Comparing two different classes of deterministic recognizable picture languages

by
David Kronenberger

Prof. Dr. Friedrich Otto, Advisor

A thesis submitted in partial fulfillment
of the requirements for the
Degree of Bachelor of Science
in Computer Science

UNIVERSITY OF KASSEL
Hesse, Germany

August 30, 2014

I declare that I have developed and written the enclosed thesis entirely by myself, and have not used sources or means without declaration in the text.

Kassel, August 30, 2014

. . . . . .

(**David Kronenberger**)

Contents

1 Introduction

In this thesis we compare two different definitions of deterministic recognizable picture languages. Furthermore, we examine some closure properties of the corresponding classes of languages.

The first definition of deterministic recognizable picture languages is from K. Reinhardt and B. Borchert and was introduced in [1] in 2006. This definition uses a domino tiling system to recognize a local picture by a deterministic process. They abbreviated their family of all deterministically recognizable picture languages as **DREC**.

In 2007 another and completely different definition was introduced by M. Anselmo, D. Giammarresi and M. Madonia in [2]. This definition uses a tiling system to recognize a picture by proceeding from one corner to the diagonally opposite corner. They also abbreviated their family of all deterministic recognizable picture languages by DREC. In 2009 V. Lonati and M. Pradella called these languages "diagonal deterministic recognizable picture languages" because of its procedure and abbreviated the class of all these languages as *Diag*-**DREC** (see [3]). To avoid ambiguity, we will also adapt the abbreviation *Diag*-DREC in this paper.

At first, we review some basic definitions of picture languages. A picture is a two-dimensional array of symbols and a picture language is a set of pictures. Furthermore, we describe some operations on pictures and languages.

The following section describes the usage of tiling systems (presented in [4]) to generate pictures. This approach is based on the idea of the description of a picture language by a set of tiles. The set of picture languages which are generated by tiling systems is denoted as the class of recognizable picture languages abbreviated with **REC**. This class can be described in many other ways e.g. with a domino system, which is presented in Section 3.2.

Every approach that generates REC recognizes a picture non-deterministically. This is the reason for the two definitions of DREC and *Diag*-DREC which will be presented in the next two chapters. We regard closure properties that are known for *Diag*-DREC and prove some missing closure properties of *Diag*-DREC and DREC. In [2] it is shown that *Diag*-DREC is closed under complement and rotation. We show that *Diag*-DREC is not closed under intersection and union. In the second part of these two chapters we prove that DREC is closed under rotation and intersection.

The next part presents a special kind of two-dimensional cellular automaton which accepts picture languages. These automata are called two-dimensional deterministic online tesselation automata, abbreviated with **2DOTA**, and were introduced in 1977 in [5] by K. Inoue and A. Nakamura. It should be noted that the non-deterministic version is another approach to describe REC.

In the last section we compare *Diag*-DREC and DREC. In so doing we show that *Diag*-DREC is a strict subset of DREC.

2 Preliminaries

This section introduces the most important definitions concerning picture languages. The notions are mainly from [6] and [7]. We assume that the reader is already familiar with basic notions of one-dimensional formal language theory.

Definition 2.1. *Let Σ be a finite, non-empty set of symbols. Then p is called a* ***picture over*** *Σ, if p is a finite rectangular array containing only symbols of Σ. The* ***set of all pictures over*** *Σ is denoted by $\Sigma^{*,*}$. Any $L \subseteq \Sigma^{*,*}$ is called a* ***picture language****.*

Let p be a picture. The **height** and **width** of p are the number of rows and the number of columns and are denoted as $l_1(p)$ and $l_2(p)$, respectively. The **size** of p is the pair $(l_1(p), l_2(p))$. The **empty picture** is the only picture of size $(0,0)$ and is denoted by λ. The set $\Sigma^{h,k} \subset \Sigma^{*,*}$ denotes the **set of pictures over Σ of size** (h,k).

If we need evaluate the value of one specific pixel within the picture, we use $p(i,j)$ to return the symbol in the picture p at the vertical position i and horizontal position j, where $1 \leq i \leq l_1(p)$ and $1 \leq j \leq l_2(p)$.

Sometimes it is necessary that a picture is surrounded by a special symbol to detect whether the border of a picture has been reached. For this purpose, we use the symbol # as border symbol. With $\hat{p}$ we denote the picture p bordered with #'s:

$$\hat{p} = \begin{array}{|c|c|c|c|c|} \hline \# & \# & \cdots & \# & \# \\ \hline \# & p(1,1) & \cdots & p(1,l_2(p)) & \# \\ \hline \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline \# & p(l_1(p),1) & \cdots & p(l_1(p),l_2(p)) & \# \\ \hline \# & \# & \cdots & \# & \# \\ \hline \end{array} .$$

Definition 2.2. *Let $p \in \Sigma^{m,n}$. $q = p((i,j),(i',j'))$ is called a* ***subpicture of*** *p if $1 \leq i < i' \leq m$ and $1 \leq j < j' \leq n$ satisfying the following condition:*

- *for each $k, r (1 \leq k \leq i' - i + 1, 1 \leq r \leq j' - j + 1)$, $q(k,r) = p(k + i - 1, r + j - 1)$ holds.*

Sometimes it is necessary to look at all subpictures with a specific size of a picture. Therefore, we denote the **set of subpictures of size** (h,k) **of a picture** $p \in \Sigma^{*,*}$ as

$$B_{h,k}(p) = \{q \in \Sigma^{h,k} \mid q \text{ is a subpicture of } p\}.$$

We are now able to look at some operations on pictures and languages. The concatenation of pictures can be performed in two directions: horizontally and vertically. Two pictures $p, q \in \Sigma^{*,*}$ can be horizontally concatenated, if $l_1(p) = l_1(q)$. We denote the **horizontal concatenation of** p **and** q by $p \oplus q$. Similarly, the two pictures p and q can be concatenated vertically, if $l_2(p) = l_2(q)$. We then write $p \ominus q$. Moreover, the empty picture λ can always be concatenated and is the neutral element for both of the concatenation operations.

Definition 2.3. *Let $L_1, L_2 \in \Sigma^{*,*}$ be two picture languages. The* ***horizontal concatenation of*** *L_1* ***and*** *L_2 is defined by*

$$L_1 \oplus L_2 = \{p \oplus q \mid p \in L_1, q \in L_2\}$$

Similarly, the vertical concatenation of picture languages is defined.

Another operation is the projection by mapping.

Definition 2.4. *Let Γ and Σ be two finite alphabets and $\pi : \Gamma \to \Sigma$ be a mapping. Let $p \in \Gamma^{m,n}$. $q \in \Sigma^{m,n}$* ***is a projection of*** *p, if $q(i,j) = \pi(p(i,j))$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$. We write $q = \pi(p)$.*

Since the projection is not required to be injective, a picture can have more than one pre-image. We define the **projection of a language** L as

$$L' = \pi(L) = \{q \mid q = \pi(p), p \in L\}.$$

Definition 2.5. *Let $p \in \Sigma^{*,*}$. The* ***clockwise rotated picture*** *p^R* ***of*** *p is*

$$p^R = \begin{array}{|c|c|c|} \hline p(l_1(p),1) & \dots & p(1,1) \\ \hline \vdots & \ddots & \vdots \\ \hline p(l_1(p),l_2(p)) & \dots & p(1,l_2(p)) \\ \hline \end{array} .$$

Counterclockwise rotation can be defined similarly. With this operator, it is possible to define the **rotation of a language** L by

$$L^R = \{p^R \mid p \in L\}.$$

Based on this idea, it is possible to define horizontal and vertical mirroring and transposing of pictures and languages.

Further operations on picture languages are union, intersection and complementation and will be understood as ordinary set operations.

3 Recognizable Picture Languages

In this section we present two approaches leading to the same class of languages. As many approaches lead to this class of languages, it is called the **family of recognizable picture languages**, abbreviated as REC. At first we examine a system that uses sets of tiles, to determine whether a picture belongs to a language or not. This approach is required to define the deterministic recognizable picture languages in Section 4. Furthermore, we describe the domino systems which are necessary to define the deterministic process in Section 5.1. Keep in mind that the domino system is also a possible approach to describe REC.

3.1 Tiling Systems

Tiling systems were introduced by D. Giammaresi and A. Restivo in [4]. They searched for a possibility to define recognizable picture languages by extending the projection of local languages from the one-dimensional case to two dimensions.

Let $p \in \Sigma^{*,*}$ be a picture. We call $B_{2,2}(\hat{p})$ the **set of tiles of the picture** p where a **tile** is a picture of size $(2,2)$. Regard that $B_{2,2}(\hat{p})$ contains any subpicture of the bordered picture $\hat{p}$ of size $(2,2)$. We can now restate the definition of two-dimensional local languages from [7]:

Definition 3.1. *Let Θ be a finite set of tiles over $\Sigma \cup \{\#\}$. A language $L \subseteq \Sigma^{*,*}$ is **local**, if L is generated by Θ as follows:*

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

We write $L = LOC(\Theta)$.

To underline the power of local languages, we review a suitable example from [6].

Example 3.2. *Let Θ be the following finite set of tiles over $\Sigma \cup \{\#\}$, where $\Sigma = \{0,1\}$.*

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

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

The first four tiles are the only tiles that can be used in the four corners. Thus, top-left and bottom-right corners always contain 1s, whereas the other corners always contain 0s. The following eight tiles are for vertical and horizontal borders. It can be seen that these tiles do not allow any other 1s except those in the appropriate corners. The last four tiles are forcing the picture to have 1s on the diagonal and 0s otherwise. Therefore

$$LOC(\Theta) = \{p \in \Sigma^{n,n} \mid n \geq 1, p(i,i) = 1, p(i,j) = 0 \text{ where } 1 \leq i,j \leq l_1(p) \text{ and } i \neq j\}$$

is the language of square-shaped pictures with 1s on the main diagonal and 0s everywhere else.

