Strategies to scan pictures with automata based on Wang tiles

Violetta Lonati(1), Matteo Pradella(2)

(1) Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano
Via Comelico 39/41, 20135 Milano – Italy,
lonati@dsi.unimi.it
(2) IEIIT, Consiglio Nazionale delle Ricerche
Via Golgi 40, 20133 Milano, Italy
pradella@elet.polimi.it

January 2010

Abstract

Wang automata are devices for picture language recognition recently introduced by us, which characterize the class REC of recognizable picture languages. Thus, Wang automata are equivalent to tiling systems or online tessellation acceptors, and are based like Wang systems on labeled Wang tiles. The present work focus on scanning strategies, to prove that the ones Wang automata are based on are those following four kinds of movements: boustrophedonic, "L-like", "U-like", and spirals.

Keywords: picture languages, 2D languages, Wang systems, 2D automata, scanning strategies.

Introduction

Recent years saw a growing interest towards picture languages, and especially tile-based models. Picture languages are a generalization to two dimensions of classical string languages, where a picture is an array of symbols taken from a finite alphabet. Tiling problems have appeared in many branches of physics and mathematics like group theory, topology, quasicrystals, symbolic dynamics. More recently Winfree et al. [10] have demonstrated the feasibility of creating tiles made from folded DNA molecules that can act as *Wang tiles* [20]. As pointed out by Brun [7], such models are self-assembling, like many biological systems, highly distributed, and parallel; they may be implemented using molecules, or a large computer network such as the Internet, thus opening several new prospectives. This idea is exploited in [8], where an approach to the design of self-adaptive service-oriented applications based on a tile-based model is presented.

Among the various classes of languages defined by tile-based models, probably the most successful, as far as theoretical characterizations are concerned, is the class of *tiling recognizable* languages, also known as REC [12]. REC is a robust class, and coincides with the class of languages generated by various kind of devices, such as *online tessellation acceptors* [13], *tiling systems* [11], and *Wang systems* [9].

Wang automata is a more recent kind of device characterizing REC, introduced by us in [18], and based on the labeled variant of Wang tiles used in Wang systems [9]. Our model in some sense combine features of online tessellation acceptors, and traditional 4-ways automata [14]. Indeed, Wang automata are like online tessellation acceptors, because their computation assigns states to each input picture position; while the similarity with 4-way automata arises from the presence of an input head, which visits the input picture following a *scanning strategy*, i.e. a procedure to visit its pixels.

We originally introduced Wang automata to study the concept of *determinism* in REC. The concept of determinism for picture languages is not as straightforward as in string languages, where one may read the input string going from left to right or vice versa. The literature contains several different deterministic subclasses of REC, starting from deterministic online tessellation acceptors [13], and going to the more recent [1, 4, 3, 17]. Indeed, Wang systems are implicitly nondeterministic: REC is not closed under complement, and the membership problem is NP-complete [15]. Wang automata overcome these issues by using a scanning strategy to move the input head: this allows us to introduce a natural and decidable notion of determinism, yielding a proper subclass of REC, called Scan-DREC, closed under complement, rotation and mirror operations. We refer the interested reader to [16], where different subclasses of Scan-DREC determined by different scanning strategies, and their relation to unambiguity, are presented and studied.

In the present work, we focus on studying scanning strategies of Wang automata. In fact, we prove that *polite* scanning strategies, the ones Wang automata are based on, are essentially only those following four kinds of movements, and their rotations and symmetrical: boustrophedonic, where the head proceeds row-by-row, in a "snake"-like fashion; "L-like", where the head scans a row, a column, and then goes back; "U-like", where the head scans a column, a row, then another column, and goes back; and spirals. This result is interesting in two ways. First, it is now easier to study the properties of deterministic Wang automata, because we only need to focus on those few strategies. Second, the definition of Wang automata is now greatly simplified, because it does no more need all the theoretical scaffolding supporting generic kind of scanning strategies.

We consider now some related formalisms and results. Another kind of tile-based automaton, in some senses similar to Wang automata, is presented in [5]: *quadrapolic automata* are like to Wang automata, in that both use variant of labeled Wang tiles for working. Differently, quadrapolic automata do not read the input picture by following a fixed scanning strategy. To our knowledge, no definition of determinism for such devices is proposed in the literature.

As far as scanning strategies for picture languages are concerned, two relevant works are [2] and [6].

In [2], an automata model called *tiling automaton* is introduced, with the aim to define a general computational model for recognizable languages. This approach is centered upon the concept of scanning strategy itself, which directly depends on the size of the picture to be scanned. This definition is very general, and may exploit the size of the picture to perform "jumps", thus allowing complex behaviors. This freedom, together with the potential knowledge of the picture size, may be exploited to exceed REC.

In [6], the considered strategies are "continuous", in the sense that the next considered position is adjacent to the current one. The actual definition of such strategies is presented in a qualitative form. This aspect could be source of some problems, since may admit different strategies depending on the picture size or shape (e.g. Peano-Hilbert curves are suitable only for square pictures). Indeed, if we consider unary languages, scanning strategies which depends on the shape or size of the input picture may be exploited to exceed REC also in this case.

The paper is structured as follows. The first section introduces some notation and tiling recognizable languages. Section 2 defines Wang automata and their scanning strategies. Section 3 presents the main results on polite scanning strategies. Finally, the last section draws the conclusions.

1 Preliminaries

The following notation and definitions are partially adapted from [12]. Let Σ be a finite alphabet. A two-dimensional array of elements of Σ is a *picture* over Σ. The set of all pictures over Σ is Σ^{++}; a picture language is a subset of Σ^{++}. For $h, k \geq 1$, $\Sigma^{h,k}$ denotes the set of pictures of size (h, k), i.e. having h rows and k columns. The *support* of a picture of size (h, k) is the set $h \times k = \{1, 2, \ldots, h\} \times \{1, 2, \ldots, k\}$. A *pixel* is an element $p(i, j)$ of p. We call (i, j) the *position* in p of the pixel.

We will sometimes consider the 90° clockwise *rotation*, the *horizontal mirror*, and the *vertical mirror* of a picture p. E.g. if $p = \begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline\end{array}$, then $\begin{array}{|c|c|}\hline c & a \\ \hline d & b \\ \hline\end{array}$, $\begin{array}{|c|c|}\hline c & d \\ \hline a & b \\ \hline\end{array}$, and $\begin{array}{|c|c|}\hline b & a \\ \hline d & c \\ \hline\end{array}$ are its rotation, horizontal mirror and vertical mirror, respectively. Naturally, the same operations can be applied to languages, and classes of languages, too.

An important class of two-dimensional languages is REC, i.e., the class of *tiling-recognizable languages*, originally defined in terms of tiling systems [11]. Here we define this class by using the equivalent notation introduced in [9], which is based on a variant of Wang tiles.

Let Σ be a finite alphabet and K be a set of colors, containing the special color # representing borders. A *labeled Wang tile* (or *tile* for short) is a unitary square with colored edges and a *label* in Σ. Formally, a tile is an element $A = (a, t, l, r, b) \in \Sigma \times K^4$, where t, b, r, l represent the colors at top, bottom, right and left edges, respectively.

