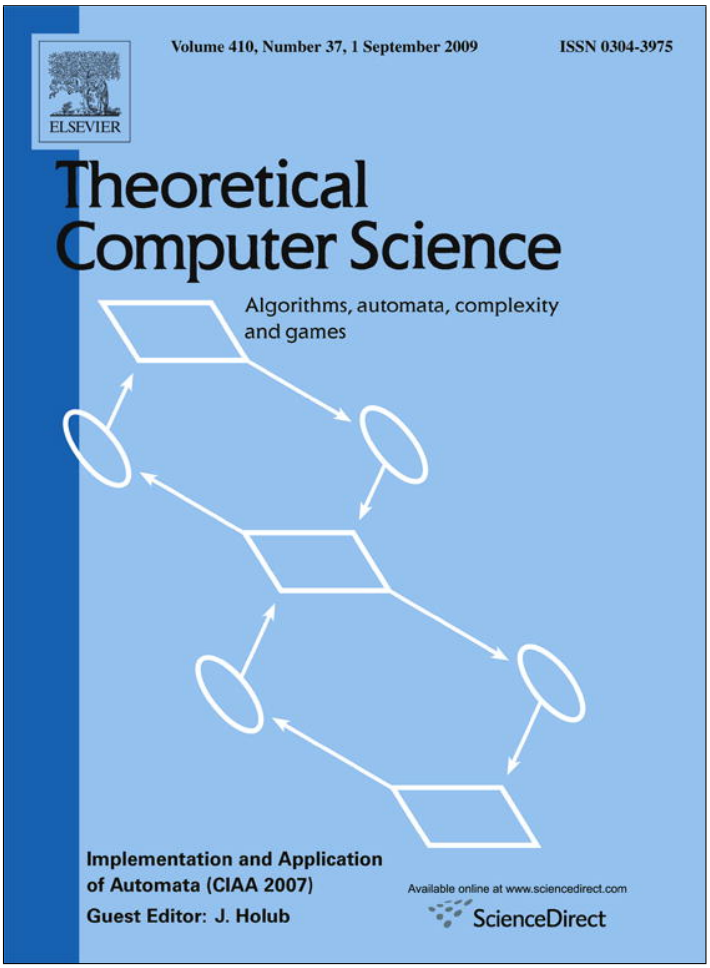


Theoretical Computer Science 410 (2009) 3520–3529

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

A computational model for tiling recognizable two-dimensional languages☆

Marcella Anselmo [a], Dora Giammarresi [b,*], Maria Madonia [c]

[a] Dip. di Informatica ed Applicazioni, Università di Salerno, I-84084 Fisciano (SA), Italy

[b] Dip. Matematica, Università di Roma "Tor Vergata", via Ricerca Scientifica, 00133 Roma, Italy

[c] Dip. Matematica e Informatica, Università di Catania, Viale Andrea Doria 6/a, 95125 Catania, Italy

ARTICLE INFO

Keywords:
Two-dimensional languages
Finite automata
Determinism

ABSTRACT

Tiling systems are a well accepted model to define recognizable two-dimensional languages but they are not an effective device for recognition unless a *scanning strategy* for the pictures is fixed. We define a *tiling automaton* as a tiling system equipped with a scanning strategy and a suitable data structure. The class of languages accepted by tiling automata coincides with the REC family. In this framework it is possible to define determinism, non-determinism and unambiguity. Then (deterministic) tiling automata are compared with the other known (deterministic) automata models for two-dimensional languages.

1. Introduction

Interest in problems such as pattern recognition has brought since the sixties to a theory of two-dimensional languages defined as sets of finite rectangular arrays of symbols from a finite alphabet. Different types of models have been introduced to generate or recognize two-dimensional objects, as a generalization of finite state automata, grammars and regular expressions. The main goal was to define a class of (finite state) recognizable two-dimensional languages that generalizes the class of regular string languages together with all its properties.

The first automaton model for two-dimensional languages was introduced by M. Blum and C. Hewitt [5] and called *four-way automaton (4FA)*. It generalizes the two-way automaton for strings by allowing the input head to move up and down besides right and left. Many other automaton-like devices have been introduced so far [4,5,13–15]. Furthermore, many other approaches appeared in the literature following other classical ways to define regular languages, namely grammars, logics and regular expressions [8]. All those models are defined by generalizing some specific definitions or properties of recognizable string languages but, as result, mostly they correspond to different classes of "recognizable" two-dimensional languages that are difficult to compare. In this paper we consider the family *REC* of *tiling recognizable picture languages* introduced in [7] (see also [8]). A two-dimensional language is *tiling system recognizable* if it can be obtained as a projection of a local picture language. This definition generalizes to two dimensions a characterization of a finite automaton for strings (local set plus a projection are effectively a description of the state-graph of an automaton). The REC family inherits several properties from the class of regular string languages and it has characterizations also in terms of logic formulas and of some kind of regular expressions [8]. It is worth remarking that tiling systems generalize *non-deterministic* automata for

☆ Partially supported by MIUR Project *"Automi e Linguaggi Formali: aspetti matematici e applicativi"* (2005), by ESF Project *AutoMathA* (2005–2010), by 60% Projects of University of Catania, Roma "Tor Vergata", Salerno (2006, 2007).

* Corresponding author.
E-mail addresses: anselmo@dia.unisa.it (M. Anselmo), giammarr@mat.uniroma2.it (D. Giammarresi), madonia@dmi.unict.it (M. Madonia).

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.03.016

strings and that, unlike the one-dimensional case, the REC family is not closed under complement. Very recently definitions of deterministic and unambiguous versions of REC families were studied in [2,3] while some necessary conditions for a language to be in REC or in REC ∪ co-REC were given in [9].

The starting motivation of this paper is the major observation that despite that a tiling system generalizes to two dimensions a conventional finite automaton, it does not correspond to an effective procedure of recognition. To verify that a picture p belongs to a given language, we have to look for a counter-image p' in the local alphabet that matches the set of tiles. Even if the set of tiles of the local language describes in one dimension the transitions of a finite automaton, to get an automaton in the usual sense we need also a procedure to process the input and to define the steps of computations (where they start, where/when they accept the input). Observe that, in the case of conventional finite automata for strings, a scanning procedure is (implicitly) fixed *a priori* for all input strings: it prescribes to read the input starting from the leftmost position and to go right at each step. Then, separately, we have a transition function that takes a pair (state, symbol) and gives the new state.

The idea is that to let tiling systems to become "real automata" we have to provide them with a scanning procedure for the input: we have to fix, for each picture, the starting position and the sequence of all positions we read. Moreover to apply the transition function described as set of tiles, it is necessary a data structure that maintains the needed information. We call this device a *tiling automaton*. We remark that the language accepted by a tiling automaton will not depend on the scanning strategy associated to the automaton (it belongs to the REC class anyway) despite the scanning strategy matters in the case of definition of *deterministic tiling automata*.

