RAIRO-Theor. Inf. Appl. 48 (2014) 505–539
DOI: 10.1051/ita/2014023

Available online at:
www.rairo-ita.org

TWO-DIMENSIONAL SGRAFFITO AUTOMATA *, **

Daniel Průša[1], František Mráz[2] and Friedrich Otto[3]

Abstract. We present a new model of a two-dimensional computing device called *Sgraffito automaton*. In general, the model is quite simple, which allows a clear design of computations. When restricted to one-dimensional inputs, that is, strings, the Sgraffito automaton does not exceed the power of finite-state automata. On the other hand, for two-dimensional inputs, it yields a family of picture languages with good closure properties that strictly includes the class REC of recognizable picture languages. The deterministic Sgraffito automata define a class of picture languages that includes the class of deterministic recognizable picture languages DREC, the class of picture languages that are accepted by four-way alternating automata, those that are accepted by deterministic one-marker automata, and the sudoku-deterministically recognizable picture languages, but the membership problem for the accepted languages is still decidable in polynomial time. In addition, the deterministic Sgraffito automata accept some unary picture languages that are outside of the class REC.

Mathematics Subject Classification. 68Q45.

Keywords and phrases. Two-dimensional language, Sgraffito automaton, bounded Turing machine, REC.

* *This paper summarizes and extends results that have been announced at DLT 2012 in Taipei, August 2012, at DLT 2013 in Paris, June 2013, and at CIAA 2013 in Halifax, July 2013. Extended abstracts appeared in the proceedings of these conferences [18, 20, 21].*

** *The first two authors were supported by the Czech Science Foundation – the first author under the project P103/10/0783, and the second author under the project P202/10/1333.*

[1] Czech Technical University, Faculty of Electrical Engineering, Karlovo náměstí 13, 12135 Prague 2, Czech Republic. prusapa1@cmp.felk.cvut.cz

[2] Charles University at Prague, Faculty of Mathematics and Physics, Department of Computer Science, Malostranské nám. 25, 11800 Prague 1, Czech Republic. frantisek.mraz@mff.cuni.cz

[3] Fachbereich Elektrotechnik/Informatik, Universität Kassel, 34109 Kassel, Germany. otto@theory.informatik.uni-kassel.de

Article published by EDP Sciences

© EDP Sciences 2015

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

1. Introduction

Many concepts and techniques from the theory of formal languages and automata have been generalized to two-dimensional languages, where the basic entity, the *string*, has been replaced by a two-dimensional rectangular array of symbols, called a *picture*. By generalizing various one-dimensional computational or generative models to the two-dimensional setting, many different classes of picture languages have been defined. One of the driving forces behind these definitions has been the quest for a two-dimensional counterpart of the Chomsky hierarchy for string languages. Typically, one asks whether the class of picture languages that is obtained by generalizing a particular characterization of a class of the Chomsky hierarchy to the two-dimensional setting has the same closure properties or algorithmic properties as the corresponding class of string languages. Unfortunately, the answer to this question is typically negative. The more complex topology of pictures causes the resulting classes of picture languages to be much more complex than the corresponding string languages.

The *four-way finite automaton* (4FA) of Blum and Hewitt [2] is a case in point. A 4FA has a finite-state control and a head that moves over an input picture by performing move-right, move-left, move-up, and move-down steps. Although this is a straightforward generalization of the two-way finite-state acceptor, the properties of the class of picture languages accepted by these automata differ sharply from those of the regular languages (see, *e.g.*, [6]). For example, it turned out that the deterministic 4FA accepts strictly less languages than the nondeterministic 4FA.

In 1991, Giammaresi and Restivo proposed the family of *recognizable picture languages* (REC) [5]. This class is defined using *tiling systems*, and it coincides with the class of picture languages that are recognized by the two-dimensional on-line tessellation automata [11], and it corresponds to the class of picture languages that are definable in existential monadic second order logic [7]. The class REC has many remarkable properties, and the type of recognizability given by this class is a very robust notion. REC is even considered as the ground-level class among the families of two-dimensional languages.

However, due to the nondeterministic nature of tiling systems, REC is quite powerful. In fact, it even contains some NP-complete languages [17], which somehow contradicts the idea of a ground level class, when taking into account the simplicity of the devices that are sufficient to recognize the (one-dimensional) regular languages. Accordingly, more restricted classes of picture languages have been proposed and studied like, for example, the class DREC of *deterministically recognizable picture languages* [1] and the *sudoku-deterministically recognizable picture languages* [3]. These types of picture languages are defined through certain deterministic variants of tiling systems – to our knowledge no characterizations in terms of (simple) automata are known for them apart from the fact that DREC coincides with the closure of the class of picture languages that are accepted by deterministic two-dimensional on-line tessellation automata under rotation.

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

Here we present a new type of a two-dimensional computing device called *Sgraffito automaton* (2SA). The name "Sgraffito automaton" is derived from a technique used in the visual arts:

> 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 on Nov. 11, 2013, from http://www.britannica.com/EBchecked/topic/537397/sgraffito).

A 2SA $\mathcal{A}$ has a finite-state control and a read/write head which works on a rectangular picture that contains symbols from some weighted alphabet Γ and that is surrounded by border markers. The 2SA $\mathcal{A}$ can move its head in all four directions, as long as it stays on the picture. In each step it replaces the symbol under its head by a symbol with smaller weight (this corresponds to the process of scratching some of the top layers of paint), it changes its state, and it moves its head to a neighbouring field. $\mathcal{A}$ accepts by entering an accepting state. In addition to the symbols of an input alphabet Σ, the working alphabet Γ may also contain some additional symbols, called *auxiliary symbols*.