As this example underlines, the generative power of local languages is restricted. It is e.g. impossible to generate square-shaped pictures with a one-letter alphabet.

Due to the fact that LOC is the natural extension of string local languages [7], we can extend the definition of one-dimensional regular languages to the two-dimensional case:

Definition 3.3. *The quadruple $\tau = (\Sigma, \Gamma, \Theta, \pi)$ is called a **tiling system** (**TS**), where*

- *Σ and Γ are two finite alphabets,*
- *$\pi : \Gamma \to \Sigma$ is a mapping and*
- *Θ is a finite set of tiles over $\Gamma \cup \{\#\}$.*

The language recognized by τ is $L(\tau) = \pi(LOC(\Theta))$.

The **family of languages generated by tiling systems** is called $\mathscr{L}$(TS). This definition solves the restriction of shaping pictures with one letter.

Example 3.4. *Let $\tau = (\{a\}, \{0,1\}, \Theta, \pi)$ be a TS with Θ from Example 3.2 and $\pi(0) = \pi(1) = a$.*

It is obvious that this TS recognizes the language $L(\tau) = \{p \in \{a\}^{n,n} \mid n \geq 0\}$, which contains square-shaped pictures of a's.

3.2 Domino Systems

At first, we will restrict the size of tiles to the sizes $(2,1)$ and $(1,2)$. These tiles are called **dominos**. We are now able to describe the languages recognized by dominos and the projection of those languages.

Definition 3.5. *Let Δ be a finite set of dominos over $\Sigma \cup \{\#\}$. A language $L \subseteq \Sigma^{*,*}$ is **hv-local** if L is generated by Δ as follows:*

$$L = \{p \in \Sigma^{*,*} \mid B_{2,1}(\hat{p}) \cup B_{1,2}(\hat{p}) \subseteq \Delta\}.$$

We write $L = hv\text{-}LOC(\Delta)$.

Like tiling systems, we can now similarly define domino systems:

Definition 3.6. *A quadruple $\tau = (\Sigma, \Gamma, \Delta, \pi)$ is called a* ***domino system (DS)****, where*

- *Σ and Γ are two finite alphabets,*
- *$\pi : \Gamma \to \Sigma$ is a mapping and*
- *Δ is a finite set of dominos over $\Gamma \cup \{\#\}$.*

The language generated by τ is $L(\tau) = \pi(hv\text{-}LOC(\Delta))$.

The family of languages generated by domino systems is denoted by $\mathscr{L}(\mathrm{DS})$.

It can be shown that $\mathscr{L}(\mathrm{DS}) = \mathscr{L}(\mathrm{TS})$ (see [6]). Furthermore, it can be shown that the non-deterministic version of on-line tessellation automata (see Section 6) is equal to $\mathscr{L}(\mathrm{TS})$. All these approaches lead to the same class of picture languages, which is called REC.

4 Diagonal Deterministic Recognizable Picture Languages

In this section we present the family of diagonal deterministic recognizable picture languages, a specialization of recognizable picture languages. Then we examine some closure properties of this class.

To define this kind of picture languages we use tiling systems. First, we will present the *c2c*-deterministic tiling systems, where $c2c \in \{tl2br, tr2bl, bl2tr, br2tl\}$ describes a scanning direction from one corner to its diagonal opposite corner. These tiling systems were introduced in 2007 by M. Anselmo, D. Giammarresi and M. Madonia in [2]. The main idea is based on the 2DOTA presented in Section 6. This kind of deterministic tiling system scans a picture $p \in \Sigma^{*,*}$ from one corner to the diagonally opposite one. While scanning p the system creates the local picture $p' \in LOC(\Theta) \subseteq \Gamma^{*,*}$. It is important that every position (i, j) of p can only be read for the *tl2br*-deterministic tiling system if all the positions above and left from (i, j) were already read i.e. all positions of the subpicture $q = p((1, 1), (i, j))$ of p without position (i, j). Formally, it is the following:

Definition 4.1. *A tiling system $\tau = (\Sigma, \Gamma, \Theta, \pi)$ is* ***tl2br-deterministic*** *if for any $\gamma_1, \gamma_2, \gamma_3 \in \Gamma \cup \{\#\}$ and $a \in \Sigma$ there exists at most one tile* $\begin{array}{|c|c|}\hline \gamma_1 & \gamma_2 \\ \hline \gamma_3 & \gamma_4 \\ \hline \end{array} \in \Theta$*, with $\pi(\gamma_4) = a$ or $\gamma_4 = \#$.*

Similarly the c2c-deterministic tiling system is defined for any corner-to-corner direction c2c.

$L \in REC$ is a **diagonal deterministic recognizable picture language** iff it can be recognized by a *c2c*-deterministic tiling system for some *c2c*. M. Anselmo, D. Giammarresi and M. Madonia have denoted the class of all diagonal deterministic recognizable picture languages by DREC in [2]. Later V. Lonati and M. Pradella called this family *Diag*-DREC in [3]. In the following we denote the **family of all diagonal deterministic recognizable picture languages** as *Diag*-DREC, too.

As an example for a tiling system that is *tl2br*-deterministic we can cite from [2] the language $L_{fr=fc}$. Let $L_{fr=fc} = \{p \in \Sigma^{*,*} \mid l_1(p) = l_2(p) \text{ and } p(1,i) = p(i,1) \text{ for } 1 \leq i \leq l_1(p)\}$ be the language of squares over a two-letter alphabet $\Sigma = \{a, b\}$, where the first row is equal to the first column. Please note that $L_{fr=fc} \in REC$. Let us consider an example of an *tl2br*-deterministic tiling system which recognizes $L_{fr=fc}$ (see [2]).

Example 4.2. *$\tau = (\Sigma, \Gamma, \Theta, \pi)$ is a tiling system, where*

- $\Gamma = \{x_y^z$ *with* $x, y \in \Sigma, z \in \{0, 1, 2\}\}$,
- $\Theta = \left\{ \begin{array}{|c|c|}\hline w_r^0 & x_r^0 \\ \hline y_s^0 & z_s^0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline w_r^0 & x_r^1 \\ \hline y_s^0 & z_s^0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline w_r^1 & x_s^2 \\ \hline y_s^0 & z_s^1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline w_r^2 & x_s^2 \\ \hline y_r^1 & z_s^2 \\ \hline \end{array}, \begin{array}{|c|c|}\hline w_r^2 & x_s^2 \\ \hline y_r^2 & z_s^2 \\ \hline \end{array}, \right.$

$\begin{array}{|c|c|}\hline \# & \# \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & z_z^1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & x_x^0 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline y_y^2 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline w_w^1 & \# \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline y_y^1 & z_z^2 \\ \hline \end{array},$

$\begin{array}{|c|c|}\hline \# & \# \\ \hline y_y^2 & z_z^2 \\ \hline \end{array}, \begin{array}{|c|c|}\hline w_w^0 & x_x^0 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline w_w^0 & x_x^1 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline w_w^2 & \# \\ \hline y_y^2 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline w_w^2 & \# \\ \hline y_y^1 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & x_x^1 \\ \hline \# & z_z^0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & x_x^0 \\ \hline \# & z_z^0 \\ \hline \end{array} \mid r, s, w, x, y, z \in \Sigma \left. \right\}$

- $\pi(x_y^z) = x$,
- *the upper index* 0 *describes a position below the diagonal,* 1 *a position on the diagonal and* 2 *a position above the diagonal,*
- *and the lower index of the upper right, lower left and lower right of each tile have at least one neighbour, which has the same lower index. Remark that the upper and the left neighbour of each lower right symbol of a tile, which has a* 1 *as the upper index, both have the same lower index as this element.*

For a better visualization of the language $L_{fr=fc}$ and the tiling system τ, an example of a picture $p \in L_{fr=fc}$ together with the corresponding local picture p' can be found below:

$$p = \begin{array}{|c|c|c|c|c|}\hline a & a & b & b & a \\ \hline a & b & b & a & a \\ \hline b & b & a & a & b \\ \hline b & b & a & a & a \\ \hline a & a & a & a & b \\ \hline \end{array}, \quad p' = \begin{array}{|c|c|c|c|c|}\hline a_a^1 & a_a^2 & b_b^2 & b_b^2 & a_a^2 \\ \hline a_a^0 & b_a^1 & b_b^2 & a_b^2 & a_a^2 \\ \hline b_b^0 & b_b^0 & a_b^1 & a_b^2 & b_a^2 \\ \hline b_b^0 & b_b^0 & a_b^0 & a_b^1 & a_a^2 \\ \hline a_a^0 & a_a^0 & a_a^0 & a_a^0 & b_a^1 \\ \hline \end{array}.$$

We can see that τ is $tl2rb$-deterministic and $L(\tau) = L_{fr=fc}$ and hence $L_{fr=fc} \in$ *Diag*-DREC.

Remark that this tiling system is not $br2tl$-deterministic, $bl2tr$-deterministic or $tr2bl$-deterministic. The two tiles $\begin{array}{|c|c|}\hline a_a^1 & a_a^2 \\ \hline a_a^0 & b_a^1 \\ \hline \end{array}$, $\begin{array}{|c|c|}\hline a_b^1 & a_a^2 \\ \hline a_a^0 & b_a^1 \\ \hline \end{array} \in \Theta$ with $\pi(a_a^1) = \pi(a_b^1) = a$ are the reason why τ is not $br2tl$-deterministic. The two tiles $\begin{array}{|c|c|}\hline a_a^2 & a_a^2 \\ \hline a_a^2 & a_a^2 \\ \hline \end{array}$, $\begin{array}{|c|c|}\hline a_a^2 & a_a^2 \\ \hline a_a^1 & a_a^2 \\ \hline \end{array} \in \Theta$ with $\pi(a_a^2) = \pi(a_a^1) = a$ are the reason why τ is not $tr2bl$-deterministic and the two tiles $\begin{array}{|c|c|}\hline a_a^0 & a_a^0 \\ \hline a_a^0 & a_a^0 \\ \hline \end{array}$, $\begin{array}{|c|c|}\hline a_a^0 & a_a^1 \\ \hline a_a^0 & a_a^0 \\ \hline \end{array} \in \Theta$ with $\pi(a_a^0) = \pi(a_a^1) = a$ are the reason why τ is not $bl2tr$-deterministic.