Recall that defining a scanning procedure for a two-dimensional object is a well studied problem in pattern recognition context. In this paper we consider a different approach that is suitable for automata definitions. We give a formal definition of a scanning strategy by means of the *next-position function* that has major properties of being computable, starting from a corner, going by contiguous positions and covering all the picture positions. In two dimensions it is possible to define several different scanning strategies with the prescribed properties: we will focus on those that follow corner-to-corner directions and give explicit examples. The computation of a tiling automaton is defined as a sequence of instantaneous configurations that include the content of a specific data structure. The transition function, given by the set of tiles, rules the passage from an instantaneous configuration to the next one. We introduce also the unambiguous and the deterministic version of the model. Then we compare tiling automata with other finite devices for recognizing two-dimensional languages as tiling systems, on-line tessellation automata and 4-way automata together with their deterministic and unambiguous counterparts.

In this paper, we first recall some basic definitions and notations for two-dimensional languages together with some results about the REC family. Then we propose motivations and intuitions for the definition of tiling automata to prepare the formal definitions given later. Finally we give all comparison results. A preliminary version of this paper appeared in [1].

2. Preliminaries

We introduce some definitions about two-dimensional languages. The notations used and more details can be mainly found in [8].

A *two-dimensional string* (or a *picture*) over a finite alphabet Σ is a two-dimensional array of elements of Σ. The set of all pictures over Σ is denoted by Σ^{**} and a *two-dimensional language* over Σ is a subset of Σ^{**}. Given a picture $p \in \Sigma^{**}$, we let $p_{(i,j)}$ denote the symbol in p with coordinates (i, j) where position $(1, 1)$ corresponds to top-left corner. Moreover if p has m rows and n columns we refer to the pair (m, n) as the *size* of the picture p. For any picture p of size (m, n), we consider the *bordered picture* $\widehat{p}$ of size $(m + 2, n + 2)$ obtained by surrounding p with a special *boundary symbol* $\# \notin \Sigma$.

A *tile* is a picture of size $(2, 2)$ and $B_{2,2}(p)$ is the set of all subpictures of size $(2, 2)$ of a picture p. Given an alphabet Γ, a two-dimensional language $L \subseteq \Gamma^{**}$ is *local* if there exists a finite set Θ of tiles over $\Gamma \cup \{\#\}$ (the set of *allowed blocks*) such that $L = \{p \in \Gamma^{**} | B_{2,2}(\widehat{p}) \subseteq \Theta\}$ and we will write $L = L(\Theta)$. A *tiling system* is a quadruple $(\Sigma, \Gamma, \Theta, \pi)$ where Σ and Γ are finite alphabets, Θ is a finite set of tiles over $\Gamma \cup \{\#\}$ and $\pi : \Gamma \to \Sigma$ is a projection. A two-dimensional language $L \subseteq \Sigma^{**}$ is *tiling recognizable* if there exists a tiling system $(\Sigma, \Gamma, \Theta, \pi)$ such that $L = \pi(L(\Theta))$ (extending π in the usual way).

We denote by *REC* the family of all *tiling recognizable* picture languages. The REC is closed under union, intersection, rotation and under some concatenation operations. The main important difference with the one dimensional case is that the REC is *not* closed under complement (see [8]). Very recently *DREC* and *UREC* classes have been introduced (see [2,7]) as deterministic and unambiguous subfamilies of REC, respectively. More exactly (see [2]) a language L is in DREC if it admits a d-deterministic tiling system for some direction d in the set of the possible directions from a corner to the opposite one, i.e. $d \in \{tl2br, tr2bl, bl2tr, br2tl\}$. Here $tl2br$ is the short form for *from top-left corner to bottom-right corner* and similarly for the other corner-to-corner directions. A tiling system $(\Sigma, \Gamma, \Theta, \pi)$ is *tl2br-deterministic* if, for any $\gamma_1, \gamma_2, \gamma_3 \in \Gamma \cup \{\#\}$ and $\sigma \in \Sigma$, there is at most one tile $\begin{array}{|c|c|}\hline \gamma_1 & \gamma_2 \\ \hline \gamma_3 & \gamma_4 \\ \hline \end{array} \in \Theta$, with $\pi(\gamma_4) = \sigma$. The definitions for the other possible directions d are analogous.

A language L is in UREC if it admits an unambiguous tiling system and a tiling system $(\Sigma, \Gamma, \Theta, \pi)$ is *unambiguous* if, for any picture $p \in L$, there exists a unique local picture $p' \in L(\Theta)$ such that $p = \pi(p')$ (see [7]). In [2] it is shown that DREC⊂UREC⊂REC with all strict inclusions.

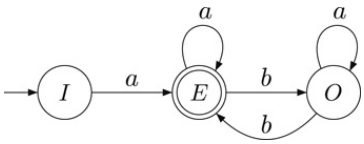


Fig. 1. Automaton in Example 2.

Example 1. Let $L_{lastr=lastc}$ be the language of squares over $\Sigma = \{a, b\}$ with the last row equal to the last column. We show that $L_{lastr=lastc} \in REC$ by describing a tiling system $(\Sigma, \Gamma, \Theta, \pi)$ recognizing it. The tiling system is such that, for any picture p, the information on each letter of the last row is brought up to the diagonal and then right towards the last column. More precisely, we use a local alphabet $\Gamma = \{x_y^0, x_y^1, x_y^2\}$ with $x, y \in \{a, b\}$ and define $\pi(x_y^0) = \pi(x_y^1) = \pi(x_y^2) = x$. The superscript symbol 0 occurs only in positions below the diagonal, symbol 1 occurs only on the diagonal and symbol 2 only above the diagonal while the subscript symbols correspond to information we are bringing from the last row to the last column (making a turn at the diagonal). Here below it is given a picture $p \in L_{lastr=lastc}$ together with a corresponding local picture p' (i.e. $\pi(p') = p$).

$p =$

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

$p' =$

b_a^1	a_a^2	b_a^2	a_a^2
a_a^0	a_b^1	b_b^2	b_b^2
a_a^0	b_b^0	b_a^1	a_a^2
a_a^0	b_b^0	a_a^0	a_a^1

An interesting model of an automaton to recognize picture languages is the *on-line tessellation automaton* (*OTA*) introduced in [10]. In a sense the OTA is an infinite array of identical finite-state automata in a two dimensional space. The computation goes by anti-diagonals starting from top-left towards bottom-right corner of the picture. A run of an OTA on a picture consists of associating a state to each position of the picture. The state for some position (i, j) is given by the transition function and depends on symbol in that position and on the states already associated to positions $(i, j-1)$, $(i-1, j-1)$ and $(i-1, j)$. An unambiguous on-line tessellation automaton (*UOTA*) is an OTA where each picture admits one successful run at most. A deterministic version of an OTA is referred to as *DOTA*. The family of languages recognized by the different versions of the model (L(*OTA*), L(*UOTA*), L(*DOTA*)) are different. Although this kind of automaton is quite difficult to manage, this is actually the machine counterpart of a tiling system: in [11], it is proved that REC= L(OTA).

Another model of two-dimensional automaton is the *4-way automaton* (*4NFA* or *4DFA* for the deterministic version). It is defined as an extension of the two-way automaton for strings (cf. [5]) by allowing it to move in four directions: *Left, Right, Up, Down*. It is proved that also for 4-way automata, the deterministic version of the model defines a class of languages (L(*4DFA*)) smaller than the corresponding one defined by non-deterministic version (L(*4NFA*)) (see [5,8]).