The class of picture languages $\mathcal{L}$(2SA) that are accepted by Sgraffito automata strictly extends the class REC, while it exhibits the same nice closure properties. However, when restricted to one-row pictures, that is, string languages, then its expressive power collapses to that of finite-state acceptors. In addition to its expressiveness, a significant advantage of the Sgraffito automaton is its simplicity. Many important computations can be described in a simple and clear way for Sgraffito automata. For example, Sgraffito automata can be used to solve certain basic tasks of digital geometry [22] like determining the connected components in a black-and-white picture, or modelling computations in an unstable environment that is effected by the presence of simple computing entities (*e.g.*, nanobots that are moving in a plane, thereby damaging the surface, or consuming it, using it as fuel). In addition, the simplicity of the model might even be of an advantage in automata based model checking in a two-dimensional setting.

An important and interesting restriction of the Sgraffito automaton is its *deterministic variant*, the 2DSA. The class of languages $\mathcal{L}$(2DSA) extends many previously considered classes of picture languages, among them the class DREC, the class $\mathcal{L}$(4FA), the sudoku-deterministically recognizable picture languages, and the class of picture languages that are accepted by *deterministic one-marker automata* [2]. In addition, the membership problem for each picture language $L \in \mathcal{L}$(2DSA) is decidable in polynomial time. Thus, the deterministic Sgraffito automata are a new and more powerful deterministic alternative to DREC. This fact is complemented by our observation that $\mathcal{L}$(2DSA) is a proper subclass of the class of picture languages that are accepted by the two-dimensional deterministic forgetting automaton (2DFA) that was introduced by Jiřička and Král in [15]. In addition, we prove that deterministic Sgraffito automata accept some unary picture languages that are outside the class REC of recognizable picture languages.

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

Finally we compare the nondeterministic two-dimensional one-marker automata to the Sgraffito automata, showing that the former accept some picture languages that cannot be accepted by the latter. However, if nondeterministic two-dimensional one-marker automata were to accept all picture languages that are accepted by Sgraffito automata or by deterministic Sgraffito automata, respectively, then the complexity class NL (nondeterministic logarithmic space) would equal the complexity class NP (nondeterministic polynomial time) or P (deterministic polynomial time), respectively. Accordingly, it is likely that the classes of picture languages accepted by these two types of nondeterministic automata are incomparable under inclusion.

This paper is structured as follows. In the next section we restate the basic notions and notation on picture languages and some operations on picture languages, and we repeat the definitions of some types of two-dimensional automata from the literature in short. Then in Section 3 we introduce our model, the Sgraffito automaton, as a special type of the two-dimensional bounded Turing machine, and we present a technical result based on crossing sequences that can be used to prove some lower bound results. In the next section we show that the class $\mathcal{L}(\mathsf{2SA})$ of picture languages that are accepted by Sgraffito automata has many nice closure properties, and we also establish some closure properties for the class $\mathcal{L}(\mathsf{2DSA})$ of picture languages that are accepted by deterministic Sgraffito automata. In the following four sections we compare the expressive power of (deterministic) Sgraffito automata to that of various other types of two-dimensional automata, establishing a taxonomy of picture languages. We will see that already the deterministic Sgraffito automaton is quite powerful. Finally, in Section 9 we consider unary picture languages that are accepted by deterministic Sgraffito automata. The paper closes with a short summary and some open problems in Section 10.

2. Preliminaries

Here we use the common notation and terms on pictures and picture languages (see, *e.g.*, [6]). If Σ is a finite alphabet, then $\Sigma^{*,*}$ is used to denote the set of rectangular pictures over Σ, that is, if $P \in \Sigma^{*,*}$, then P is a two-dimensional array of symbols from Σ. We denote the number of rows and columns of a picture 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 $(0,0)$, and for all $m, n \geq 1$, $\Sigma^{m,n}$ denotes the set of pictures of size (m,n) over Σ. A *picture language* over Σ is a subset of $\Sigma^{*,*}$. For $1 \leq i \leq \mathrm{rows}(P)$ and $1 \leq j \leq \mathrm{cols}(P)$, $P(i,j)$ (or shortly $P_{i,j}$) identifies the symbol located in the ith row and the jth column of 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 if and only if $k = m$, while the *row concatenation* $P \ominus Q$ is defined

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

if and only if $l = n$. The corresponding products are depicted below:

$$P \oplus Q = \begin{bmatrix} P_{1,1} & \dots & P_{1,l} & Q_{1,1} & \dots & Q_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ P_{k,1} & \dots & P_{k,l} & Q_{m,1} & \dots & Q_{m,n} \end{bmatrix}$$

$$\text{and } P \ominus Q = \begin{bmatrix} P_{1,1} & \dots & P_{1,l} \\ \vdots & \ddots & \vdots \\ P_{k,1} & \dots & P_{k,l} \\ Q_{1,1} & \dots & Q_{1,n} \\ \vdots & \ddots & \vdots \\ Q_{m,1} & \dots & Q_{m,n} \end{bmatrix}.$$

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

The *column closure* $L^{*\oplus}$ of a picture language L is defined as $L^{*\oplus} = \bigcup_{i \geq 0} L^{i\oplus}$, where $L^{0\oplus} = \{\Lambda\}$, $L^{1\oplus} = L$, and $L^{i\oplus} = L \oplus L^{(i-1)\oplus}$ for all $i \geq 2$. Similarly, the *row closure* $L^{*\ominus}$ is defined as $L^{*\ominus} = \bigcup_{i \geq 0} L^{i\ominus}$, where $L^{0\ominus} = \{\Lambda\}$, $L^{1\ominus} = L$, and $L^{i\ominus} = L \ominus L^{(i-1)\ominus}$ for all $i \geq 2$. These two operations can be seen as the extension of the Kleene star from string languages to picture languages.