After all this information about diagonal deterministic tiling systems we will discuss some closure properties.

4.1 Closure Properties

In [2] M. Anselmo, D. Giammarresi and M. Madonia have shown that the family *Diag*-DREC is closed under complement and rotation. In [8] M. Anselmo and M. Madonia have shown that *Diag*-DREC is not closed under union and intersection. Here we give another proof of this statement.

Theorem 4.3. *Diag-DREC is not closed under union and intersection.*

Proof. In [2] M. Anselmo, D. Giammarresi and M. Madonia show that the picture language L_{frames} is not in *Diag*-DREC.

$$\begin{aligned} L_{frames} = \{p \in \Sigma^{n,n} \mid & n \geq 0, p(1,i) = p(i,1), p(2,i) = p(n-i+1, n-1), \\ & p(n-1,i) = p(n-i+1,2) \text{ and } p(n,i) = p(i,n), \\ & \text{where } 1 \leq i \leq n\} \end{aligned}$$

is the language of all square pictures, where the first row is equal to the first column (see the language $L_{fr=fc}$), the second row is equal to the reverse of the second-last column, the second-last row is equal to the reverse of the second column and the last row equals the last column.

For example the following picture p is in L_{frames}:

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

This language can be seen as an intersection of four languages. It is easy to verify that these four languages are in *Diag*-DREC. For example, the tiling system for all square pictures, where the first row is equal to the first column, can be seen in Example 4.2.

The four intersection languages are in *Diag*-DREC, but L_{frames} is not. This is the reason why *Diag*-DREC is not closed under intersection.

Because *Diag*-DREC is closed under complement and is not closed under intersection, *Diag*-DREC is not closed under union, either. □

5 Deterministically Recognizable Picture Languages

In this section we present the family of deterministically recognizable picture languages introduced by K. Reinhardt und B. Borchert in [1]. First, we describe the definition of a determinstic process followed by the definition of the Sudoku deterministic process. Afterwards, we present DREC and regard some of its closure properties.

5.1 Deterministic Process

In [9] K. Reinhard introduced another way to describe deterministically recognizable picture languages. For this he defined a **deterministic process** that starts with a given picture p over Σ and ends with the local picture p' over Γ using a DS. One step in this process is the replacement of one symbol $s \in \Sigma$ by its local symbol $g \in \Gamma$, which can only be performed if it is locally the only possible choice. Formally this is the following:

Definition 5.1. *Let $\tau = (\Sigma, \Gamma, \Delta, \pi)$ be a DS. Extend Δ to $\Delta' = \Delta \cup \left\{ \begin{array}{|c|}\hline s \\ \hline r \\ \hline \end{array}, \begin{array}{|c|}\hline s \\ \hline f \\ \hline \end{array}, \begin{array}{|c|}\hline g \\ \hline r \\ \hline \end{array}, \right.$*

$\left. \begin{array}{|c|c|}\hline q & o \\ \hline \end{array}, \begin{array}{|c|c|}\hline q & d \\ \hline \end{array}, \begin{array}{|c|c|}\hline e & o \\ \hline \end{array} \mid \begin{array}{|c|}\hline g \\ \hline f \\ \hline \end{array}, \begin{array}{|c|c|}\hline e & d \\ \hline \end{array} \in \Delta, s = \pi(g), r = \pi(f), q = \pi(e), o = \pi(d) \right\}$ *by also allowing the image symbols in the dominos.*

*For two intermediate configuations $p, p' \in (\Sigma \cup \Gamma)^{m,n}$, $n, m \geq 1$, we allow a **replacement step** $\hat{p} \underset{\tau}{\Rightarrow} \hat{p}'$ if for all positions (i, j) of p', we have either $p'(i, j) = p(i, j)$ or $\pi(p'(i, j)) = p(i, j)$ and*

$$\begin{array}{|c|}\hline p(i-1,j) \\ \hline p'(i,j) \\ \hline \end{array}, \begin{array}{|c|}\hline p'(i,j) \\ \hline p(i+1,j) \\ \hline \end{array}, \begin{array}{|c|c|}\hline p(i,j-1) & p'(i,j) \\ \hline \end{array}, \begin{array}{|c|c|}\hline p'(i,j) & p(i,j+1) \\ \hline \end{array} \in \Delta'$$

and the choice $p'(i, j)$ was 'forced'.

Forced means that, for each $g \in \Gamma$ with $\pi(g) = p(i,j)$ and $g \neq p'(i,j)$, the replacement of $p(i,j)$ in p by g would result in at least one of these tiles to not be in Θ'.

The accepted language for a given DS τ is $L_d(\tau) := \{p \in \Sigma^{*,*} \mid \hat{p} \overset{*}{\underset{\tau}{\Rightarrow}} \hat{p}' \in (\Gamma \cup \{\#\})^{*,*}\}$. A picture language $L \subseteq \Sigma^{*,*}$ is called **deterministically recognizable** if there is a DS τ with $L = L_d(\tau)$. Remark that $L_d(\tau) \subseteq L(\tau)$.

The language of pictures over $\{a, b\}$, where all occurring b's are connected, is one example picture language that is deterministically recognizable. For the proof of this statement see [9].

5.2 Sudoku Deterministic Process

In 2006 B. Borchert and K. Reinhardt introduced a stronger version of determinism which they called Sudoku-determinism [1]. They also considered a more general deterministic process than the one presented in Section 5.1.

The **Sudoku-determinism** is defined in a process that reduces a picture step by step. The difference with the determinism described in the section above is that instead of determining the pre-image symbol on a position in one shot, they keep a set of possible pre-image symbols for each position and reduce one set per step of the process by excluding impossible pre-images.

Accordingly, let $\tau = (\Sigma, \Gamma, \Delta, \pi)$ be a DS. For a picture $p \in \Sigma$, s_p is a local picture of the same size in which every position (i,j) is initialized by the set $\pi^{-1}(p(i,j)) \in 2^\Gamma$ of possible pre-image symbols. With this information, one step in a Sudoku-deterministic process is defined as follows:

Definition 5.2. *For $s, s' \in (2^\Gamma)^{*,*}$ with $l_1(s) = l_1(s')$ and $l_2(s) = l_2(s')$, we allow a **Sudoku-deterministic step** $\hat{s} \underset{sd(\tau)}{\Rightarrow} \hat{s}'$ if the following condition holds for all positions (i,j) of s':*

$$s'(i,j) = \Bigg\{ x \in s(i,j) \mid \exists \gamma_1, \gamma_2, \gamma_3, \gamma_4 \in \Gamma \cup \{\#\} \text{ such that}$$

$$\gamma_1 \in \hat{s}(i, j+1), \gamma_2 \in \hat{s}(i, j-1), \gamma_3 \in \hat{s}(i+1, j), \gamma_4 \in \hat{s}(i-1, j)$$

$$\text{and } \boxed{x}\boxed{\gamma_1}, \boxed{\gamma_2}\boxed{x}, \begin{array}{|c|}\hline x \\ \hline \gamma_3 \\ \hline \end{array}, \begin{array}{|c|}\hline \gamma_4 \\ \hline x \\ \hline \end{array} \in \Delta \Bigg\}.$$

One step in a deterministic process which is described in Section 5.1 can be formulated as a special case (see [1]).

Definition 5.3. *For $s, s', s'' \in (2^\Gamma)^{*,*}$ with $l_1(s) = l_1(s')$ and $l_2(s) = l_2(s')$ we allow a **deterministic step** $\hat{s} \underset{d(\tau)}{\Rightarrow} \hat{s}'$ if $\hat{s} \underset{sd(\tau)}{\Rightarrow} \hat{s}''$ and, for all positions (i,j) of s', we define $s'(i,j) := s''(i,j)$ if $|s''(i,j)| = 1$ or $\hat{s}'(i,j) = \hat{s}(i,j)$.*

Remark that by reducing all positions to singletons we got the forced choice of Definition 5.1. The replacement of each symbol by one of its local symbols dependend on the dominos of Δ' of Definition 5.1, is similar to starting with all possible local symbols of one symbol and minimize them to a one element dependent on the dominos of Δ of Definition 5.3.

In the following we use the notion from [1], where $\{p\}$ describes a picture which has the same size as p and has a singleton $\{p(i,j)\}$ instead of the letter $p(i,j)$ on every position.

Let $\tau = (\Sigma, \Gamma, \Delta, \pi)$ be a DS. Then $L_{sd}(\tau) := \{p \in \Sigma^{*,*} \mid \exists p' \in L(\tau) \text{ s.t. } \hat{s}_p \overset{*}{\underset{sd(\tau)}{\Rightarrow}} \{\hat{p}'\}\}$ is a **Sudoku-deterministically recognizable picture language**. That implies that the local picture $\hat{s}_p$ can be transformed with a finite number of steps of a Sudoku-deterministic process into the picture $\{\hat{p}'\}$. Remark that the language $L_d(\tau)$ is defined in the same way using $\overset{*}{\underset{d(\tau)}{\Rightarrow}}$ instead of $\overset{*}{\underset{sd(\tau)}{\Rightarrow}}$. The **family of all Sudoku-deterministically recognizable picture languages** is denoted by **SDREC**. The **class of all deterministically recognizable picture languages** $L_d(\tau)$ is denoted by DREC.