3. Toward a definition of tiling automaton

Tiling systems are models to recognize two-dimensional languages, but really they are not an effective computation device. We now want to define a computational device for accepting two-dimensional languages based on tiling systems. The resulting device is what we call a *Tiling Automaton*. In this section we will discuss motivations and intuitions to lead the reader to the formal definition that will be given in the next section.

We try first to understand precisely the situation in a one-dimensional case to make an accurate generalization to two dimensions. Consider the proof in [6] that shows that a (string) language is accepted by a finite automaton if and only if it is the projection of a local language of words. Let $L \subseteq \Sigma^*$ and $\mathcal{A}$ be a finite automaton recognizing L. The tiling system recognizing L corresponding to $\mathcal{A}$ is $\mathcal{T}_{\mathcal{A}} = (\Sigma, \Gamma, \Theta, \pi)$ where Γ is the set of transitions, Θ is the set of pairs of consecutive transitions (plus some pairs with # symbol in correspondence to transitions from the initial state and to final states), and π is the projection that deletes the states in the transition. Deciding whether a string w belongs to L is accomplished in the automaton by looking for a successful path labeled w; while the analogous steps are accomplished via the tiling system by looking for a local string $w' \in L(\Theta)$ such that $\pi(w') = w$. Let us fix this in an example.

Example 2. Let $\Sigma = \{a, b\}$ and $L \subseteq \Sigma^*$ be the set of words starting with a and with an even number of occurrences of b. L is accepted by the automaton $\mathcal{A}$ in Fig. 1.

States are I, E, O (where I stands for initial, E for even and O for odd). Tiling system $\mathcal{T}_{\mathcal{A}} = (\Sigma, \Gamma, \Theta, \pi)$ is as follows:
$\Gamma = \{\gamma_1 = (I, a, E), \gamma_2 = (E, a, E), \gamma_3 = (E, b, O), \gamma_4 = (O, a, O), \gamma_5 = (O, b, E)\}$
$\Theta = \{(\sharp, \gamma_1), (\gamma_1, \sharp), (\gamma_2, \sharp), (\gamma_5, \sharp)\} \cup \{(\gamma_i, \gamma_j) \mid \gamma_i = (x_i, \sigma_i, y_i), \gamma_j = (x_j, \sigma_j, y_j) \text{ and } y_i = x_j\}$
$\pi((x, \sigma, y)) = \sigma$ for any $(x, \sigma, y) \in \Gamma$.

Consider for example $w = abaababb$. The word w is accepted by $\mathcal{A}$ since there is the following successful path in $\mathcal{A}$ labeled w: $(\gamma_1, \gamma_3, \gamma_4, \gamma_4, \gamma_5, \gamma_2, \gamma_3, \gamma_5)$. In a similar way, the word w is recognized by $\mathcal{T}_{\mathcal{A}}$ since it can be recovered by the following tiles: $(\sharp, \gamma_1)$, (γ_1, γ_3), (γ_3, γ_4), (γ_4, γ_4), (γ_4, γ_5), (γ_5, γ_2), (γ_2, γ_3), (γ_3, γ_5), $(\gamma_5, \sharp)$. In this way we have reconstructed a word in $L(\Theta)$, namely $w' = \gamma_1\gamma_3\gamma_4\gamma_4\gamma_5\gamma_2\gamma_3\gamma_5$ whose projection by π is $\pi(w') = w$. □

Keeping this example in mind, consider again language L and start now from the tiling system $\mathcal{T}_{\mathcal{A}}$ we have just constructed. In a sense, what we call tiling system, in one dimension, represents the transition function of an automaton where edges are actually *non-oriented* edges. The whole automaton is recovered by assuming the conventional way of

reading the string (starting from leftmost border and proceeding from left to right) and therefore by choosing a direction for such edges. Recall that we could also assume to read the string starting from the rightmost border and going left (probably more natural for Arabs!): from the same tiling system we would get another automaton that accepts the same language but processes strings from right to left (indeed it can be also viewed as the automaton that accepts the reverse of the language L reading from left to right, but this will not be our point of view).

In order to use a tiling system as a computational device for strings, what is usually (implicitly) understood are the following "instructions": "Start on the leftmost border of the string. If you are in position i then go to position $i+1$. Use local symbol γ_i associated to position i in order to compute γ_{i+1} looking for a tile $(\gamma_i, \gamma_{i+1}) \in \Theta$".

Previous instructions give an effective computation since:

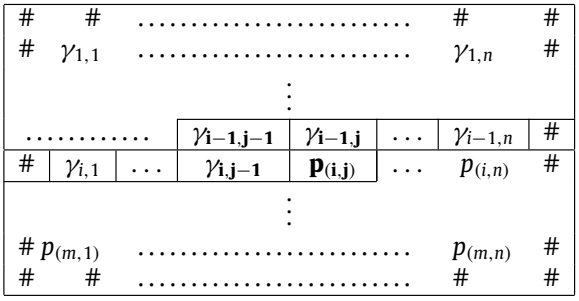
(1) each position of the string is visited (and each one exactly once)
(2) when we are in position i the local symbol γ_{i-1}, required for the computation, has already been computed
(3) we can easily keep in memory the last computed local symbol.

We are going to extend these reasonings to two dimensions and to define a computational device consisting of a tiling system equipped with a scanning strategy and a data structure that ensure properties 1, 2, 3. Remark that the "usually understood instructions" in one dimension, become much more complicated in two dimensions, but, at the same time, they will play a more important role in the recognition power of the model (see Remark 18). For a scanning strategy on a picture we have much more possibilities since we are going from writing in a single tape to writing in a whole sheet.

To fix the ideas with an example, we choose the scanning strategy that seems "the most natural one": for any picture, it goes, row by row, from left to right and from top to bottom (probably oriental people would find more natural going column by column!). So suppose to have a tiling system $\mathcal{T} = (\Sigma, \Gamma, \Theta, \pi)$ that we want to use as a computational device to recognize pictures of the language L defined by $\mathcal{T}$: with this scanning strategy, it will read any picture $p \in \Sigma^{**}$ row by row in order to decide if $p \in L$, i.e. if there exists $p' \in \Gamma^{**}$ such that $p = \pi(p')$. Let us see in detail how this could be accomplished.

First position of the scanning is the top-left corner. Then we need a function that, at any step of the computation, gives the next position to be scanned. Observe that, to handle the borders, this position depends not only on the current position but also on the size (m, n) of the picture: if the current position (i, j) is the last one in the i-th row, then the next position is the first one in the $(i+1)$-th row, otherwise the next position is $(i, j+1)$. This can be formalized by considering a *next-position function* $f(i, j, m, n)$.

Applying iteratively such a next-position function to the current position, we obtain a complete scanning sequence for the input picture. Note also that, when we have reached position (i, j), we have already visited its top-left contiguous positions (i.e. positions $(i-1, j-1)$, $(i-1, j)$, $(i, j-1)$) and so we have already "chosen" the local symbols for those positions (as in the figure below).



