Two-Dimensional Sgraffito Automata*

Daniel Průša[1] and František Mráz[2]

[1] Czech Technical University, Faculty of Electrical Engineering
Karlovo náměstí 13, 121 35 Prague 2, Czech Republic
prusapa1@cmp.felk.cvut.cz
[2] Charles University, Faculty of Mathematics and Physics
Malostranské nám. 25, 118 25 Prague 1, Czech Republic
frantisek.mraz@mff.cuni.cz

Abstract. We present a new model of a two-dimensional computing device called sgraffito automaton and demonstrate its significance. In general, the model is simple, allows a clear design of important computations and defines families exhibiting good properties. It does not exceed the power of finite-state automata when working over one-dimensional inputs. On the other hand, it induces a family of picture languages that strictly includes REC and the deterministic variant recognizes languages in DREC as well as those accepted by four-way automata.

Keywords: two-dimensional languages, sgraffito automaton, bounded Turing machine, REC.

1 Introduction

The theory of two-dimensional languages generalizes concepts and techniques from the theory of formal languages. The basic element, which is a string, is extended to a two-dimensional array, usually called a picture. Various classes of picture languages can be formed, especially by generalizing one-dimensional computational or generative models, which possibly leads to some two-dimensional variant of the Chomsky hierarchy. Naturally we can ask, whether the induced families inherit properties of their one-dimensional counterparts. The answer is typically negative. A more complex topology of pictures causes that families of picture languages are of a different founding.

A four-way finite automaton (4FA) [2] is a good example. It is a finite-state device composed of a control unit equipped with a head moving over an input picture in four directions: left, right, up and down. Even if the automaton is a simple extension of the two-way finite automaton, the formed family of languages shows properties different from those of regular languages [4].

In 1991, Giammaresi and Restivo proposed the family of recognizable languages (REC) [3]. The languages in REC are defined using tiling systems. They

* The authors were supported by the Grant Agency of the Czech Republic: the first author under the project P103/10/0783 and the second author under the projects P103/10/0783 and P202/10/1333.

H.-C. Yen and O.H. Ibarra (Eds.): DLT 2012, LNCS 7410, pp. 251–262, 2012.

also coincide with the languages recognizable by the two-dimensional on-line tessellation automata [7] or definable using existential monadic second order logic. The family is well established. It has many remarkable properties and the defined recognizability is a very robust notion. It is even presented as the ground-level class among the families of two-dimensional languages.

The non-determinism exhibited by REC makes it quite powerful. Even some NP-complete problems belong to REC [10]. It somehow contradicts the vision of a ground level class, taking into account the simplicity of resources sufficient to recognize (one-dimensional) regular languages. This fact has inspired the further proposal of DREC [1] – the family of deterministically recognizable languages.

We introduce a new two-dimensional computing device called *sgraffito automaton* (2SA).

> Sgraffito (Italian: "scratched"), in the visual arts, a technique used in painting, pottery, and glass, which consists of putting down a preliminary surface, covering it with another, and then scratching the superficial layer in such a way that the pattern or shape that emerges is of the lower colour. (Encyclopædia Britannica Online. Retrieved 20 March, 2012, from http://www.britannica.com/EBchecked/topic/537397/sgraffito)

The automaton has a finite state control and works on a picture consisting of symbols with different weights (as if they were put on its background in order from the lightest to the heaviest). 2SA can move its head over the picture in four directions. It must rewrite scanned symbol in each step and the symbol can be rewritten by a lighter symbol only (this corresponds to scratching some of the top layers). The automaton accepts by entering an accepting state.

The power of 2SAs collapses to finite-state automata when working over one-row pictures, while the induced two-dimensional family strictly includes REC and exhibits the same closure properties. A significant advantage of the model is its simplicity. The design of many important computations is simple and clear. An interesting family is settled by the deterministic variant of the automaton. It covers DREC as well as $\mathcal{L}$(4FA). Thus deterministic sgraffito automata are a new stronger deterministic alternative to DREC. This complements the result given by Jirička and Král who showed how to simulate 4FAs using deterministic forgetting automata [8].

Section 2 recalls basic notions and properties of picture languages. Sgraffito automata are introduced in Section 3 and we show there that one-dimensional sgraffito automata recognize exactly the class of regular languages. Sections 4 and 5 show several closure properties for languages accepted by nondeterministic and deterministic 2SAs. Concluding remarks are presented in Section 6.

2 Preliminaries

A *picture* P over a finite alphabet Σ is a two-dimensional matrix of elements from Σ. We denote the number of rows and columns of P by $\mathrm{rows}(P)$ and $\mathrm{cols}(P)$, respectively. The pair $(\mathrm{rows}(P), \mathrm{cols}(P))$ is called the *size* of P.