We now consider an example of a DS τ, some subpictures $t_p, t'_p, t''_p \in B_{2,2}(s_p)$, where $p \in \Sigma^{*,*}$, and how a Sudoku-deterministic and a deterministic process work with these subpictures depending on the given τ. For convenience, we only test two of the four possible neighbours of every position in the subpictures.

Example 5.4. *For a DS $\tau = (\Sigma, \Gamma, \Delta, \pi)$ with $\Gamma = \{a, b, c, d\}$ and*

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

$$\left. \begin{array}{|c|c|}\hline x & x \\ \hline \end{array}, \begin{array}{|c|c|}\hline x & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & x \\ \hline \end{array} \mid x \in \Gamma \right\}$$

holds.

$$t_p = \begin{array}{|c|c|}\hline a,b,c,d & a,b,c,d \\ \hline b,d & a,b,c,d \\ \hline \end{array} \overset{}{\underset{sd(\tau)}{\Rightarrow}} \begin{array}{|c|c|}\hline a,b,c,d & a,b,c,d \\ \hline b,d & b,d \\ \hline \end{array}$$

$$t'_p = \begin{array}{|c|c|}\hline a,b,c,d & c,d \\ \hline a,b,c,d & a,b,c,d \\ \hline \end{array} \overset{}{\underset{sd(\tau)}{\Rightarrow}} \begin{array}{|c|c|}\hline a,b,c,d & c,d \\ \hline a,b,c,d & c,d \\ \hline \end{array}$$

$$t''_p = \begin{array}{|c|c|}\hline a,b,c,d & c,d \\ \hline b,d & a,b,c,d \\ \hline \end{array} \underset{d(\tau)}{\Rightarrow} \begin{array}{|c|c|}\hline a,b,c,d & c,d \\ \hline b,d & d \\ \hline \end{array}$$

Furthermore, the language family dependencies will be talked about. The following class relationships are proved in [1]:

Theorem 5.5. *The family DREC is a subset of*

- *SDREC,*
- *REC and*
- *co-REC* $= \{L \mid \bar{L} \in REC\}$.

5.3 Closure Properties

In this chapter we discuss the closure properties of DREC. For the following proofs let $\tau_1 = (\Sigma_1, \Gamma_1, \Delta_1, \pi_1)$ and $\tau_2 = (\Sigma_2, \Gamma_2, \Delta_2, \pi_2)$ be two DS's where $\Gamma_1 \cap \Gamma_2 = \emptyset$ without any loss of generality.

Theorem 5.6. *DREC is closed under rotation.*

Proof. We have to show that for any $L \in$ DREC the rotated picture language L^R is in DREC. Let $L_d(\tau_1) = L$. We construct a DS $\tau^R = (\Sigma_1, \Gamma_1, \Delta^R, \pi_1)$ such that $L_d(\tau^R) = L^R$, where

$$\Delta^R = \left\{ \begin{array}{|c|}\hline a \\ \hline b \\ \hline \end{array} \mid \begin{array}{|c|c|}\hline a & b \\ \hline \end{array} \in \Delta_1, a, b \in \Gamma_1 \right\} \cup \left\{ \begin{array}{|c|c|}\hline a & b \\ \hline \end{array} \mid \begin{array}{|c|}\hline b \\ \hline a \\ \hline \end{array} \in \Delta_1, a, b \in \Gamma_1 \right\}.$$

Because of the construction of τ^R, we can see that τ^R is a DS, and, hence, it can be used for a deterministic process. Next, we have to show that $L_d(\tau^R) = L^R$.

- $L_d(\tau^R) \subseteq L^R$:
  Let $p \in L_d(\tau^R)$, i.e., there exists a deterministic process such that $s_p(i,j)$ will be minimized to a singleton set $\{x\}$ for $1 \leq i \leq l_1(p)$ and $1 \leq j \leq l_2(p)$. There is only one possible choice of four dominos in Δ^R such that $s_p(i,j)$ will be minimized to $\{x\}$. $\begin{array}{|c|c|}\hline a & x \\ \hline \end{array}$, $\begin{array}{|c|}\hline b \\ \hline x \\ \hline \end{array}$, $\begin{array}{|c|c|}\hline x & c \\ \hline \end{array}$ and $\begin{array}{|c|}\hline x \\ \hline d \\ \hline \end{array}$ are these four dominos, where $a \in \hat{s}_p(i, j-1)$, $b \in \hat{s}_p(i-1, j)$, $c \in \hat{s}_p(i, j+1)$ and $d \in \hat{s}_p(i+1, j)$. Because of the anticlockwise rotation of s_p it follows that $a \in \hat{s}_{p_1}(i_1+1, j_1)$, $b \in \hat{s}_{p_1}(i_1, j_1-1)$, $c \in \hat{s}_{p_1}(i_1-1, j_1)$, $d \in \hat{s}_{p_1}(i_1, j_1+1)$ and $x \in s_{p_1}(i_1, j_1)$, where s_{p_1} is the anticlockwise rotated local picture and $i_1 = l_2(p) - j + 1$ and $j_1 = i$. We can see that p is the clockwise rotated picture of the picture p_1. The picture p_1 can be accepted through the deterministic process using the DS τ_1 because of the construction of τ^R. It follows $p \in L^R$.

- $L^R \subseteq L_d(\tau^R)$:
  Let $p \in L^R$, i.e., the anticlockwise rotated picture p_1 of p is in $L_d(\tau_1)$. Any position of the local picture s_{p_1} will be minimized to a singleton set. $\boxed{a \mid x}$, $\begin{array}{|c|}\hline b \\ \hline x \\ \hline \end{array}$, $\boxed{x \mid c}$ and $\begin{array}{|c|}\hline x \\ \hline d \\ \hline \end{array}$ are the four dominos which minimize the set on position (i,j) of s_{p_1} to the singleton set $\{x\}$, where $a \in \hat{s}_{p_1}(i, j-1)$, $b \in \hat{s}_{p_1}(i-1, j)$, $c \in \hat{s}_{p_1}(i, j+1)$ and $d \in \hat{s}_{p_1}(i+1, j)$. Because of the rotation of s_{p_1}, it follows that $a \in \hat{s}_p(i_1+1, j_1)$, $b \in \hat{s}_p(i_1, j_1-1)$, $c \in \hat{s}_p(i_1-1, j_1)$, $d \in \hat{s}_p(i_1, j_1+1)$ and $x \in s_p(i_1, j_1)$, where s_p is the clockwise rotated local picture for a deterministic process, and $i_1 = j$ and $j_1 = l_1(p) - i + 1$. The clockwise rotated local picture s_p minimizes the singleton set in the same way as s_{p_1} dependend on a, b, c, d. The only difference is that the neighbours are on other positions. We can see τ^R fulfills the minimization because of the construction of Δ^R. It follows $p \in L_d(\tau^R)$.

□

Theorem 5.7. *DREC is closed under intersection.*

Proof. We have to show that for any $L_1, L_2 \in DREC$ the intersection language $L_1 \cap L_2$ is in DREC. Let $L_1 = L_d(\tau_1)$ and $L_2 = L_d(\tau_2)$. We construct a DS $\tau_\cap = (\Sigma_\cap, \Gamma_\cap, \Delta_\cap, \pi_\cap)$ such that $L_d(\tau_\cap) = L_1 \cap L_2$, where

- $\Sigma_\cap = \Sigma_1 \cap \Sigma_2$,
- $\Gamma_\cap = \{(a,b) \mid \pi_1(a) = \pi_2(b), a \in \Gamma_1, b \in \Gamma_2\}$,
- $\Delta_\cap = \left\{ \begin{array}{|c|}\hline \# \\ \hline (c,d) \\ \hline \end{array} \mid \begin{array}{|c|}\hline \# \\ \hline c \\ \hline \end{array} \in \Delta_1, \begin{array}{|c|}\hline \# \\ \hline d \\ \hline \end{array} \in \Delta_2, \right.$
  $\left. c \in \Gamma_1, d \in \Gamma_2, (c,d) \in \Gamma_\cap \right\}$
  $\cup \quad \left\{ \begin{array}{|c|}\hline (a,b) \\ \hline (c,d) \\ \hline \end{array} \mid \begin{array}{|c|}\hline a \\ \hline c \\ \hline \end{array} \in \Delta_1, \begin{array}{|c|}\hline b \\ \hline d \\ \hline \end{array} \in \Delta_2, \right.$
  $\left. a, c \in \Gamma_1, b, d \in \Gamma_2, (a,b), (c,d) \in \Gamma_\cap \right\}$

$$\cup \quad \left\{ \begin{array}{|c|}\hline (a,b) \\ \hline \# \\ \hline \end{array} \;\middle|\; \begin{array}{|c|}\hline a \\ \hline \# \\ \hline \end{array} \in \Delta_1, \begin{array}{|c|}\hline b \\ \hline \# \\ \hline \end{array} \in \Delta_2, \right.$$

$$\left. a \in \Gamma_1, b \in \Gamma_2, (a,b) \in \Gamma_\cap \right\}$$

$$\cup \quad \left\{ \begin{array}{|c|c|}\hline \# & (c,d) \\ \hline \end{array} \;\middle|\; \begin{array}{|c|c|}\hline \# & c \\ \hline \end{array} \in \Delta_1, \begin{array}{|c|c|}\hline \# & d \\ \hline \end{array} \in \Delta_2, \right.$$

$$\left. c \in \Gamma_1, d \in \Gamma_2, (c,d) \in \Gamma_\cap \right\}$$

$$\cup \quad \left\{ \begin{array}{|c|c|}\hline (a,b) & (c,d) \\ \hline \end{array} \;\middle|\; \begin{array}{|c|c|}\hline a & c \\ \hline \end{array} \in \Delta_1, \begin{array}{|c|c|}\hline b & d \\ \hline \end{array} \in \Delta_2, \right.$$

$$\left. a, c \in \Gamma_1, b, d \in \Gamma_2, (a,b), (c,d) \in \Gamma_\cap \right\}$$

$$\cup \quad \left\{ \begin{array}{|c|c|}\hline (a,b) & \# \\ \hline \end{array} \;\middle|\; \begin{array}{|c|c|}\hline a & \# \\ \hline \end{array} \in \Delta_1, \begin{array}{|c|c|}\hline b & \# \\ \hline \end{array} \in \Delta_2, \right.$$

$$\left. a \in \Gamma_1, b \in \Gamma_2, (a,b) \in \Gamma_\cap \right\} \text{ and}$$

- $\pi_\cap((a,b)) = \pi_1(a)$.

