection of the language L' over $\Gamma = \{0, 1, 2\}$ of squares whose positions in the main diagonal are covered by 1's, positions above the main diagonal are covered by 0's and positions below the main diagonal are covered by 2's. That is, L' contains pictures like the following.

1	0	0	0	0	0
2	1	0	0	0	0
2	2	1	0	0	0
2	2	2	1	0	0
2	2	2	2	1	0
2	2	2	2	2	1

Language L' is hv-local, in fact it is represented by the following set of dominoes.

$$\Delta = \left\{ \begin{array}{c} \begin{array}{|c|}\hline \# \\ \hline \# \\ \hline\end{array}, \begin{array}{|c|}\hline \# \\ \hline 1 \\ \hline\end{array}, \begin{array}{|c|}\hline \# \\ \hline 0 \\ \hline\end{array}, \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline\end{array}, \begin{array}{|c|}\hline 0 \\ \hline 1 \\ \hline\end{array}, \begin{array}{|c|}\hline 0 \\ \hline 0 \\ \hline\end{array}, \begin{array}{|c|}\hline 2 \\ \hline 2 \\ \hline\end{array}, \begin{array}{|c|}\hline 2 \\ \hline \# \\ \hline\end{array}, \begin{array}{|c|}\hline 1 \\ \hline \# \\ \hline\end{array}, \\ \\ \begin{array}{|c|c|}\hline \# & \# \\ \hline\end{array}, \begin{array}{|c|c|}\hline \# & 1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline 1 & 0 \\ \hline\end{array}, \begin{array}{|c|c|}\hline 0 & 0 \\ \hline\end{array}, \begin{array}{|c|c|}\hline 0 & \# \\ \hline\end{array}, \\ \\ \begin{array}{|c|c|}\hline \# & 2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline 2 & 2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline 2 & 1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline 1 & \# \\ \hline\end{array} \end{array} \right\}$$

Then $L = \pi(L')$ where $\pi : \Gamma \to \Sigma$, $\pi(0) = \pi(1) = \pi(2) = a$.

7.5 Generalizations of local languages

In Section 7.4 we considered a special subclass of local languages, namely the hv-local languages, and we proved that the family $\mathcal{L}(\text{DS})$ of languages that are projections of hv-local languages coincides with the family $\mathcal{L}(\text{TS})$ of languages that are projections of local languages.

In the string language theory there exist two important generalizations of local languages: the *locally testable* languages (cf. [27]) and the *threshold locally testable* languages. In this section we extend these two notions to the two-dimensional case and we show that the corresponding two-dimensional languages and their projections are all recognizable by tiling systems.

Let Σ be a finite alphabet: for any pair of integers $h, k \geq 1$, we define an equivalence relation on Σ^{**} denoted by $\sim_{hk}$ as follows.

$$\forall p, q \in \Sigma^{**}, \quad p \sim_{h,k} q \Leftrightarrow B_{h,k}(\hat{p}) = B_{h,k}(\hat{q})$$

In other words, two pictures p and q are $\sim_{hk}$ *−equivalent* if the corresponding pictures with border, have the same set of blocks of size (h, k).

We can now give the following definition.

Definition 7.5 *A two-dimensional language L over Σ is* locally testable *if it is union of $\sim_{hk}$-equivalence classes for some h and k.*

The family of all locally testable two-dimensional languages is denoted by LT. The following theorem holds (see [13] for the proof).

Theorem 7.8 *The family LT is properly included in the family $\mathcal{L}$(TS).*

Remark: Family LT is an example of sub-family of family $\mathcal{L}(\text{TS})$ that is closed under complement. $\square$

A further generalization can be obtained starting from the locally testable languages and adding the extra condition of counting, up to a fixed threshold, the number of occurrences of the sub-pictures of size (h, k). This gives the definition of *locally threshold testable* two-dimensional languages. More formally, let $t \geq 1$ be a threshold number and let $h, k \geq 1$. Given a picture p, for any picture σ of size (h, k), we define $occ^t_\sigma(p)$ to be the number of occurrences of σ in $\widehat{p}$ if σ occurs less than t times in p or to be equal to t otherwise.

Given two pictures p and q, we say that they are $\sim^t_{h,k}$ *−equivalent* if, for every picture σ of size (h, k), $occ^t_\sigma(p) = occ^t_\sigma(q)$. The relation $\sim^t_{h,k}$ is an equivalence relation.

Definition 7.6 *A two-dimensional language L over Σ is* locally threshold testable *if it is the union of $\sim^t_{h,k}$-equivalence classes for some h, k and t.*

The family of all locally threshold testable two-dimensional languages is denoted by LTT. The following theorem holds (see [15] for the proof).

Theorem 7.9 *Every language in LTT is a projection of a locally testable language.*

The following corollary is an easy consequence of Theorem 7.9.

Corollary 7.1 *The family LTT is properly included in the family $\mathcal{L}$(TS).*

8 Equivalence Theorems

The families of two-dimensional languages defined in the previous sections are based on different approaches to recognize or generate pictures that were all generalizations from the one-dimensional languages theory. It turns out that equivalence theorems hold among these families and that such theorems are, in some sense, the analogous of fundamental equivalence theorems among the families of recognizable string languages.

8.1 Tiling systems and automata

Tiling systems for picture languages were defined generalizing to the two–dimensional case a characterization of finite automata for strings in terms of local sets and projections (see Section 7.2). In [22] K. Inoue and I. Takanami proved that tiling systems can be viewed as machine devices. More specifically, a tiling system can simulate an on-line tesselation automaton and vice versa. This is the contents of the following theorem.

Theorem 8.1 *$\mathcal{L}$(2OTA) = $\mathcal{L}$(TS).*

To make to proof of the theorem easier to read, we split the theorem in two lemmas corresponding to the two inclusions in the theorem.

Lemma 8.1 *If a language is recognized by two-dimensional on-line tesselation automata then it is recognized by finite tiling systems ($\mathcal{L}$(2OTA)$\subseteq$ $\mathcal{L}$(TS)).*

Proof: Let $L \subseteq \Sigma^{**}$ be a language recognized by a two-dimensional on-line tesselation automaton $\mathcal{A} = (\Sigma, Q, I, F, \delta)$. We have to show that there exists a tiling system $\mathcal{T}$ that recognizes L. Let $\mathcal{T} = (\Sigma, \Gamma, \Theta, \pi)$ be a tiling system such that:

- $\Gamma = \Sigma \cup \{\#\} \times Q$;
- $\Theta = \Theta_m \cup \Theta_t \cup \Theta_b \cup \Theta_l \cup \Theta_r \cup \Theta_{tl} \cup \Theta_{tr} \cup \Theta_{bl} \cup \Theta_{br}$ (where *m, t, b, l, r* stand for "middle", "top", "bottom", "left", "right", respectively) with:

$$\Theta_m = \left\{ \begin{array}{|c|c|} \hline (a,r) & (b,s) \\ \hline (c,t) & (d,q) \\ \hline \end{array} \mid a,b,c,d \neq \# \text{ and } q \in \delta(s,t,d) \right\};$$

$$\Theta_t = \left\{ \begin{array}{|c|c|} \hline (\#,q_0) & (\#,q_0) \\ \hline (a,s) & (b,q) \\ \hline \end{array} \mid a,b \neq \#,\ q_0 \in I \text{ and } q \in \delta(q_0,s,b) \right\};$$

$$\Theta_b = \left\{ \begin{array}{|c|c|} \hline (a,s) & (b,q) \\ \hline (\#,q_0) & (\#,q_0) \\ \hline \end{array} \mid a,b \neq \# \text{ and } q_0 \in I \right\};$$

$$\Theta_l = \left\{ \begin{array}{|c|c|} \hline (\#,q_0) & (a,s) \\ \hline (\#,q_0) & (b,q) \\ \hline \end{array} \mid a,b \neq \#,\ q_0 \in I \text{ and } q \in \delta(s,q_0,b) \right\};$$

$$\Theta_r = \left\{ \begin{array}{|c|c|} \hline (a,s) & (\#,q_0) \\ \hline (b,q) & (\#,q_0) \\ \hline \end{array} \mid a,b \neq \# \text{ and } q_0 \in I \right\};$$

$$\Theta_{tl} = \left\{ \begin{array}{|c|c|} \hline (\#,q_0) & (\#,q_0) \\ \hline (\#,q_0) & (a,q) \\ \hline \end{array} \mid a \neq \#,\ q_0 \in I \text{ and } q \in \delta(q_0,q_0,a) \right\};$$

$$\Theta_{tr} = \left\{ \begin{array}{|c|c|} \hline (\#,q_0) & (\#,q_0) \\ \hline (a,q) & (\#,q_0) \\ \hline \end{array} \mid a \neq \# \text{ and } q_0 \in I \right\};$$

$$\Theta_{bl} = \left\{ \begin{array}{|c|c|} \hline (\#,q_0) & (a,q) \\ \hline (\#,q_0) & (\#,q_0) \\ \hline \end{array} \mid a \neq \# \text{ and } q_0 \in I \right\};$$

$$\Theta_{br} = \left\{ \begin{array}{|c|c|} \hline (a,q_f) & (\#,q_0) \\ \hline (\#,q_0) & (\#,q_0) \\ \hline \end{array} \mid a \neq \# \text{ and } q_0 \in I, q_f \in F \right\};$$

- $\pi : (\Sigma \cup \{\#\}) \times Q \longrightarrow \Sigma$ such that $\pi(a,q) = a$, $\forall a \in \Sigma \cup \{\#\}$, $q \in Q$.