In addition, we introduce the *clockwise rotation* P^{R}, the *transposition* P^T, the *vertical mirroring* P^{VM}, and the *horizontal mirroring* P^{HM} of a picture $P \in \Sigma^{m,n}$:

$$P^{\mathrm{R}} = \begin{bmatrix} P_{m,1} & \dots & P_{1,1} \\ \vdots & \ddots & \vdots \\ P_{m,n} & \dots & P_{1,n} \end{bmatrix}, \quad P^T = \begin{bmatrix} P_{1,1} & \dots & P_{m,1} \\ \vdots & \ddots & \vdots \\ P_{1,n} & \dots & P_{m,n} \end{bmatrix},$$

$$P^{\mathrm{VM}} = \begin{bmatrix} P_{1,n} & \dots & P_{1,1} \\ \vdots & \ddots & \vdots \\ P_{m,n} & \dots & P_{m,1} \end{bmatrix}, P^{\mathrm{HM}} = \begin{bmatrix} P_{m,1} & \dots & P_{m,n} \\ \vdots & \ddots & \vdots \\ P_{1,1} & \dots & P_{1,n} \end{bmatrix}.$$

Let $\pi : \Sigma \to \Gamma$ be a mapping, where Γ is an alphabet. Then π induces a mapping from $\Sigma^{*,*}$ to $\Gamma^{*,*}$ by sending $P \in \Sigma^{m,n}$ to $P' \in \Gamma^{m,n}$ such that $P'(i,j) = \pi(P(i,j))$ for each $1 \leq i \leq m$ and $1 \leq j \leq n$. This mapping is called a *projection* from $\Sigma^{*,*}$ to $\Gamma^{*,*}$, and P' is simply written as $\pi(P)$.

Note that each of the above operations naturally extends to languages.

Let $\mathcal{S} = \{\vdash, \dashv, \top, \bot, \#\}$ be a set of five special markers, called *sentinels*. In what follows, we will always assume implicitly that $\Sigma \cap \mathcal{S} = \emptyset$ for any alphabet Σ considered. In order to enable our automata to be defined below to detect the border of a picture P easily, we define the *boundary picture* $\widehat{P}$ over $\Sigma \cup \mathcal{S}$ of size $(m+2) \times (n+2)$, which is illustrated by Figure 1. Here the symbols $\vdash, \dashv, \top$ and $\bot$ uniquely identify the corresponding borders (left, right, top, bottom) of $\widehat{P}$, while the symbol # marks the corners of $\widehat{P}$.

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

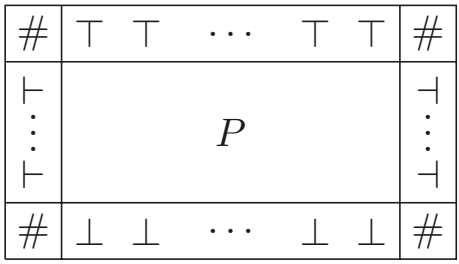


FIGURE 1. The boundary picture $\widehat{P}$.

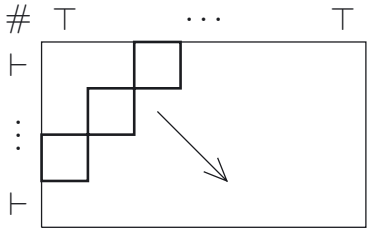


FIGURE 2. A 2OTA: The 3-rd diagonal and the direction of the computation are depicted.

As mentioned in the introduction, a large variety of accepting devices has been formulated and studied in the literature. Here we restate some of them in short.

The *4-way finite automaton* (4FA) is the two-dimensional variant of the finite-state acceptor. Such an automaton is defined by a tuple $\mathcal{A} = (Q, \Sigma, q_0, q_a, q_r, \delta)$, where Q is a finite set of states, $q_0, q_a, q_r \in Q$ are the initial state, the accepting state, and the rejecting state, and $\delta : ((Q \setminus \{q_a, q_r\}) \times \Sigma) \to 2^{(Q \times \{\mathrm{L,R,U,D}\})}$ is the transition relation [2]. Here L, R, U, and D denote directions (left, right, up, down). The 4FA $\mathcal{A}$ accepts a picture $P \in \Sigma^{*,*}$ if it has an accepting computation on P starting in its initial state q_0 from position $(1, 1)$. If δ is a (partial) function, then $\mathcal{A}$ is a *deterministic 4-way finite automaton* (4DFA). It is known that, in contrast to the one-dimensional case, $\mathcal{L}$(4DFA) is a proper subclass of $\mathcal{L}$(4FA).

The *two-dimensional on-line tessellation automaton* (2OTA) is a restricted type of cellular automaton [11]. For an input $P \in \Sigma^{*,*}$, a cell is placed at each position (i, j) of P. Then a computation is performed that consists in $\mathrm{rows}(P) + \mathrm{cols}(P) - 1$ many parallel steps. During the kth step, each cell at coordinates (i, j), where $i + j - 1 = k$, performs a state-transition that depends on $P(i, j)$ and on the states of its left and top neighbour cells. If a neighbour lies at the border of P, it is a fictive cell the state of which is defined as the corresponding symbol of $\widehat{P}$. The result of the computation is determined by the final state of the cell at the bottom-right corner of P. This way of operation is illustrated by Figure 2.