Because of the construction of $\tau_\cap$ we see that $\tau_\cap$ is a DS, and, hence, it can be used for a deterministic process. Next we have to show that $L_d(\tau_\cap) = L_1 \cap L_2$.

- $L_d(\tau_\cap) \subseteq L_1 \cap L_2$:
  Let $p \in L_d(\tau_\cap)$, i.e., any position in s_p is minimized by the deterministic process of $\tau_\cap$. Let (i,j) be one of these positions, and let (x,y) be the element in the singleton set on (i,j). Let (x_1,y_1), (x_2,y_2), (x_3,y_3) and (x_4,y_4) be four elements that are in the neighbour sets, and let $\begin{array}{|c|c|}\hline (x_1,y_1) & (x,y) \\ \hline \end{array}$, $\begin{array}{|c|c|}\hline (x,y) & (x_2,y_2) \\ \hline \end{array}$, $\begin{array}{|c|}\hline (x_3,y_3) \\ \hline (x,y) \\ \hline \end{array}$

and $\begin{array}{|c|}\hline (x,y) \\ \hline (x_4,y_4) \\ \hline \end{array}$ be the corresponding dominos that $\tau_\cap$ used to minimize the set on position (i,j). We now take a look at the following domino: $\begin{array}{|c|c|}\hline (x_1,y_1) & (x,y) \\ \hline \end{array}$. This domino is only in $\Delta_\cap$ if $\begin{array}{|c|c|}\hline x_1 & x \\ \hline \end{array} \in \Delta_1$, $\begin{array}{|c|c|}\hline y_1 & y \\ \hline \end{array} \in \Delta_2$, $\pi_1(x_1) = \pi_2(y_1)$ and $\pi_1(x) = \pi_2(y)$. This means that DS τ_1 and DS τ_2 can "accept" the symbol $\pi(x)$ on position (i,j) because $\pi(x_1)$ is the left neighbour in p. Considering all four dominos, it is easy to understand that this deterministic step is only possible if this deterministic step is possible with DS τ_1 and τ_2. Consequentially, all the positions in s_p will be minimized accordingly and, thus, p is in L_1 and in L_2. It follows that $p \in L_1 \cap L_2$.

- $L_1 \cap L_2 \subseteq L_d(\tau_\cap)$:
  Let $p \in L_1 \cap L_2$, that is, $p \in L_1$ and $p \in L_2$. Then, we have two local pictures of p. s_{p_1} is the local picture of p for DS τ_1 and s_{p_2} is the local picture of p for DS τ_2. Because $p \in L_1$ and $p \in L_2$, the two corresponding local languages will be reduced to a picture which has a singleton set on each position. Let (i,j) be one of the positions in p. In the deterministic process using DS τ_1, let x_1, x_2, x_3 and x_4 be some elements of the sets in the neighbourhood of position (i,j) and $\begin{array}{|c|c|}\hline x_1 & x \\ \hline \end{array}$, $\begin{array}{|c|c|}\hline x & x_2 \\ \hline \end{array}$, $\begin{array}{|c|}\hline x_3 \\ \hline x \\ \hline \end{array}$ and $\begin{array}{|c|}\hline x \\ \hline x_4 \\ \hline \end{array}$ the corresponding dominos which minimize the set on position (i,j) to $\{x\}$. In the deterministic process using DS τ_2, let y_1, y_2, y_3 and y_4 some elements of the sets in the neighbourhood of position (i,j) and $\begin{array}{|c|c|}\hline y_1 & y \\ \hline \end{array}$, $\begin{array}{|c|c|}\hline y & y_2 \\ \hline \end{array}$, $\begin{array}{|c|}\hline y_3 \\ \hline y \\ \hline \end{array}$ and $\begin{array}{|c|}\hline y \\ \hline y_4 \\ \hline \end{array}$ the corresponding dominos which minimize the set on position (i,j) to $\{y\}$. Because $\pi_1(x_1) = \pi_2(y_1)$, $\pi_1(x_2) = \pi_2(y_2)$, $\pi_1(x_3) = \pi_2(y_3)$, $\pi_1(x_4) = \pi_2(y_4)$ and $\pi_1(x) = \pi_2(y)$, the following dominos are in $\Delta_\cap$:

$$\begin{array}{|c|c|}\hline (x_1,y_1) & (x,y) \\ \hline \end{array}, \quad \begin{array}{|c|c|}\hline (x,y) & (x_2,y_2) \\ \hline \end{array}, \quad \begin{array}{|c|}\hline (x_3,y_3) \\ \hline (x,y) \\ \hline \end{array} \text{ and } \begin{array}{|c|}\hline (x,y) \\ \hline (x_4,y_4) \\ \hline \end{array}.$$

Because the deterministic step with DS τ_1 minimizes the set on position (i, j) to $\{x\}$ and the deterministic step with DS τ_2 minimizes the set on position (i, j) to $\{y\}$, the deterministic step with DS $\tau_\cap$ minimizes the set on position (i, j) to $\{(x, y)\}$ using the above dominos.This is similar for any position in the picture p. Thus, p can be accepted by a deterministic process of the DS $\tau_\cap$.

□

6 Deterministic On-line Tessellation Automata

In this chapter we present the definition of the deterministic on-line tesselation automata. This automata model is needed to show the inclusion of *Diag*-DREC in DREC. In the following $L(M)$ describes the language of all pictures p that are accepted by an automaton M.

The following automata model is a restricted type of a two-dimensional cellular automaton with the name two-dimensional online tessellation automaton abbreviated as **2OTA**. A cellular automaton is an array of "cells", where each cell contains a pixel of the input picture and a state. The automaton moves diagonally over the picture to compute the state of each cell. After the computation, the automaton accepts a picture if the cell at the bottom right corner contains an accepting state. The 2OTA was introduced by K. Inoue and A. Nakamura in 1977(see [5]).

In this automata model, each cell only receives its state at one step at a time. Before this time step the state of these cells are undefined. Exactly at that moment when the two neighbours on the top and the left of a cell c have a defined state, c can define its own state depending on its symbol and the states of these neighbours.

Definition 6.1. *A **deterministic two-dimensional online tessellation automaton**, abbreviated as 2DOTA, is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where*

- *Q is a finite set of states,*
- *Σ is a finite set of input symbols,*
- *$q_0 \in Q$ is the initial state,*
- *$F \subseteq Q$ is the set of accepting states and*
- *$\delta : Q \times Q \times \Sigma \to Q$ is the control function.*

The automaton M starts at time step $t = 0$ on a picture $p \in \Sigma^{*,*}$. At this time step, the initial state q_0 is associated with every position of the first row and the first column of $\hat{p}$. One time step later, the top left corner of p associates the state $\delta(q_0, q_0, p(1, 1))$. At time step $t = 2$, the position $p(1, 2)$ and $p(2, 1)$ define their states and so on. That means that M moves along the first main diagonal over the picture to define the state of each cell which is parallel to the second main diagonal dependent of the current position. The automaton M accepts a picture p if an accepting state is associated to position $(l_1(p), l_2(p))$ after the computation.

Now we show an example of a 2DOTA. First let $S = \{p \mid p \in \Sigma^{*,*}, l_1(p) = l_2(p)\}$ be the language of all quadratic pictures over $\Sigma = \{0\}$. The following definition of a 2DOTA M is one possible construction such that $L(M) = S$ holds. It is obtained from [6].

Example 6.2. $M = (\{q_0, q_1, q_2, q_a\}, \Sigma, \delta, q_0, \{q_a\})$
Remark that the following table of transitions does not contain symbols, because M has a unary alphabet:

δ	q_0	q_1	q_2	q_a
q_0	q_a	-	q_2	q_2
q_1	q_1	q_1	-	-
q_2	-	q_a	q_2	q_2
q_a	q_1	q_1	-	-

M associates the final state q_a to every position in the first main diagonal of an input picture p. For all positions in p above the first main diagonal M associates the state q_1 and for all positions below the state q_2. If, after the execution of M, the final state q_a is at the right bottom corner, the input picture is a square.