The *empty picture* Λ is defined as the only picture of size $(0,0)$. The set of all pictures over Σ is denoted by $\Sigma^{*,*}$, while $\Sigma^{m,n}$ denotes the subset of pictures of size (m,n). A *picture language* over Σ is a subset of $\Sigma^{*,*}$. Assuming $1 \leq i \leq \mathrm{rows}(P)$ and $1 \leq j \leq \mathrm{cols}(P)$, $P(i,j)$ (or shortly $P_{i,j}$) identifies the symbol located in the i-th row and the j-th column in P.

Two (partial) binary operations are used to concatenate pictures. Let P and Q be pictures over Σ of sizes (k,l) and (m,n), respectively. The *column concatenation* $P \oplus Q$ is defined iff $k = m$, the *row concatenation* $P \ominus Q$ is defined iff $l = n$. The corresponding products are depicted below:

$$P \oplus Q = \begin{array}{cccc} P_{1,1} & \dots & P_{1,l} & Q_{1,1} & \dots & Q_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ P_{k,1} & \dots & P_{k,l} & Q_{m,1} & \dots & Q_{m,n} \end{array} \qquad P \ominus Q = \begin{array}{ccc} P_{1,1} & \dots & P_{1,l} \\ \vdots & \ddots & \vdots \\ P_{k,1} & \dots & P_{k,l} \\ Q_{1,1} & \dots & Q_{1,n} \\ \vdots & \ddots & \vdots \\ Q_{m,1} & \dots & Q_{m,n} \end{array}$$

We also define $\Lambda \ominus P = P \ominus \Lambda = \Lambda \oplus P = P \oplus \Lambda = P$ for any picture P.

In addition, we introduce the *clockwise rotation* P^{R}, *vertical mirroring* P^{VM} and *horizontal mirroring* P^{HM}.

$$P^{\mathrm{R}} = \begin{array}{ccc} P_{m,1} & \dots & P_{1,1} \\ \vdots & \ddots & \vdots \\ P_{m,n} & \dots & P_{1,n} \end{array} \qquad P^{\mathrm{VM}} = \begin{array}{ccc} P_{1,n} & \dots & P_{1,1} \\ \vdots & \ddots & \vdots \\ P_{m,n} & \dots & P_{m,1} \end{array} \qquad P^{\mathrm{HM}} = \begin{array}{ccc} P_{m,1} & \dots & P_{m,n} \\ \vdots & \ddots & \vdots \\ P_{1,1} & \dots & P_{1,n} \end{array}$$

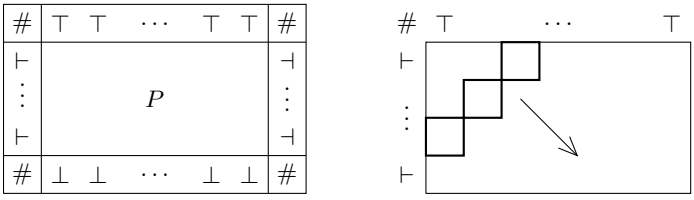
Let $\pi : \Sigma \to \Gamma$ be a mapping between two alphabets. The *projection* by π of $P \in \Sigma^{m,n}$ is $P' \in \Gamma^{m,n}$ such that $P'(i,j) = \pi\left(P(i,j)\right)$ for each admissible i, j. Note that each introduced operation can be naturally extended to languages.

Let $\mathcal{S} = \{\vdash, \dashv, \top, \bot, \#\}$ be a set of special markers (*sentinels*). In the text we always implicitly assume that $\Sigma \cap \mathcal{S} = \emptyset$ for any alphabet Σ. For $P \in \Sigma^{m,n}$, we define a *boundary picture* $\widehat{P}$ over $\Sigma \cup \mathcal{S}$ of size $(m+2, n+2)$. Its symbols are given by Figure 1(a).

Usually, only # is used to mark the border. Our version simplifies the definition of bounded computations, keeping the recognition abilities unchanged.

The two-dimensional on-line tessellation automaton (2OTA), depicted in Figure 1(b), is a restricted type of a cellular automaton. For an input $P \in \Sigma^{*,*}$, a computation is performed in $\mathrm{rows}(P) + \mathrm{cols}(P) - 1$ parallel steps. During the k-th step, each cell at coordinates (i,j), where $i + j - 1 = k$, performs a state-transition depending on $P(i,j)$ and the final states of the left and top neighbor cells. If the neighbor lies at the border of P, it is a fictive cell whose final state is defined as the corresponding symbol in $\widehat{P}$. The result of the computation is determined by the final state of the bottom-right cell.

Tiling systems (TS) [4] specify *tiling recognizable* languages. Since it holds $\mathcal{L}(\mathsf{TS}) = \mathcal{L}(\mathsf{2OTA})$, the related languages are referred simply as *recognizable languages* and the family is denoted by REC. The deterministic variant DREC



(a) Boundary picture. (b) 2OTA.

Fig. 1. (a) A scheme for the boundary picture. (b) 2OTA example. The 3-rd diagonal and the direction of spreading computation are depicted.

[1] coincides with the closure under rotation of $\mathcal{L}(\mathsf{2DOTA})$. It holds $\mathcal{L}(\mathsf{4DFA}) \not\subseteq \mathsf{DREC}$, where 4DFA abbreviates a deterministic 4FA.