For better readability, we represent labeled Wang tiles as

$$A = l \, \overset{t}{\underset{b}{\boxed{a}}} \, r \, . \tag{1}$$

Dirs is the set of four directions $\rightarrow, \downarrow, \leftarrow, \uparrow$. For any direction $d \in \mathit{Dirs}$, A_d is the color of the edge of A towards direction d. We also use $-d$ for referring to the direction opposite to d. Also, $\lambda(A)$ refers to the label of tile A. For example, in the case of A given by (1), $A_\downarrow = b$ and $\lambda(A) = a$. The set of tiles with labels in Σ and colors in K is Σ_{4K}.

We also consider *partial* tiles, where some colors may be undefined: the set of partial tiles is denoted by Σ_K. The *domain* of a tile A is the set Δ_A of directions where A is defined. Given two partial tiles A, B, we say that B extends A if $B_d = A_d$ for every $d \in \Delta_A$. When we need to emphasize the fact that a tile is not partial, we will call it *complete*.

Labeled Wang tiles in Σ_{4K} can be used to build pictures over Σ, by using colors to check compatibility: two tiles may be adjacent only if the color of the touching edges is the same. A picture $P \in \Sigma_{4K}^{++}$ is called a *Wang picture* if all borders are colored with # and

$$\begin{array}{rll} P(i,j)_\downarrow = P(i+1,j)_\uparrow & \text{for every} & 1 \leq i < n, \\ P(i,j)_\rightarrow = P(i,j+1)_\leftarrow & \text{for every} & 1 \leq j < m, \end{array}$$

where (n, m) is the size of P. We call $W(P)$ the set of Wang tiles contained in a Wang picture P. The *label of a Wang picture P* over Σ_{4K} is the picture $p = \lambda(P) \in \Sigma^{++}$ having for pixels the labels of pixels of P, i.e., $p(i, j) = \lambda(P(i, j))$. Next (on the left), the reader may find the example of a Wang picture of size $(2, 2)$ with its label (in the middle). For better readability, we represent Wang pictures by writing each common color only once, as in the figure on the right.

$$P = \begin{array}{|c|c|}\hline \begin{array}{ccc} & \# & \\ \# & \boxed{a} & 4 \\ & 1 & \end{array} & \begin{array}{ccc} & \# & \\ 4 & \boxed{b} & \# \\ & 3 & \end{array} \\ \hline \begin{array}{ccc} & 1 & \\ \# & \boxed{b} & 2 \\ & \# & \end{array} & \begin{array}{ccc} & 3 & \\ 2 & \boxed{a} & \# \\ & \# & \end{array} \\ \hline \end{array}, \qquad \lambda(P) = \begin{array}{|c|c|}\hline a & b \\ \hline b & a \\ \hline \end{array}, \qquad \begin{array}{|ccccc|}\hline & \# & & \# & \\ \# & \boxed{a} & 4 & \boxed{a} & \# \\ & 1 & & 3 & \\ \# & \boxed{b} & 2 & \boxed{a} & \# \\ & \# & & \# & \\ \hline \end{array}.$$

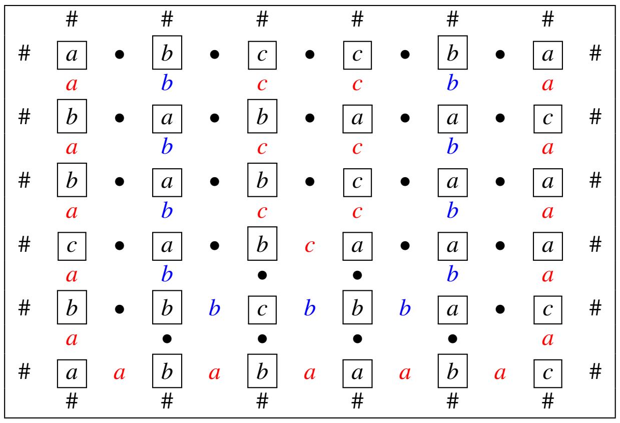
A *Wang system* is a triple $\omega = \langle \Sigma, K, \Theta \rangle$, where Σ is a finite alphabet, K is a set of colors, Θ is a subset of Σ_{4K}. The language generated by ω is the language $L(\omega) \subseteq \Sigma^{++}$ of the labels of all Wang pictures in Θ^{++}. REC is the class of picture languages generated by Wang systems.

Example 1.1 *Consider the language $L_{half} \subset \Sigma^{++}$ of pictures of size $(n, m), n \geq m \geq 4$, with the first row like $w \cdot \bar{w}$, where $\bar{w}$ is the reverse of w. Then L_{half} is recognized by the Wang system $\langle \Sigma, K, \Theta \rangle$ where $K = \Sigma \cup \{\bullet, \#\}$ and*

$$\Theta = \left\{ \begin{array}{l} \begin{array}{ccc} & \# & \\ \# & \boxed{x} & \bullet \\ & x & \end{array}, \begin{array}{ccc} & x & \\ \# & \boxed{y} & \bullet \\ & x & \end{array}, \begin{array}{ccc} & x & \\ \# & \boxed{y} & x \\ & \# & \end{array}, \begin{array}{ccc} & \bullet & \\ x & \boxed{y} & x \\ & \# & \end{array}, \begin{array}{ccc} & x & \\ x & \boxed{y} & \# \\ & \# & \end{array}, \begin{array}{ccc} & x & \\ \bullet & \boxed{y} & \# \\ & x & \end{array}, \\ \begin{array}{ccc} & \# & \\ \bullet & \boxed{x} & \# \\ & x & \end{array}, \begin{array}{ccc} & \# & \\ \bullet & \boxed{x} & \bullet \\ & x & \end{array}, \begin{array}{ccc} & x & \\ \bullet & \boxed{y} & \bullet \\ & x & \end{array}, \begin{array}{ccc} & x & \\ \bullet & \boxed{y} & x \\ & \bullet & \end{array}, \begin{array}{ccc} & \bullet & \\ x & \boxed{y} & x \\ & \bullet & \end{array}, \begin{array}{ccc} & x & \\ x & \boxed{y} & \bullet \\ & \bullet & \end{array} \end{array} \;\middle|\; x, y \in \Sigma \right\}.$$

The colors are used to connect each letter in w to the corresponding letter in $\bar{w}$, along nested paths following a U-like form. Next (on the left), we show an example of picture $p \in L_{half}$, together with the corresponding Wang picture P over Θ (on the right). The actual colors in P are used in the figure only to emphasize the U-like form of the resulting paths.

a	b	c	c	b	a
b	a	b	a	a	c
b	a	b	c	a	a
c	a	b	a	a	a
b	b	c	b	a	c
a	b	b	a	b	c



2 Wang automata

2.1 Two-dimensional scanning strategies

Here we recall the notion of 2D scanning strategies as introduced in [18] and in particular the definition of blind scanning strategy.

Definition 2.1 *A* scanning strategy *is a family $\mu = \{\mu_{n\times m} : \{1, 2, \ldots\} \to n \times m\}_{n,m}$ where $\mu_{m\times n}$ is a partial function called the* scanning function over support $n \times m$. *A scanning strategy is said to be* continuous *if $\mu_{n\times m}(t + 1)$ is adjacent to $\mu_{n\times m}(t)$ for every n, m, t; it is said to be* one-pass *if each scanning function $\mu_{n\times m}$ restricted to $\{1, 2, \ldots, nm\}$ is a bijection.*

Intuitively, a scanning strategy provides a method to visit positions in any picture support: $\mu_{n\times m}(t)$ is the position visited in $n \times m$ at time t. One-pass strategies are those that visit each position in each support exactly once.