Now, at this step of the computation, it is possible to choose a suitable tile for the four positions (i, j), $(i-1, j-1)$, $(i-1, j)$ and $(i, j-1)$ and compute a local symbol for (i, j). Remark that for the computation, it is necessary to remember some of the local symbols associated to the positions of p already scanned. In particular when we have reached position (i, j), we need to remember the local symbols in the positions of the $(i-1)$-th row, from the $(j-1)$-th column to the last one, and the local symbols in the positions of the i-th row, from the first column to the $(j-1)$-th one (the ones pointed out in figure above). We do this with a suitable data structure. For example, we can use a list of $n+3$ symbols. After applying the transition function (i.e. using a tile to get the next local symbol) the data structure is consequently updated. Referring to the figure above, the data structure at that step is the following: $(\gamma_{i-1,j-1}, \gamma_{i-1,j}, \ldots, \gamma_{i-1,n}, \#, \#, \gamma_{i,1}, \ldots, \gamma_{i,j-1})$. Then using tile $\begin{array}{|c|c|}\hline \gamma_{i-1,j-1} & \gamma_{i-1,j} \\ \hline \gamma_{i,j-1} & \gamma_{i,j} \\ \hline \end{array}$ with $\pi(\gamma_{i,j}) = p_{(i,j)}$ it will be updated as follows: $(\gamma_{i-1,j}, \ldots, \gamma_{i-1,n}, \#, \#, \gamma_{i,1}, \ldots, \gamma_{i,j-1}, \gamma_{i,j})$. Remark that at each step, symbols $\gamma_{i-1,j-1}$, $\gamma_{i-1,j}$, $\gamma_{i,j-1}$ necessary to compute the tile, can be easily extracted from the list as the first, the second and the last element in the list.

To sum up briefly, our computational device will be defined by a tiling system plus a scanning strategy (that uses a next-position function) plus a data structure (supporting some operations). We remark that we can adopt other scanning strategies: we will get a different tiling automaton but the class of recognized languages will be the same, namely the REC. The main difference will arise in the case of a definition of deterministic automaton. In particular (see Remark 18) it can be proved that, depending on the chosen scanning strategy, deterministic tiling automata define different classes of languages.

All these considerations give the evidence that a tiling automaton should be defined without fixing a particular (natural) scanning strategy for the pictures, but indeed in a much more general setting.

4. Tiling automata: Formal definition and examples

At this point we have collected all the ingredients (motivations + intuitive notion) and thus we are ready to formalize a definition of an automaton for two-dimensional languages based on a tiling system. We will define first a next-position function, then a scanning strategy, and finally the corresponding tiling automaton with the support of a data structure that maintains all necessary information.

We start with some terminology. A *position* in a picture of size (m, n) is any pair in $P(m, n) = \{0, 1, \ldots, m+1\} \times \{0, 1, \ldots, n+1\}$. *External positions* are pairs (i, j) with $i = 0$, $i = m+1$, $j = 0$ or $j = m+1$, while *internal positions* are the non-external ones. *Corner positions* are the ones in the set $\{(0, 0), (0, n+1), (m+1, 0), (m+1, n+1)\}$. Given a position (i, j) with $i = 1, \ldots, m+1$ and $j = 1, \ldots, n+1$ its *top-left-* (*tl-* for short) *contiguous positions* are the positions: $(i, j-1)$, $(i-1, j-1)$, and $(i-1, j)$. In a similar way for *tr*, *bl*, and *br*. Here *t*, *b*, *l*, and *r* are used for *top*, *bottom*, *left* and *right* respectively. For any internal position, its *contiguous positions* are all the *tl-*, *tr-*, *br-*, and *bl*-ones.

Definition 3. A *next-position function* for pictures is a computable partial function $f : \mathbb{N}^4 \to \mathbb{N}^2$ that associates to a quadruple (i, j, m, n), with $(i, j) \in P(m, n)$, a position $(i', j') \in P(m, n)$.

The sequence of *visited positions* by f at step k, starting from position (i_0, j_0), is sequence $V_{f,k}(m, n) = (v_1(m, n), v_2(m, n), \ldots, v_{k-1}(m, n))$ where $v_1(m, n) = (i_0, j_0)$ and for any $h = 2, \ldots, k-1$, $v_h(m, n) = f(i, j, m, n)$ with $(i, j) = v_{h-1}(m, n)$.

Definition 4. A *scanning strategy* is a next-position function $\mathcal{S}$ such that for any $m, n \in \mathbb{N}$ the sequence of visited positions by $\mathcal{S}$ at step $(m+2)(n+2)+1$ starting from a corner, $V(m, n) = (v_1(m, n), v_2(m, n), \ldots, v_{(m+2)(n+2)}(m, n))$, satisfies:

(1) $V(m, n)$ is a permutation of $P(m, n)$
(2) for any $k = 2, \ldots, (m+2)(n+2)$, the tl- (or tr-, or bl-, or br- resp.) contiguous positions of $v_k(m, n)$ (when defined) are all in $V_{\mathcal{S},k}(m, n)$.
    Furthermore, a scanning strategy is said to be *continuous* when it satisfies condition 3 below; it is said to be *normalized* when it satisfies condition 4.
(3) for any $k = 2, \ldots, (m+2)(n+2)$, $v_k(m, n)$ is a contiguous position of $v_{k-1}(m, n)$ unless eventually when $v_{k-1}(m, n)$ is an external position and, in this case, $v_k(m, n)$ is an external position too.
(4) $v_{(m+2)(n+2)}(m, n)$ is a corner.

Note that when a next-position function is given, there is one starting corner at most, verifying conditions 1 and 2. Property 3 forbids "taking the pen off" the picture unless on external positions; in this case we may jump only to some other external position. This condition avoids that two non-contiguous regions of a picture are both scanned during a scanning process. Note that some spiral-like scanning strategies (as in Example 5) is continuous, while allowing the presence of "holes" in the scanned region. Properties 3 and 4 together forbid the existence of "holes" in the picture during the scanning process, because once a hole is filled we could not jump to a final corner. The following examples show the richness and extent of possibilities that arise when going from one to two dimensions.

Example 5. We list some possible scanning strategies for pictures.

(1) The strategy of scanning pictures row by row from left to right and from top to bottom (as in the previous section) can be formalized as follows. The next-position function $\mathcal{S}_{row}$ is defined on (i, j, m, n) such that $(i, j) \in P(m, n)$ and $(i, j) \neq (m+1, n+1)$ by:
$$\mathcal{S}_{row}(i, j, m, n) = \begin{cases} (i, j+1) & \text{if } j \leq n \\ (i+1, 1) & \text{if } j = n+1, i \leq m \end{cases}$$
In this case, for any $m, n \in \mathbb{N}$, $v_1(m, n) = (0, 0)$. $\mathcal{S}_{row}$ is continuous and normalized.
In a similar way one can also depict a scanning strategy $\mathcal{S}_{col}$ that starts in the bottom-right corner and proceeds, column by column, from bottom to top and from right to left.
(2) The scanning strategy $\mathcal{S}_{diag}$ starts in the top-left corner and proceeds following the counter-diagonal direction from top-right to bottom-left. The next-position function $\mathcal{S}_{diag}$ is defined on (i, j, m, n) such that $(i, j) \in P(m, n)$ and $(i, j) \neq (m+1, n+1)$ by:
$$\mathcal{S}_{diag}(i, j, m, n) = \begin{cases} (i+1, j-1) & \text{if } j \neq 0 \text{ and } i \neq m+1 \\ (0, i+j+1) & \text{if } j = 0 \text{ or } i = m+1, i+j \leq n \\ (i+j-n, n+1) & \text{otherwise} \end{cases}$$
In this case, for any $m, n \in \mathbb{N}$, $v_1(m, n) = (0, 0)$. $\mathcal{S}_{diag}$ is continuous and normalized.
(3) Let $\mathcal{S}_{snake}$ be the scanning strategy that starts in position $(0, 0)$ and proceeds snake-like. The next-position function $\mathcal{S}_{snake}$ is defined on (i, j, m, n) such that $(i, j) \in P(m, n)$ and $(i, j) \neq (m+1, n+1)$ if m is odd, $(i, j) \neq (m+1, 0)$ if m is even, as follows:

$$\mathcal{S}_{snake}(i,j,m,n) = \begin{cases} (i, j+1) & \text{if } i \text{ is even and } j \leq n \\ (i+1, n+1) & \text{if } i \text{ is even and } j = n+1 \\ (i, j-1) & \text{if } i \text{ is odd and } j \geq 1 \\ (i+1, 0) & \text{if } i \text{ is odd and } j = 0 \end{cases}$$

$\mathcal{S}_{snake}$ satisfies conditions 1–4. Note that, contrarily to previous cases, for positions of i-th rows with even i, the tl-contiguous positions are already visited, while for odd i, the tr-contiguous positions are in the sequence of visited positions.

(4) Consider a next-position function defined as S_{snake} on the left half-part of the picture, and any on the remaining right half-part. This is not a scanning strategy since, after scanning position $(0, \lfloor n/2 \rfloor)$ the next position $(1, \lfloor n/2 \rfloor)$ has not 3 contiguous positions (against property 2).

(5) A spiral-like way of proceeding is a continuous but not normalized scanning strategy, since it does not stop in a corner. During the spiral-like scanning process there are holes.

(6) Consider proceeding row by row from top to bottom scanning each row from left to right until the "middle" position is reached, and then from right to left. This is a scanning strategy, but it is neither continuous nor normalized.

In order to introduce tiling automata, we now investigate, among all possible scanning strategies, which ones have the properties that fit our purposes. As it will be proved later (see Remark 18), when determinism is considered, deterministic tiling automata define different classes of languages, depending on the chosen scanning strategy. On the other hand, to have a definition that generalizes the one-dimensional case, and to make the definition more effective (where determinism is a decidable property, see Corollary 15), we will restrict to a particular type of scanning strategies, that somehow follow a main corner-to-corner direction. Remark that while in a one-dimensional case, there are only two possible directions (from the left and from the right) and once a direction is fixed there is only one scanning strategy compatible with that direction, in two-dimensional case, even if a main direction is fixed, there is an huge number of scanning strategies along that direction.

Definition 6. A scanning strategy $\mathcal{S}$ is *tl2br-directed* if for any $(m, n) \in \mathbb{N} \times \mathbb{N}$ and $k = 1, \ldots, (m+2)(n+2)$ the tl-contiguous positions of $v_k(m, n)$ (when defined) are in the set of visited positions at step k. In a similar way define d-directed scanning strategies for any corner-to-corner direction d. A scanning strategy is said *corner-to-corner directed* (*c2c-directed*) if it is d-directed for some corner-to-corner direction.

Example 7. Referring to Example 5, the scanning strategies $\mathcal{S}_{row}$ and $\mathcal{S}_{diag}$ are $tl2br$-directed, $\mathcal{S}_{col}$ is $br2tl$-directed, while $\mathcal{S}_{snake}$ is not $c2c$-directed.

According to a scanning strategy, a tiling system becomes a device able to effectively process a picture and decide whether it has to be accepted or not, whenever we can (easily) keep track of all information needed for the next steps of the computation. In other words for any scanning strategy, we need a proper data structure that supports operations of retrieval of the three local symbols defined in the three contiguous positions and of updating of structure itself. In the simple example in Section 3, such a data structure was a list.

Definition 8. A *tiling automaton* of type $tl2br$ is a quadruple $\mathcal{A} = (\mathcal{T}, \mathcal{S}, D_0, \delta)$ where $\mathcal{T} = (\Sigma, \Gamma, \Theta, \pi)$ is a tiling system, $\mathcal{S}$ is a tl2br-directed scanning strategy, D_0 is the initial content of a data structure that supports operations $\texttt{state}_1\texttt{(D)}$, $\texttt{state}_2\texttt{(D)}$, $\texttt{state}_3\texttt{(D)}$, $\texttt{update(D,}\gamma\texttt{)}$, for $\gamma \in \Gamma \cup \{\#\}$, and $\delta : (\Gamma \cup \{\#\})^3 \times (\Sigma \cup \{\#\}) \to 2^{\Gamma \cup \{\#\}}$ is a partial function such that $\gamma_4 \in \delta(\gamma_1, \gamma_2, \gamma_3, \sigma)$ if the tile $\begin{array}{|c|c|}\hline \gamma_1 & \gamma_2 \\ \hline \gamma_3 & \gamma_4 \\ \hline \end{array} \in \Theta$ and $\pi(\gamma_4) = \sigma$ if $\sigma \in \Sigma$, $\gamma_4 = \#$, otherwise.

Similarly, we define a tiling automaton of type d for any corner-to-corner direction d. A tiling automaton (*TA* for short) is a tiling automaton of any type d.

Definition 9. A tiling automaton $\mathcal{A} = (\mathcal{T}, \mathcal{S}, D_0, \delta)$ is *deterministic* if for any $\gamma_1, \gamma_2, \gamma_3 \in \Gamma \cup \{\#\}$ and $\sigma \in \Sigma \cup \{\#\}$ there exists at most one symbol γ_4 such that $\gamma_4 \in \delta(\gamma_1, \gamma_2, \gamma_3, \sigma)$.

Let us define how the computation of a tiling automaton goes on a picture and when it accepts. An *instantaneous configuration* of $\mathcal{A} = (\mathcal{T}, \mathcal{S}, D_0, \delta)$ is a quadruple (p, i, j, D) where p is a picture, $(i, j) \in \mathbb{N} \times \mathbb{N}$, and D is the content of the data structure. The initial instantaneous configuration for an input picture p of size (m, n) is (p, i_0, j_0, D_0) with $(i_0, j_0) = v_1(m, n)$; D_0 plays the role of an "initial state". The next instantaneous configuration is given by the relation $(p, i, j, D) \vdash (p, i', j', D')$ defined as follows. When $f(i, j, m, n)$ and $\delta(\texttt{state}_1\texttt{(D)}, \texttt{state}_2\texttt{(D)}, \texttt{state}_3\texttt{(D)}, p_{(i,j)})$ are both defined, $(p, i, j, D) \vdash (p, i', j', D')$ if $(i', j') = f(i, j, m, n)$ and D' is the content of the data structure after calling $\texttt{update(D,}\gamma_4\texttt{)}$, with $\gamma_4 \in \delta(\texttt{state}_1\texttt{(D)}, \texttt{state}_2\texttt{(D)}, \texttt{state}_3\texttt{(D)}, p_{(i,j)})$. Then, as usual, one can denote by $\vdash^*$ the reflexive and transitive closure of $\vdash$.

