Snake-deterministic tiling systems[⋆]

Violetta Lonati[1] and 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 – matteo.pradella@polimi.it

Abstract The concept of determinism, while clear and well assessed for string languages, is still matter of research as far as picture languages are concerned. We introduce here a new kind of determinism, called *snake*, based on the *boustrophedonic* scanning strategy, that is a natural scanning strategy used by many algorithms on 2D arrays and pictures. We consider a snake-deterministic variant of tiling systems, which defines the so-called *Snake-DREC* class of languages. Snake-DREC properly extends the more traditional approach of diagonal-based determinism, used e.g. by deterministic tiling systems, and by online tessellation automata. Our main result is showing that the concept of snake-determinism of tiles coincides with row (or column) unambiguity.

Keywords: picture language, 2D language, tiling systems, online tessellation automata, determinism, unambiguity.

1 Introduction

Picture languages are a generalization of string languages to two dimensions: a picture is a two-dimensional array of elements from a finite alphabet. Several classes of picture languages have been considered in the literature [8,10,6,12]. In particular, here we refer to class REC introduced in [8] with the aim to generalize to 2D the class of regular string languages. REC is a robust class that has various characterizations; in particular, it is the class of picture languages that can be generated by *tiling systems*, a model introduced in [7], where pictures are specified as alphabetic projection of a local 2D language defined by a set of tiles.

For string regular languages, two central notions are those of *determinism* and *unambiguity*. Going towards 2D, the concept of unambiguity is straightforward and yields to class UREC [7]. UREC defines unambiguously tiling recognizable languages, whose pictures are the projection of a unique element in the corresponding local language. In an effort to go towards determinism, the authors of [1] introduced an intermediate notion of "line" unambiguity, embodied in classes Row-UREC and Col-UREC, and based on backtracking at most linear in one dimension of the picture.

[⋆] This work has been supported by the MIUR PRIN project "Mathematical aspects and emerging applications of automata and formal languages", and CNR RSTL 760 "2D grammars for defining pictures".

The concept of determinism for picture languages is far from being well understood. The most relevant difficulty is that in 2D any notion of determinism seems to require some pre-established "scanning strategy" for reading the picture. Tiling systems are implicitly nondeterministic: REC is not closed under complement, and the membership problem is NP-complete [11]. Clearly, this latter fact severely hinders the potential applicability of the notation. The identification of a reasonably "rich" deterministic subset of REC would spur its application, since it would allow linear parsing w.r.t. the number of pixels of the input picture.

In past and more recent years, several different deterministic subclasses of REC have been studied, e.g. the classes defined by deterministic 4-way automata [10] or deterministic online tessellation automata [9]. This latter model inspired the notion of determinism of [1], that relies on four diagonal-based scanning strategies, each starting from one of the four corners of the picture. To mark this aspect, in this paper we will call the corresponding deterministic class Diag-DREC[1].

In a effort to generalize their approach, the same authors in [2] suggest other kinds of strategies. Inspired by their work, we introduce here a new kind of determinism for tiles, based on a boustrophedonic scanning strategy, that is a natural scanning strategy used by many algorithms on pictures and 2D arrays (such as shearsort) [4,2,5]. This leads to a class called Snake-DREC, which can be defined equivalently in terms of tiling systems or online tessellation acceptors.

Snake-DREC properly extends Diag-DREC while keeping some important closure properties. For instance, it is still closed under complement, rotation and symmetries. However, like Diag-DREC, it is not closed under intersection. When pictures of only one row (or column) are considered, this model reduces to deterministic finite state automata. Quite surprisingly, we found that our notion of determinism coincides with line unambiguity of Row-UREC (or Col-UREC): our main result is showing that the languages of this class can actually be recognized deterministically by following a boustrophedonic scanning strategy.

The paper is organized as follows. In Section 2 we recall some basic definitions and properties on two-dimensional languages and tiling systems. In Section 3 we introduce snake-deterministic tiling systems. In Section 4 we present our main result. In the last section we define and characterize class Snake-DREC.

2 Preliminaries

2.1 Tiling recognizable picture languages

The following definitions are taken and adapted from [8].

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^{++}. If C denotes some kind of picture-accepting device, then $\mathcal{L}(\mathrm{C})$ denotes the class of picture languages recognized by such devices.

For $h, k \geq 1$, $\Sigma^{h,k}$ denotes the set of pictures of size (h, k); $\# \notin \Sigma$ is used when needed as a *boundary symbol*; $\hat{p}$ refers to the bordered version of picture p. That is, for $p \in \Sigma^{h,k}$,

[1] The original name is DREC.

it is

$$p = \begin{array}{|c|c|c|}\hline p(1,1) & \dots & p(1,k) \\ \hline \vdots & \ddots & \vdots \\ \hline p(h,1) & \dots & p(h,k) \\ \hline \end{array} \qquad \hat{p} = \begin{array}{|c|c|c|c|c|}\hline \# & \# & \dots & \# & \# \\ \hline \# & p(1,1) & \dots & p(1,k) & \# \\ \hline \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline \# & p(h,1) & \dots & p(h,k) & \# \\ \hline \# & \# & \dots & \# & \# \\ \hline \end{array}.$$

A *pixel* is an element $p(i, j)$ of p. We call (i, j) the *position* in p of the pixel. We will sometimes use position (i, j) with i or j equal to 0, or $h + 1$, or $k + 1$ for referring to borders.

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.

We call *tile* a square picture of size (2,2). We denote by $T(p)$ the set of all tiles contained in a picture p.

Let Σ be a finite alphabet. A (two-dimensional) language $L \subseteq \Sigma^{++}$ is *local* if there exists a finite set Θ of tiles over the alphabet $\Sigma \cup \{\#\}$ such that $L = \{p \in \Sigma^{++} \mid T(\hat{p}) \subseteq \Theta\}$. We will refer to such language as $L(\Theta)$.