In particular, we are interested in scanning strategies that satisfy some further properties: uniformity with respect to the picture support, no memory about how the scanning strategy visited the past positions, and independence with respect to the actual contents of the picture. The notion of *blindness* of a scanning strategy was introduced in [18] to this aim. Basically, our idea of blind strategy is based on local properties on the "shape" of visited positions of the input picture. To recall such notion we shall need some notations.

For every position y, and $d \in Dirs$, the edge of y in direction d is denoted by y_d, and the position adjacent to y in direction d is denoted by $y \boxplus d$. Moreover, given a position y, Edges(y) denotes the set of 4 edges adjacent to y. The top-leftmost, top-rightmost, bottom-rightmost, and bottom-leftmost corners of any picture domain are denoted by 1, 2, 3, 4, respectively.

A *next-position function* is a partial function $\eta : 2^{Dirs} \times Dirs \to Dirs$ such that $\eta(D, d) = \perp$ if $-d \notin D$. Informally, the meaning of η is that, for a given position, we have a set of already considered edges, given by the set D of directions, and d, the direction from the "last-considered" edge. η is used to choose where to go next, i.e. the direction towards the position to visit next.

Now fix any next-position function η, any starting corner $c_s \in \{1, 2, 3, 4\}$ and any starting direction $d_s \in Dirs$. Then, for every support $n \times m$, consider the following scanning function $\mu_{n\times m}$ over $n \times m$.

- The starting position is

$$\mu_{n\times m}(1) = \begin{cases} (1,1) & \text{if } c_s = 1 \\ (1,m) & \text{if } c_s = 2 \\ (n,1) & \text{if } c_s = 4 \\ (n,m) & \text{if } c_s = 3 \end{cases}$$

  moreover we define E_1 as the set of outer edges (i.e. those adjacent to borders) of the picture support $n \times m$, and we set $d_1 = d_s$.

- The inductive definition[1] of $\mu_{n\times m}(t + 1)$ for $t \geq 1$ is given by:

$$\begin{aligned} D_t &= \{d \in Dirs : (\mu_{n\times m}(t))_d \in E_t\} \\ E_{t+1} &= E_t \cup \text{Edges}(\mu_{n\times m}(t)) \\ d_{t+1} &= \eta(D_t, d_t) \\ \mu_{n\times m}(t+1) &= \mu_{n\times m}(t) \boxplus d_{t+1} \end{aligned}$$

  Notice that $\mu_{n\times m}(1)_{d_1}$ must be in E_1 for $\eta(D_1, d_1)$ to be defined.

We say that $\mu = \{\mu_{n\times m}\}_{n,m}$ is the scanning strategy *induced by* the triple $\langle \eta, c_s, d_s \rangle$.

Definition 2.2 *A scanning strategy is* blind *if it is induced by a triple $\langle \eta, c_s, d_s \rangle$, where η is a next-position function, c_s a starting corner, and d_s a starting direction.*

Notice that, in general, a blind scanning strategy is not one-pass. However, it is continuous and satisfies the other requirements we need. First, all scanning functions are defined by the same triple $\langle \eta, c_s, d_s \rangle$ for every picture support; second, the next position to visit always depends only on this information: which neighboring positions have already been visited, and which direction we are moving from. This yields the following definition.

Definition 2.3 *A scanning strategy is called* polite *if it is blind and one-pass.*

Example 2.4 *Some one-pass scanning strategies are illustrated in Figure 1. Actually they are not fully defined: only the function $\mu_{3\times4}$ is depicted whereas the other functions should be defined analogously; each position y in 3×4 contains the number t such that $y = \mu_{3\times4}(t)$.*

[1] In the definition, also d_t, D_t, and E_t depend on n and m. For better readability, this dependence is not explicit.

1	6	7	12
2	5	8	11
3	4	9	10

(a) snake ($\mathcal{S}$)

1	10	11	12
2	9	8	7
3	4	5	6

(b) L-like ($\mathcal{J}$)

1	12	9	8
2	11	10	7
3	4	5	6

(c) U-like ($\mathcal{U}$)

1	10	9	8
2	11	12	7
3	4	5	6

(d) spiral ($\mathcal{C}$)

1	2	11	10
4	3	12	9
5	6	7	8

(e) mixed

1	2	3	4
5	6	7	8
9	10	11	12

(f) row by row

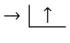
4	3	2	1
5	8	9	12
6	7	10	11

(g) snake with bootstrap

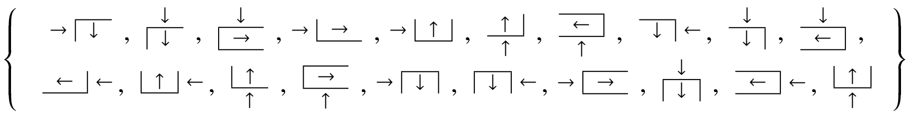
Figure 1: Some one-pass scanning strategies: the number in each pixel denotes its scanning order.

We denote strategies from (a) to (d) as $\mathcal{S}$, $\mathcal{J}$, $\mathcal{U}$, $\mathcal{C}$, respectively: $\mathcal{S}$ has a boustrophedonic (snake-like) behavior, $\mathcal{J}$ draws nested L-like path, $\mathcal{U}$ draws nested U-like paths, $\mathcal{C}$ has a spiral behavior. They are all polite and turn out to be the basic ones (see Theorem 4.5).

Strategy (e) combines the behavior of a rotation of $\mathcal{S}$ in the first half of the picture and $\mathcal{C}$ in the second one; it is not blind, since it exploits the knowledge of the width of picture, to change direction when reaching its half. Strategy (f) visits one row after the other, from left to right and from top to bottom. Also (f) is not blind, since it is not continuous and uses the knowledge of picture's width, after reaching the end of a row, to "jump" back to the beginning of the next row. The blind strategy (g) is like (a), but for a "bootstrap", i.e. a row scan going from corner 2 to corner 1.

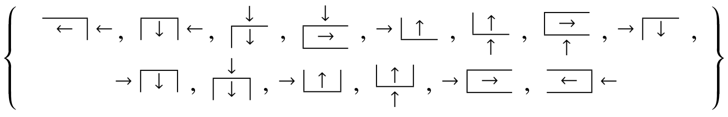
For better readability, we introduce a concise and intuitive notation for the next-position function $\eta(D, d) = d'$, where a set of directions (i.e. D) is graphically depicted as a partially outlined rectangle, the incoming direction d is shown as an arrow entering the rectangle, and d' is put inside the rectangle. For instance, to represent $\eta(\{\leftarrow, \downarrow\}, \rightarrow) = \uparrow$ we will use:  . We will also call such writings *configurations* of the next-position function.

Example 2.5 *$\mathcal{U}$ is induced by the triple $\langle \eta_{\mathcal{U}}, 1, \rightarrow\rangle$, where the next-position function $\eta_{\mathcal{U}}$ is given by the following set of possible configurations:*



The last six configurations are used to scan the pixels in the last column or row to visit.