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

Also for the 2OTA, the deterministic variant, 2DOTA, is strictly weaker than the nondeterministic variant, and $\mathcal{L}$(4FA) is properly contained in $\mathcal{L}$(2OTA), while $\mathcal{L}$(2DOTA) is incomparable to both, $\mathcal{L}$(4DFA) and $\mathcal{L}$(4FA), with respect to inclusion [11].

Finally, we turn to tiling systems. A *tile* is a square picture of size (2,2), and for a picture P, $B_{2,2}(\widehat{P})$ denotes the set of all tiles that are subpictures of $\widehat{P}$. Now a picture language $L \subseteq \Sigma^{*,*}$ is called a *local language*, if there exists a finite set of tiles Θ such that $L = \{ P \in \Sigma^{*,*} \mid B_{2,2}(\widehat{P}) \subseteq \Theta \}$, that is, $P \in L$ if and only if all tiles that are subpictures of $\widehat{P}$ belong to Θ. This fact is expressed as $L = L(\Theta)$.

A *tiling system* (TS) is given through a tuple $\mathcal{T} = (\Sigma, \Gamma, \Theta, \pi)$, where Σ and Γ are two finite alphabets, Θ is a finite set of tiles over $\Gamma \cup \mathcal{S}$, and $\pi : \Gamma \to \Sigma$ is a projection. Then the language $L = L(\mathcal{T}) \subseteq \Sigma^{*,*}$ that is defined by $\mathcal{T}$ is the projection $\pi(L(\Theta))$ of the local language specified by Θ. $\mathcal{L}$(TS) is called the class of *tiling recognizable languages* [5]. Since $\mathcal{L}$(TS) = $\mathcal{L}$(2OTA) [12], and since this class has many nice closure properties, it is simply being referred to as REC.

The deterministic variant DREC (see [1]) coincides with the closure of $\mathcal{L}$(2DOTA) under rotation. Notice, however, that $\mathcal{L}$(4DFA) $\not\subseteq$ DREC holds.

3. Sgraffito automata

First we restate the definition of the two-dimensional bounded Turing machine, as the Sgraffito automaton is a special instance of it. Let $\mathcal{H} = \{\mathrm{R}, \mathrm{L}, \mathrm{D}, \mathrm{U}, \mathrm{Z}\}$ be the set of possible *head movements*, where the first four elements denote directions (right, left, down, up), while Z stands for zero (no) movement. Furthermore, let $\nu : \mathcal{S} \to \mathcal{H}$ denote the mapping that is defined by

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

that is, for each occurrence of a sentinel from $\{\vdash, \dashv, \top, \bot\}$ within a boundary picture $\widehat{P}$, ν yields the direction that leads to the nearest field of the proper picture P.

Definition 3.1. A *(nondeterministic) two-dimensional bounded Turing machine (*2BTM*)* is given by 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 such that the following properties are satisfied for each pair $(q, a) \in (Q \setminus Q_F) \times (\Gamma \cup \mathcal{S})$ and each transition $(q', a', d) \in \delta(q, a)$:
  - if $a \in \mathcal{S}$, then $d = \nu(a)$ and $a' = a$, and
  - if $a \notin \mathcal{S}$, then $a' \notin \mathcal{S}$.

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

If $|\delta(q,a)| \leq 1$ for all $q \in Q$ and $a \in \Gamma \cup \mathcal{S}$, then $\mathcal{M}$ is a *deterministic* 2BTM.

The notions of configuration and computation of the machine $\mathcal{M}$ are defined as usual. Let $P \in \Sigma^{*,*}$ be an input. In the initial configuration of $\mathcal{M}$ on input P, the working tape contains the boundary picture $\widehat{P}$, $\mathcal{M}$ is in its initial state q_0, and its head scans the top-left corner of P, that is, cell $(2,2)$ of $\widehat{P}$. If $P = \Lambda$, then the head is assumed to scan the bottom-right corner of $\widehat{P}$ containing #. The machine $\mathcal{M}$ is said to *accept* P if there is a computation of $\mathcal{M}$ that starts in the initial configuration on input P and that finishes in a state from Q_F. By $L(\mathcal{M})$ we denote the picture language that consists of all pictures that are accepted by $\mathcal{M}$.

Obviously, the 2BTM is the two-dimensional variant of the *linear bounded automaton*, as given a picture P as input, the 2BTM $\mathcal{M}$ cannot leave the part of the tape that contains the input P and its border. Now the Sgraffito automaton is obtained as a restricted variant of the 2BTM.

Definition 3.2. A *two-dimensional Sgraffito automaton (*2SA*)* is given by 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; and
- $\mu : \Gamma \to \mathbb{N}$ is a weight function such that the condition $\mu(a') < \mu(a)$ is satisfied for all transitions $(q', a', d) \in \delta(q, a)$ satisfying $a \in \Gamma$.

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

Let $P \in \Sigma^{*,*}$ be a given input picture for the 2SA $\mathcal{A}$. Then $\mathcal{A}$ starts in its initial state q_0 with its head scanning the top-left corner of P, while its tape contains the boundary picture $\widehat{P}$. Now $\mathcal{A}$ walks across this boundary picture, but as it is a 2BTM, it cannot leave the space covered by $\widehat{P}$. Furthermore, when executing a transition step at a position (i,j) of P, then it must replace the current symbol, say a, at that position by a symbol, say a', that has strictly smaller weight than a. It follows that $\mathcal{A}$ can visit each position of P, and therewith of $\widehat{P}$, at most $|\Gamma|$ many times. Accordingly, the 2SA can be seen as the two-dimensional extension of the one-dimensional *constant-visit machines* studied by Hennie [8]. This observation is underlined by the following technical result.