Let $\pi : \Gamma \to \Sigma$ be a mapping between two alphabets. Given a picture $p \in \Gamma^{++}$, the *projection* of p by π is the picture $\pi(p) \in \Sigma^{++}$ such that $\pi(p)\ (i, j) = \pi(p(i, j))$ for every position (i, j). Analogously, the projection of a language $L \subseteq \Gamma^{++}$ by π is the set $\pi(L) = \{\pi(p) \mid p \in \Gamma^{++}\} \subseteq \Sigma^{++}$.

A *tiling system* (TS) is a 4-tuple $\tau = \langle \Sigma, \Gamma, \Theta, \pi \rangle$ where Σ and Γ are two finite alphabets, Θ is a finite set of tiles over the alphabet $\Gamma \cup \{\#\}$ and $\pi : \Gamma \to \Sigma$ is a projection. A picture language $L \subseteq \Sigma^{++}$ is *tiling recognizable* if there exists a tiling system $\langle \Sigma, \Gamma, \Theta, \pi \rangle$ such that $L = \pi(L(\Theta))$. We say that τ generates L and denote by REC the class of picture languages that are tiling recognizable, i.e, REC $= \mathcal{L}(\text{TS})$. Notice in particular that any local language is tiling recognizable.

Example 1. The language L_{center} of square pictures over $\{0, 1\}$ with odd size, greater than 2, and having 1 only in the center is generated by the tiling system $\langle \Sigma, \Gamma, \Theta, \pi \rangle$, where: $\Gamma = \{1, \diagdown, \diagup, \cdot\}$; $\pi(1) = 1$, $\pi(x) = 0$ for $x \neq 1$, and the set of tiles is $\Theta = T(\hat{p})$,

$$p = \begin{array}{|c|c|c|c|c|c|c|}\hline \diagdown & \cdot & \cdot & \cdot & \cdot & \cdot & \diagup \\ \hline \cdot & \diagdown & \cdot & \cdot & \cdot & \diagup & \cdot \\ \hline \cdot & \cdot & \diagdown & \cdot & \diagup & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ \hline \cdot & \cdot & \diagup & \cdot & \diagdown & \cdot & \cdot \\ \hline \cdot & \diagup & \cdot & \cdot & \cdot & \diagdown & \cdot \\ \hline \diagup & \cdot & \cdot & \cdot & \cdot & \cdot & \diagdown \\ \hline \end{array}.$$

Notice that it is straightforward to extend the previous tiling system to define the language L'_{center} of square pictures with odd size, and having 1 not only in the center, but possibly elsewhere. E.g., we may set $\Gamma = (\{0,1\} \times \{\diagdown, \diagup, \cdot\}) \cup \{(1,1)\}$ and π: $\pi(0,y) = 0, \pi(1,x) = 1$.

REC coincides with the class of languages recognized by online tessellation acceptors (OTA), that are special acceptors related to cellular automata [9]. Informally, an online tessellation acceptor can be described as an infinite two-dimensional array of identical finite-state automata, where the computation proceeds by counter-diagonals starting from top-left towards bottom-right corner of the input picture. A run of a OTA on a picture consists in associating a state to each position of the picture. At the beginning, an initial state is assigned to all top and left border positions. The state at position (i, j) is given by the transition function and depends both on the symbol of the picture at that position, and on the states already associated with positions $(i, j-1)$, $(i-1, j-1)$ and $(i-1, j)$. The picture is accepted if the state associated with the bottom-right corner is final.

A natural subclass of REC, already introduced in [7], is UREC consisting of the tiling recognizable languages whose pictures are the projection of a unique element in the corresponding local language. Formally, a tiling system $\langle\Sigma, \Gamma, \Theta, \pi\rangle$ is called *unambiguous* if, for every $q, q' \in L(\Theta)$, $\pi(q) = \pi(q')$ implies $q = q'$. UREC is the class of all unambiguous languages. It is known that UREC $\subset$ REC and that it is undecidable whether a tiling system is unambiguous [3].

2.2 Diagonal-deterministic languages

Here we present the notion of determinism proposed in [1]. This is inspired by the deterministic version of online tessellation acceptors [9], which are directed according to a corner-to-corner direction (namely, from top-left to bottom-right, or *tl2br*).

Consider a scanning strategy that respects the tl2br direction: any position (x, y) is read only if all the positions that are above and to the left of (x, y) have already been read. Roughly speaking, tl2br determinism means that, given a picture $p \in \Sigma^{++}$, its preimage $p' \in L(\Theta) \subseteq \Gamma^{++}$ can be build deterministically when scanning p with any such strategy. Formally, a tiling system $\tau = \langle\Sigma, \Gamma, \Theta, \pi\rangle$ is called *tl2br-deterministic* if for any $X, Y, Z \in \Gamma \cup \{\#\}$ and $a \in \Sigma$, there exists at most one tile

$$\begin{array}{|c|c|}\hline X & Y \\ \hline Z & A \\ \hline\end{array} \in \Theta \qquad \text{with } \pi(A) = a.$$

By rotation, one can define *d-deterministic* tiling systems (*d*-DTS) for any corner-to-corner direction d in {tl2br, tr2bl, bl2tr, br2tl}, where t, b, l, and r stand for top, bottom, left, and right, respectively.

Example 2. The language $L_{fr=fc}$ of square pictures where the first row equals the first column is in $\mathcal{L}$(tl2br-DTS) $\cap$ $\mathcal{L}$(tr2bl-DTS) $\cap$ $\mathcal{L}$(bl2tr-DTS), but does not belong to $\mathcal{L}$(br2tl-DTS) [1].

We use Diag-DREC to denote the family of languages recognized by some d-DTS (for all corner-to-corner directions d). Diag-DREC is equal to the closure by rotation of the class of languages recognized by deterministic OTAs (denoted as DOTAs).