Notice that set Θ is defined in a way that a picture p' of the underlying local language of $L(\mathcal{T})$ describes exactly a run of the 2OTA $\mathcal{A}$ on $p = \pi(p')$. Then, it is easy to verify that $L(\mathcal{A}) = L(\mathcal{T})$. $\square$

Lemma 8.2 *If a language is recognizable by finite tiling systems then it is recognizable by two-dimensional on-line tesselation automata ($\mathcal{L}$(TS)$\subseteq$ $\mathcal{L}$(2OTA)).*

Proof: Let $L \subseteq \Sigma^{**}$ be a language recognized by the tiling system $(\Sigma, \Gamma, \Theta, \pi)$ and let L' the underling local language represented by the set of tiles Θ (i.e., $\pi(L') = L$). Since $\mathcal{L}$(2OTA) is closed under projection (see Theorem 4.5), it suffices to show that there exists a 2OTA recognizing $L' \subseteq \Gamma^{**}$.

Let $\mathcal{A} = (\Gamma, Q, I, F, \delta)$ be a 2OTA such that:

- $Q = \Theta$;
- $I = \left\{ \begin{array}{|c|c|}\hline \# & b \\ \hline c & d \\ \hline\end{array} \mid b, c, d \in \Gamma \cup \{\#\} \text{ and } \begin{array}{|c|c|}\hline \# & b \\ \hline c & d \\ \hline\end{array} \in \Theta \right\}$;
- $F = \left\{ \begin{array}{|c|c|}\hline a & \# \\ \hline \# & \# \\ \hline\end{array} \mid \begin{array}{|c|c|}\hline a & \# \\ \hline \# & \# \\ \hline\end{array} \in \Theta \right\}$;
- $\delta : Q \times Q \times \Sigma \longrightarrow 2^Q$ such that: $\forall \begin{array}{|c|c|}\hline x & y \\ \hline a & b \\ \hline\end{array}, \begin{array}{|c|c|}\hline z & a \\ \hline t & c \\ \hline\end{array} \in Q$:

$$\delta\left(\begin{array}{|c|c|}\hline x & y \\ \hline a & b \\ \hline\end{array}, \begin{array}{|c|c|}\hline z & a \\ \hline t & c \\ \hline\end{array}, a \right) = \left\{ \begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline\end{array} \mid \begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline\end{array} \in Q \right\}.$$

Notice that the transition function δ is defined in a way that the run of $\mathcal{A}$ over a picture p simulates a tiling of p by elements of $Q = \Theta$. The initial states correspond to the tiles in the first row and in the first column of $\widehat{p}$ and the final state to accept p correspond to the tile in the bottom-right corner of $\widehat{p}$. The transition $\delta\left(\begin{array}{|c|c|}\hline x & y \\ \hline a & b \\ \hline\end{array}, \begin{array}{|c|c|}\hline z & a \\ \hline t & c \\ \hline\end{array}, a \right)$, that computes the state for position (i, j), corresponds to the following "portion" of tiling of p (position (i, j) is in the center):

$$\begin{array}{|c|c|c|}\hline & x & y \\ \hline z & a & b \\ \hline t & c & d \\ \hline\end{array}$$

Then, it can be easily verified that $L' = L(\mathcal{A})$. $\square$

8.2 Tiling systems and logic formulas

Finite tiling systems have also a natural logic meaning. In this section we prove that an analogous of Büchi's theorem for strings (cf. [3], [4] or [37]) holds also for two-dimensional languages. More precisely, in [15] it is shown that the family of languages recognized by finite tiling systems and the family of languages defined by existential monadic second order formulas coincide. This is the contents of the following theorem.

Theorem 8.2 *$\mathcal{L}$(TS)=$\mathcal{L}$(EMSO)*

Again, we split the theorem in two lemmas corresponding to the two inclusion relations in the theorem.

Lemma 8.3 *If a language is recognizable by finite tiling systems then it is definable by existential monadic second order formulas ($\mathcal{L}$(TS) $\subseteq$ $\mathcal{L}$(EMSO)).*

Proof: Let $L \subseteq \Sigma^{**}$ be recognized by the finite tiling system $(\Sigma, \Gamma, \Theta, \pi)$. Without loss of generality we assume that $\Gamma = \Sigma \times Q$ and $\pi : \Sigma \times Q \to \Sigma$ is the canonical projection. Moreover, to indicate a picture p' of the local language $L' = L(\Theta)$, we use the notations $p' = p \times c$, where p and c are two pictures of the same size over the alphabets Σ and Q, respectively.

We have:

$p \in L$ iff there is a picture $c \in Q^{**}$ of the same size as p such that the blocks of size $(2,2)$ of $\widehat{p \times c}$ belong to Θ.

We have to formalize the right-hand side by an EMSO-formula to be interpreted in $\underline{p}$. Let $Q = \{q_1, \ldots, q_k\}$: we use set variables $X_1, \ldots, X_k$ where $X_l(x)$ has the meaning $c(x) = q_l$. The above equivalence can then be reformulated as follows.

$p \in L$ iff $\underline{p}$ satisfies the following condition:
$\exists X_1 \ldots \exists X_k (X_1, \ldots, X_k$ form a partition of $dom(p)$
and for the picture c given by $c(i,j) = q_l$ iff $(i,j) \in X_l$
the blocks of size $(2,2)$ of $\widehat{p \times c}$ belong to Θ.

The partition condition on $X_1, \ldots, X_k$ corresponds to the following formula.

$$\varphi_{part}(X_1, \ldots, X_k) : \forall z \; (X_1(z) \vee \ldots \vee X_k(z)) \wedge \bigwedge_{i \neq j} \neg(X_i(z) \wedge X_j(z)).$$

Next we have to express that each sub-picture of size $(2,2)$ of $\widehat{p \times c}$ (where c is defined by a given partition $X_1, \ldots, X_k$ as above) belongs to Θ. Each tile $\theta \in \Theta$ can be numerated in the form:

$$\theta = \begin{array}{|c|c|} \hline \theta_1 & \theta_2 \\ \hline \theta_3 & \theta_4 \\ \hline \end{array}$$

with $\theta_i \in (\Sigma \times Q) \cup \{\#\}$ for $i \in \{1, \ldots 4\}$. We divide Θ into nine disjoint sets:

$$\Theta = \Theta_m \,\dot\cup\, \Theta_t \,\dot\cup\, \Theta_b \,\dot\cup\, \Theta_l \,\dot\cup\, \Theta_r \,\dot\cup\, \Theta_{tl} \,\dot\cup\, \Theta_{tr} \,\dot\cup\, \Theta_{bl} \,\dot\cup\, \Theta_{br}$$

where Θ_m contains all "middle tiles", i.e., those without $\#$; Θ_t contains the "top tiles", i.e., those with $\theta_1 = \theta_2 = \#$ and $\theta_3, \theta_4 \in \Sigma \times Q$, and so on until Θ_{br} that contains all "bottom-right tiles", i.e., those with $\theta_1 \neq \#$ and $\theta_2 = \theta_3 = \theta_4 = \#$.