The language (of all pictures) accepted by $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of pictures p such that $(p, i_0, j_0, D_0) \vdash^* (p, i, j, D)$ and $f(i, j, m, n)$ is not defined. Note that particular attention should be paid when dealing with external positions.

Example 10. Let $\mathcal{T} = (\Sigma, \Gamma, \Theta, \pi)$ be a tiling system for a language L. We construct a tiling automaton $\mathcal{A}_{col} = (\mathcal{T}, \mathcal{S}_{col}, D_{col_0}, \delta_{col})$, for L, based on $\mathcal{T}$ and on the scanning strategy $\mathcal{S}_{col}$, as in Example 5. We need to define the data structure associated to $\mathcal{S}_{col}$ and how it can (efficiently) support the operations $\texttt{state}_1\texttt{(D)}$, $\texttt{state}_2\texttt{(D)}$, $\texttt{state}_3\texttt{(D)}$, $\texttt{update(D,}\gamma\texttt{)}$, as in Definition 8.

As in Section 3, we can use a list of size $m+3$ for a picture p of size (m,n), with pointers to the first, second and last elements. At the beginning D_{col_0} is empty and reading the last column (column of index $n+1$) $m+2$ symbols # are inserted. Then, when position (i,j) of p is considered, the list D stores (in order) the symbols associated to the visited positions $(i+1,j+1)$, $(i,j+1)$, …, $(0,j+1)$, $(m+1,j)$, …, $(i+1,j)$. The call to $\mathtt{state}_1(\mathtt{D})$, $\mathtt{state}_2(\mathtt{D})$, $\mathtt{state}_3(\mathtt{D})$, will return (in O(1) time) the first, second and last symbols in the list that are the ones in positions $(i+1,j+1)$, $(i,j+1)$ and $(i+1,j)$, say γ_1,γ_2 and γ_3, respectively. Value $\gamma_4 \in \delta_{col}(\gamma_1,\gamma_2,\gamma_3,p_{(i,j)})$ is such that $\begin{array}{|c|c|}\hline \gamma_4 & \gamma_2 \\ \hline \gamma_3 & \gamma_1 \\ \hline \end{array} \in \Theta$ and $\pi(\gamma_4)=p_{(i,j)}$ if $p_{(i,j)} \in \Sigma$, $\gamma_4 = \#$, otherwise. A call to $\mathtt{update}(\mathtt{D},\gamma_4)$ deletes the first symbol in the list (γ_1) and inserts γ_4 as the last element (in O(1) time).

Example 11. Let $\mathcal{T} = (\Sigma,\Gamma,\Theta,\pi)$ be a tiling system for a language L. We sketch a tiling automaton $\mathcal{A}_{diag} = (\mathcal{T},\mathcal{S}_{diag},D_{diag_0},\delta_{diag})$ recognizing L, based on $\mathcal{T}$ and on the scanning strategy $\mathcal{S}_{diag}$ as in Example 5. As data structure we use a list of size $min\{m,n\}+2$ at most for a picture p of size (m,n), with pointers to the first, second and last elements. At the beginning, D_{diag_0} is empty. Suppose position (i,j) of p is considered and $\gamma_{h,k}$ is the symbol associated to the visited position (h,k). At each time we have to keep track of the symbols already associated to some positions in the two counter-diagonals above position (i,j). So we store in the list D, in a proper order, all the computed tiles with bottom-right element in some positions on the previous two counter-diagonals. This way we know whether the list increases, decreases or is constant in length, following whether we have just inserted a corner tile or not. Increment and decrement are handled on external positions. For all internal positions the call to $\mathtt{state}_1(\mathtt{D})$, $\mathtt{state}_2(\mathtt{D})$, $\mathtt{state}_3(\mathtt{D})$, will return (in O(1) time) the symbols $\gamma_{i-1,j-1}$ and $\gamma_{i-1,j}$ from the first tile in the list, and $\gamma_{i,j-1}$ from the second tile in the list. A value $\gamma_4 \in \delta_{diag}(\gamma_{i-1,j-1},\gamma_{i-1,j},\gamma_{i,j-1},p_{(i,j)})$ of the transition function is a symbol such that the tile $\begin{array}{|c|c|}\hline \gamma_{i-1,j-1} & \gamma_{i-1,j} \\ \hline \gamma_{i,j-1} & \gamma_4 \\ \hline \end{array} \in \Theta$ and $\pi(\gamma_4)=p_{(i,j)}$.

Example 12. Let $L_{lastr=lastc}$ be the language of squares over a two-letter alphabet $\Sigma=\{a,b\}$ with last row equal to the last column. Consider the tiling system as in Example 1, and the tiling automata $\mathcal{A}_{col}$ and $\mathcal{A}_{diag}$ based on $\mathcal{T}$, as in Examples 10 and 11. We have that $\mathcal{A}_{col}$ is a deterministic tiling automaton, whereas $\mathcal{A}_{diag}$ is not. In fact there exist $\gamma_1=b^1_a$, $\gamma_2=a^2_a$, $\gamma_3=a^0_a$, $\gamma_4=a^1_b$, $\gamma'_4=a^1_a\in\Gamma$, $\sigma=a\in\Sigma$ such that $\gamma_4\neq\gamma'_4$ and $\gamma_4,\gamma'_4\in\delta_{diag}(\gamma_1,\gamma_2,\gamma_3,\sigma)$.

Let us now briefly discuss some complexity issues. Recall that in [15] it was proved that the problem of parsing in the REC is NP-complete. In the examples shown, the next-position functions are all computable in $O(1)$ time; whereas the data structure used for the computation of a picture of size (m,n) occupies $O(m+n)$ space and the operations for retrieving information and updating are achieved in time $O(1)$ each. Recall that scanning strategies are defined so that each position is scanned once and only once. Hence when such conditions hold, the parsing of a picture of size (m,n) by a deterministic tiling automaton will require $O(mn)$ time (i.e. linear in size) with $O(m+n)$ extra space. This is stated in the following proposition.

Proposition 13. *Let $\mathcal{A}$ be a deterministic tiling automaton where the next-position function is computable in $O(1)$ time, the data structure used for the computation of a picture of size (m,n) occupies $O(m+n)$ space and the operations $\mathtt{state}_1(\mathtt{D})$, $\mathtt{state}_2(\mathtt{D})$, $\mathtt{state}_3(\mathtt{D})$, $\mathtt{update}(\mathtt{D},\gamma)$, require $O(1)$ time each. Then the parsing of a picture of size (m,n) by $\mathcal{A}$ requires $O(mn)$ time with $O(m+n)$ extra space.*

5. Languages of tiling automata

Tiling automata are the computational model for tiling systems. In this section we are interested in their power of recognition of languages. We are going to establish some relations among the languages accepted by tiling automata and other classes of two-dimensional languages defined by finite devices. In particular we will consider recognizability by tiling systems, on-line tessellation automata and 4-way automata and their deterministic and unambiguous counterparts.