Lemma 3.3. *Let $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Q_F)$ be a* 2BTM*. Assume that there exists a positive integer k such that during each computation of $\mathcal{M}$ on 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, then so is $\mathcal{A}$.*

Proof. Let $\mathcal{A} = (Q, \Sigma, \Gamma_1, \delta_1, q_0, Q_F, \mu)$ be the 2SA that is obtained from the given 2BTM $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Q_F)$ and the integer k by taking $\Gamma_1 = \Sigma \cup (\Gamma \times \{1, \ldots, k\})$, and by taking the following set of instructions for all $q \in Q$,

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

$s \in \mathcal{S}$, and $a, a' \in \Gamma$:

$$\begin{aligned}
\delta_1(q,s) &= \delta(q,s),\\
\delta_1(q,a) &= \{(q',(a',1),d) \mid (q',a',d) \in \delta(q,a)\},\\
\delta_1(q,(a,i)) &= \{(q',(a',i+1),d) \mid (q',a',d) \in \delta(q,a)\} \text{ for all } 1 \le i < k.
\end{aligned}$$

Finally, we define

$$\begin{aligned}
\mu(a) &= k+1 \qquad \text{for all } a \in \Sigma,\\
\mu((a,i)) &= k+1-i \text{ for all } (a,i) \in \Gamma \times \{1,\dots,k\}.
\end{aligned}$$

It is easily verified that $L(\mathcal{A}) = L(\mathcal{M})$ holds. In addition, if $\mathcal{M}$ is deterministic, that is, if δ is a partial function, then δ_1 is a (partial) function, too, that is, $\mathcal{A}$ is a deterministic 2SA. □

The advantage of our model is its constructiveness. While it is undecidable in general whether a given Turing machine is a Hennie machine (see, *e.g.*, [19]), the property of being weight-reducing can easily be checked algorithmically.

Lemma 3.3 says that, instead of designing a 2SA, it suffices to describe a 2BTM for which the number of transitions over each tape field is bounded by a constant. This observation will be utilized in the constructive proofs we present below.

Next we define the notion of *horizontal crossing sequence* for two-dimensional bounded Turing machines. This definition is a generalization of the notion of *crossing sequence* defined in [9] to the two-dimensional setting.

Definition 3.4. Let $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Q_F)$ be a 2BTM, let P be an input picture, and let $\mathcal{C}$ be a finite computation of $\mathcal{M}$ over P. The *horizontal crossing sequences* $\mathcal{K}_j$ for $\mathcal{C}$ between columns j and $j+1$, $1 \le j \le \mathrm{cols}(P)$, are constructed as follows by induction over the length m of $\mathcal{C}$:

(1) For $m = 0$, $\mathcal{K}_j := \emptyset$ for all $1 \le j \le \mathrm{cols}(P)$.
(2) Assume that $\mathcal{C} = (c_0, c_1, \dots, c_{m-1}, c_m)$, and let $\mathcal{K}'_j$ be the horizontal crossing sequence for $\mathcal{C}' = (c_0, c_1, \dots, c_{m-1})$ between columns j and $j+1$ for all $1 \le j \le \mathrm{cols}(P)$. Consider the computation step of $\mathcal{M}$ that transforms c_{m-1} into c_m. Assume that during this step, the head of $\mathcal{M}$ moves from position (r,j) to $(r,j+1)$ or from position $(r,j+1)$ to position (r,j), where $1 \le r \le \mathrm{rows}(P)$, and assume that state q is entered during this step. Then $\mathcal{K}_j = \mathcal{K}'_j \circ (q,r)$, that is, the pair (q,r) is appended to the sequence $\mathcal{K}'_j$, and $\mathcal{K}_s = \mathcal{K}'_s$ for all $s \ne j$.

The horizontal crossing sequence $\mathcal{K}_j$ records all head movements of $\mathcal{M}$ between columns j and $j+1$. Each element of $\mathcal{K}_j$ records the state in which $\mathcal{M}$ performed the corresponding move, and the row number in which the head crossed the border between these two columns. Note that the notion of *vertical crossing sequence* could be defined analogously for crossings between neighbouring rows.

The following technical result is an immediate consequence of the above definition.

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

Proposition 3.5. *Let $\mathcal{M}$ be a* 2BTM *that accepts the pictures $P = P_1 \oplus P_2$ and $R = R_1 \oplus R_2$, where* $\mathrm{rows}(P) = \mathrm{rows}(R)$*, and 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 $\mathcal{M}$ also accepts the picture $P_1 \oplus R_2$.*

As the Sgraffito automaton is a specialization of the 2BTM, the notion of crossing sequence also applies to Sgraffito automata. In particular, Proposition 3.5 holds for Sgraffito automata, too.

As our first result on Sgraffito automata we show that, when restricted to one-row pictures, that is, strings, 2SAs accept exactly the class of regular (one-dimensional) languages. Actually, this result is only a slight generalization of a theorem by Hennie [8].

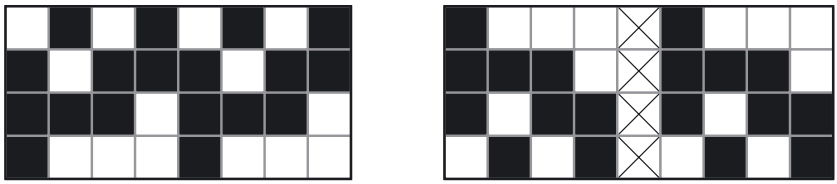
Theorem 3.6. *Let $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Q_F, \mu)$ be a* 2SA *that accepts a one-dimensional picture language $L(\mathcal{A}) \subseteq \Sigma^{1,*} = \Sigma^*$. Then $L(\mathcal{A})$ is a regular (string) language.*

Proof. As $\mathcal{A}$ is supposed to accept one-row pictures only, it is possible to eliminate all its vertical moves without changing the result of its computations. For example, for some $q, q' \in Q$ and $a, a' \in \Gamma$, if $\delta(q, a)$ contains the instruction (q', a', U), then we can replace it by the set of all instructions of the form (q'', a', Z), where $(q'', \top, \mathrm{D})$ is in $\delta(q', \top)$. Hence, we can assume that $\mathcal{A}$ does not make any vertical moves at all. Furthermore, we modify $\mathcal{A}$ in such a way that it can enter a final state only when returning from $\dashv$ to the rightmost field of the input (for a nonempty input).