Example 2.6 *The scanning strategy depicted in Figure 1(g) is induced by the triple $\langle \eta, 2, \leftarrow\rangle$, where η is given by the following set of possible configurations:*



The last six configurations are used to scan the pixels in the last column or row to visit.

2.2 Definition and semantics of Wang automata

A Wang automaton can be seen as having a head that visits a picture, by moving from a position to an adjacent one, and coloring at each step the edges of the position it is visiting. For each accepting computation, the automaton produces a Wang picture whose label is equal to the input picture. The movements of the head are lead by a scanning strategy μ, which is independent of the input picture, whereas the coloring operations the automaton performs are determined by a finite control formalized by function δ, depending also on the current input pixel.

Definition 2.7 *A μ-*directed Wang automaton *(μ-WA) is a tuple $\langle \Sigma, K, \delta, \mu, F \rangle$ where:*

- *Σ is a finite input alphabet,*
- *K is a finite set of colors*
- *$F \subset \Sigma_{4K}$,*
- *$\delta : \Sigma_K \times Dirs \to 2^{\Sigma_{4K}}$ is a partial function such that each tile in $\delta(A, d)$ extends A,*
- *μ is a blind scanning strategy induced by some $\langle \eta, c_s, d_s \rangle$ such that $\delta(A, d) \neq \emptyset$ implies $\eta(\Delta_A, d) \neq \perp$.*

A Wang automaton can be seen as having a head that visits a picture, by moving from a position to an adjacent one, and coloring at each step the edges of the position it is visiting (in a sense, the elements of $\Sigma_K \times Dirs$ are the *states* of the automaton). For each accepting computation, the automaton produces a Wang picture whose label is equal to the input picture. The movements of the head are lead by the scanning strategy μ, whereas the coloring operations the automaton performs are determined by a finite control formalized by function δ. Since the scanning strategy μ is blind, the automaton visits the picture positions independently of the input symbols, and only the choice of colors to assign to edges is nondeterministic.

More precisely, the behavior of a μ-directed Wang automaton $\mathcal{A} = \langle \Sigma, K, \delta, \mu, F \rangle$ over an input picture $p \in \Sigma^{m,n}$ can be described as follows.

Configuration of the WA: $\langle cf, dr, ps \rangle$, where

$$cf \in \Sigma_K^{m,n}, \;\; dr \in Dirs, \;\; ps = (i, j), \text{ with } 1 \leq i \leq m, 1 \leq j \leq n.$$

Initial configuration: $\langle cf_s, d_s, c_s \rangle$, where $cf_s \in \Sigma_K^{m,n}$ is such that $\lambda(cf_s) = p$, with coloring totally undefined, except for the borders. Note that μ is induced by $\langle \eta, c_s, d_s \rangle$.

Transition: $\langle cf, dr, ps \rangle \vdash_{\mathcal{A}} \langle cf', dr', ps' \rangle$ is such that

$$\begin{array}{rcl} dr' & = & \eta(\Delta_{cf(ps)}, dr), \\ ps' & = & ps \boxplus dr', \\ cf'(ps) & \in & \delta(cf(ps), dr), \\ cf'(ps \boxplus u)_{-u} & = & cf'(ps)_u, \quad \forall u \in Dirs \setminus \Delta_{cf(ps)}. \end{array}$$

Final configuration: $\langle cf_F, d_F, ps_F\rangle$, where $cf_F \in \Sigma_{4K}^{m,n}$, and $cf_F(ps_F) \in F$.

Informally, at the beginning the head of the automaton points at the position in the starting corner c_s and the current direction is set to d_s. When the current direction is dr, the head is placed at position ps, the pixel and the colors of borders of p at position ps are summarized by $cf(ps)$, then let $dr' = \eta(\Delta_{cf(ps)}, dr)$ and $A' \in \delta(cf(ps), dr)$. Hence the automaton may execute this move: color the borders at position ps according to A', set the current direction to dr', move to position $ps \boxplus dr'$, and extend cf to the Wang picture cf' with $cf'(ps) = A'$.

If no move is possible, the automaton halts. The input picture p is accepted if there is a computation such that the borders of the final position are colored according to some Wang tile in F.

Example 2.8 *Consider the language L_{half} presented in Example 1.1. Starting from the Wang system sketched in the same example, one can define an equivalent $\mathcal{C}$-WA as described in Table 1. Note that $\delta(A, d)$ has at most one element, for any A, d (i.e. it is* deterministic*), so in the table these are not represented as sets. For better readability, the table also presents the next-position function η. The accepting tiles are of the form*

$$\begin{array}{ccc} & x & \\ \bullet & \boxed{y} & \bullet \\ & x & \end{array}$$

with $x \in K$. It is interesting to note that the set of complete tiles coincides with tile-set Θ of Example 1.1.

For nondeterministic Wang automata the choice of the scanning strategy (as long as it is polite) is not relevant from the point of view of the recognizing power of the device: for every polite scanning strategy μ, the class of picture languages recognized by μ-WA equals REC [18, Theorem 1]. This is no longer true when determinism is concerned. It is therefore useful to study the actual behavior of polite scanning strategies, because depending on it we obtain different deterministic subclasses of REC - some of them are presented in [16].

3 A concise representation of polite scanning strategies

From now on, let μ induced by $\langle \eta, c_s, d_s\rangle$ be a blind scanning strategy. To better present the main issues related to polite scanning strategies, we need to introduce some notation. First of all, we need to extend the original concept of corner introduced in Section 2.1. We will call a *corner position*, or simply a *corner*, a position of the picture where only one other adjacent position is yet to be visited. For instance, if we arrive from left to the top-rightmost position of a picture, the only visitable adjacent position is the one immediately below. We distinguish four kinds of corners: 1, 2, 3, 4, corresponding respectively to the top-leftmost, top-rightmost, bottom-rightmost, and bottom-leftmost corners. Note that these are corners of the part of the picture which is yet to be visited, so they correspond to actual corners of the picture only at the beginning of the scanning procedure; moreover, this definition does not take into account

A,d	$\begin{array}{ccc} & \# & \\ \# & \boxed{x} & \\ & & \end{array} \rightarrow$	$\begin{array}{ccc} & x & \\ \# & \boxed{y} & \\ & & \end{array} \downarrow$	$\begin{array}{ccc} & x & \\ \# & \boxed{y} & \\ & \# & \end{array} \downarrow$	$\begin{array}{ccc} & & \\ x & \boxed{y} & \\ & \# & \end{array} \rightarrow$	$\begin{array}{ccc} & & \\ x & \boxed{y} & \# \\ & \# & \end{array} \rightarrow$
δ,η	$\begin{array}{ccc} & \# & \\ \# & \boxed{x} & \bullet \\ & x & \end{array} \downarrow$	$\begin{array}{ccc} & x & \\ \# & \boxed{y} & \bullet \\ & x & \end{array} \downarrow$	$\begin{array}{ccc} & x & \\ \# & \boxed{y} & x \\ & \# & \end{array} \rightarrow$	$\begin{array}{ccc} & \bullet & \\ x & \boxed{y} & x \\ & \# & \end{array} \rightarrow$	$\begin{array}{ccc} & x & \\ x & \boxed{y} & \# \\ & \# & \end{array} \uparrow$