3 Sgraffito Automata

We give a definition of bounded 2D Turing machines first, since sgraffito automata are their special instances. Let $\mathcal{H} = \{\mathrm{R}, \mathrm{L}, \mathrm{D}, \mathrm{U}, \mathrm{Z}\}$ be the set of the *head movements*, where the first four elements denote directions (right, left, down, up) and Z stands for zero (none) movement. Furthermore, let us define a mapping $\nu : \mathcal{S} \to \mathcal{H}$ such that

$$\nu(\vdash) = \mathrm{R}, \quad \nu(\dashv) = \mathrm{L}, \quad \nu(\top) = \mathrm{D}, \quad \nu(\bot) = \mathrm{U} \ \text{ and } \ \nu(\#) = \mathrm{Z}.$$

Definition 1. *A (nondeterministic)* two-dimensional bounded Turing machine (2BTM) *is a tuple* $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Q_F)$ *where*

- Σ *is an input alphabet,*
- Γ *is a working alphabet such that* $\Sigma \subseteq \Gamma$*,*
- Q *is a finite, nonempty set of states,*
- $q_0 \in Q$ *is the initial state,*
- $Q_F \subseteq Q$ *is the set of final states, and*
- $\delta : (Q \setminus Q_F) \times (\Gamma \cup \mathcal{S}) \to 2^{Q \times (\Gamma \cup \mathcal{S}) \times \mathcal{H}}$ *is a transition relation.*

Moreover, for any pair $(q, a) \in Q \times (\Gamma \cup \mathcal{S})$*, every* $(q', a', d) \in \delta(q, a)$ *fulfils*

- $a \in \mathcal{S}$ *implies* $d = \nu(a) \wedge a' = a$*, and*
- $a \notin \mathcal{S}$ *implies* $a' \notin \mathcal{S}$*.*

If $\forall q \in Q, \forall a \in \Gamma \cup \mathcal{S} : |\delta(q, a)| \leq 1$*, we say* $\mathcal{M}$ *is a* deterministic 2BTM*.*

The notions like configuration and computation of the machine $\mathcal{M}$ are easily defined as usual. Let $P \in \Sigma^{*,*}$ be an input. In the initial configuration of $\mathcal{M}$ on P, its working tape contains $\widehat{P}$, its control unit is in state q_0 and the head scans the top-left corner of P. When $P = \Lambda$, the head scans the bottom-right corner of $\widehat{P}$ containing #. The machine accepts P iff there is a computation of $\mathcal{M}$ starting in the initial configuration on P and finishing in a state from Q_F.

Definition 2. A two-dimensional sgraffito automaton (2SA) *is a tuple* $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Q_F, \mu)$ *where*

- $(Q, \Sigma, \Gamma, \delta, q_0, Q_F)$ *is a* 2BTM,
- $\mu : \Gamma \to \mathbb{N}$ *is a weight function and the transition relation satisfies*

$$(q', a', d) \in \delta(q, a) \Rightarrow \mu(a') < \mu(a) \quad \text{for all } q, q' \in Q, d \in \mathcal{H}, a, a' \in \Gamma.$$

$\mathcal{A}$ *is a* deterministic 2SA (2DSA) *if the underlying* 2BTM *is deterministic.*

Lemma 1. *Let* $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Q_F)$ *be a* 2BTM. *Let* $k \in \mathbb{N}$ *be an integer such that during each computation of* $\mathcal{M}$ *over any picture from* $\Sigma^{*,*}$, *each tape field is scanned by the head of* $\mathcal{M}$ *in at most* k *configurations. Then, there is a* 2SA $\mathcal{A}$ *such that* $L(\mathcal{A}) = L(\mathcal{M})$. *Moreover, if* $\mathcal{M}$ *is deterministic,* $\mathcal{A}$ *is deterministic too.*

Proof. Let $\mathcal{A} = (Q, \Sigma, \Gamma_2, \delta_2, q_0, Q_F, \mu)$ be a 2SA, where $\Gamma_2 = \Sigma \cup (\Gamma \times \{1, \ldots, k\})$ and each instruction $(q, a) \to (q', a', d)$ from δ is represented in δ_2 by the following set of instructions:

$$\begin{aligned} (q, a) &\to (q', (a', 1), d), \\ (q, (a, i)) &\to (q', (a', i+1), d) \quad \forall i \in \{1, \ldots, k-1\}. \end{aligned}$$

Finally, we define

$$\begin{aligned} \mu(a) &= k+1 && \forall a \in \Sigma, \\ \mu((a, i)) &= k+1-i && \forall (a, i) \in \Gamma \times \{1, \ldots, k\}. \end{aligned}$$

It is easy to see that $L(\mathcal{A}) = L(\mathcal{M})$ and if δ is deterministic, then it produces deterministic δ_2. □

Lemma 1 says that, instead of designing a 2SA, it is sufficient to describe a 2BTM for which the number of transitions over each tape field is bounded by a constant. This will be utilized in the constructive proofs we present. Note that one-dimensional constant-visit machines were already studied by Hennie [5].