We now have a look at how M will operate with the following picture $p \in S$:

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

First, M appends the initial state to all cells of the first row and first column of $\hat{p}$. The following time steps should be self-explanatory:

$\overset{t=0}{\rightarrow}$

$\#q_0$	$\#q_0$	$\#q_0$	$\#q_0$	$\#q_0$
$\#q_0$	0	0	0	#
$\#q_0$	0	0	0	#
$\#q_0$	0	0	0	#
$\#q_0$	#	#	#	#

$\overset{t=1}{\rightarrow}$

$\#q_0$	$\#q_0$	$\#q_0$	$\#q_0$	$\#q_0$
$\#q_0$	$0q_a$	0	0	#
$\#q_0$	0	0	0	#
$\#q_0$	0	0	0	#
$\#q_0$	#	#	#	#

$\overset{t=2}{\rightarrow}$

$\#q_0$	$\#q_0$	$\#q_0$	$\#q_0$	$\#q_0$
$\#q_0$	$0q_a$	$0q_1$	0	$\#$
$\#q_0$	$0q_2$	0	0	$\#$
$\#q_0$	0	0	0	$\#$
$\#q_0$	$\#$	$\#$	$\#$	$\#$

$\overset{t=3}{\rightarrow}$

$\#q_0$	$\#q_0$	$\#q_0$	$\#q_0$	$\#q_0$
$\#q_0$	$0q_a$	$0q_1$	$0q_1$	$\#$
$\#q_0$	$0q_2$	$0q_a$	0	$\#$
$\#q_0$	$0q_2$	0	0	$\#$
$\#q_0$	$\#$	$\#$	$\#$	$\#$

$\overset{t=4}{\rightarrow}$

$\#q_0$	$\#q_0$	$\#q_0$	$\#q_0$	$\#q_0$
$\#q_0$	$0q_a$	$0q_1$	$0q_1$	$\#$
$\#q_0$	$0q_2$	$0q_a$	$0q_1$	$\#$
$\#q_0$	$0q_2$	$0q_2$	0	$\#$
$\#q_0$	$\#$	$\#$	$\#$	$\#$

$\overset{t=5}{\rightarrow}$

$\#q_0$	$\#q_0$	$\#q_0$	$\#q_0$	$\#q_0$
$\#q_0$	$0q_a$	$0q_1$	$0q_1$	$\#$
$\#q_0$	$0q_2$	$0q_a$	$0q_1$	$\#$
$\#q_0$	$0q_2$	$0q_2$	$0q_a$	$\#$
$\#q_0$	$\#$	$\#$	$\#$	$\#$

.

As we can see, M has finished its run over p after five time steps. Note that every computation of a 2DOTA consists of $l_1(p) + l_2(p) - 1$ steps.

One important theorem for the proof of Theorem 7.3, which states that *Diag*-DREC is a subset of DREC, is the following theorem. It was proven by M. Anselmo, D. Giammarresi and M. Madonia in [2].

Theorem 6.3. *The class Diag-DREC is equal to the closure by rotation of* $\mathscr{L}$*(2DOTA).*

More information about this type of model can be found in [6].

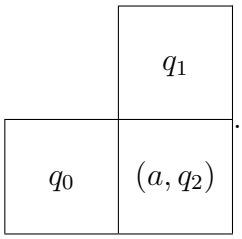
7 Comparing DREC and *Diag*-DREC

In this section we show that *Diag*-DREC is a proper subset of DREC. Therefore, we first show the inclusion of $\mathscr{L}$(2DOTA) in DREC.

Lemma 7.1. $\mathscr{L}$*(2DOTA)* $\subseteq$ *DREC.*

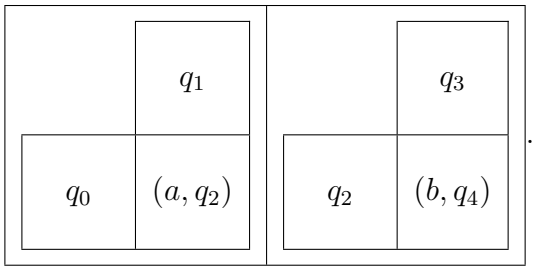
Proof. Let $L \in \mathscr{L}$(2DOTA) and $M = (Q, \Sigma, \delta, q_0, F)$ be a 2DOTA accepting L. We want to construct a DS $\tau = (\Sigma, \Gamma, \Delta, \pi)$ such that $L_d(\tau) = L$. First, we take $\Gamma = Q \times Q \times (\Sigma \times Q)$ as the set of 3-tuples which represent the set of all possible transitions

from $Q \times Q \times \Sigma$ to Q. Imagine one element of Γ as an L mirrored on the vertical axis in the following abbreviated with ◱. The first element of the tuple is on the left, the second element is at the top and the 2-tuple is in the center. For example, the 3-tuple $(q_0, q_1, (a, q_2)) \in \Gamma$ is depicted as follows:



Next, we define the mapping π as $\pi(q_1, q_2, a, q_3) = a$ for all $q_1, q_2, q_3 \in Q$ and $a \in \Sigma$. Finally, let $\Delta = \Delta' \cup \Delta_{\#_{\text{lt}}} \cup \Delta_{\#_{\text{rb}}}$ be the set of dominos which consists of three different sets of dominos. $\Delta' \subseteq \Gamma^{2,1} \cup \Gamma^{1,2}$ is the set of dominos which fulfills the two conditions (I) and (II) presented subsequently. $\Delta_{\#_{lt}}, \Delta_{\#_{rb}} \subseteq (\Gamma \cup \{\#\})^{1,2} \cup (\Gamma \cup \{\#\})^{2,1}$ are the sets of dominos which include exactly one #. How $\Delta_{\#_{lt}}$ arises is described in condition (III), and how $\Delta_{\#_{rb}}$ arises is described in condition (IV).

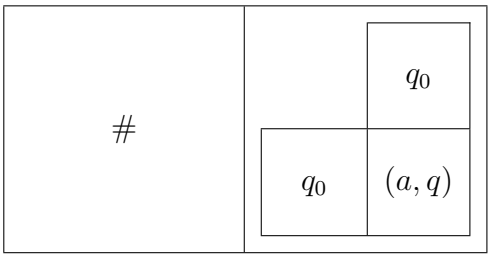
(I) The two ◱'s of each domino must be consistent. This means that, for each horizontal domino, the state in the center of the left ◱ must be the same as the state on the left of the right ◱. For every vertical domino the state in the center of the upper ◱ must be the same as the state on the top of the lower ◱. For example the following horizontal domino fulfills this condition:



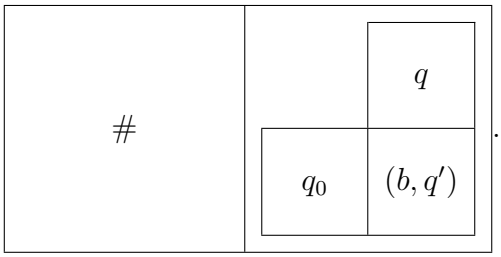
(II) Each ◱ on a domino corresponds to a transition of δ, i.e. if the tuple $(q_1, q_2, (a, q_3))$ corresponds to one ◱ of the dominos, then $\delta(q_1, q_2, a) = q_3$ for all $q_1, q_2, q_3 \in Q$ and $a \in \Sigma$.

(III) Any horizontal domino has on its left side the border symbol # and on its right side a ◱. Any ◱ has on its left side the initial state q_0. Furthermore, the top and the center of each ◱ result from an inductive procedure.
We start with the first set of horizontal dominos. For any $a \in \Sigma$ there exists exactly one domino such as the ◱ has on its top the initial state q_0 and on its center the tuple $(a, \delta(q_0, q_0, a))$. Remark that any such ◱ has on its left side the initial state q_0. For any domino d in this set of horizontal dominos, we add a horizontal domino for any $a \in \Sigma$ such that the ◱ has on its top the state q, which is listed in the center of the ◱ of d, and on its center the tuple $(a, \delta(q_0, q, a))$. This

procedure will be repeated unitl no new horizontal domino will be added to the set.
Note that Σ and Q are finite sets which implies that the inductive procedure is finite.
The upper side of any vertical domino includes # and the lower side a ⊡, which has on its top the state q_0. The left and the center field of any ⊡ of these vertical dominos will be filled in a similar inductive procedure like in the set of horizontal dominos.
For example, the following horizontal domino is one of the first dominos of the set of horizontal dominos described above. Without any loss of generality let $\delta(q_0, q_0, a) = q$.



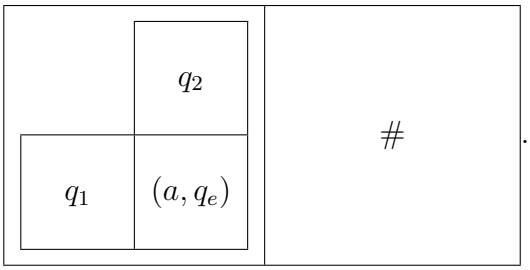
The following horizontal domino is one of the dominos that follows from the above horizontal domino, if $\delta(q_0, q, b) = q'$:



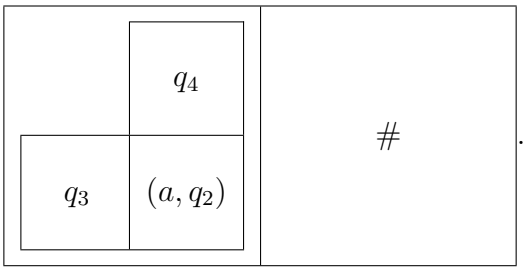
.

(IV) Any horizontal domino of this condition has on its right side the border symbol # and on its left side a ⊡. These ⊡'s arise from an inductive process.
We start with the first set of dominos. For any $(q_1, q_2, a) \in \delta^{-1}(q_e)$ and any $q_e \in F$ we add a horizontal domino with a ⊡, which has on its left the state q_1, on its top the state q_2 and on its center the tupel (a, q_e).
After this, we add a horizontal domino to the set of horizontal dominos with a ⊡ on its left, which has the state q_1 on its left, the state q_2 on its top and on its center the tuple (a, q), for any state q, which is on the top of a ⊡ of the dominos in the current set of horizontal dominos and any $(q_1, q_2, a) \in \delta^{-1}(q)$. This procedure will be repeated until no new horizontal domino will be added to the set.
The lower side of any vertical domino includes # and the upper side a ⊡. The ⊡ of these vertical dominos will be generated in a similar inductive procedure as the set of horizontal dominos. The difference is that we add a vertical domino to the current set of vertical dominos for each state that is on the left of a ⊡ of the

dominos in the current set.
For example, the following horizontal domino is one of the first dominos iff $\delta(q_1, q_2, a) = q_e$:



The following horizontal domino is one of the dominos that follows from the domino above if $\delta(q_3, q_4, a) = q_2$:



Starting on a given picture p, the only possible choice of the deterministic process using the domino system τ is the pixel on the upper left corner of s_p like in the workflow of M over p. Without any loss of generality let $p(1,1) = a$. Hence, by condition (III) the set on pixel $s_p(1,1)$ loses all $\boxplus$'s without the state q_0 at its top and its left and the tuple (a, q) on its center for all $q \in \delta(q_0, q_0, a)$. Because M is deterministic, there exists only one $q \in \delta(q_0, q_0, a)$. This is the reason why $s_p(1,1)$ can be minimized to a singleton set and the deterministic process can make a deterministic step. Whenever the upper pixel and the left pixel of a pixel x of s_p is minimized to a singleton set through the deterministic process, the deterministic process can uniquely determine x using conditions (I) and (II). In this way the deterministic process moves over the picture like a 2DOTA. p is only in $L_d(\tau)$ if all pixels along the right and lower border of local picture s_p can be covered by dominos described by condition (IV). Otherwise, the deterministic process cannot make a step because all elements of the set of a pixel at the border would disappear.