A,d	$\begin{array}{ccc} & & \\ & \boxed{y} & \# \\ & x & \end{array} \uparrow$	$\begin{array}{ccc} & \# & \\ & \boxed{x} & \# \\ & x & \end{array} \uparrow$	$\begin{array}{ccc} & \# & \\ & \boxed{x} & \bullet \\ & & \end{array} \leftarrow$	$\begin{array}{ccc} & \# & \\ \bullet & \boxed{x} & \bullet \\ & & \end{array} \leftarrow$	$\begin{array}{ccc} & x & \\ \bullet & \boxed{y} & \\ & & \end{array} \downarrow$	$\begin{array}{ccc} & x & \\ \bullet & \boxed{y} & \\ & \bullet & \end{array} \downarrow$
δ,η	$\begin{array}{ccc} & x & \\ \bullet & \boxed{y} & \# \\ & x & \end{array} \uparrow$	$\begin{array}{ccc} & \# & \\ \bullet & \boxed{x} & \# \\ & x & \end{array} \leftarrow$	$\begin{array}{ccc} & \# & \\ \bullet & \boxed{x} & \bullet \\ & x & \end{array} \leftarrow$	$\begin{array}{ccc} & \# & \\ \bullet & \boxed{x} & \bullet \\ & x & \end{array} \downarrow$	$\begin{array}{ccc} & x & \\ \bullet & \boxed{y} & \bullet \\ & x & \end{array} \downarrow$	$\begin{array}{ccc} & x & \\ \bullet & \boxed{y} & x \\ & \bullet & \end{array} \rightarrow$

A,d	$\begin{array}{ccc} & & \\ x & \boxed{y} & \\ & \bullet & \end{array} \rightarrow$	$\begin{array}{ccc} & & \\ x & \boxed{y} & \bullet \\ & \bullet & \end{array} \rightarrow$	$\begin{array}{ccc} & & \\ & \boxed{y} & \bullet \\ & x & \end{array} \uparrow$	$\begin{array}{ccc} & x & \\ & \boxed{y} & \bullet \\ & x & \end{array} \uparrow$	$\begin{array}{ccc} & x & \\ & \boxed{y} & \bullet \\ & & \end{array} \leftarrow$	$\begin{array}{ccc} & x & \\ \bullet & \boxed{y} & \bullet \\ & & \end{array} \leftarrow$
δ,η	$\begin{array}{ccc} & \bullet & \\ x & \boxed{y} & x \\ & \bullet & \end{array} \rightarrow$	$\begin{array}{ccc} & x & \\ x & \boxed{y} & \bullet \\ & \bullet & \end{array} \uparrow$	$\begin{array}{ccc} & x & \\ \bullet & \boxed{y} & \bullet \\ & x & \end{array} \uparrow$	$\begin{array}{ccc} & x & \\ \bullet & \boxed{y} & \bullet \\ & x & \end{array} \leftarrow$	$\begin{array}{ccc} & x & \\ \bullet & \boxed{y} & \bullet \\ & x & \end{array} \leftarrow$	$\begin{array}{ccc} & x & \\ \bullet & \boxed{y} & \bullet \\ & x & \end{array} \downarrow$

Table 1: $\mathcal{C}$-WA for L_{half}: δ and η stand respectively for $\delta(A, d)$ and $\eta(\Delta_A, d)$; $x, y \in \Sigma$, $K = \Sigma \cup \{\bullet, \#\}$.

configurations found when scanning the last row or column, where this condition holds in every position. This is not a problem, because we intend to study the general behavior of the strategy, while the scan of the last positions is always constrained.

We will distinguish horizontal and vertical kind of directions of movement of the strategy, that will be denoted by the symbols h, and v, respectively. Quite naturally, $\rightarrow, \leftarrow$ are of kind h, while $\uparrow, \downarrow$ are of kind v. h is called the *dual movement* of v and vice versa; we will also write $\overline{h} = v$ or $\overline{v} = h$.

We also write $c \sim_m c$ if a corner of kind c' can be reached from a corner of kind c by a movement of kind m, i.e. $1 \sim_h 2$, $3 \sim_h 4$, $1 \sim_v 4$ and $2 \sim_v 3$.

3.1 Polite configurations of the next-position function

Proposition 3.1 *If a next-position function $\eta(D, d)$ is defined for $D = \{\uparrow\}$ (and its rotations, i.e. $|D| = 1$), and the corresponding configuration is reachable, then η induces a blind scanning strategy which is not one-pass.*

Proof. Consider w.l.o.g to start from the top-left position of the picture, with configuration $\overset{\downarrow}{\ulcorner_{\rightarrow}}$ (note that configuration $\rightarrow\ulcorner_{\rightarrow}$ would force us to proceed towards the right border, being η a function). To reach a configuration with $D = \{\uparrow\}$, we must now go down, i.e. the next configuration must be: $\rightarrow\begin{array}{|c}\hline \downarrow\end{array}$. At the next step, either we turn right $\begin{array}{c}\downarrow\\ \hline \rightarrow\end{array}$, or left $\begin{array}{c}\downarrow\\ \hline \leftarrow\end{array}$ or we proceed straight ahead $\begin{array}{c}\downarrow\\ \hline \downarrow\end{array}$.

If we turn left, the scanning proceeds in a two-step snake-like fashion, considering two positions in the same row, then going down. In this case, η is not one-pass, because, if the picture has an even number of rows, the last position considered is that at the bottom-left, thus leaving untouched all the rest of the picture.

If we turn right, either we go down, i.e. $\rightarrow\begin{array}{|c}\downarrow\end{array}$ thus proceeding in diagonal towards the lower-right part of the picture, or we go up, i.e. $\rightarrow\begin{array}{|c}\uparrow\end{array}$. In this last case, it is easy to see that we are following a two-step snake-like strategy analogous to the one considered before, so the same arguments apply.

For all the other configurations, we proceed towards either the left border of the picture, or the bottom border. Let us consider w.l.o.g the bottom border. In this case, η must comprise either the configuration $\begin{array}{c}\downarrow\\ \hline \leftarrow\\ \hline\end{array}$ or $\begin{array}{c}\downarrow\\ \hline \rightarrow\\ \hline\end{array}$. But, either way, we are partitioning the picture in two sections, one of which cannot be reached without going back to an already-considered position, and this means that η is not one-pass. $\square$

Corollary 3.2 *A polite scanning strategy is induced by a next-position function η which comprises at most the reachable configurations*

$$L_0 = \rightarrow\begin{array}{|c}\hline \rightarrow\end{array} \qquad L_1 = \rightarrow\begin{array}{|c}\hline \downarrow\end{array} \qquad U_0 = \begin{array}{c}\downarrow\\ \hline \multicolumn{1}{|c|}{\downarrow}\end{array} \qquad U_1 = \rightarrow\begin{array}{|c|}\hline \downarrow\end{array}$$

and all their rotations and symmetrical configurations.

Given a configuration $S \in \{L_0, L_1, U_0, U_1\}$, any configuration which is a rotation or symmetrical of S is called S-shaped. Notice that subscript 1 represents the presence of a direction change, whereas letter L or U represents the shape of borders (i.e., D).