Definition 3. *Let* $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Q_F)$ *be a* 2BTM, P *an input picture,* j *an integer such that* $1 \leq j \leq \mathrm{cols}(P)$, *and let* $\mathcal{C}$ *be a finite computation of* $\mathcal{M}$ *over* P. *The* horizontal crossing sequence *for* $\mathcal{C}$ *between columns* j *and* $j+1$, *denoted* $\mathcal{K}_j$, *is constructed by the following procedure:*

1. *Initialize* $\mathcal{K}_j := \emptyset$.
2. *Iterate through all configurations in* $\mathcal{C} = (c_0, c_1, \ldots, c_m)$ *except the last one. Let* c_k *be the current one. Consider the computation step of* $\mathcal{M}$ *that changes* c_k *to* c_{k+1}. *If the head is moved from the* j*-th to* $j+1$*-st column or vice versa and it occurs in the* r*-th row of* P, *append* (q, r) *to* $\mathcal{K}_j$ *where* q *is the state entered by the control unit in* c_{k+1}. *Continue by the next iteration.*

The definition is a two-dimensional generalization of the crossing sequence defined in [6]. It records all activities performed between two columns and thus allows to combine computations done over different pictures as it is given by the proposition which follows. Note also that the vertical crossing sequence could be defined analogously for crossings between two neighboring rows.

Proposition 1. *Let $\mathcal{M}$ be a* 2BTM *accepting pictures $P = P_1 \oplus P_2$ and $R = R_1 \oplus R_2$ where* $\mathrm{rows}(P) = \mathrm{rows}(R)$. *Let $\mathcal{C}_P$ and $\mathcal{C}_R$ be accepting computations of $\mathcal{M}$ over P and R, respectively. If the horizontal crossing sequence for $\mathcal{C}_P$ between columns* $\mathrm{cols}(P_1)$ *and* $\mathrm{cols}(P_1)+1$ *is identical to the horizontal crossing sequence for $\mathcal{C}_R$ between columns* $\mathrm{cols}(R_1)$ *and* $\mathrm{cols}(R_1)+1$, *then $\mathcal{M}$ accepts $P_1 \oplus R_2$.*

Next we show that 2SAs accepting one-row pictures only accept exactly the class of regular (one-dimensional) languages. Actually, the result is only a slight generalization of a theorem by Hennie [5].

Theorem 1. *Let $\mathcal{A}_1 = (Q, \Sigma, \Gamma, \delta, q_0, Q_F, \mu)$ be a* 2SA *accepting a one-dimensional picture language ($L(\mathcal{A}_1) \subseteq \Sigma^{1,*} = \Sigma^*$). There is a finite-state automaton $\mathcal{A}_2$ such that $L(\mathcal{A}_2) = L(\mathcal{A}_1)$.*

Proof. When $\mathcal{A}_1$ works over a one-row picture, it is possible to eliminate its vertical moves without changing the result of the computation. E.g. if $\delta(q, a)$ contains an instruction moving up (q', a', U) (for some $q, q' \in Q, a, a' \in \Gamma$), we can replace it by the set of all instructions of the form (q'', a', Z) such that $(q'', \top, \mathrm{D})$ is in $\delta(q', \top)$. Hence, we can assume $\mathcal{A}_1$ makes no vertical moves. Further, we modify $\mathcal{A}_1$ in such a way that it can enter a final state only when returning from $\dashv$ to the rightmost input symbol (for a nonempty input).

We show how to construct a nondeterministic finite state automaton $\mathcal{A}_2$ accepting $\{\vdash\} \cdot L(\mathcal{A}_1) \cdot \{\dashv\}$. On input $\vdash w \dashv$, the automaton $\mathcal{A}_2$ guesses a computation of $\mathcal{A}_1$ on w by guessing all the horizontal crossing sequences between columns of $\widehat{w}$ and checking if the crossing sequences correspond to an accepting computation. If the guessed and verified computation is accepting, $\mathcal{A}_2$ accepts, otherwise it rejects. The length of any horizontal crossing sequence of $\mathcal{A}_1$ on any one-dimensional input picture is limited by the constant $2 \cdot \max_{a \in \Sigma} \mu(a)$. Hence we can include all such possible crossing sequences into the set of states of $\mathcal{A}_2$.

$\mathcal{A}_2$ starts by reading $\vdash$, guessing a crossing sequence between the first two columns of $\widehat{w}$ and entering the state corresponding to this crossing sequence. The sequence must have an even number length (possibly zero). $\mathcal{A}_2$ also distinguishes in states, whether it scans the first symbol of w or not. It continues as follows. Let s be the crossing sequence corresponding to its current state. $\mathcal{A}_2$ reads the next input symbol a (in a column j) and enters a state corresponding to a nonempty crossing sequence s' representing crossings between the columns j and $j+1$. The sequence s' has to be consistent with s and a. To check that, $\mathcal{A}_2$ guesses a sequence of instructions performed by $\mathcal{A}_1$ while visiting the j-th column and verifies that the induced head movements match the sequences s, s'. If $j = 1$, $\mathcal{A}_1$ knows that the first instruction must start in the state q_0, otherwise the state before applying an instruction is determined by s or s'.