By nine corresponding formulas $\psi_m, \psi_t, \ldots, \psi_{br}$ we describe in each case which of the four positions x_1, x_2, x_3, x_4 of a tile should match the picture (excluding the boundary $\#$). (The formulas $\psi_m, \psi_t, \ldots$ are variants of the formulas $\varphi_t(x), \varphi_b(x), \ldots$ described in Section 6. We set:

$\psi_m(x_1, x_2, x_3, x_4) := x_1 S_1 x_3 \wedge x_1 S_2 x_2 \wedge x_3 S_2 x_4 \wedge x_2 S_1 x_4$
$\psi_t(x_3, x_4) := x_3 S_2 x_4 \wedge \neg \exists x \, x S_1 x_3 \wedge \neg \exists x \, x S_1 x_4$

$\vdots$

$\psi_{br}(x_1) := \neg \exists x \, x_1 S_1 x \wedge \neg \exists x \, x_1 S_2 x.$

Then, we have to express that, for each quadruple satisfying one of the formulas above, the corresponding sub-picture belongs to the corresponding set in Θ. For example, let us focus on middle tiles. For each quadruple (i,j), $(i{+}1,j)$, $(i,j{+}1)$, $(i{+}1,j{+}1)$ of positions of $dom(p \times c)$, i.e., for each quadruple satisfying $\psi_m(x_1, x_2, x_3, x_4)$, the tile

$(p \times c)(i,j)$	$(p \times c)(i,j{+}1)$
$(p \times c)(i{+}1,j)$	$(p \times c)(i{+}1,j{+}1)$

belongs to Θ_m. More formally, we use nine formulas $\chi_m, \chi_t, \ldots, \chi_{br}$ expressing this for the nine possible cases. If for a tile θ we have $\theta_i = (a, q_l)$, where $i \in \{1, \ldots, 4\}$, we let $\varphi_{\theta_i}(x)$ be an abbreviation for $P_a(x) \wedge X_l(x)$. Now let:

$$\chi_m : \psi_m(x_1,x_2,x_3,x_4) \to \bigvee_{\begin{array}{|c|c|}\hline \theta_1 & \theta_2 \\ \hline \theta_3 & \theta_4 \\ \hline \end{array} \in \Theta_m} (\varphi_{\theta_1}(x_1) \wedge \varphi_{\theta_2}(x_2) \wedge \varphi_{\theta_3}(x_3) \wedge \varphi_{\theta_4}(x_4))$$

$$\chi_t : \psi_t(x_3,x_4) \to \bigvee_{\begin{array}{|c|c|}\hline \# & \# \\ \hline \theta_3 & \theta_4 \\ \hline \end{array} \in \Theta_t} (\varphi_{\theta_3}(x_3) \wedge \varphi_{\theta_4}(x_4))$$

$$\vdots$$

$$\chi_{br} : \psi_{br}(x_1) \to \bigvee_{\begin{array}{|c|c|}\hline \theta_1 & \# \\ \hline \# & \# \\ \hline \end{array} \in \Theta_{br}} \varphi_{\theta_1}(x_1).$$

Thus we obtain an existential monadic second-order sentence defining L:

$$\exists X_1 \ldots \exists X_k (\varphi_{part} \ \wedge \forall x_1 \ldots \forall x_4 \ (\chi_m \wedge \chi_t \wedge \chi_b \wedge \chi_l \wedge \chi_r \wedge \chi_{tl} \wedge \chi_{tr} \wedge \chi_{bl} \wedge \chi_{br})).$$

□

To prove the other inclusion of Theorem 8.2, i.e., that an EMSO-definable picture language is recognized by finite tiling system, we make use of the following theorem that holds in general for different structures (a proof that refers directly to picture languages can be found in [15]).

Theorem 8.3 *A language is first-order definable if and only if it is locally threshold testable.*

Lemma 8.4 *If a language is definable by existential monadic second order formulas then it is recognizable by finite tiling systems ($\mathcal{L}$(EMSO)$\subseteq \mathcal{L}$(TS)).*

Proof: Let $L \subseteq \Sigma^{**}$ be defined by the following existential monadic second-order formula.

$$\varphi : \exists X_1 \ldots \exists X_k \psi(X_1, \ldots, X_k)$$

where ψ is a first-order formula. Notice that $\psi(X_1, \ldots, X_k)$ is satisfied in picture models of the form $(\underline{p}, Q_1, \ldots, Q_k)$ with $Q_i \subseteq dom(p)$ for $i = 1, \ldots, k$. Such an expanded picture model corresponds to a picture over the extended alphabet $\Gamma = \Sigma \times \{0,1\}^k$ where the m-th additional component is 1 at position (i,j) if and only if $(i,j) \in Q_m$; otherwise it is 0. Let us call L' the language over Γ defined by $\psi(X_1, \ldots, X_k)$. By Theorem 8.3, the picture language L' is locally threshold testable and therefore, by Theorem 7.9, it is recognized by tiling systems.

Let π be the canonical projection from $\Sigma \times \{0,1\}^k$ to Σ. Then $L = \pi(L')$. Since $\mathcal{L}$(TS) is closed under projection, this implies that also L is recognized by tiling systems. □

It is interesting to remark that, as a technical application of Theorem 8.2 together with Theorem 7.5, we easily obtain that the class of picture languages definable in existential monadic second-order logic is not closed under complement. This situation is in some contrast to the involved proof of Fagin (cf. [10]) (applying an extension of Ehrenfeucht-Fraisse games) which shows nonclosure under complement of existential monadic second-order logic with respect to the class of arbitrary finite graphs.

8.3 Tiling systems and regular expressions

Let $L \subseteq \Sigma^{**}$ be a language defined by a finite tiling system. By Lemma 7.1, L can be defined as well by domino systems, i.e., L is a projection of a hv-local language K. We first analyze the relations between hv-local languages and languages denoted by regular expressions.

Theorem 8.4 *The family of hv-local languages is included in the family $\mathcal{L}$(CFRE).*

Proof: If $K \subseteq \Gamma^{**}$ is hv-local, then there exists a finite set Δ of dominoes over $\Gamma \cup \{\#\}$ such that $K = \{p \in \Gamma^{**} \mid B_{1,2}(\widehat{p}) \cup B_{2,1}(\widehat{p}) \subseteq \Delta\}$.

Let Δ_h (Δ_v respectively) denote the set of horizontal (vertical, resp.) dominoes of Δ (i.e., $\Delta = \Delta_h \cup \Delta_v$). Denote by K_h (K_v, resp.) the hv-local language obtained replacing Δ with Δ_h (Δ_v, resp.). Now, one can associate to the picture language K_h a local string language $S_h \subseteq \Gamma^*$ defined by the set of dominoes Δ_h, considered as a subset of $(\Gamma \cup \{\#\})^2$. Similarly, one can associate to the picture language K_v the local string language $S_v \subseteq \Gamma^*$, defined by the set of dominoes Δ_v considered as subset of $(\Gamma \cup \{\#\})^2$. It is easy to verify that a picture p over Γ is in K if and only if each row of p (taken as a string) is in S_h and each column of p (taken as a string) is in S_v. This corresponds to the following equality (see Section 2 for the definition of $\oplus$).

$$K = S_h \oplus S_v$$

Since S_h (S_v resp.) is a local string language, it is also regular, i.e., there exists a regular (string) expression α_h (α_v resp.) denoting S_h (S_v resp.). Let β_h (β_v resp.) the regular (picture) expression obtained replacing in α_h (α_v resp.) the concatenation with the operation $\oplus$ ($\ominus$ resp.) and the $*$ operation with the operation $*\oplus$ ($*\ominus$ resp.). Then the language K corresponds to the following complementation–free regular expression:

$$K = (\beta_h)^{*\ominus} \cap (\beta_v)^{*\oplus}.$$

□

Using this result, one can obtain the following Kleene-like characterization for the family $\mathcal{L}(TS)$.

Theorem 8.5 $\mathcal{L}(TS) = \mathcal{L}(PCFRE)$.

Proof: We prove first the inclusion $\mathcal{L}(TS) \subseteq \mathcal{L}(PCFRE)$. We recall (see Theorem 7.7) that, for any language $L \subseteq \Sigma^{**}$ belonging to $\mathcal{L}(TS)$, there exists an alphabet Γ, a projection $\pi : \Gamma \to \Sigma$ and a hv-local language K over Γ such that $L = \pi(K)$.Then, by Theorem 8.4, $K \in \mathcal{L}(CFRE)$ and therefore $L \in \mathcal{L}(PCFRE)$.