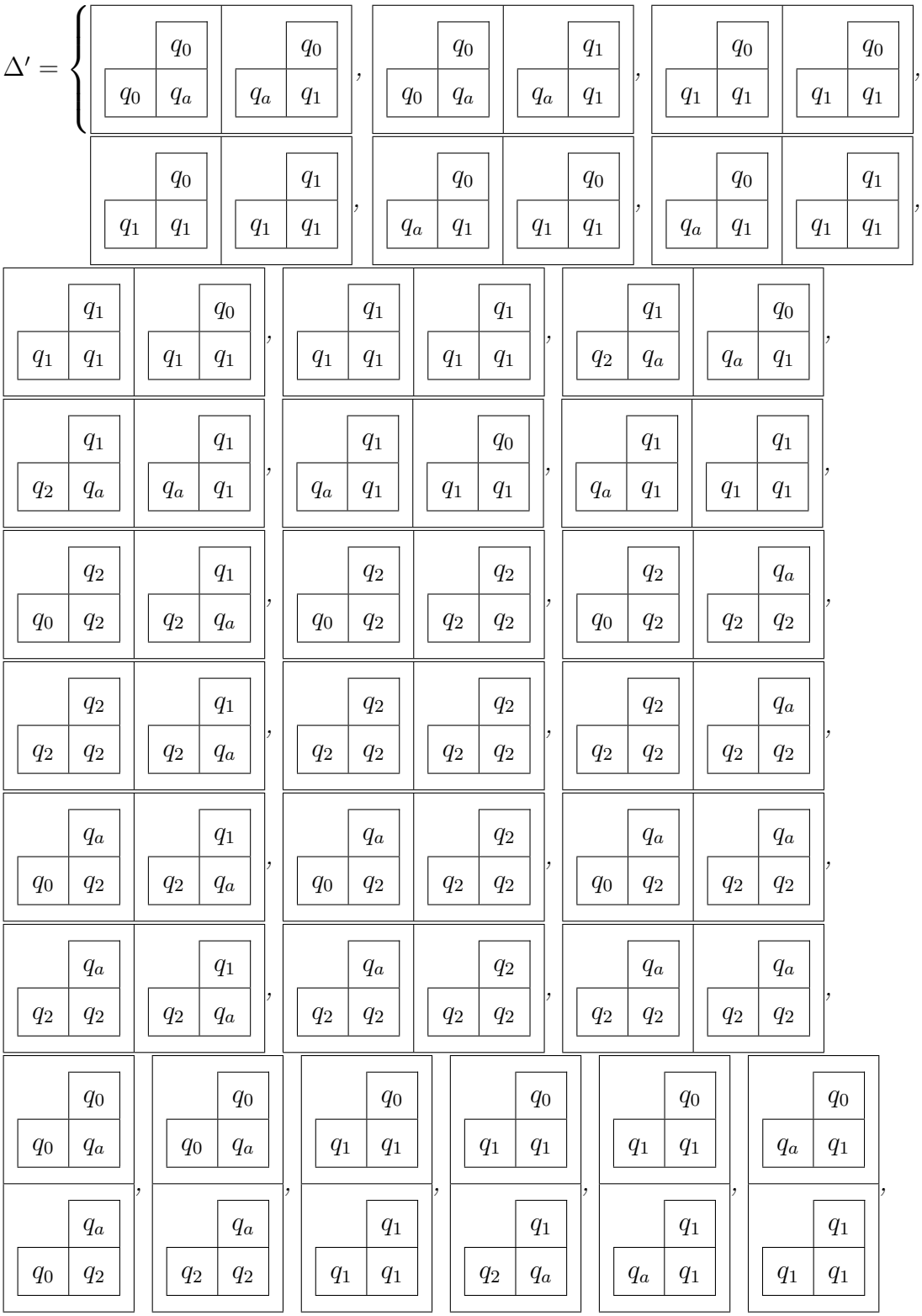
Because we have shown that $L_d(\tau) = L$, the theorem follows. □

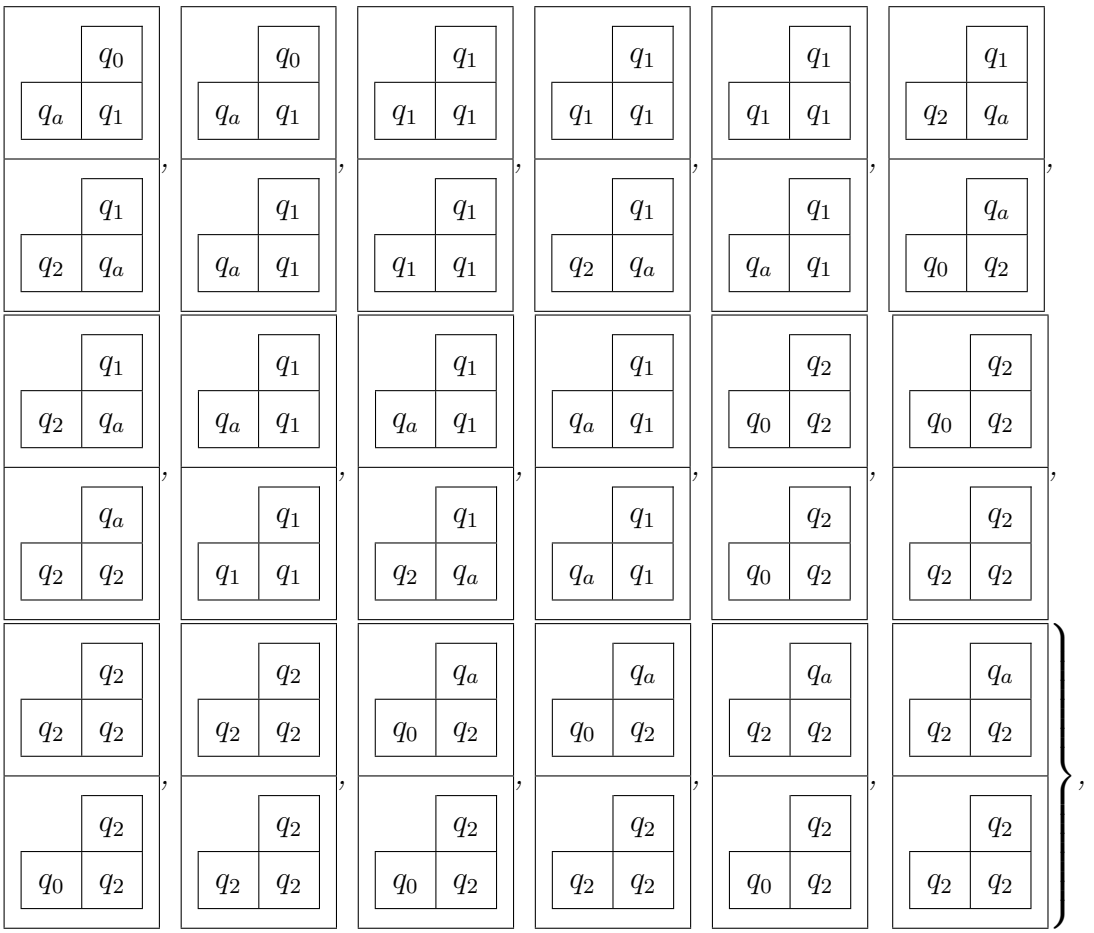
In Example 6.2 we have seen that the language S is an element of $\mathscr{L}$(2DOTA). We now want to use the construction described above to create a DS τ such that $L(\tau) = S$.