Example 3. The language $L_{\exists r=lr}$ of square pictures where there is one row that equals the last one cannot be recognized neither by any tl2br-DTS [9, Theorem 3.1], nor (symmetrically) by any tr2bl-DTS. However, $L_{\exists r=lr} \in$ Diag-DREC since one can prove that it is recognized both by bl2tr-DTS and br2tl-DTS.

Diag-DREC is properly included in UREC, as the following example testifies.

Example 4. Let $L_{\text{frames}} = L_{fr=fc} \cap L_{lr=lc} \cap L_{2fr=\overline{2lc}} \cap L_{2lr=\overline{2fc}}$ be the language of square pictures such that: the first row equals the first column, the last row equals the last column, the second row equals the reverse of the second last column, the second last row equals the reverse of the second column. Then L_{frames} is in UREC but not in Diag-DREC [1].

2.3 Row and column unambiguity

In [1] a hierarchy of classes between determinism and unambiguity is also exhibited. Here we adapt the basic definitions, results, and examples from [1].

Consider the side-to-side direction t2b (from top to bottom). A tiling system $\langle \Sigma, \Gamma, \Theta, \pi \rangle$ is called t2b-*unambiguous* if, for any rows $\mathbf{X} = (X_1, X_2, \cdots, X_m) \in \Gamma^{1,m} \cup \{\#\}^{1,m}$ and $\mathbf{a} = (a_1, a_2, \cdots, a_m) \in \Sigma^{1,m}$, there exists at most one row $\mathbf{A} = (A_1, A_2, \cdots, A_m) \in \Gamma^{1,m}$ such that

$$\pi(\mathbf{A}) = \mathbf{a} \quad \text{and} \quad T\left(\begin{array}{|c|c|c|c|c|c|} \hline \# & X_1 & X_2 & \ldots & X_m & \# \\ \hline \# & A_1 & A_2 & \ldots & A_m & \# \\ \hline \end{array} \right) \subseteq \Theta . \tag{1}$$

Example 5. Let $L = L_{\exists c=fc} \cap L_{\exists c=lc}$ of square pictures where there are one column that equals the first one and one column that equals the last one. L is not in Diag-DREC, but it can be recognized by a t2b-unambiguous tiling system.

Similar properties define d-unambiguous tiling systems (d-UTS) for any side-to-side direction $d \in \{t2b, b2t, l2r, r2l\}$. Row-UREC (resp. Col-UREC) denotes the class of *row-unambiguous* (resp. *column-unambiguous*) languages, i.e., the languages generated by t2b-UTS or b2t-UTS (resp. l2r-UTS or r2l-UTS). The following relation holds, with all strict inclusions:

$$\text{Diag-DREC} \subset (\text{Col-UREC} \;\cap\; \text{Row-UREC}) \subset (\text{Col-UREC} \;\cup\; \text{Row-UREC}) \subset \text{UREC}. \tag{2}$$

In the rest of the paper we will use the informal term *line unambiguity* for referring both to row and to column unambiguity.

3 Snake-deterministic tiling systems

Given a tiling system $\tau = \langle \Sigma, \Gamma, \Theta, \pi \rangle$ and a picture $p \in \Sigma^{++}$, imagine to build one preimage $p' \in L(\Theta)$, $\pi(p') = p$, by scanning p with a boustrophedonic strategy. More precisely, start from the top-left corner, scan the first row of p rightwards, then scan the second row leftwards, and so on, like in the following picture, where the number in each pixel denotes its scanning order:

$$\begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & 4 \\ \hline 8 & 7 & 6 & 5 \\ \hline 9 & 10 & 11 & 12 \\ \hline \end{array} \;.$$

This means that we scan odd rows rightwards and even row leftwards, assigning a symbol in Γ to each position. The choice of the symbol clearly depends on the symbols in the neighbourhood, and it is determined by a tile in Θ. In general, the choice is not unique and hence the procedure may not be deterministic. We introduce the following definition to guarantee such condition.

Definition 1. *A tiling system $\tau = \langle \Sigma, \Gamma, \Theta, \pi \rangle$ is* snake-deterministic *if Γ and Θ can be partitioned as $\Gamma = \Gamma_1 \cup \Gamma_2$, $\Theta = \Theta_1 \cup \Theta_2$, where*

- *$\langle \Sigma, \Gamma, \Theta_1, \pi \rangle$ is tl2br-deterministic and Θ_1 contains only tiles like*

$$\begin{array}{|c|c|}\hline a_2 & b_2 \\ \hline a_1 & b_1 \\ \hline \end{array}\,, \quad \text{with } a_i, b_i \in \Gamma_i \cup \{\#\} \text{ for } i = 1, 2;$$

- *$\langle \Sigma, \Gamma, \Theta_2, \pi \rangle$ is tr2bl-deterministic and Θ_2 contains only tiles like*

$$\begin{array}{|c|c|}\hline a_1 & b_1 \\ \hline a_2 & b_2 \\ \hline \end{array}\,, \quad \text{with } a_i, b_i \in \Gamma_i \cup \{\#\} \text{ for } i = 1, 2, \text{ and } (a_1, b_1) \neq (\#, \#).$$

Snake-deterministic tiling systems are abbreviated as snake-DTS.

In the following we shall always use this notation: symbols on light gray background belong to $\Gamma_1 \cup \{\#\}$, symbols on dark gray background belong to $\Gamma_2 \cup \{\#\}$. Hence tiles in Θ_1 or Θ_2 will appear as $\begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline \end{array}$ or $\begin{array}{|c|c|}\hline c & d \\ \hline a & b \\ \hline \end{array}$, respectively.

In [1], it is proved that tl2br-deterministic tiling systems are equivalent to deterministic online tessellation acceptors (DOTA). Analogously, here we introduce a similar model of acceptor equivalent to snake-deterministic tiling systems.