To prove the converse, we first remark that (as it is easy to verify) the atomic languages belong to $\mathcal{L}(TS)$. Moreover, since $\mathcal{L}(TS)$ is closed under all the operations in $\mathcal{R}_1$ plus projection (see Section 7.3), one concludes that $\mathcal{L}(PCFRE) \subseteq \mathcal{L}(TS)$. □

The previous theorem can be restated also as follows: *"Family $\mathcal{L}(TS)$ coincides with the smallest family of languages that contains the atomic languages and is closed under regular operations in $\mathcal{R}_1$ plus projection"*. In other words, a language in $\mathcal{L}(TS)$ can be expressed by a formula containing regular operations (without complementation) and projection.

In the course of the proofs of previous theorems we also proved the following result which is of independent interest, since it allows to define tiling recognizable picture languages in terms of recognizable string languages. This characterization will be useful for further generalizations (cf. Section 11).

Theorem 8.6 *A picture language L over an alphabet Σ is tiling recognizable if and only if there exist two recognizable string languages S_1 and S_2 over an alphabet Γ and a projection $\pi : \Gamma \to \Sigma$ such that $L = \pi(S_1 \oplus S_2)$.*

As a consequence of this theorem, a tiling recognizable language can be specified by a triple $(\mathcal{A}_1, \mathcal{A}_2, \pi)$, where $\mathcal{A}_1$ and $\mathcal{A}_2$ are two (standard) finite automata and π is a projection.

8.4 Comparing all families

We summarize in the following theorem the equivalence results obtained in the previous sections.

Theorem 8.7 *Given a two-dimensional language L, the following conditions are equivalent.*

(i) L is defined by a complementation–free regular expression with projection ($L \in \mathcal{L}$(PCFRE)).

(ii) L is recognized by an on-line tesselation automaton ($L \in \mathcal{L}$(2OTA)).

(iii) L is defined by an existential monadic second order formula ($L \in \mathcal{L}$(EMSO)).

(iv) L is recognized by a finite tiling system ($L \in \mathcal{L}$(TS)).

For the sequel, the family of two-dimensional languages defined in the theorem above will be denoted by REC and the elements of REC will be simply referred as *recognizable two-dimensional languages*.

Remark that Theorem 8.7 indicates the "robustness" of this notion of "finite-state" recognizability for two-dimensional languages. In fact, it can be defined in terms of machine models, regular expressions, logic formulas and tiling systems.

In the next part of this section we summarize the inclusion relationships between the families of two-dimensional languages introduced in this chapter and defined in terms of different formal models for recognizing or generating languages.

As far as machine models are concerned, in Section 4 we showed that $\mathcal{L}$(4DFA) is properly included in $\mathcal{L}$(4NFA) (cf. Theorem 4.1) and that $\mathcal{L}$(2DOTA) is properly included in $\mathcal{L}$(2OTA) (cf. Theorem 4.4). Moreover, $\mathcal{L}$(4NFA) is properly included in $\mathcal{L}$(2OTA) (cf. Theorem 4.7) whereas the family $\mathcal{L}$(2DOTA) is incomparable with $\mathcal{L}$(4DFA) and $\mathcal{L}$(4NFA).

In connection with tiling systems it has been proved in [13] that the family of locally testable languages LT is properly included in $\mathcal{L}$(4DFA) and we showed (cf. Theorem 8.3) that the family of locally threshold testable languages LTT coincides with family $\mathcal{L}$(FO) of first-order formulas definable languages.

Regarding grammar models, we proved (cf. Theorem 5.1) that $\mathcal{L}$(2RLG) is properly included in $\mathcal{L}$(DFA). Moreover $\mathcal{L}$(2RLG) is not comparable with LOC and LT. Indeed it is easy to give examples of languages in $\mathcal{L}$(2RLG) that are not in LOC. On the other hand, the fact that the family $\mathcal{L}$(2RLG) is closed under projection (cf. Proposition 5.1) and that is properly included in REC, implies that LOC cannot be included in $\mathcal{L}$(2RLG). We do not know whether there is some inclusion relation between $\mathcal{L}$(2RLG) and $\mathcal{L}$(2DOTA).

Some interesting open problems in the theory arises for two-dimensional languages defined by regular expressions. The following inclusions trivially hold.

$$\mathcal{L}(\mathrm{SFRE}) \subseteq \mathcal{L}(\mathrm{RE})$$

$$\mathcal{L}(\mathrm{CFRE}) \subseteq \mathcal{L}(\mathrm{RE})$$

It is easy to verify that the first inclusion is strict. We are not able to prove that also the second inclusion is strict. A related open problem is whether $\mathcal{L}$(SFRE) is included in $\mathcal{L}$(CFRE). This inclusion holds and it is strict in the one-dimensional case.

In Section 8.3 we showed that $\mathcal{L}$(CFRE) is included in REC and such inclusion is strict (it can be seen by considering, for instance, the language of squares). The main open problem concerning regular expressions is whether $\mathcal{L}$(RE) is included in REC. We know that REC is not closed under complementation (cf. Theorem 7.5); on the other hand we do not have any example of languages in $\mathcal{L}$(RE) and not in REC.

Relations between larger families of regular expressions and recognizable picture languages are studied in [26].

9 Properties of recognizable languages

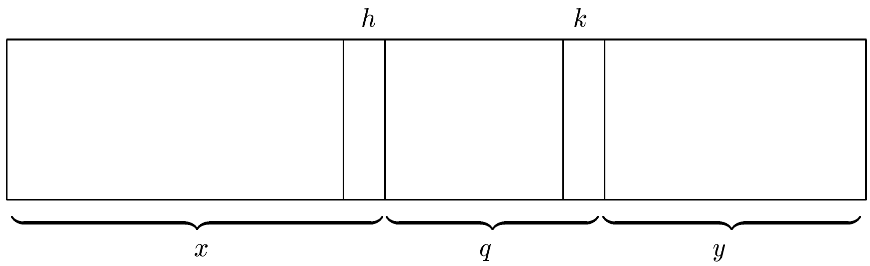
9.1 Necessary conditions for recognizability

In this section we provide some tools that can be used to show the *non*-recognizability of some picture languages and to show that a given language is infinite. All the proofs will use the tiling system characterization for the family REC.

We first state an analogous to the "pumping lemma" for recognizable string language (cf. [17]) that holds for the family REC. Roughly speaking, this lemma says that, if a language contains a picture whose number of columns is sufficiently larger than its number of rows then the language contains an infinite number of pictures with that number of rows.

Lemma 9.1 (Horizontal Iteration Lemma) *Let L be a recognizable two-dimensional language. Then there is a function $\varphi : \mathbb{N} \to \mathbb{N}$ such that if p is a picture in L such that $\ell_1(p) = n, \ell_2(p) > \varphi(n)$, we may write $p = x \obar q \obar y$ with $\ell_2(x \obar q) \leq \varphi(n)$ and $\ell_2(y) \geq 1$; and for all $i \geq 0$, picture $x \obar q^{i\obar} \obar y$ is in L. Furthermore, $\varphi(n) \leq \gamma^n$, $n \in \mathbb{N}$, where γ is the size of any local alphabet used to represent L.*

Proof: Let $(\Sigma, \Gamma, \Theta, \pi)$ be a tiling system for the language L and let L' be the underling local language for L defined by set Θ. Given a picture $p' \in L'$ with n rows, p' can have, at most, γ^n distinct columns, where γ is the size of local alphabet Γ. Then, if $\ell_2(p') > \gamma^n$ there exist two columns, say the $h-$th and the $k-$th ones with $h < k$, that are equal.



We decompose $p' = x \obar q \obar y$, that is we write p' as column concatenations of three pictures x, q and y that are the blocks of p' corresponding to columns from 1 to h, columns from $h+1$ to k and columns from $k+1$ to $\ell_2(p')$ respectively. By definition of local language, $p' \in L'$ means that its blocks of size $(2,2)$ belong to set Θ. It is not difficult to verify that therefore also the picture $x \obar y$ is in L. Moreover, for any $i = 2, 3, \ldots$, pictures $p'_i = x \obar q^{i\obar} \obar y$ are defined by the same set of tiles as p' and therefore they all belong to L . This concludes the proof of the lemma. □

In a similar fashion one can state a *"Vertical Iteration Lemma"* that refers to languages containing pictures whose number of rows is larger than an exponential function of its number of columns. (In this case the Lemma states that the language contains an infinite number of pictures with that number of columns.)