Example 7.2. *Remark that instead of the tuple on the center of each $\boxplus$ we write only the state of this tuple, because M of Example 6.2 has an unary alphabet.*
$\tau = (\{0\}, \Gamma, \Delta, \pi)$, *where*

- $\Gamma = \left\{ \begin{array}{|c|c|}\hline & b \\ \hline a & c \\ \hline \end{array} \mid a, b, c \in Q \right\}$,

- $\Delta = \Delta' \cup \Delta_{\#_{lt}} \cup \Delta_{\#_{rb}}$ *with*

$\Delta' =$ 



$$\Delta_{\#_{lt}} = \left\{ \begin{array}{|c|}\hline \# \\ \hline t \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & l \\ \hline \end{array} \mid \right.$$

$$t \in \left\{ \begin{array}{|cc|}\hline & q_0 \\ \hline q_0 & q_a \\ \hline \end{array}, \begin{array}{|cc|}\hline & q_0 \\ \hline q_a & q_1 \\ \hline \end{array}, \begin{array}{|cc|}\hline & q_0 \\ \hline q_1 & q_1 \\ \hline \end{array} \right\},$$

$$l \in \left\{ \left\{ \begin{array}{|cc|}\hline & q_0 \\ \hline q_0 & q_a \\ \hline \end{array}, \begin{array}{|cc|}\hline & q_a \\ \hline q_0 & q_2 \\ \hline \end{array}, \begin{array}{|cc|}\hline & q_2 \\ \hline q_0 & q_2 \\ \hline \end{array} \right\} \right\} \textit{ and}$$

$$\Delta_{\#_{rb}} = \left\{ \begin{array}{|c|}\hline b \\ \hline \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline r & \# \\ \hline \end{array} \mid \right.$$

$$b \in \left\{ \begin{array}{|cc|}\hline & q_0 \\ \hline q_0 & q_a \\ \hline \end{array}, \begin{array}{|cc|}\hline & q_a \\ \hline q_0 & q_2 \\ \hline \end{array}, \begin{array}{|cc|}\hline & q_1 \\ \hline q_2 & q_a \\ \hline \end{array}, \begin{array}{|cc|}\hline & q_2 \\ \hline q_0 & q_2 \\ \hline \end{array}, \begin{array}{|cc|}\hline & q_a \\ \hline q_2 & q_2 \\ \hline \end{array}, \begin{array}{|cc|}\hline & q_2 \\ \hline q_2 & q_2 \\ \hline \end{array} \right\},$$

$$r \in \left\{ \left\{ \begin{array}{|cc|}\hline & q_0 \\ \hline q_0 & q_a \\ \hline \end{array}, \begin{array}{|cc|}\hline & q_0 \\ \hline q_a & q_1 \\ \hline \end{array}, \begin{array}{|cc|}\hline & q_1 \\ \hline q_2 & q_a \\ \hline \end{array}, \begin{array}{|cc|}\hline & q_0 \\ \hline q_1 & q_1 \\ \hline \end{array}, \begin{array}{|cc|}\hline & q_1 \\ \hline q_a & q_1 \\ \hline \end{array}, \begin{array}{|cc|}\hline & q_1 \\ \hline q_1 & q_1 \\ \hline \end{array} \right\} \right\}$$

and

- $\pi(x) = 0$ *for all* $x \in \Gamma$.

We now have a look at how τ will operate with the following picture $p \in S$ and $\hat{s}_p$:

$$p = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}, \quad \hat{s}_p = \begin{array}{|c|c|c|c|c|}\hline \# & \# & \# & \# & \# \\ \hline \# & \Gamma & \Gamma & \Gamma & \# \\ \hline \# & \Gamma & \Gamma & \Gamma & \# \\ \hline \# & \Gamma & \Gamma & \Gamma & \# \\ \hline \# & \# & \# & \# & \# \\ \hline \end{array}.$$

The only possible choice for the deterministic process to start, is at position $s_p(1,1)$:

$\hat{s}_p \underset{d(\tau)}{\Rightarrow}$

#	#	#	#	#
#	q_0 / q_0 q_a	Γ	Γ	#
#	Γ	Γ	Γ	#
#	Γ	Γ	Γ	#
#	#	#	#	#

.

The positions $s_p(1,2)$ and $s_p(2,1)$ are the next positions in the process. We can see that the determinstic process moves over the picture to minimize the pixels to singleton sets along the first main diagonal. Dependent on the current position, it minimizes all cells, which are parallel to the second main diagonal, like the automaton M of Example 6.2. The following steps should be self-explanatory:

#	#	#	#	#
#	q_0 / q_0 q_a	q_0 / q_a q_1	Γ	#
#	q_a / q_0 q_2	Γ	Γ	#
#	Γ	Γ	Γ	#
#	#	#	#	#

$\Rightarrow_{d(\tau)}$

#	#	#	#	#
#	q_0 / q_0 q_a	q_0 / q_a q_1	q_0 / q_1 q_1	#
#	q_a / q_0 q_2	q_1 / q_2 q_a	Γ	#
#	q_2 / q_0 q_2	Γ	Γ	#
#	#	#	#	#

$\Rightarrow_{d(\tau)}$

$\underset{d(\tau)}{\Rightarrow}$

#	#	#	#	#
#	$\begin{array}{cc} & q_0 \\ q_0 & q_a \end{array}$	$\begin{array}{cc} & q_0 \\ q_a & q_1 \end{array}$	$\begin{array}{cc} & q_0 \\ q_1 & q_1 \end{array}$	#
#	$\begin{array}{cc} & q_a \\ q_0 & q_2 \end{array}$	$\begin{array}{cc} & q_1 \\ q_2 & q_a \end{array}$	$\begin{array}{cc} & q_1 \\ q_a & q_1 \end{array}$	#
#	$\begin{array}{cc} & q_2 \\ q_0 & q_2 \end{array}$	$\begin{array}{cc} & q_a \\ q_2 & q_2 \end{array}$	Γ	#
#	#	#	#	#

$\underset{d(\tau)}{\Rightarrow}$

#	#	#	#	#
#	$\begin{array}{cc} & q_0 \\ q_0 & q_a \end{array}$	$\begin{array}{cc} & q_0 \\ q_a & q_1 \end{array}$	$\begin{array}{cc} & q_0 \\ q_1 & q_1 \end{array}$	#
#	$\begin{array}{cc} & q_a \\ q_0 & q_2 \end{array}$	$\begin{array}{cc} & q_1 \\ q_2 & q_a \end{array}$	$\begin{array}{cc} & q_1 \\ q_a & q_1 \end{array}$	#
#	$\begin{array}{cc} & q_2 \\ q_0 & q_2 \end{array}$	$\begin{array}{cc} & q_a \\ q_2 & q_2 \end{array}$	$\begin{array}{cc} & q_1 \\ q_2 & q_a \end{array}$	#
#	#	#	#	#

.

As any position of s_p includes a singleton set the picture p is in $L_d(\tau)$.

After we showed that $\mathscr{L}(\text{2DOTA}) \subseteq \text{DREC}$, the following lemma is easy to prove.

Lemma 7.3. *Diag-DREC* $\subseteq$ *DREC*

Proof. In Theorem 6.3 we have seen that *Diag*-DREC is equal to the closure by rotation of $\mathscr{L}$(2DOTA). In Theorem 5.6 we have seen that DREC is closed under rotation and in Lemma 7.1 we have seen that $\mathscr{L}$(2DOTA) $\subseteq$ DREC. By combining these three theorems we obtain that *Diag*-DREC is a subset of DREC. □

The last theorem of this paper shows that *Diag*-DREC is a proper subset of DREC.

Theorem 7.4. *Diag-DREC* $\subset$ *DREC.*

Proof. In Theorem 4.3 we have seen that *Diag*-DREC is not closed under intersection, i.e., there exist two languages $L_1, L_2 \in$ *Diag*-DREC such that $L_1 \cap L_2 \notin$ *Diag*-DREC. Because DREC is closed under intersection (see Theorem 5.7) and *Diag*-DREC $\subseteq$ DREC, the intersection language of L_1 and L_2 is in DREC. That means that there exists a language L such that $L \notin$ *Diag*-DREC and $L \in$ DREC. Therefore, the inclusion is proper. □

8 Conclusion

In this thesis we have proven that the class *Diag*-DREC is a proper subset of DREC. Furthermore, we have proven that the family *Diag*-DREC is not closed under intersection and union, and that DREC is closed under rotation and intersection. It can be shown that DREC is also closed under vertical and horizontal concatenation, vertical and horizontal mirroring and transposing. These properties were ignored in this thesis because they are irrelevant for the comparison between *Diag*-DREC and DREC.

It is an open problem whether DREC is closed under complement, union and projection. Remark that if it could be shown that DREC is not closed under union or projection, DREC would be a proper subset of REC.

References

[1] Bernd Borchert, Klaus Reinhardt, et al. *Deterministically and sudoku-deterministically recognizable picture languages.* Willhelm-Schickhard-Institut-2006-09. Universitätsbibliothek Tübingen, Tübingen, 2006.

[2] Marcella Anselmo, Dora Giammarresi, and Maria Madonia. From determinism to non-determinism in recognizable two-dimensional languages. In Tero Harju, Juhani Karhumki, and Arto Lepist, editors, *Developments in Language Theory*, volume 4588 of *Lecture Notes in Computer Science*, pages 36–47. Springer Berlin Heidelberg, 2007.

[3] Violetta Lonati and Matteo Pradella. Snake-deterministic tiling systems. In *Proceedings of the 34th International Symposium on Mathematical Foundations of Computer Science 2009*, Mathematical Foundations of Computer Science '09, pages 549–560, Berlin, Heidelberg, 2009. Springer-Verlag.

[4] Dora Giammarresi and Antonio Restivo. Recognizable picture languages. *International Journal of Pattern Recognition and Artificial Intelligence*, 6(02n03):241–256, 1992.

[5] Katsushi Inoue and Akira Nakamura. Some properties of two-dimensional on-line tessellation acceptors. *Information Sciences*, 13(2):95 – 121, 1977.

[6] Dora Giammarresi and Antonio Restivo. Two-dimensional languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, pages 215–267. Springer Berlin Heidelberg, 1997.

[7] Alessandra Cherubini and Matteo Pradella. Picture languages: From wang tiles to 2d grammars. In Symeon Bozapalidis and George Rahonis, editors, *Algebraic Informatics*, volume 5725 of *Lecture Notes in Computer Science*, pages 13–46. Springer Berlin Heidelberg, 2009.

[8] Marcella Anselmo and Maria Madonia. Deterministic and unambiguous two-dimensional languages over one-letter alphabet. *Theoretical Computer Science*, 410(16):1477–1485, 2009.

[9] Klaus Reinhardt. On some recognizable picture-languages. In Luboš Brim, Jozef Gruska, and Jiří Zlatuška, editors, *Mathematical Foundations of Computer Science 1998*, volume 1450 of *Lecture Notes in Computer Science*, pages 760–770. Springer Berlin Heidelberg, 1998.