Definition 2. *A* deterministic snake online tessellation acceptor *(ZOTA) is a 7-tuple $\langle \Sigma, Q_1, Q_2, q_{01}, q_{02}, F, \delta \rangle$ where:*

- *Σ is the input alphabet;*
- *Q_1 and Q_2 are two disjoint set of states;*
- *$q_{0i} \in Q_i$ are the initial states;*
- *$F \subset Q_1 \cup Q_2 = Q$;*
- *$\delta : Q \times Q \times Q \times \Sigma \mapsto Q$ is the transition function satisfying $\delta(p_1, q_1, p_2, a) \in Q_2$, $\delta(p_2, q_2, p_1, a) \in Q_1$, for every $p_i, q_i \in Q_i$ and $a \in \Sigma$.*

A run of any ZOTA on a picture consists in scanning the picture following the snake-like strategy, associating, at each step, a state with the current position in the picture. At step 0, the initial state q_{01} is assigned to all the border positions $(0, j)$ and $(i, 0)$ with i even, whereas the initial state q_{02} is assigned to all positions $(i, 0)$ with i odd. The state at position (i, j) is given by the transition function and depends on the input symbol at that position and on the states already associated with some of the neighbouring positions: for odd i, the positions considered are $(i-1, j)$, $(i-1, j-1)$, and $(i, j-1)$; for even i, $(i-1, j)$, $(i-1, j+1)$, and $(i, j+1)$. The picture is accepted if the state associated with the last position (i.e. the bottom-rightmost for pictures with an odd number of rows, the bottom-leftmost otherwise) is in F.

Reasoning as in [8, Theorem 8.1] one can easily prove that deterministic snake tessellation automata are equivalent to snake-deterministic tiling systems.

Proposition 1. *$\mathcal{L}$(ZOTA) = $\mathcal{L}$(snake-DTS).*

Proposition 2. *$\mathcal{L}$(snake-DTS) is a boolean algebra.*

Proof (sketch). If L is recognized by a ZOTA, then so is its complement (it is sufficient to exchange final states with non-final ones). Hence, by Proposition 1, $\mathcal{L}$(snake-DTS) is closed under complement.

Moreover, given two snake-DTSs recognizing two languages L_1 and L_2 respectively, one can follow the construction defined in [8] to build a new TS recognizing the intersection $L_1 \cap L_2$. Such construction preserves snake-determinism, hence we get the closure under intersection. □

The following example shows how a snake-DTS, or, equivalently, a ZOTA, can "propagate signals" both in tl2br and tr2bl corner-to-corner directions. This property does not hold in general for tl2br-DTSs or tr2bl-DTSs. Indeed, we will show in Section 4 that they are strictly less powerful than snake-DTSs.

Example 6. The language L_{center} described in Example 1 is recognized by the following snake-DTS. $\Gamma = \Gamma_1 \cup \Gamma_2$ where

$$\Gamma_0 = \{\diagdown, \diagup, \searrow, \swarrow, \rightarrow, \leftarrow, -, \cdot, 1\}, \qquad \Gamma_1 = \{1\} \times \Gamma_0, \qquad \Gamma_2 = \{2\} \times \Gamma_0,$$

and π is such that $\pi(x, 1) = 1$, $\pi(x, y) = 0$, for $y \neq 1$. $\Theta = T(\hat{p}) \cup T(\hat{q})$ where p and q are the following pictures. For better readability, the first component of symbols in Γ_1 is depicted as a light gray background, instead of the symbol 1; analogously, the first component of symbols in Γ_2 is depicted as a dark gray background.

$$p = \begin{array}{|c|c|c|c|c|c|c|c|c|}
\hline
\diagdown & \rightarrow & - & - & - & - & - & - & - \\
\hline
\cdot & \diagdown & \cdot & \cdot & \cdot & \cdot & \leftarrow & \diagup & - \\
\hline
\cdot & \cdot & \diagdown & \rightarrow & \cdot & \cdot & \diagup & \cdot & \cdot \\
\hline
\cdot & \cdot & \cdot & \searrow & \leftarrow & \diagup & \cdot & \cdot & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & 1 & \rightarrow & \cdot & \cdot & \cdot \\
\hline
\cdot & \cdot & \leftarrow & \diagup & \cdot & \diagdown & \cdot & \cdot & \cdot \\
\hline
\cdot & \cdot & \diagup & \cdot & \cdot & \cdot & \diagdown & \rightarrow & \cdot \\
\hline
\leftarrow & \diagup & \cdot & \cdot & \cdot & \cdot & \cdot & \diagdown & \cdot \\
\hline
\diagup & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \diagdown \\
\hline
\end{array}, \qquad q = \begin{array}{|c|c|c|c|c|c|c|}
\hline
\diagdown & \rightarrow & - & - & - & - & - \\
\hline
\cdot & \diagdown & \cdot & \cdot & \leftarrow & \diagup & - \\
\hline
\cdot & \cdot & \diagdown & \rightarrow & \swarrow & \cdot & \cdot \\
\hline
\cdot & \cdot & \leftarrow & 1 & \cdot & \cdot & \cdot \\
\hline
\cdot & \cdot & \diagup & \cdot & \diagdown & \rightarrow & \cdot \\
\hline
\leftarrow & \diagup & \cdot & \cdot & \cdot & \diagdown & \cdot \\
\hline
\diagup & \cdot & \cdot & \cdot & \cdot & \cdot & \diagdown \\
\hline
\end{array}.$$

The basic mechanism of this tiling system is the same as the one of Example 1: the two diagonals are used to identify the center. To make the tiles snake-deterministic, we have first to distinguish odd and even rows, by using in Γ a first component 1, and 2, respectively. First, notice that we have to use two prototypal pictures p and q to define the tile-set: p represents pictures in which the center symbol is found during a left-to-right scan, while q represents the other direction. Symbols $\rightarrow$ and $\leftarrow$ mark the fact that there is a diagonal at the position immediately below. This information is needed for the right-to-left component of the boustrophedonic movement, to mark the top-left to bottom-right diagonal (symbol $\rightarrow$), and analogously for the other direction and diagonal (symbol $\leftarrow$). Symbol $-$ is used to identify the start of the top-right to bottom-left diagonal. Symbols $\searrow$ and $\swarrow$ are used both to mark diagonals, and to state that at the following row the center will be found. Notice that the language generated by this tiling system does not contain pictures having side less than 7 - it is clearly straightforward to extend it to cover those cases as well.