$\mathcal{A}_2$ will enter an accepting state after reading $a = \dashv$ if s is consistent with crossings between two last columns of $\widehat{w}$ and if it ends by $(q_f, 1)$, where $q_f \in Q_F$.

It is easy to verify that $L(\mathcal{A}_2) = \{\vdash\} \cdot L(\mathcal{A}_1) \cdot \{\dashv\}$. Hence, $L(\mathcal{A}_2)$ is a regular language. As the class of regular languages is closed under the left and the right quotient [6], $L(\mathcal{A}_1)$ is regular too. □

4 Closure Properties

Theorem 2. *Both* $\mathcal{L}(\mathsf{2SA})$ *and* $\mathcal{L}(\mathsf{2DSA})$ *are closed under union, intersection, rotation and mirroring.*

Proof. Let $\mathcal{A}_1$, $\mathcal{A}_2$ be two 2SAs and let $L_1 = L(\mathcal{A}_1)$, $L_2 = L(\mathcal{A}_2)$. We can construct a 2SA that starts to compute as $\mathcal{A}_1$ and when $\mathcal{A}_1$ finishes, it computes as $\mathcal{A}_2$. The recognition of $L_1 \cap L_2$ or $L_1 \cup L_2$ requires to accept iff both simulations accept or at least one of the simulations accepts, respectively. For recognizing L_1^R, a 2SA moves its head to the top-right corner of the input and simulates $\mathcal{A}_1$, treating columns as rows and vice versa. Similarly, in order to recognize L_1^{VM} or L_1^{HM}, a 2SA moves its head to the top-right or the bottom-left corner, respectively, and simulates $\mathcal{A}_1$, taking rows or columns, respectively, in the reversed order.

If $\mathcal{A}_1$, $\mathcal{A}_2$ are deterministic, the designed automata are deterministic too. □

Theorem 3. $\mathcal{L}(\mathsf{2SA})$ *is closed under row and column concatenations and projection.*

Proof. Let $\mathcal{A}_1$, $\mathcal{A}_2$ be two 2SAs and let $L_1 = L(\mathcal{A}_1)$, $L_2 = L(\mathcal{A}_2)$. To recognize e.g. $L_1 \Phi L_2$, a 2SA nondeterministically chooses a column in the input and marks it. Then it simulates $\mathcal{A}_1$ over the left part (including the marked column) and, after that, $\mathcal{A}_2$ over the right part (excluding the marked column).

Let π be a projection. For an input P, a 2SA accepting $\pi(L_1)$ guesses and writes down P' such that $\pi(P') = P$. Then it simulates $\mathcal{A}_1$ over P'. □

Theorem 4. $\mathcal{L}(\mathsf{2DSA})$ *is closed under complement.*

Proof. A 2DSA $\mathcal{A}$ rejects an input iff it reaches a state q and scans some a such that $\delta(q, a)$ is empty. Since it is a deterministic automaton, it can be modified to accept the complement of $L = L(\mathcal{A})$, i.e. the language $\overline{L} = \Sigma^{*,*} \setminus L$. □

We use two languages over $\Sigma = \{0, 1\}$ to demonstrate additional properties of sgraffito automata. Their variants were already introduced in [4] and [9]. The language of "duplicates" L_{dup} consists of all pictures $Q \Phi Q$, where Q is a nonempty square over Σ. The language of "permutations" L_{perm} is a subset of L_{dup} and consists of those pictures $Q \Phi Q$, where each row and each column of Q contains symbol 1 exactly once. Examples are shown in Figure 2.

Proposition 2 ([4,9]). L_{dup} *and* L_{perm} *are not in* REC*, while their complements are in* REC.

(a)

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

(b)

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

Fig. 2. Sample pictures from (a) L_{dup} and (b) L_{perm}

Lemma 2. *L_{dup} is not accepted by any* 2SA*.*

Proof. By contradiction, let $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Q_F, \mu)$ be a 2SA accepting L_{dup}. Let $c = \max_{a \in \Sigma} \mu(a)$ and let $L_{\mathrm{dup}}(n)$ be the subset of L_{dup} consisting of pictures whose size is $(n, 2n)$. Moreover, for $P \in L_{\mathrm{dup}}$, let $\mathrm{seq}(P)$ be the crossing sequence of $\mathcal{A}$ on P between columns $\mathrm{cols}(P)/2$ and $\mathrm{cols}(P)/2 + 1$ for some (arbitrarily chosen) accepting computation.