U_1-shaped configurations are used when we reach a corner, L_1-shaped configurations always follow a U_1-shaped configuration at the end of a row (or column), L_0-shaped configurations are used repeatedly when visiting a whole row (or column), U_0-shaped configurations are used only when visiting the last unvisited row (or column). Hence the previous corollary means that a polite configuration visits whole rows (and columns) without changing direction halfway.

U_0-shaped configurations are used only to scan the pixels in the last row (or column) and are not relevant for our discussion, so from now on we consider only the other configurations. Every configuration can be uniquely identified by its shape, the movement (horizontal or vertical) of its incoming direction and the corner it involves. More precisely, the corner involved by a L_x-shaped configuration is the corresponding D, whereas the corner involved by U_1-shaped configurations is given by $D \setminus \{d\}$ (for instance $\rightarrow\begin{array}{|c|}\hline \downarrow\end{array}$ is used when we reach a corner of kind 2). This leads to the following definition.

Definition 3.3 *Given $S \in \{L_0, L_1, U_1\}$, $m \in \{h, v\}$, and $c \in \{1, 2, 3, 4\}$, notation $S[m, c]$ denotes the S-shaped configuration with c as involving corner and whose incoming direction has m as kind of movement.*

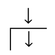
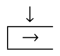
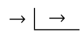
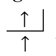
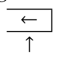
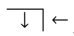
The following remarks are easy consequences of the previous definition and the fact that the next-position function is an actual function.

Remark 3.4 *A polite scanning strategy cannot admit both configurations $L_0\,[m,c]$ and $L_1\,[m,c]$ for some movement m and corner c.*

Remark 3.5 *When applying a polite scanning strategy to some picture support, except when there is only one row or column left to visit, configuration $L_1\,[m,c]$ is always followed by $L_0\,[\overline{m},c]$. Similarly, configuration $U_1\,[m,c]$ is always followed by the same configuration, which is either $L_0\,[\overline{m},c]$ or $L_1\,[\overline{m},c]$, according to the definition of the strategy.*

Example 3.6 *The first configurations defining scanning strategy $\mathfrak{U}$ (see Example 2.5) can be rewritten as follows:*

$$\begin{array}{lllllll} L_1\,[h,1]\,, & L_0\,[v,1]\,, & U_1\,[v,4]\,, & L_0\,[h,4]\,, & U_1\,[h,3]\,, & L_0\,[v,3]\,, & U_1\,[v,2]\,, \\ L_1\,[h,2]\,, & L_0\,[v,2]\,, & U_1\,[v,3]\,, & L_0\,[h,3]\,, & U_1\,[h,4]\,, & L_0\,[v,4]\,, & U_1\,[v,1]\,. \end{array}$$

The configurations are listed in the order they appear when applying $\mathfrak{U}$ to any (big enough) picture support, and such order respects Remark 3.5. For instance, after visiting the leftmost unvisited column downwards with $L_0\,[v,1]$ =  *, one gets in configuration $U_1\,[v,4]$ =*  *; at this point, since η admits configuration $L_0\,[h,4]$ =*  *, then the strategy prescribes to turn and scan the downmost unvisited row rightwards. After visiting the rightmost unvisited column upwards with $L_0\,[v,3]$ =*  *, configuration $U_1\,[v,2]$ =*  *is reached; this time, since η admits configuration $L_1\,[h,2]$ =*  *, the strategy prescribes to go back and scan the rightmost unvisited column downwards.*

Example 3.7 *The first configurations defining scanning strategy μ of Example 2.6, and depicted in Figure 1(g), can be rewritten as follows:*

$$\begin{array}{llllll} L_0\,[h,2]\,, & U_1\,[h,1]\,, & & & & \\ L_0\,[v,1]\,, & U_1\,[v,4]\,, & L_1\,[h,4]\,, & L_0\,[v,4]\,, & U_1\,[v,1]\,, & L_1\,[h,1]\,. \end{array}$$

The configurations are listed in the order they appear when applying μ to any (big enough) picture support, and such order respects Remark 3.5: the first two configurations are used to scan the first row leftwards, whereas the other ones are used cyclically to scan the rest of the picture, column by column, in a boustrophedonic way.

By the previous remarks and examples, it should be clear that any polite scanning strategy behaves as follows: it starts from a corner in some L_x-shaped configuration, visits one whole row (resp. column) staying in some L_0-shaped configuration, reaches a corner with a U_1-shaped configuration, then can either turn the corner and visit the whole adjacent unvisited column (resp. row) with a repeated L_0-shaped configuration, or go back visiting entirely the next unvisited row (resp. column) passing through a L_1-shaped configuration followed by some repeated L_0-shaped configurations; either way, it reaches another corner with a U_1-shaped configuration and so on and so forth.

3.2 Graph representation of polite scanning strategies

Given any polite scanning strategy induced by the triple $\langle \eta, c_s, d_s \rangle$, with d_s of kind m_s, consider the partial function $f_\eta : \{v, h\} \times \{1, 2, 3, 4\} \to \{v, h\}$ defined by setting

$$f_\eta(m, c) = \begin{cases} m & \text{if } \eta \text{ admits } U_1\,[m, c] \text{ and } L_1\,[\overline{m}, c] \\ \overline{m} & \text{if } \eta \text{ admits } U_1\,[m, c] \text{ and } L_0\,[\overline{m}, c] \\ \perp & \text{otherwise} \end{cases}$$

Intuitively, $f_\eta(m, c)$ denotes the kind of movement to execute each time one arrives at a corner of kind c with movement m. We will call such function the *movement function* associated to η. Notice that f_η is well-defined by Remark 3.4.

Example 3.8 *The following tables define the movement functions $f_{\mathfrak{U}}$ associated with $\eta_{\mathfrak{U}}$, and f' associated with the scanning strategy defined in Example 2.6, respectively: the value of $f_\eta(m, c)$ appears in row indexed by m and column indexed by c:*

$f_{\mathfrak{U}}$	1	2	3	4
h	$\perp$	$\perp$	v	v
v	v	v	h	h

f'	1	2	3	4
h	v	$\perp$	$\perp$	$\perp$
v	v	$\perp$	$\perp$	v

Intuitively, the movement function associated to a next-position function η is sufficient to describe how η works: when the head arrives in a corner position c with movement m, then $f_\eta(m, c) = \overline{m}$ represents a change of direction of the head (from v to h or vice versa), meaning that in the corresponding η configurations there is a U_1 followed by a L_0. On the other hand, $f_\eta(m, c) = m$ stands for a U_1-shaped configuration followed by a L_1, and then necessarily a L_0; this means that the head is "going back" - e.g. if it was scanning a row, then it is going to scan the subsequent row in a backward direction.

The movement function f_η of a polite scanning strategy can be represented by a graph having four vertices, corresponding to the four corners, marked 1 to 4, and arcs, labeled by h or v. There exists an arc starting from vertex c with label m if and only if $f_\eta(m, c)$ is defined. In this case: if $f_\eta(m, c) = h$, then such arc is horizontal (i.e. it connects vertices 1 and 2, or 3 and 4) if $f_\eta(m, c) = v$, then it is vertical (i.e. it connects vertices 1 and 4, or 2 and 3). We will call such graph the *movement graph* associated to η. Clearly, diagonal arcs connecting other pairs of vertices are forbidden. Moreover, f_η being a function, from each vertex cannot start more than one arc with a given label.