A simple extension of the same structure can be used to define a snake-deterministic tiling system for the language L'_{center}, mentioned at the end of Example 1. □

4 Snake determinism is equivalent to line unambiguity

In this section we prove our main result, showing that snake-deterministic tiling systems are equivalent to t2b-unambiguous tiling systems.

Theorem 1. *$\mathcal{L}$(snake-DTS) = $\mathcal{L}$(t2b-UTS).*

In one direction, the result is easy (any snake-deterministic tiling system is also t2b-unambiguous). The converse is less intuitive; in order to prove it, from now on let $\tau = \langle \Sigma, \Gamma, \Theta \pi \rangle$ be a t2b-UTS.

First of all, let Γ_1 (resp. Γ_2) be the set of symbols in Γ that may appear only in odd (resp. even) rows, and w.l.o.g assume that Γ_1 and Γ_2 are disjoint. (Otherwise we can mark with subscript i all elements that may appear in Γ_i, possibly duplicating symbols and tiles.) Consequently, as in the definition of snake-DTS, split the set of tiles into two sets Θ_1 and Θ_2. If the resulting tiling system is not snake-deterministic, then we build a snake-deterministic tiling system $\tilde{\tau} = \langle \Sigma, \tilde{\Gamma}, \tilde{\Theta}, \tilde{\pi} \rangle$ that simulates τ. Before formally defining $\tilde{\tau}$, let us first point out some important remarks.

Given any $\mathbf{X} = (X_1, X_2, \ldots, X_m) \in \Gamma^{1,m} \cup \{\#\}^{1,m}$ and $\mathbf{a} = (a_1, a_2, \ldots, a_m) \in \Sigma^{1,m}$, there exists at most one preimage $\mathbf{A} = (A_1, A_2, \ldots, A_m) \in \Gamma^{1,m}$ satisfying relation (1). However, we have no guarantees that $\mathbf{A}$ can be built from left to right deterministically. For instance, for $m = 4$, τ may allow the choices represented in Figure 1 (left).

$$\begin{tikzcd}
\# \arrow[r] \arrow[dr] \arrow[ddr] & A_1 \arrow[r] \arrow[dr] & A_2 \arrow[r] & A_3 \arrow[r] \arrow[dr] & A_4 \arrow[r] & \# \\
& A'_1 \arrow[dr] & A'_2 \arrow[r] & A'_3 & A'_4 & \\
& A''_1 \arrow[r] & A''_2 \arrow[r] & A''_3 & &
\end{tikzcd}$$

$$\begin{tikzcd}
\# \arrow[r] \arrow[dr] \arrow[ddr] & A_1 \arrow[r] \arrow[dr] & A_2 \arrow[r] & A_3 \arrow[r] \arrow[dr] & A_4 \arrow[r] & \# \\
& A'_1 & A'_2 \arrow[r] & A'_3 & A'_4 & \\
& A''_1 & & & &
\end{tikzcd}$$

Figure1. Graph (left) and tree (right) of the preimages of row (a_1, a_2, a_3, a_4).

Clearly, τ being t2b-unambiguous, only one branch of the graph ends with #: the one corresponding to $\mathbf{A}$. In the other cases, a backtracking linear in the length of the row is always sufficient to (eventually) determine $\mathbf{A}$.

Remark 1. Since we are building a preimage of a fixed row $\mathbf{a}$, at each position we can choose among symbols that all have the same image through π. E.g., $\pi(A_1) = \pi(A'_1) = \pi(A''_1) = a_1$.

Remark 2. Since τ is t2b-unambiguous, the branch corresponding to $\mathbf{A}$ cannot contain symbols with in-degree greater that one (otherwise there would exist two different preimages of row $\mathbf{a}$ satisfying relation (1), a contradiction). In other words, if two or more branches “collapse”, the successive symbols may be ignored. Then we can assume that the graph of the preimages of **a** is actually a tree, where each symbol has exactly one predecessor. We call it the *tree of the preimages of **a** in Γ*. For the previous example, the tree is depicted in Figure 1 (right).

Similar remarks can be done if we try to build the preimage of row **a** from right to left.

To simulate τ deterministically on a picture p, we proceed as follows. Let p' be the unique preimage of p in $L(\Theta)$. When scanning rightwards the first row of p, we compute and keep trace of the tree of its preimages in Γ; at the end of the row, we determine which branch is successful (i.e, the one that corresponds to the first row of p'). When scanning the second row backwards, we use such information (together with the traces we left in the previous scan) to reconstruct backwards the successful branch and, at the same time, we compute and keep trace of the tree of the preimages in Γ of the current row. This procedure continues till the last row has been scanned.

To represent locally the tree of preimages of the current row, we store at each position the set of symbols of Γ that may appear at the corresponding position of p', together with their predecessors in the tree. To represent the correspondence between a symbol and its predecessors, which is unique by Remark 2, we use partial functions. For instance, the tree of the previous example is represented by the sequence of partial functions $(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)$, where $\alpha_1(A_1) = \alpha_1(A'_1) = \alpha_1(A''_1) = \#$, $\alpha_2(A_2) = \alpha_2(A'_2) = A_1$, $\alpha_3(A_3) = A_2, \alpha_3(A'_3) = A'_2$, $\alpha_4(A_4) = \alpha_4(A'_4) = A_3$, and $\alpha_5(\#) = A_4$.