We describe the construction of a nondeterministic finite-state automaton $\mathcal{B}$ for the language $\{\vdash\} \cdot L(\mathcal{A}) \cdot \{\dashv\}$. On input $\vdash w \dashv$, the automaton $\mathcal{B}$ guesses a computation of $\mathcal{A}$ on w by guessing all the horizontal crossing sequences between columns of $\widehat{w}$ and by checking whether the crossing sequences correspond to an accepting computation. If this computation is accepting, then $\mathcal{B}$ accepts, otherwise it rejects. The length of any horizontal crossing sequence of $\mathcal{A}$ 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 in the set of states of $\mathcal{B}$.

$\mathcal{B}$ 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. This sequence must consist of an even number of pairs (possibly zero). $\mathcal{B}$ continues as follows. Assume that $\mathcal{K}_s$ is the crossing sequence that corresponds to its current state. Then $\mathcal{B}$ reads the next input symbol a (in column j, say) and enters a state that corresponds to a nonempty crossing sequence $\mathcal{K}_{s'}$, which represents crossings of $\mathcal{A}$ between columns j and $j + 1$. The sequence $\mathcal{K}_{s'}$ must be consistent with the sequence $\mathcal{K}_s$ and the symbol a read. To check this, $\mathcal{B}$ guesses a sequence of instructions that are performed by $\mathcal{A}$ while visiting the jth column and verifies that the induced head movements match the sequences $\mathcal{K}_s$ and $\mathcal{K}_{s'}$. For $j = 1$,

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press



(a) (b)

FIGURE 3. Sample pictures from (a) L_{copy} and (b) $\hat{L}_{\text{copy}}$.

$\mathcal{B}$ knows that the first instruction must start in the initial state q_0, otherwise the state before applying an instruction is determined from $\mathcal{K}_s$ or $\mathcal{K}_{s'}$.

$\mathcal{B}$ will enter an accepting state after reading $a = \dashv$, if $\mathcal{K}_s$ is consistent with crossings between the last two columns of $\widehat{w}$, and if this sequence ends with a pair $(q_f, 1)$ for some $q_f \in Q_F$.

It is easily verified that the regular language $L(\mathcal{B})$ satisfies the equality $L(\mathcal{B}) = \{\vdash\} \cdot L(\mathcal{A}) \cdot \{\dashv\}$. As the class of regular languages is closed under left and right quotients [9], $L(\mathcal{A})$ is regular, too. □

We close this section with an important technical result that can be used to show that a picture language is not accepted by any Sgraffito automaton.

In order to derive this result, we consider the two-dimensional variants L_{copy} of the *copy language* and $\hat{L}_{\text{copy}}$ of the *marked copy language*. The former contains those pictures over $\Sigma = \{\square, \blacksquare\}$ that are of the form $U \Phi U$, where U is a square picture over Σ, and the latter is the picture language over $\hat{\Sigma} = \{\square, \boxtimes, \blacksquare\}$ that consists of all pictures $U \Phi C \Phi U$, where U is a square picture over Σ, and C is a column of symbols $\boxtimes$ (see Fig. 3). L_{copy} and $\hat{L}_{\text{copy}}$ are widely used examples of rather complicated picture languages [6].

The following technical result states that any sublanguage L of L_{copy} that is accepted by a 2SA can at most contain a certain number of pictures of size $n \times 2n$ for all sufficiently large values of n.

Lemma 3.7. *Let L be a picture language over Σ accepted by a* 2SA*, and let $f : 2\Sigma^{*,*} \times \mathbb{N} \to \mathbb{N}$ be the function that is defined by taking $f(L, n)$ to be the number of pictures in L of size $n \times 2n$. If $L \subseteq L_{\text{copy}}$, then $f(L, n) \in 2^{\mathcal{O}(n \log n)}$.*

Proof. Let $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Q_F, \mu)$ be a 2SA that accepts a language $L \subseteq L_{\text{copy}}$. Let $c = \max_{a \in \Sigma} \mu(a)$, and let $L(n)$ be the subset of L that consists of the pictures of size $(n, 2n)$ from L. Moreover, for $P \in L(n)$, let $\text{seq}(P)$ be the crossing sequence of $\mathcal{A}$ on P between columns n and $n + 1$ for some (arbitrarily chosen) accepting computation.

For a fixed value of n, we estimate the size of the set $\{\, \text{seq}(P) \mid P \in L(n) \,\}$. 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

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

is bounded from above by the number $2cn$. Hence, there are at most

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

many different crossing sequences.

Assume that there are two different pictures $P_1 = Q_1 \Phi Q_1$ and $P_2 = Q_2 \Phi Q_2$ in $L(n)$ such that $\mathrm{seq}(P_1) = \mathrm{seq}(P_2)$. By Proposition 3.5, this means that $\mathcal{A}$ also accepts on input $P_3 = Q_1 \Phi Q_2$. This, however, contradicts our choice of $\mathcal{A}$, as $P_3 \notin L_{\mathrm{copy}}$. It follows that $f(L,n) \leq |\{\, \mathrm{seq}(P) \mid P \in L(n) \,\}| \leq 2^{\mathcal{O}(n \log n)}$, which completes the proof. □

By using essentially the same proof it can be shown that this lemma also holds for $\hat{L}_{\mathrm{copy}}$.