For a fixed n, we estimate the size of the set $\{\mathrm{seq}(P) \mid P \in L_{\mathrm{dup}}(n)\}$. The head can move horizontally in n different rows. Each crossing sequence contains at most $2c$ elements with an identical row index, thus the length of each sequence is not greater than $2cn$. Hence, there are at most

$$\sum_{i=0}^{2cn} (|Q| \cdot n)^i = 2^{\mathcal{O}(n \log n)}$$

different crossing sequences. Since $|L_{\mathrm{dup}}(n)| = 2^{n^2}$, for a sufficiently large n there are two different pictures $P_1 = Q_1 \Phi Q_1$, $P_2 = Q_2 \Phi Q_2$ in $L_{\mathrm{dup}}(n)$ such that $\mathrm{seq}(P_1) = \mathrm{seq}(P_2)$. By Proposition 1, $\mathcal{A}$ accepts $P_3 = Q_1 \Phi Q_2$, but $P_3 \notin L_{\mathrm{dup}}$. □

Since $\mathcal{L}$(2DSA) is closed under complement, we obtain the following corollary.

Corollary 1. *$\overline{L}_{\mathrm{dup}}$ is not accepted by any* 2DSA*.*

Theorem 5. *$\mathcal{L}$(2SA) is not closed under complement. $\mathcal{L}$(2DSA) is not closed under row, neither column concatenation.*

Proof. We will prove that $\overline{L}_{\mathrm{dup}} \in \mathcal{L}(\mathsf{2SA})$. To do it, we use the decomposition of $\overline{L}_{\mathrm{dup}}$ given in [4]. Let $\Sigma = \{0, 1\}$. We have $\overline{L}_{\mathrm{dup}} = L_1 \cup L_2$, where

$$L_1 = \{P \in \Sigma^{*,*} \mid \mathrm{cols}(P) \neq 2 \cdot \mathrm{rows}(P)\},$$

$$L_2 = \{Q_1 \Phi Q_2 \mid Q_1, Q_2 \in \Sigma^{*,*} \wedge \mathrm{cols}(Q_1) = \mathrm{cols}(Q_2) = \mathrm{rows}(Q_1) \wedge Q_1 \neq Q_2\}.$$

L_2 can be further expressed as

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

where

$$L_3 = \{P \in \Sigma^{*,*} \mid \mathrm{cols}(P) = 2 \cdot \mathrm{rows}(P)\},$$

$$L_4 = \{P \in \Sigma^{*,*} \mid \mathrm{cols}(P) = \mathrm{rows}(P) + 1\},$$

$$L_5 = \{P \in \Sigma^{*,*} \mid \mathrm{rows}(P) = 1 \wedge P(1,1) \neq P(1,\mathrm{cols}(P))\}.$$

L_5 contains one-row pictures only and is regular. $\Sigma^{*,*}$ is trivially in $\mathcal{L}(\mathsf{4DFA})$. The languages L_1, L_3, L_4 are recognizable by a 4DFA which checks the condition on size by passing the input diagonally. Thus, the already proved closure properties of $\mathcal{L}(\mathsf{2SA})$ guarantee $\overline{L}_{\mathrm{dup}}$ is in $\mathcal{L}(\mathsf{2SA})$.

By Corollary 1, $\overline{L}_{\mathrm{dup}} \notin \mathcal{L}(\mathsf{2DSA})$, hence $\mathcal{L}(\mathsf{2DSA})$ is not closed under (w.l.o.g.) the row concatenation. It holds

$$P_1 \ominus P_2 = \left(\left(\left(P_2^{\mathrm{R}} \Phi P_1^{\mathrm{R}}\right)^{\mathrm{R}}\right)^{\mathrm{R}}\right)^{\mathrm{R}},$$

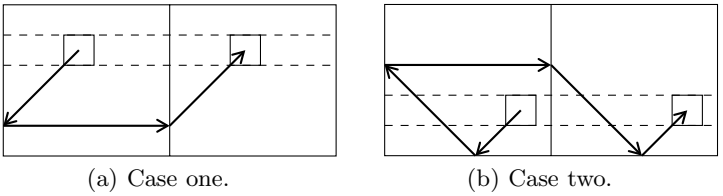
thus $\mathcal{L}(\mathsf{2DSA})$ is not closed under the column concatenation as well. □

Theorem 6. $\mathcal{L}(\mathsf{2DSA})$ *is not closed under projection.*

Proof. Let $\Sigma_1 = \{0,1\}$, $\Sigma_2 = \{\overline{0},\overline{1}\}$, $\Sigma = \Sigma_1 \cup \Sigma_2$ and let $\pi : \Sigma \to \Sigma_1$ be a mapping such that $\pi(0) = \pi(\overline{0}) = 0$, $\pi(1) = \pi(\overline{1}) = 1$. Define a language L_1 over Σ consisting of all pictures of the form $Q_1 \Phi Q_2$, where Q_1 is a square over Σ containing exactly one symbol from Σ_2 (at some position (i,j)), and Q_2 is a square over Σ such that $\pi(Q_2(i,j)) \neq \pi(Q_1(i,j))$. Next, define

$$L_2 = \{P \in \Sigma^{*,*} \mid \mathrm{cols}(P) \neq 2 \cdot \mathrm{rows}(P)\} \text{ and } L = L_1 \cup L_2.$$