Let *L(TA)* (*L(DTA)*, resp.) denote the class of languages accepted by tiling automata (deterministic tiling automata, resp.). Then consider *unambiguous tiling automata (UTA)*, where as usual, a tiling automaton is said to be unambiguous when, for any picture, there exists at most one accepting computation. Denote *L(UTA)*, the class of languages accepted by unambiguous tiling automata. We show that tiling automata are exactly the machine counterpart of tiling systems: further they are the computational model capturing the notions of determinism and unambiguity, as introduced in the formalism of tiling systems.

Proposition 14. *The following properties hold.*

- *L(TA) = REC*
- *L(UTA)=UREC*
- *L(DTA)=DREC*

Proof. L(TA) $\subseteq$ REC: let $L=L(\mathcal{A})$ where $\mathcal{A}=(\mathcal{T},\mathcal{S},D_0,\delta)$ is a tiling automaton. Then L is recognized by the tiling system $\mathcal{T}$. REC$\subseteq$ L(TA): let $L\in$REC and $\mathcal{T}$ a tiling system recognizing L. Then the tiling automaton obtained from $\mathcal{T}$ equipped with any scanning strategy realizable by a data structure (as in Examples 10 and 11) accepts L.

Furthermore, these constructions preserve unambiguity: the existence of two different accepting computations in $\mathcal{A}$ holds iff there are two different local counter-images of some picture referred to $\mathcal{T}$. Hence L(UTA)=UREC.

L(DTA) $\subseteq$ DREC: let $L = L(\mathcal{A})$ where $\mathcal{A} = (\mathcal{T}, \mathcal{S}, D_0, \delta)$ is a deterministic tiling automaton and without loss of generality suppose it is of type *tl2br*. Then by definition, the tiling system $\mathcal{T}$, is necessarily tl2br-deterministic: the existence of two different tiles contradicting the determinism of $\mathcal{T}$, would imply the existence of two different values of the transition function δ on same variables. For similar reasons, if $L \in$DREC and $\mathcal{T}$ is a deterministic tiling system recognizing L, say a tl2br one, then the tiling automaton obtained from $\mathcal{T}$ equipped with any tl2br-directed scanning strategy realizable by a data structure will accept L. □

Using these equivalences between the two models (tiling systems and tiling automata) in their power of recognizing languages, we easily obtain the following results.

Corollary 15. *The following properties hold.*

- *L(DTA)⊂ L(UTA)⊂ L(TA), where the inclusions are strict*
- *L(DTA) is closed under complementation*
- *It is decidable whether a tiling automaton is deterministic.*

Proof. The first two statements come from the analogous results stated in [2,3] with the REC, DREC, UREC in place of L(TA), L(UTA), L(DTA), and the equivalence in Proposition 14.

Then a tiling automaton of type d, for some corner-to-corner direction d, is deterministic iff the underlying tiling system is d-deterministic. Hence it is decidable whether a tiling automaton is deterministic since it is decidable whether a tiling system is d-deterministic [2]. □

Tiling automata can be viewed as a more general model than an OTA, since the computation done by an OTA can be simulated by a tiling automaton equipped with any *tl2br*-directed scanning strategy. Nevertheless the OTA and tiling automata have the same recognition power, that is they recognize the same class of languages (namely REC). On the contrary, when restricted to their deterministic counterparts, the DOTA are less powerful than deterministic tiling automata. Let us denote *tl2br-TA* (*tl2br-DTA*, resp.) a tiling automaton (deterministic tiling automaton, resp.) of type tl2br; similarly for the other corner-to-corner directions. Moreover denote by *L(tl2br-TA)* the corresponding family of accepted languages and similarly for all the types of tiling automata.

Proposition 16. *Any OTA can be simulated by a tl2br-TA.*

Proof. We here refer to the classical construction of a tiling system accepting the same language as a given OTA, as in [11, 8], where the tiles are obtained from the transitions of the OTA. The run of an OTA is usually defined as going in parallel following the counter-diagonals (see Example 11), but indeed the tiles corresponding to the transitions of an OTA could be used by any tiling automaton of type *tl2br*. □

Proposition 17. *The following properties hold.*

- *L(OTA)= L(TA)*
- *L(UOTA)= L(UTA)*
- *L(DOTA)= L(tl2br-DTA)*
- *L(DTA) is equal to the closure by rotation of L(DOTA)*

Proof. L(OTA)= L(TA) follows from Proposition 14 and the equivalence L(OTA)=REC [11,8].

L(UOTA)= L(UTA) follows from L(UOTA)=UREC [3,16] and L(UTA)=UREC (Proposition 14).

Then if $L \in L$(DOTA), the tiling automaton constructed as in Proposition 16 is tl2br-deterministic. On the other hand, if $L \in L$(tl2br-DTA), the classical construction of an OTA accepting the same language as the tiling system on which is based the given tl2br-DTA, as in [8,11], preserves determinism and thus yields a DOTA.

To prove the last statement recall that L(DTA)= DREC (Proposition 14) and DREC is equal to closure by rotation of L(DOTA) [2]. □

Remark 18. The recognition power of a tiling automaton is independent from the scanning strategy we choose: L(TA) coincides with the REC family (Proposition 14). On the other hand, when determinism is concerned, the result is different (Proposition 19): the class of accepted languages may depend on the type of scanning strategy we choose. Let us emphasize that this situation does not hold in one-dimensional language theory. Observe that also in the one-dimensional case, determinism is an oriented notion: if the automaton that reads strings from right to left is deterministic, we use the term of co-deterministic automaton. Nevertheless a string language is accepted by a deterministic automaton if and only if it is accepted by a co-deterministic one.

Proposition 19. *L(tl2br-DTA)⊂ L(DTA), where the inclusion is strict.*

Proof. Language $L_{lastr=lastc}$, as introduced in Example 12, is an example of the strict inclusion. In fact it belongs to L(DTA) (see Example 12) but it cannot be accepted by a DOTA [10] and thus it cannot be accepted by a deterministic tiling automaton of type *tl2br* (see Proposition 17). □

Consider now 4-way automata. The tiling automaton is a model conceptually different from a 4-way automaton exactly as, in one dimension, conventional finite automata differ from two-way automata. Indeed, while the next move of a 4-way automaton is determined from the pair (state, symbol) (i.e. there is a unique function for the next-position and state), in a tiling automaton the direction of the computation is fixed in advance (by the scanning strategy). As a consequence the sequence of picture positions may be different for different inputs. Furthermore 4-way automata can visit the same position many times, while this is forbidden in tiling automata. And in fact L(4NFA) is strictly contained in L(TA)=REC [10]. When restricting to determinism, the two models diverge as stated in Proposition 20.