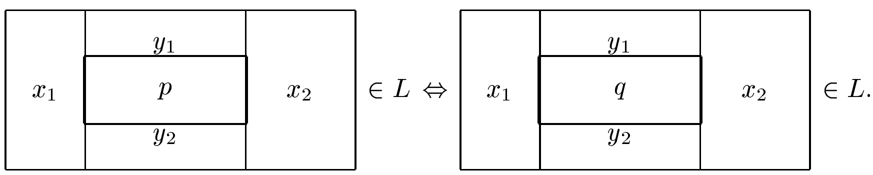
An application of Lemma 9.1 will be given later, in Section 10.

We now give a definition of an equivalence relation among pictures. Let Σ be a finite alphabet.

Definition 9.1 *Let $L \subseteq \Sigma^{**}$ be a language and let $p, q \in L$ be two pictures of the same size (k, r). Pictures p and q are* syntactically equivalent modulo L *($p \sim_L q$) if for any $x_1, x_2, y_1, y_2 \in \Sigma^{**}$ it holds that*

$$x_1 \obar (y_1 \ominus p \ominus y_2) \obar x_2 \in L \quad \Leftrightarrow \quad x_1 \obar (y_1 \ominus q \ominus y_2) \obar x_2 \in L$$

that corresponds to the following:



Notice that the syntactic equivalence can be defined only among pictures of the same size. If the alphabet Σ has σ symbols, then the number of different pictures of size (k, r) over Σ is $\sigma^{k \cdot r}$. We denote by $f_L(k, r)$ the number of equivalence classes of pictures of size (k, r). The following lemma gives an upper bound on the number $f_L(k, r)$.

Lemma 9.2 (Syntactic Equivalence Lemma) *Let L be a recognizable language, then there exists a positive integer c such that $f_L(k, r) \leq c^{k+r}$.*

Proof: Let $L' \subseteq \Gamma^{**}$ be an underling local language for L and let γ be the size of the local alphabet Γ. We set $c = \gamma^2$.

For any picture p, we call *perimeter of p* the set composed by all positions of the bottom and the top rows and of the leftmost and the rightmost columns of p. Let $p, q \in L$ be two pictures of size (k, r) and let $p', q' \in L'$ be their corresponding pictures in L' (that is $\pi(p') = p$ and $\pi(q') = q$). It is not difficult to verify that, if p' and q' have the same perimeter, then pictures p and q are syntactically equivalent.

Then, the lemma follows by noticing that the number of different perimeters of pictures of size (k, r) over Γ is $\gamma^{2(k+r)-4}$. $\square$

Remark: The proof of non-closure under complement of family REC (see Theorem 7.5) is actually based on the Syntactic Equivalence Lemma. In fact, in such proof it is shown that there exists a constant c such that if n is greater than c then we can find two square pictures p and q ($p \neq q$) of side n that are syntactically equivalent. This implies that picture $p \ominus q$ belongs to language L_n and gives the contradiction. $\square$

9.2 Undecidability results

In this section we analyze some properties that the family of recognizable picture languages does not share with the corresponding family in one dimension. We already saw one of these properties in Section 7.3 where we proved that family REC is not closed under complement. As in the previous section, all the proofs will refer to the tiling systems characterization of family REC.

We consider the decidability of the emptiness problem. In the one-dimensional case it is trivially decidable whether the language recognized by a finite automaton is empty (cf., for example, [17]). The corresponding problem in two dimensions can be formulated as follows.

Emptiness Problem for recognizable picture languages: *Given a tiling system $\mathcal{T} = (\Sigma, \Gamma, \Theta, \pi)$, decide whether the corresponding picture language $L = L(\mathcal{T})$ is empty.*

We give first an example.

Example 9.1 Let Θ be the following set of tiles over $\Gamma = \{0, 1\}$.

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

The local picture language $L = L(\Theta)$ is an empty set (there are no left-border (or right-border) tiles that allow us to "switch" from the $1s$ in the top to the $0s$ in the bottom of the pictures).

In the case of Example 9.1, it is quite easy to verify that the language corresponding to the given set of tiles Θ is empty. Nevertheless, this is not always the case. We will prove that, in general, the Emptiness Problem for recognizable two-dimensional languages is undecidable.

Theorem 9.1 *The Emptiness Problem for the family REC is undecidable.*

Proof: We use the known fact that, "Given a Turing Machine $\mathcal{M}$, it is undecidable whether the language $L(\mathcal{M})$ recognized by $\mathcal{M}$ is empty" (see, for example, [17]). We describe a procedure that, given any Turing Machine $\mathcal{M}$, define a set of tiles $\Theta_{\mathcal{M}}$ such that *"Language $L(\mathcal{M})$ is empty if and only if the local language $L(\Theta_{\mathcal{M}})$ is empty"*. This will give, as a consequence, that the emptiness problem is undecidable for the family LOC and therefore, in general, (since LOC $\subset$ REC) it is undecidable for the family REC.

Given a Turing Machine $\mathcal{M}$, let Γ and Q be the alphabet and the set of states of $\mathcal{M}$, respectively. An instantaneous configuration of $\mathcal{M}$ is given by a string uqv where $uv \in \Gamma^*$ is the contents of the tape between the leftmost and the rightmost non-blank symbol, state $q \in Q$ is the current state and the first letter of v is the one scanned by the tape-head. We consider a new alphabet Γ' in a one-to-one correspondence with Γ such that if $x \in \Gamma$ then x' is the corresponding letter in Γ'. Then, we can rewrite any instantaneous configuration uqv as uqv', where v' is the string over Γ' corresponding to v.

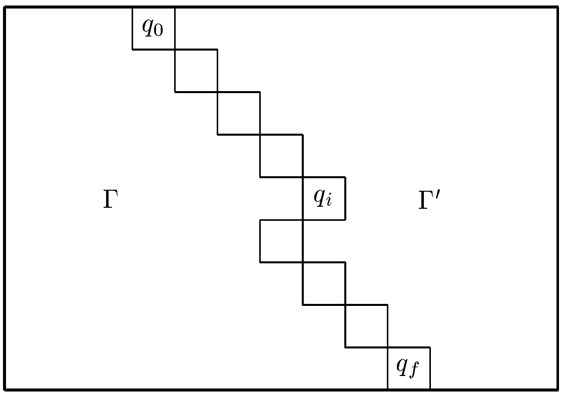
A successful computation of $\mathcal{M}$ can be described as a finite sequence of strings $w = (w_1, w_2, \ldots, w_m)$ where w_i is the ith instantaneous configuration of $\mathcal{M}$. Given a successful computation w, we add as many as needed B and B' "blank" symbols to the beginning and to the end, respectively, of each instantaneous configuration w_i in a way that, for $i = 1, 2, \ldots, m$:

- the first symbol of w_i is the blank B;
- the last symbol of w_i is the blank B';
- all strings w_i have the same length n.

Notice that, if q_0 and q_f are the initial and the final state, respectively, of $\mathcal{M}$, the following holds:

- $w_1 \in B^* q_0 \Gamma^* B'^*$
- $w_f \in B^* \Gamma^* q_f \Gamma'^* B'^*$
- if $w_i = uqv'$ and q has position j in w_i, then $w_{i+1} = u_1 p v_1$ and p has position $j-1, j$ or $j+1$ in w_{i+1} according to the corresponding transition.

Let p_w be a picture of size (m, n) obtained by writing all strings in w in a way that w_{i+1} is below w_i for $i = 2, \ldots, m$. Then all possible blocks of size $(2, 2)$ of $\widehat{p_w}$ are compatible with the transitions of the Turing Machine $\mathcal{M}$.



Let $\Theta_{\mathcal{M}}$ be the set of tiles over $\Gamma \cup \Gamma' \cup Q \cup \{\#\}$ that can appear as blocks of pictures p_w for some successful computation w of $\mathcal{M}$. Then, it is not difficult to verify that:

- If $\mathcal{M}$ has successful computation, i.e. if $\mathcal{L}(\mathcal{M}) \neq \emptyset$, then there exist a picture whose blocks of size $(2, 2)$ belongs to $\Theta_{\mathcal{M}}$. i.e. $\mathcal{L}(\Theta_{\mathcal{M}}) \neq \emptyset$.
- If $\mathcal{L}(\Theta_{\mathcal{M}}) \neq \emptyset$ then $\mathcal{M}$ has successful computation.