It is should be clear that $\pi(L) = \overline{L}_{\mathrm{dup}}$. To finish the proof we will construct a 2DSA $\mathcal{A}$ accepting L. Given some input P, $\mathcal{A}$ checks the size of P. When it is $(n, 2n)$, it marks the last column of the left half of P and verifies that this half contains just one symbol from Σ_2 (at a position (i,j)). $\mathcal{A}$ marks the whole i-th row as *working* and moves the head back to position (i,j). Then it locates the corresponding tape field in the right half of P, at position $(i, n+j)$. To do it, a bouncing traversal style shown in Figure 3 is performed until the working row is reached during the final phase of the movement. Finally, $\mathcal{A}$ checks whether $\pi(P(i,j)) \neq \pi(P(i,n+j))$. □



(a) Case one. (b) Case two.

Fig. 3. Locating the corresponding field in the opposite half using a bouncing style. Dashed lines denote the marked working row. Oblique directions make an angle of 45°.

5 A Taxonomy of Picture Languages

Theorem 7. $\mathcal{L}(\mathsf{4FA})$ *is included in* $\mathcal{L}(\mathsf{2DSA})$.

Proof. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, Q_F)$ be a 4FA and P an input over Σ. Define a directed graph $G = (V, E)$ as follows.

- vertices are all triples of the form (q, i, j), where $1 \leq i \leq \mathrm{rows}(\widehat{P})$, $1 \leq j \leq \mathrm{cols}(\widehat{P})$ and $q \in Q$,
- $((q_1, i_1, j_1), (q_2, i_2, j_2))$ is an edge iff δ contains $(q_1, \widehat{P}(i_1, j_1)) \to (q_2, d)$ and (i_2, j_2) is the coordinate next to (i_1, j_1) in the direction given by d.

$\mathcal{A}$ accepts P iff, for some $q_f \in Q_F$, there is a vertex (q_f, i, j) reachable from $(q_0, 2, 2)$. To decide this, it suffices to perform a depth first search in G. We give a related procedure that labels visited nodes and edges. Vertices are in two states – *unexplored* and *explored*, edges in three states – *unexplored*, *tree edge*, *cross edge*. All elements are initially in the unexplored state.

1: $v := (q_0, 2, 2)$
2: label v as explored
3: **if** v represents an accepting configuration **then**
4: **ACCEPT**
5: **end if**
6: **if** there is an unexplored edge $e = (v, w)$ **then**
7: **if** w is unexplored **then**
8: label e as tree edge, move to w, set $v := w$
9: **goto** 2
10: **else**
11: label e as cross edge
12: **goto** 6
13: **end if**
14: **else if** there is an incoming tree edge (u, v) **then**
15: move to u, set $v := u$
16: **goto** 6
17: **end if**
18: **REJECT**

Labels of a vertex (q, i, j) and of its outgoing edges are recorded in the tape field storing $\widehat{P}(i, j)$. The exception are vertices on the border, their labels are represented in the nearest tape field storing an inner part of $\widehat{P}$. Since each vertex has the number of outgoing edges limited by $|\mathcal{H}| \cdot |Q|$, the proposed algorithm can be performed by a 2DSA, a constant memory usage as well as a constant number of traversals are guaranteed for each tape field. □

Theorem 8. REC *is included in* $\mathcal{L}(\mathsf{2SA})$, DREC *is included in* $\mathcal{L}(\mathsf{2DSA})$.

Proof. Let L be a language in REC. It is accepted by a 2OTA $\mathcal{A}_1$. We can easily construct a 2SA $\mathcal{A}_2$ simulating $\mathcal{A}_1$. It goes trough the input e.g. row by row,

retrieves info needed to simulate a transition at each cell and represents the final state in the corresponding tape field. It nondeterministically branches when $\mathcal{A}_1$ does so. If $\mathcal{A}_1$ is a 2DOTA, then $\mathcal{A}_2$ is a 2DSA, thus $\mathcal{L}(\mathsf{2DOTA}) \subseteq \mathcal{L}(\mathsf{2DSA})$. Since $\mathcal{L}(\mathsf{2DSA})$ is closed under rotation, it includes the closure by rotation of $\mathcal{L}(\mathsf{2DOTA})$ which equals DREC (shown in [1]). □

Lemma 3. *L_{perm} is accepted by a* 2DSA.

Proof. We construct a 2DSA $\mathcal{A}$ recognizing L_{perm}. It starts by checking if an input P is of size $(n, 2n)$ and marks the n-th column. After that, it verifies if the both halves Q_1, Q_2 represent permutations, i.e. if each their row and column contains exactly one occurrence of symbol 1. This is done traversing P row by row first, followed column by column.