Lemma 3.8. *Let L be a picture language over $\hat{\Sigma}$ accepted by a* 2SA*, and let $f : 2\Sigma^{*,*} \times \mathbb{N} \to \mathbb{N}$ be the function that is defined by taking $f(L,n)$ to be the number of pictures in L of size $n \times (2n+1)$. If $L \subseteq \hat{L}_{\mathrm{copy}}$, then $f(L,n) \in 2^{\mathcal{O}(n \log n)}$.*

As an immediate application of Lemmas 3.7 and 3.8 we obtain the following negative result, as L_{copy} ($\hat{L}_{\mathrm{copy}}$) contains 2^{n^2} many pictures of size $n \times 2n$ ($n \times (2n+1)$) for all $n \geq 1$.

Corollary 3.9. *L_{copy} and $\hat{L}_{\mathrm{copy}}$ are not accepted by any* 2SA.

4. Closure properties

Next we concentrate on the closure properties for the language classes that are accepted by 2SA and 2DSA. Our first result shows that both these classes are closed under the basic operations on pictures described above.

Theorem 4.1. *The language classes $\mathcal{L}$(2SA) and $\mathcal{L}$(2DSA) are closed under union, intersection, rotation, transposition, and mirroring.*

Proof. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two 2SAs, and let $L_1 = L(\mathcal{A}_1)$ and $L_2 = L(\mathcal{A}_2)$. We construct a 2SA $\mathcal{A}$ that simulates both, $\mathcal{A}_1$ and $\mathcal{A}_2$, in turn. The 2SA $\mathcal{A}$ starts its computation by simulating $\mathcal{A}_1$. While doing so, an extended alphabet is used such that, while $\mathcal{A}_1$ is being simulated, the original input letter is stored together with the current symbol produced by $\mathcal{A}_1$ in each cell. On reaching the end of a computation of $\mathcal{A}_1$, $\mathcal{A}$ returns to the cell at position $(1,1)$, and then it starts to simulate a computation of $\mathcal{A}_2$ on the same input picture, using the original input letter stored in each cell. For accepting the union $L_1 \cup L_2$, it is required that $\mathcal{A}$ accepts as soon as the computation of $\mathcal{A}_1$ or the computation of $\mathcal{A}_2$ it simulates accepts. For accepting the intersection $L_1 \cap L_2$, it is required that $\mathcal{A}$ accepts if and only if the computation of $\mathcal{A}_1$ and the computation of $\mathcal{A}_2$ simulated both accept.

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

For recognizing L_1^R, a 2SA is used that first moves its head to the top-right corner of the input, and that then simulates $\mathcal{A}_1$, treating columns as rows and *vice versa*. Analogously, for recognizing L_1^T, a 2SA is used that starts at the top-left corner of the input, and that simply interchanges move-right and move-left steps with move-down and move-up steps, respectively. 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 reversed order.

Finally, observe that the 2SAs $\mathcal{A}$ designed for the new languages are deterministic, if both $\mathcal{A}_1$ and $\mathcal{A}_2$ are deterministic. □

For the nondeterministic Sgraffito automata, we also have the following important closure properties.

Theorem 4.2. *The language class* $\mathcal{L}(\mathsf{2SA})$ *is closed under the operations of row and column concatenation, row and column closure, and projection.*

Proof. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two 2SAs, and let $L_1 = L(\mathcal{A}_1)$ and $L_2 = L(\mathcal{A}_2)$. To recognize the language $L_1 \oplus L_2$, a 2SA nondeterministically chooses a column of the input picture and marks it. Then it simulates $\mathcal{A}_1$ on the left-hand part (including the marked column) and, after that, it simulates $\mathcal{A}_2$ on the right-hand part (excluding the marked column). For the language $L_1 \ominus L_2$, the construction is similar in that a row is marked, and then $\mathcal{A}_1$ is simulated on the upper part and $\mathcal{A}_2$ on the lower part of the given input picture.

In order to recognize the language $L_1^{*\oplus}$, a 2SA repeats the above process of marking columns for a number of times chosen nondeterministically. Then it simulates $\mathcal{A}_1$ on each of the subpictures obtained in this way, proceeding from left to right. In addition, it must accept the empty picture λ. For the row closure $L_1^{*\ominus}$, the construction is similar.

Finally, let π be a projection. For an input picture P, a 2SA for accepting $\pi(L_1)$ simply guesses and stores a picture P' of the same size as P such that $\pi(P') = P$. Then it simulates $\mathcal{A}_1$ on P' and accepts, if and when $\mathcal{A}_1$ accepts. □

For the deterministic Sgraffito automaton, the following closure property is expected.

Theorem 4.3. *The language class* $\mathcal{L}(\mathsf{2DSA})$ *is closed under complement.*

Proof. A 2DSA $\mathcal{A}$ rejects an input if and only if it reaches a state q while scanning a symbol a such that $\delta(q, a)$ is empty. Since $\mathcal{A}$ is deterministic, we can easily modify it in order to obtain a 2DSA for accepting the complement of $L = L(\mathcal{A})$ by replacing all undefined steps by steps that enter a (new) final accepting state, and by turning all final states of $\mathcal{A}$ into nonfinal states. Obviously, the resulting 2DSA accepts the language $\overline{L} = \Sigma^{*,*} \setminus L$. □

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

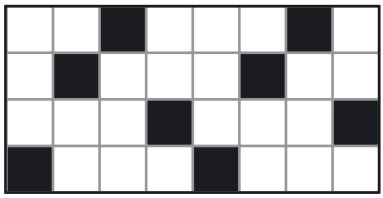


FIGURE 4. A sample picture from L_{perm}.