This concludes the proof. $\Box$

A problem that is complementary to the Emptiness Problem is the so-called *Universe Problem*. It can be stated as follows: "Given a tiling system $(\Sigma, \Gamma, \Theta, \pi)$ for a two-dimensional language over an alphabet Σ, decide whether the language $L(\Sigma, \Gamma, \Theta, \pi)$ coincides with the whole Σ^{**}." We recall that the family LT of locally testable two-dimensional languages (see Remark 7.5) is closed under complement and it contains the family LOC of local languages. Moreover, notice that in the previous proof we actually showed that the emptiness problem is undecidable for the family LOC. This proves the following corollary.

Corollary 9.1 *The Universe Problem for recognizable picture languages is undecidable.*

10 Recognizable functions

In the theory of string languages the case of one-letter alphabet plays a special role and leads to the theory of recognizable set of numbers (cf. [7]).

In this section we consider two-dimensional languages over one-letter alphabets. In this case a picture can be actually identified with a pair (m, n) of natural numbers, where m is the height (i.e., the number of rows) of the picture and n is its length (i.e., the number of columns). With this assumptions the study of a picture language over a one-letter alphabets can be performed as well by investigating the corresponding set of integer pairs. On the other hand, every function $f : \mathbb{N} \rightarrow \mathbb{N}$ defined over the set of natural numbers is actually a set of integer pairs and therefore it can be viewed as a two-dimensional language over a one-letter alphabet.

We can "export" the definition of recognizability from picture languages to functions and we say that a function is *recognizable* if it is so the corresponding picture language. Of course we can define functions recognized by 4-way automata, by grammars, by tiling systems, and so on for any definition given in this chapter. Here we refer to family REC and, in particular, we use the tiling systems characterization of REC. Our aim is identifying classes of recognizable functions and finding particular characterizations for them. Most of the results in this section are from [12].

We establish first some notations and give formal definitions. Let $\mathbb{N}$ indicate the set of natural numbers. Given a function $f : \mathbb{N} \rightarrow \mathbb{N}$, the *two-dimensional language associated to* f is defined as

$$L_f = \{p \in \Sigma^{**} | \ell_1(p) = n \text{ and } \ell_2(p) = f(n),\, n \in \mathbb{N}\}$$

where Σ is a one-letter alphabet.

Definition 10.1 *A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is* recognizable *if the associated two-dimensional language L_f is recognizable.*

We denote by $\mathcal{F}_{\mathrm{REC}}$ the family of all functions recognizable by finite tiling systems. In the sequel, the two notations $f \in \mathcal{F}_{\mathrm{REC}}$ and $L_f \in$REC will be used indifferently with the meaning of "function f is recognizable (by finite tiling systems)". As simple examples of such recognizable functions, let us consider the following.

Example 10.1 The "constant function", e.g. function $f(n) = c$, $c \in \mathbb{N}$, is recognizable. In fact, language L_f corresponds to that one in Example 7.4. Furthermore, the "identity function", e.g. function $f(n) = n$, is recognizable: language L_f corresponds to the language of squares in in Example 7.2.

Given two functions $f, g : \mathbb{N} \rightarrow \mathbb{N}$, we indicate by $f+g$ the function defined as $(f+g)(n) = f(n)+g(n)$. Similarly, if $c \in \mathbb{N}$, we indicate by $c \cdot f$ the function defined as $(c \cdot f)(n) = c \cdot f(n)$. We show that family $\mathcal{F}_{\mathrm{REC}}$ is closed under these operations of sum and product for a positive constant.

Lemma 10.1 *Let $f, g : \mathbb{N} \rightarrow \mathbb{N}$ be two functions and let $c \in \mathbb{N}$. If $f, g \in \mathcal{F}_{\mathrm{REC}}$ then $f + g \in \mathcal{F}_{\mathrm{REC}}$ and $c \cdot f \in \mathcal{F}_{\mathrm{REC}}$.*

Proof: Remark first that, the two-dimensional language associated to a function has the property that, for each $n \in \mathbb{N}$, it contains only one rectangle of height n. This implies that $L_{f+g} = L_f \oplus L_g$. Then, since family REC is closed under column concatenation, $L_{f+g} \in$REC, that is function $f + g$ is recognizable. Similarly, it holds that $L_{c \cdot f} = L_f \oplus L_f \oplus \ldots \oplus L_f$. c times. Then $L_{c \cdot f} \in$REC, that is function $c \cdot f$ is recognizable. $\square$

Another results that we can easily establish is about inverse functions. Recall that all the functions to whom we are concerned have values in $\mathbb{N}$, therefore all the "inverse" functions should be considered in this context.

Lemma 10.2 *Let $f : \mathbb{N} \to \mathbb{N}$ be a function such that its inverse f^{-1} is also a function. If $f \in \mathcal{F}_{\mathrm{REC}}$ then its inverse $f^{-1} \in \mathcal{F}_{\mathrm{REC}}$.*

Proof: The result holds because of the closure under rotation of family REC (cf. Theorem 7.6). $\square$

We now give sufficient conditions for a function to be recognizable. Next we will show families of recognizable functions.

Theorem 10.1 *Let $f : \mathbb{N} \longrightarrow \mathbb{N}$ be a function satisfying the following recurrence equation*

$$\begin{cases} f(1) & = c \\ f(n+1) & = k \cdot f(n) + g(n) \end{cases} \tag{1}$$

where $c, k \in \mathbb{N}$ and $g \in \mathcal{F}_{\mathrm{REC}}$. Then, function $f \in \mathcal{F}_{\mathrm{REC}}$.

Proof: By definition of recognizable function and of recognizable picture language, it suffices to give a local two-dimensional language whose rectangles have size $(n, f(n))$, for $n \in \mathbb{N}$. Since function g is recognizable, then we have local rectangles of size $(n, g(n))$, for $n \in \mathbb{N}$.

We explain first how to construct the rectangles for f in the particular case when $k = 1$. The basic idea is to think the rectangles as built recursively from $n = 1, 2, \ldots$ as if they were built following the computation for the corresponding values of function f. We define first a rectangle for $f(1)$. Then, given the rectangle for $f(n)$, we construct that one corresponding to $f(n+1)$ by performing a column concatenation with the rectangle corresponding to $g(n)$ and adding a row of 1's below. (We assume that symbol 1 does not belong to the local alphabet for L_g.)

In the figures that will follow we indicate by $f(n)$ also the corresponding rectangle of the local two-dimensional language (e.g., the rectangle of size $(n, f(n))$. We give below the rectangle corresponding to value $f(4)$. Notice that this rectangle contains inside the rectangle corresponding to values $f(1)$ (explicitly written), the rectangle corresponding to values $f(2)$ (the one of size $(2, 10)$ in the top-right corner), and so on.

<table>
<tr><td>$f(1)$</td><td>$g(1)$</td><td rowspan="2">$g(2)$</td><td rowspan="3">$g(3)$</td></tr>
<tr><td colspan="2">1 1 1 1 1 1 1 1 1 1</td></tr>
<tr><td colspan="3">1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td></tr>
<tr><td colspan="4">1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td></tr>
</table>

If we choose in an appropriate way a representation for function $g(n)$, this kind of rectangles are actually local. The "acceptance conditions" can be given by defining in a convenient way the tiles of the border. For example, a rectangle that belongs to L_f will have all 1's in the last row. This implies that tiles for the bottom border will be only $\begin{array}{|c|c|}\hline 1 & 1 \\ \hline \# & \# \\ \hline \end{array}$. Moreover, we do not include tile $\begin{array}{|c|c|}\hline 1 & \# \\ \hline 1 & \# \\ \hline \end{array}$ among the tiles for the right border. This guarantees that, at each step of the recursive construction, we can add only one row of 1's.

As example we construct explicitly the local language for the function that satisfy the following equations.

$$\begin{cases} f(1) & = 1 \\ f(n+1) & = f(n) + n \end{cases}$$

We give below the rectangle corresponding to value $f(6)$. Notice that it contains inside the representations for values $f(1)$ up to $f(5)$.