We shall need some notation. Given a partial function $f : X \to Y$, we set $I_f = f(\Delta_f)$; moreover, we write $f(x) =\perp$ if $f(x)$ is not defined, set $\Delta_f = \{x \in X \mid f(x) \neq\perp\}$, and say that f is non-empty if $\Delta_f \neq \emptyset$. For $i = 1, 2$, let $\hat{\Gamma}_i = \Gamma_i \cup \{\#\}$ and call Φ_i the set of non-empty partial functions $\varphi : \hat{\Gamma}_i \to \hat{\Gamma}_i$ such that $|\pi(\Delta_\varphi)| = |\pi(I_\varphi)| = 1$ (this last condition is the formalization of Remark 1). In particular, for every $A \in \Gamma_i$, let $\sharp_A$ be the function in Φ_i with domain $\{\#\}$ such that $\sharp_A(\#) = A$; moreover, let $\sharp$ be the function with domain $\{\#\}$ such that $\sharp(\#) = \#$. Finally, we abbreviate $\pi(\Delta_\varphi)$ by $\pi(\varphi)$.

Recall that during the simulation we perform two operations at the same time: we reconstruct the successful branch in the tree of preimages of the previous row, and compute the tree of preimages of the current row. Hence the local alphabet of $\tilde{\tau}$ must contain both pieces of information. This leads to the following definition:

$$\tilde{\Gamma} = \tilde{\Gamma}_1 \cup \tilde{\Gamma}_2 \text{ where } \tilde{\Gamma}_1 = \hat{\Gamma}_2 \times \Phi_1, \text{ and } \tilde{\Gamma}_2 = \hat{\Gamma}_1 \times \Phi_2, \tag{3}$$

$$\forall (A, \varphi) \in \tilde{\Gamma} : \ \tilde{\pi}(A, \varphi) = \pi(\varphi). \tag{4}$$

The role of symbol (A, φ) is the following: A is the correct symbol that one should have chosen when scanning the above position (i.e., the symbol appearing at that position in p'), whereas φ keeps trace of all possible symbols that may appear in the current position, together with their predecessors in the computation. Notice that w.l.o.g we define more than one border symbol in $\tilde{\Gamma}$, i.e., all pairs (A, φ) with $\pi(\varphi) = \#$.

In order to define the set of tiles, we need some other notations. For any $b \in \Sigma \cup \{\#\}$, we introduce the partial function r-next$_b$: $\hat{\Gamma}_2 \times \hat{\Gamma}_2 \times \Phi_1 \to \Phi_1$ by setting r-next$_b(X, Y, \alpha) = \beta$, where, for every $B \in \hat{\Gamma}_1$:

$$\beta(B) = \begin{cases} A & \text{if } \pi(B) = b \text{ and } A \text{ is the unique element in } \Delta_\alpha \text{ s.t. } \begin{array}{|c|c|}\hline X & Y \\ \hline A & B \\ \hline\end{array} \in \Theta_1, \\ \perp & \text{otherwise.} \end{cases}$$

Informally, r-next$_b(X, Y, \alpha)$ represents all possible symbols that can appear in next position, when going rightwards, reading symbol b, and given previous neighbours like $\begin{array}{|c|c|}\hline X & Y \\ \hline A & \\ \hline \end{array}$, with $A \in \Delta_\alpha$.

Symmetrically, for any $d \in \Sigma$ let l-next$_d : \hat{\Gamma}_1 \times \hat{\Gamma}_1 \times \Phi_2 \to \Phi_2$ be the partial function defined by l-next$_d(A, B, \gamma) = \delta$, where, for every $D \in \hat{\Gamma}_2$:

$$\delta(D) = \begin{cases} C & \text{if } \pi(D) = d \text{ and } C \text{ is the unique element in } \Delta_\gamma \text{ s.t. } \begin{array}{|c|c|}\hline A & B \\ \hline D & C \\ \hline \end{array} \in \Theta_2, \\ \bot & \text{otherwise.} \end{cases}$$

Lemma 1. *Let $\tau = \langle \Sigma, \Gamma, \Theta, \pi \rangle$ be t2b-unambiguous and let $\boldsymbol{X} = (X_1, X_2, \ldots, X_m)$, $\boldsymbol{A} = (A_1, A_2, \ldots, A_m)$ and $\boldsymbol{a} = (a_1, a_2, \ldots, a_m)$ satisfying relation (1), with $\boldsymbol{X}_i \in \Gamma_2 \cup \{\#\}$ for every i. Moreover set*

$$\alpha_1 = \textit{r-next}_{a_1}(\#, X_1, \sharp), \qquad \forall j = 2, \ldots, m : \ \alpha_j = \textit{r-next}_{a_j}(X_{j-1}, X_j, \alpha_{j-1})$$

Then, for every $j = 1, 2, \ldots, m$, $A_j \in \Delta_{\alpha_i}$, $\alpha_1(A_1) = \#$, $\alpha_j(A_j) = A_{j-1}$ for $j \neq 1$. Similar results hold for the symmetric direction.