The second stage compares the content of Q_1 and Q_2 row by row. Consider processing an i-th row. The whole row is marked as working. The leftmost symbol 1 is located in the row. Let it be in a j-th column. Now, $\mathcal{A}$ moves the head to the top of this column (coordinate $(1, j)$). Next, the field at the coordinate $(1, n+j)$ is located using the bouncing style given by Figure 3(a). Finally, the position $(i, n+j)$ is reached by moving the head down and stopping at the working row. If there is symbol 1, the iteration finishes by clearing the used markers in the i-th row and the process is ready to be started on the next row.

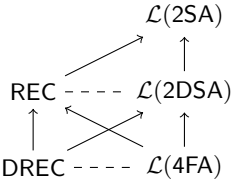
It remains to show that $\mathcal{A}$ visits each field of the working tape constantly many times and thus it is correctly defined (Lemma 1). Constantly many traversals trough P are performed during the first stage. In the second stage, each iteration works in a unique row and column. Especially the column uniqueness ensures that different paths are always used to locate the tape field in the right half. Thus, a constant number of visits is achieved on each field again. □

Theorem 9. *Families $\mathcal{L}(\mathsf{2DSA})$ and* REC *are incomparable.*

Proof. After summarizing Proposition 2, Corollary 1 and Lemma 3, we get

$$\overline{L}_{\mathrm{dup}} \in (\mathsf{REC} \setminus \mathcal{L}(\mathsf{2DSA})) \quad \text{and} \quad L_{\mathrm{perm}} \in (\mathcal{L}(\mathsf{2DSA}) \setminus \mathsf{REC}) .$$

□



(a) Families hierarchy.

	∪	∩	\	∋, ∌	π	R,VM,HM
REC	yes	yes	no	yes	yes	yes
$\mathcal{L}$(2SA)	yes	yes	no	yes	yes	yes
DREC	yes	yes	yes	no	no	yes
$\mathcal{L}$(2DSA)	yes	yes	yes	no	no	yes

(b) Closure properties.

Fig. 4. (a) Relationships between REC, DREC, $\mathcal{L}$(4FA) and the families recognizable by sgraffito automata. Proper inclusions are denoted by arrows, the dashed lines connect incomparable classes. (b) A summary of closure properties.

6 Conclusions

We have introduced a new computational model called sgraffito automaton and investigated its properties. The hierarchy formed by the induced classes of picture languages, REC and DREC is shown in Figure 4(a), which is based on new as well as already known theorems. If the automaton is restricted to work over one-row pictures only, the recognition power degenerates to the power of finite-state automaton. Such results give the families a great importance and entitle us to see them as alternative ground levels in the two-dimensional hierarchy. This is also well justified by the results on closure properties. The table in Figure 4(b) demonstrates how they coincide with the properties of REC and DREC.

In our opinion, sgraffito automata deserve to be the subject of further research. A special attention should be paid to 2DSAs, since they simulate 4FAs and define thus an interesting deterministic family. The study of the automata can provide additional insight on the fundamental differences between one-dimensional and two-dimensional languages.

References

1. Anselmo, M., Giammarresi, D., Madonia, M.: From Determinism to Non-determinism in Recognizable Two-Dimensional Languages. In: Harju, T., Karhumäki, J., Lepistö, A. (eds.) DLT 2007. LNCS, vol. 4588, pp. 36–47. Springer, Heidelberg (2007)
2. Blum, M., Hewitt, C.: Automata on a 2-dimensional tape. In: Proceedings of the 8th Annual Symposium on Switching and Automata Theory (SWAT 1967), FOCS 1967, pp. 155–160. IEEE Computer Society, Washington, DC (1967)
3. Giammarresi, D., Restivo, A.: Recognizable picture languages. International Journal of Pattern Recognition and Artificial Intelligence 6(2-3), 32–45 (1992)
4. Giammarresi, D., Restivo, A.: Two-dimensional languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 3, pp. 215–267. Springer-Verlag New York, Inc., New York (1997)
5. Hennie, F.: One-tape, off-line Turing machine computations. Information and Control 8(6), 553–578 (1965)
6. Hopcroft, J., Ullman, J.: Formal languages and their relation to automata. Addison-Wesley (1969)
7. Inoue, K., Nakamura, A.: Some properties of two-dimensional on-line tessellation acceptors. Information Sciences 13, 95–121 (1977)
8. Jiřička, P., Král, J.: Deterministic forgetting planar automata are more powerful than non-deterministic finite-state planar automata. In: Rozenberg, G., Thomas, W. (eds.) Developments in Language Theory, pp. 71–80. World Scientific (1999)
9. Kari, J., Moore, C.: New Results on Alternating and Non-deterministic Two-Dimensional Finite-State Automata. In: Ferreira, A., Reichel, H. (eds.) STACS 2001. LNCS, vol. 2010, pp. 396–406. Springer, Heidelberg (2001)
10. Lindgren, K., Moore, C., Nordahl, M.: Complexity of two-dimensional patterns. Journal of Statistical Physics 91(5-6), 909–951 (1998)