0	b	b	c	b	c	c	b	c	c	c	b	c	c	c	c
1	1	a	b	a	b	c	a	b	c	c	a	b	c	c	c
1	1	1	1	a	a	b	a	a	b	c	a	a	b	c	c
1	1	1	1	1	1	1	a	a	a	b	a	a	a	b	c
1	1	1	1	1	1	1	1	1	1	1	a	a	a	a	b
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Notice that, differently from before (cf. Example 7.2), to represent the function $g(n) = n$ we have taken a local languages of squares over the three-letter alphabet $\{a, b, c\}$. This is because we need to define also concatenations between those squares. The set Θ that represents L_f will contain all the sub-blocks of size $(2, 2)$ of the rectangle above (completed with the border of # symbols).

The general case, e.g. when $k \geq 1$, can be dealt with by extending the same idea as in the case of $k = 1$. We give here only an intuition about how to proceed. Again we think the rectangles of the corresponding local language as constructed recursively, but this time, at the $(n+1)$th step we have to concatenate k different copies of rectangles corresponding to $f(n)$. This can be achieved by defining the set Θ in such a way that, for all n, we can construct k different rectangles corresponding to $f(n)$ that have symbols $1, 2, \ldots, k$, respectively, in the last row. For example, if $k = 4$, the rectangle corresponding to $f(n+1)$ will be the following.

<table>
<tr><td>$f(n)$
1 1 1 1 1 1 1 1</td><td>$f(n)$
2 2 2 2 2 2 2 2</td><td>$f(n)$
3 3 3 3 3 3 3 3</td><td>$f(n)$
4 4 4 4 4 4 4 4</td><td>$g(n)$</td></tr>
<tr><td colspan="5">1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td></tr>
</table>

Moreover, as before, we define border-tiles that will enforce us to accept only rectangles that have 1's in all positions of the last row. $\square$

Notice that the Equation 1 is quite general, that is by putting particular values in c, k and $g(n)$, we obtain a very large number of functions. In this sense, Theorem 10.1 can be considered as a *sufficient* condition for the recognizability of functions.

As application, let us consider Equation 1 with $c = 2$, $k = 1$ and $g(n) = 2n + 1$. This corresponds to function $f(n) = n^2$ (because of the equality $(n+1)^2 = n^2 + 2n + 1$). Then, Theorem 10.1 says that function $f(n) = n^2$ is recognizable. In general, using the classical Newton's formula for computing the n-th power of a binomial

$$(n+1)^k = n^k + \binom{k}{1} n^{k-1} + \binom{k}{2} n^{k-2} + \ldots + \binom{k}{k-1} n + 1$$

and proceeding by induction we obtain that functions $f(n) = n^k$ are recognizable for all $k \geq 2$. Therefore, because of Lemma 10.1, all polynomial functions with positive integer coefficients (e.g. the functions of the form $f(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_0$ with $k \geq 2$, $a_i \in \mathbb{N}$, $i = 0, \ldots, k$) are recognizable.

As another particular case, consider Equation 1 with $c = k = 2, 3, \ldots$ and $g(n) = 0$, $n \in \mathbb{N}$. The corresponding functions result of the form $f(n) = k^n$. Then the exponential functions, e.g. the functions of the form $f(n) = k^n$, $k \geq 2$, are all recognizable. Moreover, because the inverse of a recognizable function is also recognizable (cf. Lemma10.2), it holds that functions $f(n) = \sqrt[m]{n}$, $m \in \mathbb{N}$ as well as functions $f(n) = \log_b n$, $b \in \mathbb{N}$, are all recognizable.

Remark that logarithmic and exponential functions are tight upper and lower bounds for recognizable functions. This holds as immediate consequence of Iteration Lemma (cf. Lemma 9.1). In fact, let $f : \mathbb{N} \rightarrow \mathbb{N}$ be a super-exponential function (that is f is a function that grows faster than any exponential function; for example $f(n) = n!$) and let L_f be the language associates to f. Recall that the language associated to a function contains exactly one picture with n rows for each $n \in \mathbb{N}$. Then, since the function f is super-exponential, there are pictures in L_f satisfying the hypothesis of the Iteration Lemma. Consequently, L_f contains an infinite number of pictures with the same number of rows and therefore function f is not recognizable.

We state the following corollary.

Corollary 10.1 *All functions that grow slower than any logarithmic function or faster than any exponential function, are not recognizable.*

As final note, we mention that some work has been devoted in identifying classes of recognizable functions with respect to the definition of four-way automata recognizability. In [20] it is proved that the functions $f(n) = n^2$, $f(n) = k^n$, $f(n) = n!$ are all not acceptable by four-way finite automata. Then we can conclude that the family of picture languages defined by four-way automata is strictly included in REC even when restricted to the case of one-letter alphabet.

11 Beyond finite-state recognizability

In this chapter we have investigated the notion of finite state recognizability for two-dimensional languages. This notion has been introduced by using different formal models like automata, grammars, regular expressions, logic formulas and tiling systems. The equivalence stated between some of these different approaches indicates that the corresponding notion of recognizability is a "robust" one and it can be taken as the natural candidate for being the right generalization of the one-dimensional case.

The situation is completely different for the higher levels of the Chomsky hierarchy. In particular, a comparable theory of context-free picture languages has not been yet developed. Several attempts to extend this notion to two dimensions have been tried, but the theory lacks of elegant connections between the different approaches. We shortly indicate some of these proposals.

A first natural approach is in terms of grammars. In Section 5, we proposed a model of grammar (known as "matrix grammar") consisting of two sets of rewriting rules: horizontal and vertical rules, respectively. This model can be extended by considering contex-free rules instead of the right-linear ones. This can be done in several ways and gives rise to different notions of context–free picture languages (cf. [36, 29]).

As mentioned in Section 5, there exists another model to generate pictures: the "array grammars". In an array grammar, a derivation operates by replacing sub–arrays by sub–arrays. Also in this model it is possible to define something like "context-free rules". The families of languages obtained by array grammars are nevertheless very different from those obtained by matrix grammars. Some attempts to put together matrix grammars and array grammars were carried out in [39].

As far as the approach in terms of (sequential) automata is concerned, various model have been proposed that generalize 4FA's (cf. [21] for a surveys on the subject). However, it is not yet known any deep result analogous to that (of one-dimensional case) connecting the "push–down" acceptance and the "context–free" generation.

Let us finally indicate two new proposals generalizing some ideas and formal methods that have been successfully applied in this chapter to define finite-state recognizability.

The first proposal, is to extend to the context-free case the characterization of recognizable picture languages given in Theorem 8.6. This theorem states that a picture language is recognizable if and only if it can be obtained as a projection of the row-column composition of two recognizable string languages. Then, one can introduce the following definition.

Definition 11.1 *A picture language L over an alphabet Σ is* contex–free *if there exist two contex–free string languages S_1 and S_2 over an alphabet Γ and a projection $\pi : \Gamma \longrightarrow \Sigma$ such that $L = \pi(S_1 \oplus S_2)$.*

In this way context–free picture language L over Σ can be described by a triple (G_1, G_2, π), where G_1 and G_2 are two context–free grammars over the (terminal) alphabet Γ and $\pi : \Gamma \longrightarrow \Sigma$ is a projection.

As a consequence of Theorem 8.6, the family REC is included in the family of context–free picture languages just defined. It is easy to show that the inclusion is strict. For instance, if S_1 is a non-recognizable context–free string language over Σ and $S_2 = \Sigma$, then language $L = S_1 \oplus S_2$ contains one–row pictures only, and therefore corresponds to S_1 viewed as a picture language. Then L is not recognizable. However, remark that there exist non–recognizable context–free string languages S_1 and S_2 such that $L = S_1 \oplus S_2$ is a recognizable picture language. Consider, for instance, the following languages over the alphabet $\Sigma = \{a, b, c, d\}$.

$S_1 = \{a^n b^n \,|\, n \geq 1\} \cup \{c^n d^n \,|\, n \geq 1\}$

$S_2 = \{a^n c^n \,|\, n \geq 1\} \cup \{b^n d^n \,|\, n \geq 1\}$

Then $L = S_1 \oplus S_2$ is the set of all squares over Σ of the form:

a	a	a	a	b	b	b	b
a	a	a	a	b	b	b	b
a	a	a	a	b	b	b	b
a	a	a	a	b	b	b	b
c	c	c	c	d	d	d	d
c	c	c	c	d	d	d	d
c	c	c	c	d	d	d	d
c	c	c	c	d	d	d	d

It is easy to verify that $L \in$REC.