Proposition 20. *The class L(DTA) is incomparable with L(4DFA).*

Proof. We exhibit a language in L(4DFA) but not in L(DTA) and a language in L(DTA) but not in L(4DFA). For this, consider language L_{frames} equal to the language of all square pictures over a two-letter alphabet $\Sigma = \{0, 1\}$ with the last row equal to the last column, the second row equal to the reverse of the second-last column, the first row equal to the first column and the second-last row equal to the reverse of the second column. In [2] it is shown that $L_{frames} \notin$ DREC, and DREC$=$ L(DTA) (see Proposition 14). On the other hand, it is easy to construct a 4DFA for L_{frames} so that $L_{frames} \in L$(4DFA). In fact it is known (cf. [10]) that $L_{lastr=lastc} \in L$(4DFA) and analogously the language of squares with the second row equal to the reverse of the second-last column is in L(4DFA). Further L(4DFA) is closed under rotation and intersection (cf. [8]). Moreover, in [10], an example of a language in L(DOTA) (and then in L(DTA)) but not in L(4DFA) is given and this concludes the proof. □

Proposition 21. *L(DTA) ∪L(4DFA) ⊂ L(UTA), where the inclusion is strict.*

Proof. The inclusion L(DTA)$\subseteq$ L(UTA) easily follows: the uniqueness of the next local symbol implies the uniqueness of the computation on any picture.

Let us show that L(4DFA) $\subseteq$ L(UTA). For this, we refer to the proof in [10] (Lemma 4.1), that L(4NFA) $\subseteq$ L(OTA). Informally, the authors associate to any picture accepted by a given 4NFA, a set of pictures over an expanded alphabet, called "satisfactory description tapes", which describes the possible accepting computations of the 4NFA on the original picture. Such a set can be recognized by a properly constructed OTA. The resulting OTA is not in general deterministic, even if the starting 4-way automaton is: it cannot keep the local property of determinism true along the different ways of visiting the pictures. Nevertheless, in the case where such an automaton is deterministic (and hence unambiguous), the satisfactory description tape, associated to a picture recognized by the 4-way automaton, is unique. So L(4DFA)$\subseteq$ L(UOTA). The equivalence L(UOTA)$=$ L(UTA) in Proposition 17 provides the inclusion.

An example of the strict inclusion is the language $L \subseteq \Sigma^{**}$, where Σ is a two-letter alphabet, containing the pictures of size (m, n) whose first column is equal to the i-th column and the last column is equal to the j-th one, for some $1 < i \leq n$, $1 \leq j < n$. In [2] it is shown that $L \in$UREC\DREC, that is $L \in L$(UTA)\L(DTA). Further L cannot be recognized by a 4DFA. Indeed we show that L cannot be recognized by a 4NFA, following similar arguments used in [10,12]. Suppose there exists some 4NFA M with a fixed number of states accepting L. Observe that (as more precisely stated in [10,12]) the number $c(n)$ of M-equivalent pictures, that is pictures of same size (m, n) that are distinguishable by the action of M that enters them in a given state in a given position on the first or last row, is upper bounded by $2^{\Theta(m^2)}$. On the other hand, the number $S(m)$ of different sets of columns of height m on Σ is $S(m) = 2^{2^m} - 1$. Hence, for a large enough m, there exist two subsets Q, Q' of columns such that $col_1, col_2 \in Q$, while $col_1, col_2 \notin Q'$ and the pictures z, z' constructed as below are M-equivalent: the first column of both z, z' is col_1; the last column of both z, z' is col_2; the set of remaining columns of z is Q; and the set of remaining columns of z' is Q'. By construction, picture $z \in L$, while $z' \notin L$, against their M-equivalence. □

6. Conclusions

Tiling systems are a well-founded theoretical formalism for recognition of pictures that defines the family REC. In [15] it was proved that the problem of parsing in REC is NP-complete and this fact has limited experimentation and application of tiling systems, in spite of a large amount of theoretical work. Very recently a first attempt to overcome this problem has been done in [17] where the authors have implemented a recognizer/generator for languages of pictures defined by tiling systems in a very attractive, unconventional way, by transforming the tiling problem into a boolean satisfiability one, then using an efficient off-the-shelf SAT-solver. Another important research direction regards the investigation of restricted but more feasible models. In this paper we have introduced a computational model that implements tiling systems by transforming them in a sort of enriched finite automaton. In this automaton model there can be defined the notions of determinism and unambiguity and, in Proposition 13, we have shown that the parsing time with these deterministic devices, under certain conditions, is linear in the size of the pictures.

References

[1] M. Anselmo, D. Giammarresi, M. Madonia, Tiling automaton: A computational model for recognizable two-dimensional languages, in: Procs. CIAA07, in: LNCS, vol. 4783, Springer Verlag, 2007, pp. 290–302.
[2] M. Anselmo, D. Giammarresi, M. Madonia, From determinism to non-determinism in recognizable two-dimensional languages, in: Procs. DLT07, in: LNCS, vol. 4588, Springer Verlag, 2007, pp. 36–47.
[3] M. Anselmo, D. Giammarresi, M. Madonia, A. Restivo, Unambiguous recognizable two-dimensional languages, RAIRO — Informatique Theorique et Applications 40 (2006) 277–293.

[4] M. Anselmo, M. Madonia, Simulating two-dimensional recognizability by pushdown and queue automata, in: J. Farré, et al. (Eds.), CIAA 2005, in: LNCS, vol. 3845, Springer-Verlag, 2006, pp. 43–53.
[5] M. Blum, C. Hewitt, Automata on a two-dimensional tape, IEEE Symposium on Switching and Automata Theory (1967) 155–160.
[6] S. Eilenberg, Automata, Languages and Machines, vol. A, Academic Press, 1974.
[7] D. Giammarresi, A. Restivo, Recognizable picture languages, International Journal of Pattern Recognition and Artificial Intelligence 6 (2–3) (1992) 241–256.
[8] D. Giammarresi, A. Restivo, Two-dimensional languages, in: G. Rozenberg, et al. (Eds.), Handbook of Formal Languages, vol. III, Springer Verlag, 1997, pp. 215–268.
[9] D. Giammarresi, A. Restivo, Matrix-based complexity functions and recognizable picture languages, in: Logic and Automata: History and Perspectives, in: E. Gradel, J. Flum, T. Wilke (Eds.), Texts in Logic and Games, vol. 2, University Press, Amsterdam, 2007, pp. 315–337.
[10] K. Inoue, A. Nakamura, Some properties of two-dimensional on-line tessellation acceptors, Information Sciences 13 (1977) 95–121.
[11] K. Inoue, I. Takanami, A Characterization of recognizable picture languages, in: Proc. 2nd Int. Coll. on Parallel Image Processing, in: A. Nakamura (Ed.), LNCS, vol. 654, Springer-Verlag, 1993.
[12] K. Inoue, I. Takanami, A. Nakamura, A note on two-dimensional finite automata, Information Processing Letters 7–1 (1978) 49–52.
[13] K. Inoue, I. Takanami, H. Taniguchi, Two-dimensional alternating Turing machines, Theoretical Computer Science 27 (1983) 61–83.
[14] E.B. Kinber, Three-way automata on rectangular tapes over a one-letter alphabet, Information Sciences 35 (1985) 61–77.
[15] K. Lindgren, C. Moore, M. Nordahl, Complexity of two-dimensional patterns, Journal of Statistical Physics 91 (5–6) (1998) 909–951.
[16] I. Mäurer, Weighted Picture Automata and Weighted Logics, in: B. Durand, et al. (Eds.), Procs. STACS 2006, in: LNCS, vol. 3885, Springer-Verlag, 2006, pp. 313–324.
[17] M. Pradella, S. Crespi Reghizzi, A SAT-based parser and completer for pictures specified by tiling, Pattern Recognition 41 (2008) 555–566.