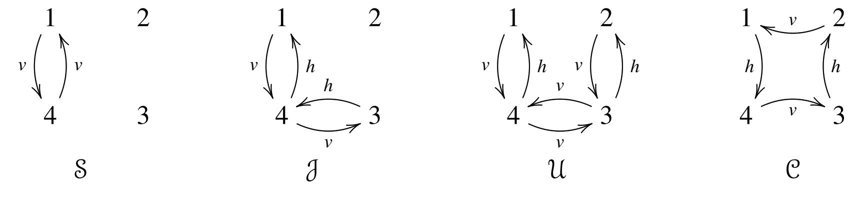


Figure 2: Diagrams of scanning strategies $\mathcal{S}$, $\mathfrak{J}$, $\mathfrak{U}$, and $\mathfrak{C}$.

Example 3.9 *The movement graphs corresponding to $\mathcal{S}$, $\mathcal{J}$, $\mathcal{U}$, and $\mathcal{C}$ are shown in Figure 2.*

4 Characterization of polite scanning strategies

First, we note that in the movement graph associated to a scanning strategy there cannot exist two non-parallel arcs starting from the same vertex and labeled according to their own direction. This fact is formalized by the following proposition.

Proposition 4.1 *Given a polite scanning strategy with next-position function η, for any movement $m \in \{h, v\}$ and kind of corner $c \in \{1, 2, 3, 4\}$, $f_\eta(m, c) = m$ implies $f_\eta(\overline{m}, c) \neq \overline{m}$.*

Proof. By definition, $f_\eta(m, c) = m$ implies that η admits the configurations $U_1[m, c]$, $L_1[\overline{m}, c]$ and, as a consequence of Remark 3.5, $L_0[m, c]$. Similarly, $f_\eta(\overline{m}, c) = \overline{m}$ implies $U_1[\overline{m}, c]$ and $L_1[m, c]$, yielding a contradiction by Remark 3.4. □

Let μ be a polite scanning strategy having movement function f and starting from corner c_0 with direction of kind m_0. Then let

$$m_1 = \begin{cases} \overline{m_0} & \text{if } \eta \text{ admits } L_1[m_0, c_0] \\ m_0 & \text{if } \eta \text{ admits } L_0[m_0, c_0] \\ \perp & \text{otherwise} \end{cases}$$

and consider the (eventually infinite) sequence

$$c_0\, m_1\, c_1\, m_2\, c_2 \ldots \tag{2}$$

where $c_i \in \{1, 2, 3, 4\}$, $m_i \in \{h, v\}$, $m_{i+1} = f(m_i, c_i)$ and $c_{i+1} \sim_{m_i} c_i$ for every $i \geq 0$.

An interesting interpretation of such sequence is the following. When applying a scanning strategy to a domain support (except, as usual, for the last row or column to be visited), one moves from corner to corner, executing horizontal or vertical movements in between; the sequence of such alternating corners and movements is actually a prefix of sequence (2): symbols m_i represent the kind of movements executed, whereas symbols c_i represent the kind of corners reached.

In the graph representing f, such sequence (without c_0) corresponds to a path starting from corner c_1 and leaving it through the arc labeled m_1. Clearly, m_2 correspond to the direction of such arc, and so on.

Example 4.2 *$\mathcal{C}$ produces the sequence $1v4h3v2h1v4h3v2h\ldots$ which can be written as $(1v4h3v2h)^\omega$. Similarly, $\mathcal{S}$, $\mathcal{J}$, and $\mathcal{U}$ produce the sequences $(1v4v)^\omega$, $(1v4h3h4v)^\omega$, and $(1v4h3v2v3h4v)^\omega$, respectively. The scanning strategy of Example 2.6 produces the sequence $2h1v4v1v4v1v\ldots$ which can be written as $2h(1v4v)^\omega$.*

Being the number of possible configurations of next-position functions finite, the sequence can always be decomposed in a starting part (or *bootstrap*) and a cyclic part. The cyclic part is the most significant, because by its nature the bootstrap is always finite, thus pertaining only to a limited part of the picture. Hence, for convenience in the rest we will only represent the cyclic part of sequences in graphs, discarding the bootstrap. If there is an arc a starting from a vertex c with label m, then there must be another arc ending in c directed according to m, if we want arc a to be reachable, and therefore part of the cycle. In this case, the cyclic part of the sequence contains factor mcm', where $m' = f(m, c)$.

Proposition 4.3 *Let f be a movement function. If, for some m and c, $f(m,c) = f(\overline{m}, c) = m$, then the cyclic part of the corresponding sequence (2) cannot contain both the factor mcm and the factor $\overline{m}cm$.*

Proof. Let μ be the polite scanning strategy having movement function f and starting from corner c_0 with direction of kind m_0.

First, let us suppose w.l.o.g that $f(h, 1) = h$ and $f(v, 1) = h$, so that there exists a sequence $m_0\ c_0\ m_1\ c_1\ m_2\ c_2 \ldots$ corresponding to a polite scanning strategy, and containing both $h1h$ and $v1h$ in its cyclic part. This means that, if we render f with a diagram, there is an arc going from 1 to 2 which is marked both h and v - see Figure 3 (i). We will show that it is impossible to build a cycle containing all these arcs. For $v1h$, we need a cycle which reaches 4, but being f a function, there cannot be an arrow going from 1 to 4. Therefore our graph must be like the one in Figure 3 (ii). In fact, the arc from 4 to 1 must bear a h symbol, because we cannot have a vertical arc from 1 to 4. Now, how should we mark the arc from 2 to 1? If it were h, then 4 would be unreachable from 1, because this would mean going back from 2 to 1. So we mark it v and add the corresponding arc from 3 to 2. Moreover, f being a function, the arc from 2 to 3 has to be marked h, and to reach 4 from 2, we have to label the arc from 3 to 4 v – see Figure 3 (iii). Again, f is a function, hence the arc from 3 to 2 has label h, but this means that there is an arc from 4 to 3, labeled v, as in Figure 3 (iv). But this last arc is unreachable because, by construction, we do not have the 1 to 4 arc. Hence, it is impossible to have both $h1h$ and $v1h$ in the cyclic part of the sequence. □

$$\begin{tikzcd}
1 \arrow[r, bend right, "{h,v}"'] & 2 \arrow[l, bend right] \\
4 \arrow[u, bend left] &
\end{tikzcd}
\qquad
\begin{tikzcd}
1 \arrow[r, bend right, "{h,v}"'] & 2 \arrow[l, bend right] \arrow[d, bend left] \\
4 \arrow[u, bend left, "h"'] & 3 \arrow[l, bend right]
\end{tikzcd}
\qquad
\begin{tikzcd}
1 \arrow[r, bend right, "{h,v}"'] & 2 \arrow[l, bend right, "v"'] \arrow[d, bend left, "h"] \\
4 \arrow[u, bend left, "h"'] & 3 \arrow[l, bend right, "v"'] \arrow[u, bend right]
\end{tikzcd}
\qquad
\begin{tikzcd}
1 \arrow[r, bend right, "{h,v}"'] & 2 \arrow[l, bend right, "v"'] \arrow[d, bend left, "h"] \\
4 \arrow[u, bend left, "h"'] \arrow[r, bend right, "v?"'] & 3 \arrow[l, bend right, "v"'] \arrow[u, bend right, "h"']
\end{tikzcd}$$

(i) (ii) (iii) (iv)

Figure 3: Diagrams for proof of Proposition 4.3.