Proof. We reason by induction on $j = 1, 2, \ldots, m$. For sake of brevity, we use Δ_i to denote Δ_{α_i}. Clearly, $A_1 \in \Delta_1$ with $\alpha_1(A_1) = \#$. Now, assuming that the statement holds for $k \leq j$, we prove it for $j+1$. We have $\begin{array}{|c|c|}\hline X_j & X_{j+1} \\ \hline A_j & A_{j+1} \\ \hline \end{array} \in \Theta$. If $\begin{array}{|c|c|}\hline X_j & X_{j+1} \\ \hline A'_j & A_{j+1} \\ \hline \end{array} \in \Theta$ for some other $A'_j \in \Delta_j$ then, setting $A'_{k-1} = \alpha_k(A'_k)$ for every $k = j, \ldots, 2, 1$, we obtain that $\begin{array}{|c|c|c|c|c|c|c|c|}\hline \# & X_1 & \ldots & X_j & X_{j+1} & \ldots & X_m & \# \\ \hline \# & A'_1 & \ldots & A'_j & A_{j+1} & \ldots & A_m & \# \\ \hline \end{array} \in \tilde{\Theta}_1$. This yields a contradiction, since also relation (1) holds but τ is 2tb-unambiguous. Thus, A_j is unique and hence $A_{j+1} \in \Delta_{j+1}$ with $\alpha_{j+1}(A_{j+1}) = A_j$. □

We are ready to prove Theorem 1, as a straightforward consequence of the following proposition.

Proposition 3. *Given a t2b-unambiguous tiling system $\tau = \langle \Sigma, \Gamma, \Theta, \pi \rangle$, let $\tilde{\tau}$ be the tiling system $= \langle \Sigma, \tilde{\Gamma}, \tilde{\Theta}, \tilde{\pi} \rangle$ where $\tilde{\Gamma}$ and $\tilde{\pi}$ are defined as in (3) and (4), while $\tilde{\Theta} = \tilde{\Theta}_1 \cup \tilde{\Theta}_2$, where*

$$\tilde{\Theta}_1 = \left\{ \begin{array}{|c|c|}\hline (A,\delta) & (B,\gamma) \\ \hline (D,\lambda) & (\delta(D),\mu) \\ \hline \end{array} \;\middle|\; \begin{array}{c} (\pi(\delta), \pi(\gamma)) = (\#,\#) \ \Rightarrow \ (A,B) = (\#,\#), \\ D \in \Delta_\delta,\ \delta(D) \in \Delta_\gamma,\ \pi(\lambda) = \# \ \Rightarrow \ \lambda = \sharp, \\ \mu = \textit{r-next}_{\tilde{\pi}(\mu)}(D, \delta(D), \lambda) \end{array} \right\}$$

$$\tilde{\Theta}_2 = \left\{ \begin{array}{|c|c|}\hline (X,\alpha) & (Y,\beta) \\ \hline (\beta(B),\delta) & (B,\gamma) \\ \hline \end{array} \;\middle|\; \begin{array}{c} (\pi(\alpha), \pi(\beta)) \neq (\#,\#), \\ B \in \Delta_\beta,\ \beta(B) \in \Delta_\alpha,\ \pi(\gamma) = \# \ \Rightarrow \ \gamma = \sharp, \\ \delta = \textit{l-next}_{\tilde{\pi}(\delta)}(\beta(B), B, \gamma) \end{array} \right\}$$

Then, $\tilde{\tau}$ is a snake-DTS equivalent to τ.

Proof. The TS $\tilde{\tau}$ is snake-deterministic by definition. We prove that $\tilde{\pi}(L(\tilde{\Theta}) = \pi(L(\Theta))$. First let $\tilde{p} \in L(\tilde{\Theta})$. W.l.o.g assume that the number of rows of $\tilde{p}$ is odd; then $\widehat{\tilde{p}}$ is as in

$(\#, \sharp)$	$(\#, \sharp)$	$(\#, \sharp)$	$\cdots$	$(\#, \sharp)$	$(\#, \sharp)$
$(\#, \sharp)$	$(\#, \alpha_{1,1})$	$(\#, \alpha_{1,2})$	$\cdots$	$(\#, \alpha_{1,m})$	$(\#, \sharp_{A_{1,m}})$
$(\#, \sharp_{A_{2,1}})$	$(A_{1,1}, \alpha_{2,1})$	$(A_{1,2}, \alpha_{2,2})$	$\cdots$	$(A_{1,m}, \alpha_{2,m})$	$(\#, \sharp)$
$(\#, \sharp)$	$(A_{2,1}, \alpha_{3,1})$	$(A_{2,2}, \alpha_{3,2})$	$\cdots$	$(A_{2,m}, \alpha_{3,m})$	$(\#, \sharp_{A_{3,m}})$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$(\#, \sharp)$	$(A_{n-1,1}, \alpha_{n,1})$	$(A_{n-1,2}, \alpha_{n,2})$	$\cdots$	$(A_{n-1,m}, \alpha_{n,m})$	$(\#, \sharp_{A_{n,m}})$
$(\#, \sharp)$	$(A_{n,1}, \sharp)$	$(A_{n,2}, \sharp)$	$\cdots$	$(A_{n,m}, \sharp)$	$(\#, \sharp)$

#	#	#	$\cdots$	#	#	#
#	$A_{1,1}$	$A_{1,2}$	$\cdots$	$A_{1,m-1}$	$A_{1,m}$	#
#	$A_{2,1}$	$A_{2,2}$	$\cdots$	$A_{2,m-1}$	$A_{2,m}$	#
#	$A_{3,1}$	$A_{3,2}$	$\cdots$	$A_{3,m-1}$	$A_{3,m}$	#
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
#	$A_{n,1}$	$A_{n,2}$	$\cdots$	$A_{n,m-1}$	$A_{n,m}$	#
#	#	#	$\cdots$	#	#	#

Figure2. Examples of bordered pictures in $L(\tilde{\Theta})$ (left), and $L(\Theta)$ (right).

Figure 2 (left). By the definition of $\tilde{\Theta}$, this implies that the picture p in Figure 2 (right) belongs to $L(\Theta)$. Moreover, one can easily see that $\pi(p) = \tilde{\pi}(\tilde{p})$. Hence, $\tilde{\pi}(L(\tilde{\Theta})) \subseteq \pi(L(\Theta))$.