An interesting problem is to investigate the relationships between this definition of context–free picture language and those given in terms of grammars.

Another possible approach is suggested by the notion of tiling recognizability. Recall that this notion takes as a starting point the characterization of recognizable string languages as a projection of local languages. For context–free string languages there exists a similar characterization, the Chomsky–Schützenberger theorem (cf. [5] or [2]), stating that a string language S is context–free if and only if there exist a Dyck language D_n over n pairs of "parentheses", a recognizable language R and an alphabetic morphism ϕ such that:

$$S = \phi(D_n \cap R).$$

If we take this characterization as a starting point for a generalization of the notion of context–free language, the main problem is to devise a suitable definition of "two–dimensional Dyck language". For instance, a first possible choice is to define a two-dimensional Dyck language $D_{m,n}$ as the row–column composition of two Dyck languages: $D_{m,n} = D_m \oplus D_n$. A minor problem in this approach is to extend the notion of alphabetic morphism to two dimensions. Indeed, alphabetic morphisms can erase symbols while in a two–dimensional array the erasing can create holes. If we limit ourselves to consider non–cancellative morphism, i.e., projection, we obtain only pictures of even size.

A possible research direction is to search for other candidates which generalize the notion of Dyck language in a purely two–dimensional fashion (i.e., not as a derivation of the corresponding one-dimensional notion as in the previous definition) that further avoid the restrictions on the size of the pictures.

Acknowledgments

Several people contributed to this chapter. We are indebted to W. Thomas and S. Seibert for many useful discussions on the relationship between logic and tiling systems. We are grateful to M. Latteux and D. Simplot for sending us a copy of [25], and for many useful comments on domino systems, and to V. Bruyere for sending us a copy of A. Gilon's Thesis [16]. K. Inoue sent us many bibliographic references on two-dimensional automata. Finally, we thank O. Matz for his careful reading and valuable comments.

References

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

[2] J. Berstel. *Transductions and Contex-Free Languages.* Teubner. Stuttgart, 1979.

[3] J. R. Büchi. Weak second-order arithmetic and finite automata. *Z. Math, Logik Grundlagen Math.*. vol. 6, pp. 66–92, 1960.

[4] J. R. Büchi. On a decision methods in restricted second oreder arithmetic. Proc. Intern. Cong, in *Logic, Methodology and Philosophy of Science.* E. Nagel et al. Eds. Stanford Univ. Press, CA. pp. 1–11, 1961.

[5] N.Chomsky and M.P. Shützenberger. The algebraic theory of context-free languages. in *Computer Programming and Formal Systems.* P. Bradford and D. Hirschberg. Eds. North Holland. Amsterdam 1963, pp. 118–161.

[6] B. Courcelle. Graph rewriting: An algebraic and logic approach, in: *Handbook of Theoretical Computer Science*, Vol. B (J. v. Leeuwen, ed.), Elsevier, Amsterdam 1990, pp. 193-242.

[7] S. Eilenberg. *Automata, Languages and Machines.* Vol. A, Academic Press, 1974.

[8] H.D. Ebbinghaus, J. Flum and W. Thomas. *Mathematical Logic*, Springer-Verlag, Berlin, Heidelberg, New York 1984.

[9] R. Fagin. Monadic generalized spectra, *Z. math. Logik Grundl. Math.* 21 (1975), pp. 89-96.

[10] R. Fagin, L. Stockmeyer and M.Y. Vardi. On monadic NP vs. monadic co-NP. In: *Proc. 8th IEEE Conf. on Structure in Complexity Theory*, 1993, pp. 19-30.

[11] K.S. Fu. *Syntactic methods in pattern recognition.* Academic Press, New York 1974.

[12] D. Giammarresi. Two-dimensional languages and recognizable functions. In *Proc. Developments in language theory*, Finland, 1993. G.Rozenberg and A. Salomaa (Eds), pag. 290–301. *World Scientific Publishing Co.* , 1994.

[13] D. Giammarresi and A. Restivo. Recognizable picture languages. *International Journal Pattern Recognition and Artificial Intelligence.* Special issue on *Parallel Image Processing.* M. Nivat, A. Saoudi and P. S. P. Wang (Eds.), pages 31–46, 1992. Also in *Proc. First International Colloquium on Parallel Image Processing*, 1991. Same journal, Vol. 6, No. 2& 3, pages 241 –256, 1992.

[14] D. Giammarresi and A. Restivo. Two-dimensional finite state recognizability. *Fundamenta Informaticae.* Special Issue: *Formal Language Theory*, vol. 25, no. 3,4 (1996), 399–422.

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

[16] A. Gilon. *Langages d'images* Mémoire de Licenciée en Science Mathématiques réalisé sous la direction de V. Bruyere. Université de Mons-Hainaut, 1995.

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

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

[19] K. Inoue and A. Nakamura. Nonclosure properties of two-dimensional on-line tessellation acceptors and one-way parallel/sequential array acceptors. *Transaction of IECE of Japan*, Vol. 6, pages 475–476, 1977.

[20] K. Inoue and A. Nakamura. Two-dimensional finite automata and unacceptable functions. *Intern. J. Comput. Math.*, Sec. A, Vol. 7, pages 207– 213, 1979.

[21] K. Inoue and I. Takanami. A Survey of two-dimensional automata theory. In *Proc. 5th Int. Meeting of Young Computer Scientists*, J. Dasson and J. Kelemen (Eds.), pages 72–91. Lecture Notes in Computer Science 381, Springer-Verlag, Berlin, 1990.

[22] K. Inoue and I. Takanami. A Characterization of recognizable picture languages. In *Proc. Second International Colloquium on Parallel Image Processing*, A. Nakamura et al. (Eds.), Lecture Notes in Computer Science 654, Springer-Verlag, Berlin 1993.

[23] K. Inoue, I. Takanami and A. Nakamura. A note on two-dimensional finite automata. *Information Processing Letters*, Vol. 7, No. 1, pages 49–52, 1978.

[24] E.B. Kinber. Three-way automata on rectangular tapes over a one-letter alphabet. *Information Sciences*, Vol. 35, pages 61–77, 1985.

[25] M. Latteux and D. Simplot. Recognizable Picture Languages and Domino Tiling. Internal Report IT-94-264. *Laboratoire d'Informatique Fondamentale de Lille. Université de Lille*, France.

[26] O. Matz. *Classification of Picture Languages with Rational Expressions, Grammars and Logic Formulas* . Diploma thesis. Christian Albrechts Universitaet Institut fuer Informatik und Praktische Mathematik Kiel, Germany, 1996. (In German).

[27] R. Mc Naughton and S. Papert. *Counter-free Automata.* M.I.T. Press, 1971.

[28] M. Minski and S. Papert. *Perceptron.* M.I.T. Press, 1969.

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

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

[31] A. Rosenfeld. *Picture Languages (Formal Models for Picture Recognition).* Academic Press, New York, 1979.

[32] R. Siromoney. On equal matrix languages. *Info. and Control* 14, pp. 135–151, 1969.

[33] R. Siromoney. Advances in array languages. In *Graph-Grammars and Their Applications to Computer Science*, Ehrig et al. (Eds.), pages 549–563. Lecture Notes in Computer Science 291, Springer-Verlag, Berlin, 1987.

[34] R. Siromoney and K.Krithivasan. Parallel context-free languages. *Info. and Control* 24, pp. 155–162, 1974.

[35] G. Siromoney, R. Siromoney, K.Krithivasan. Picture languages with array rewriting rules. *Info. and Control* 22, pp. 447–470, 1973.

[36] R. Siromoney, K.Subramanian, K.Rangarajan. Parallel/Sequential rectangular arrays with tables. *Intern. Journ. Computer Math.* 6, pp. 143–158. 1977.

[37] H. Straubing. *Finite automata, Formal logic and Circuit complexity.* Birkhäusen, 1994.

[38] W. Thomas. On Logics, Tilings, and Automata. In *Proc. 18th Int. Colloquium on Automata, Languages and Programming*, pages 441–453. Lecture Notes in Computer Science no. 510, Springer-Verlag, Berlin, 1991.

[39] P.S. Wang. Sequential/Parallel matrix array languages. *Journal of Cybernetics*, vol. 5 pages 19–36, 1975.