Example 4.4 *Let f be the movement function of the scanning strategy defined in Example 2.6. Then we have $f(h, 1) = f(v, 1) = v$ but, according to the previous proposition, the cyclic part of the associated sequence contains factor $v1v$ but not factor $h1v$. Actually, the movement graph of such cycle is exactly the first one depicted in Figure 2. This is not a coincidence, as we prove in the following theorem.*

Theorem 4.5 *The movement graph of the cyclic part of any polite scanning strategy is one of those of Figure 2, up to duality and symmetry.*

Proof. By Proposition 4.3 there cannot be an arc labeled both h and v. As a consequence, given an arc a starting from vertex c in direction m, we have only two possibilities: if a is labeled h, then there is an horizontal arc ending in c (i.e. there is a factor hcm in the associated sequence); else if a is labeled by v, then there is a vertical arc ending in c (i.e. there is a factor vcm). Applying this constraint, one can verify that it is possible to build (up to duality and symmetry) only the graphs depicted in Figure 2, or the following graph, which does not satisfy Proposition 4.1 for $c = 1$, and hence has to be ignored.

$$\begin{tikzcd}
1 \arrow[r, bend right, "h"'] \arrow[d, bend right, "v"'] & 2 \arrow[l, bend right, "v"'] \arrow[d, bend right, "h"'] \\
4 \arrow[u, bend right, "h"'] \arrow[r, bend right, "v"'] & 3 \arrow[l, bend right, "v"'] \arrow[u, bend right, "h"']
\end{tikzcd}$$

□

In practice this means that, except for possibly a limited initial part, every polite scanning strategy behaves like one of $\mathcal{S}$, $\mathcal{J}$, $\mathcal{U}$, and $\mathcal{C}$, up to duality and symmetry.

5 Conclusions

In this work we have proved that the behavior of the cyclic part of blind scanning strategies for Wang automata follows either a boustrophedonic, or “L-like”, or “U-like”, or spiral pattern. The set of all possible scanning strategies is understandably finite, so we have also been able to mechanically check the main result through the bounded model checker Zot [19], by expressing strategies as simple linear temporal logic formulae.[2]

Clearly this fact permits a simplification of the model, because the actually usable strategies are simpler and more easily describable than the original generic ones presented in Section 2.1.

This result is also interesting from a theoretical point of view, since different scanning strategies define possibly different subclasses of REC, that we usually call μ-DREC, where μ is the chosen strategy. For instance in [16], we proved that $\mathcal{S}$-DREC and $\mathcal{C}$-DREC are incomparable. Moreover, we know from [17] that the closure w.r.t. rotations of $\mathcal{S}$-DREC is a remarkable class, because it coincides with the class of row- or column-unambiguous subclasses of REC defined in [1]. Therefore, the interesting and still open questions are about $\mathcal{J}$-DREC, and $\mathcal{U}$-DREC, and their relations with $\mathcal{S}$-DREC and $\mathcal{C}$-DREC.

As far as other possible future activities are concerned, one could extend the behavior of Wang automata by considering for instance scanning strategies which are “less blind”, or based on a finite-state device, or permitting a bounded number of visits of the same position (i.e. not one-pass, but “bounded pass”). Clearly, many of these natural extensions could exceed REC, so they have to be handled with care.

[2]The interested reader may find the related Zot script at
http://home.dei.polimi.it/pradella/strat.lisp.

References

[1] M. Anselmo, D. Giammarresi, and M. Madonia. From determinism to non-determinism in recognizable two-dimensional languages. In *Proc. DLT 2007*, volume 4588 of *Lecture Notes in Computer Science*, pages 36–47. Springer, 2007.

[2] M. Anselmo, D. Giammarresi, and M. Madonia. Tiling automaton: A computational model for recognizable two-dimensional languages. In *Proc. CIAA 2007*, volume 4783 of *Lecture Notes in Computer Science*, pages 290–302. Springer, 2007.

[3] A. Bertoni, M. Goldwurm, and V. Lonati. On the complexity of unary tiling-recognizable picture languages. *Fundamenta Informaticae*, 91(2):231–249, 2009.

[4] B. Borchert and K. Reinhardt. Deterministically and sudoku-deterministically recognizable picture languages. In *Proc. LATA 2007*, 2007.

[5] S. Bozapalidis and A. Grammatikopoulou. Recognizable picture series. *Journal of Automata, Languages and Combinatorics*, 10(2/3):159–183, 2005.

[6] R. Brijder and H. J. Hoogeboom. Perfectly quilted rectangular snake tilings. *Theoretical Computer Science*, 410(16):1486–1494, 2009.

[7] Y. Brun. Solving NP-complete problems in the tile assembly model. *Theor. Comput. Sci.*, 395(1):31–46, 2008.

[8] L. Cavallaro, E. Di Nitto, C. A. Furia, and M. Pradella. A tile-based approach for self-assembling service compositions. In *Proc. ICECCS'10*, St. Anne's College, University of Oxford, March 2010. To appear.

[9] L. de Prophetis and S. Varricchio. Recognizability of rectangular pictures by Wang systems. *Journal of Automata, Languages and Combinatorics*, 2(4):269–288, 1997.

[10] L. W. E. Winfree, F. Liu and N. Seeman. Design and self-assembly of two-dimensional DNA crystals. *Nature*, 394:439–544, 1998.

[11] D. Giammarresi and A. Restivo. Recognizable picture languages. *International Journal Pattern Recognition and Artificial Intelligence*, 6(2-3):241–256, 1992. Special Issue on *Parallel Image Processing*.

[12] D. Giammarresi and A. Restivo. Two-dimensional languages. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 3, Beyond Words, pages 215–267. Springer-Verlag, Berlin, 1997.

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

[14] K. Inoue and I. Takanami. A survey of two-dimensional automata theory. *Information Sciences*, 55(1-3):99–121, 1991.

[15] K. Lindgren, C. Moore, and M. Nordahl. Complexity of two-dimensional patterns. *Journal of Statistical Physics*, 91(5-6):909–951, June 1998.

[16] V. Lonati and M. Pradella. Deterministic recognizability of picture languages with Wang automata. Submitted. Available at `http://lonati.dsi.unimi.it/lavori/automatha_lonatipradella.pdf`.

[17] V. Lonati and M. Pradella. Snake-deterministic tiling systems. In *Proc. MFCS 2009*, volume 5734 of *Lecture Notes in Computer Science*, pages 549–560. Springer, 2009.

[18] V. Lonati and M. Pradella. Picture recognizability with automata based on Wang tiles. In *Proc. SOFSEM 2010*, volume 5901 of *Lecture Notes in Computer Science*, pages 576–587. Springer, 2010.

[19] M. Pradella, A. Morzenti, and P. San Pietro. The symmetry of the past and of the future: Bi-infinite time in the verification of temporal properties. In *Proc. of 6th joint meeting of ESEC/FSE*, Dubrovnik, Croatia, September 2007.

[20] H. Wang. Proving theorems by pattern recognition II. *Bell Systems Technical Journal*, 40:1–42, 1961.