On the other hand, consider a picture p as in Figure 2 (right). Then, let $\tilde{p}$ be a picture as in Figure 2 (left), where symbols $A_{i,j}$ are from p, whereas the partial functions $\alpha_{i,j}$ are defined inductively according to the boustrophedonic order of positions (i, j):

$$\begin{array}{lll}
\alpha_{1,1} = \text{r-next}_{\pi(A_{1,1})}(\#, \#, \sharp), & \alpha_{1,j} = \text{r-next}_{\pi(A_{1,j})}(\#, \#, \alpha_{1,j-1}) & j = 2, \ldots, m, \\
\alpha_{2,m} = \text{l-next}_{\pi(A_{2,m})}(A_{1,m}, \#, \sharp), & \alpha_{2,j} = \text{l-next}_{\pi(A_{2,j})}(A_{1,j}, A_{a,j+1}, \alpha_{2,j+1}) & j = m-1, \ldots, 2, \\
\alpha_{3,1} = \text{r-next}_{\pi(A_{3,1})}(\#, A_{2,1}, \sharp), & \alpha_{3,j} = \text{r-next}_{\pi(A_{3,j})}(A_{2,j-1}, A_{2,j}, \alpha_{1,j-1}) & j = 2, \ldots, m, \\
\ldots & &
\end{array}$$

One can verify that each α_{ij} is well defined. Indeed, using Lemma 1 one can prove that, for every $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$, $A_{i,j} \in \Delta_{\alpha_{i,j}}$ and $\alpha_{i,j}(A_{i,j})$ is $A_{i,j-1}$ if i is odd, or $A_{i,j+1}$ if i is even. By the definition of $\tilde{\Theta}$, this implies that $\tilde{p} \in L(\tilde{\Theta})$. Since obviously $\tilde{\pi}(\tilde{p}) = \pi(p)$, we get $\pi(L(\Theta) \subseteq \tilde{\pi}(L(\tilde{\Theta}))$ and this concludes the proof. $\square$

5 Class Snake-DREC

Theorem 1 implies that snake-DTS can simulate both tl2br-DTS and tl2br-DTS. Actually, this extension is proper as shown in next proposition.

Proposition 4. *$\mathcal{L}$(snake-DTS) properly extends $\mathcal{L}$(tl2br-DTS) $\cup$ $\mathcal{L}$(tr2bl-DTS).*

Proof. Since both tl2br-DTS and tl2br-DTS are t2b-unambiguous, the inclusion is a consequence of Theorem 1. The inclusion is proper as testified by the language $L = L_{\exists c=fc} \cap L_{\exists c=lc}$ described in Example 5. $\square$

Notice that $\mathcal{L}$(snake-DTS) does not extend the whole class Diag-DREC. For instance the language $L_{\exists r=lr}$ described in Example 3 is in Diag-DREC but, reasoning as in [1], one can prove that it does not belong to $\mathcal{L}$(snake-DTS). On the contrary, by Proposition 4 we have that the closure under horizontal mirror of $\mathcal{L}$(snake-DTS) properly includes Diag-DREC. However, it is not closed by rotation: for instance $L_{\exists c=lc} \in \mathcal{L}$(snake-DTS) since it is in $\mathcal{L}$(tr2bl-DTS), but its rotation is not (see again Example 3). This leads to the following definition.

Definition 3. *Snake-DREC is the closure under rotation of $\mathcal{L}$(snake-DTS). The languages in Snake-DREC are called* snake-deterministic.

We conclude characterizing Snake-DREC and summarizing its properties in the following theorem.

Theorem 2. *Snake-DREC = Row-UREC $\cup$ Col-UREC. Snake-DREC is properly included between Diag-DREC and UREC. Snake-DREC is closed under complement, rotation and mirrors, but not under intersection.*

Proof. The first identity follows by Theorem 1, by applying rotations. Then, the inclusions are a straightforward consequence of relation (2). Proposition 2 implies the closure under complement; the closure under rotation is obvious by definition; the closure under mirrors follows by the closure under mirrors of both Row-UREC and Col-UREC. $L_{\exists r=fr}$ is in Snake-DREC, but its intersection with all its rotations is not [1]. □

Acknowledgments. We thank Alberto Bertoni and Massimiliano Goldwurm for their useful comments.

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. A computational model for recognizable two-dimensional languages. *Theoretical Computer Science*, 2009. To appear.
3. M. Anselmo, D. Giammarresi, M. Madonia, and A. Restivo. Unambiguous recognizable two-dimensional languages. *Theoretical Informatics and Applications*, 40(2):277–293, 2006.
4. P. Behrooz. *Introduction to Parallel Processing: Algorithms and Architectures*. Kluwer Academic Publishers, Norwell, MA, USA, 1999.
5. A. Bertoni, M. Goldwurm, and V. Lonati. On the complexity of unary tiling-recognizable picture languages. *Fundamenta Informaticae*, 91(2):231–249, 2009.
6. A. Cherubini, S. Crespi Reghizzi, and M. Pradella. Regional languages and tiling: A unifying approach to picture grammars. In *Proc. MFCS 2008*, volume 5162 of *Lecture Notes in Computer Science*, pages 253–264. Springer, 2008.
7. 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*.
8. 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.
9. K. Inoue and A. Nakamura. Some properties of two-dimensional on-line tessellation acceptors. *Information Sciences*, 13:95–121, 1977.
10. K. Inoue and I. Takanami. A survey of two-dimensional automata theory. *Information Sciences*, 55(1-3):99–121, 1991.
11. K. Lindgren, C. Moore, and M. Nordahl. Complexity of two-dimensional patterns. *Journal of Statistical Physics*, 91(5-6):909–951, June 1998.
12. O. Matz. On piecewise testable, starfree, and recognizable picture languages. In M. Nivat, editor, *Proc. FoSSaCS'98*, volume 1378 of *Lecture Notes in Computer Science*, pages 203–210. Springer, 1998.