In the following we use the language L_{copy} considered before and its sublanguage L_{perm} to demonstrate some additional properties of Sgraffito automata. Variants of these languages were already introduced in [6,16]. The language of "permutations" L_{perm} is the subset of L_{copy} that consists of those pictures $Q \oplus Q$ for which each row and each column of Q contains the symbol ■ exactly once. An example is shown in Figure 4. Concerning these languages, the following results are known.

Proposition 4.4. [6,16]
L_{copy} and L_{perm} are not in REC*, while their complements are in* REC.

Since $\mathcal{L}(\text{2DSA})$ is closed under complement, and since $\hat{L}_{\text{copy}}$ is not accepted by any Sgraffito automaton (Cor. 3.9) we obtain the following corollary.

Corollary 4.5. *The language $\overline{L}_{\text{copy}}$ is not accepted by any* 2DSA.

However, we will see below that the language $\overline{L}_{\text{copy}}$ is accepted by a (non-deterministic) Sgraffito automaton, which will give us the following non-closure property.

Theorem 4.6. *The language class $\mathcal{L}(\text{2SA})$ is not closed under complement.*

Proof. We prove that $\overline{L}_{\text{copy}} \in \mathcal{L}(\text{2SA})$. To do so, we use the decomposition of $\overline{L}_{\text{copy}}$ given in [6]. Let $\Sigma = \{\square, ■\}$. We have $\overline{L}_{\text{copy}} = L_1 \cup L_2$, where

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

and

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

The language L_2 can further be expressed as

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

where

$$L_3 = \{\, P \in \Sigma^{*,*} \mid \text{cols}(P) = 2 \cdot \text{rows}(P) \,\},$$
$$L_4 = \{\, P \in \Sigma^{*,*} \mid \text{cols}(P) = \text{rows}(P) + 1 \,\}, \text{ and}$$
$$L_5 = \{\, P \in \Sigma^{*,*} \mid \text{rows}(P) = 1 \wedge P(1,1) \neq P(1, \text{cols}(P)) \,\}.$$

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

The language L_5 contains one-row pictures only, and it is regular. The set $\Sigma^{*,*}$ is trivially in $\mathcal{L}(\mathsf{2SA})$. The languages L_1, L_3, L_4 are recognizable by 2SAs which check the size conditions by scanning the input along the main diagonal. Thus, the closure properties of $\mathcal{L}(\mathsf{2SA})$ already established show that $\overline{L}_{\mathrm{copy}}$ is in $\mathcal{L}(\mathsf{2SA})$. □

In addition, we obtain the following non-closure results.

Theorem 4.7. *The language class $\mathcal{L}(\mathsf{2DSA})$ is neither closed under row nor under column concatenation.*

Proof. The languages $\Sigma^{*,*}, L_1, L_3, L_4, L_5$ from the proof above are all accepted by 2DSAs, and $\mathcal{L}(\mathsf{2DSA})$ is closed under union and intersection. However, by Corollary 4.5, $\overline{L}_{\mathrm{copy}} \notin \mathcal{L}(\mathsf{2DSA})$. It follows that $\mathcal{L}(\mathsf{2DSA})$ is not closed under both, row and column concatenation. Assume that it were closed under row concatenation. As it is closed under rotation, and as

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

closure under column concatenation would follow. Analogously, closure under column concatenation would imply closure under row concatenation. It follows that $\mathcal{L}(\mathsf{2DSA})$ is neither closed under row nor under column concatenation. □

We complete this section by still another non-closure result.

Theorem 4.8. *The language class $\mathcal{L}(\mathsf{2DSA})$ is not closed under projection.*

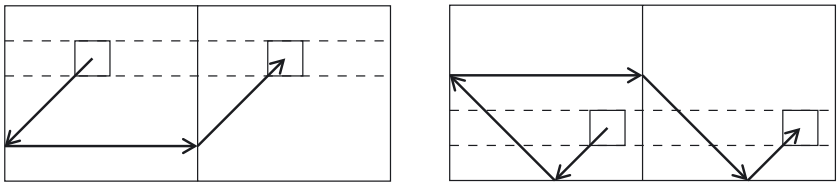
Proof. Let $\Sigma_1 = \{0, 1\}$, $\Sigma_2 = \{\overline{0}, \overline{1}\}$, $\Sigma = \Sigma_1 \cup \Sigma_2$, and let $\pi : \Sigma \to \{\square, \blacksquare\}$ be the mapping defined by $\pi(0) = \pi(\overline{0}) = \square$ and $\pi(1) = \pi(\overline{1}) = \blacksquare$. We define the language L_1 over Σ to consist 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, we define

$$L_2 = \{\, P \in \Sigma^{*,*} \mid \mathrm{cols}(P) \neq 2 \cdot \mathrm{rows}(P) \,\},$$

and we take $L = L_1 \cup L_2$. It is clear that $\pi(L) = \overline{L}_{\mathrm{copy}}$. To complete the proof we now construct a 2DSA $\mathcal{A}$ accepting L. Given some input picture P, $\mathcal{A}$ first checks the size of P. If the size is not of the form $(n, 2n)$, then $\mathcal{A}$ simply accepts, if, however, P is of size $(n, 2n)$, then $\mathcal{A}$ marks the last column of the left half of P, and it checks whether this half contains just a single symbol from Σ_2 (at a position (i, j)). In the affirmative, $\mathcal{A}$ marks the whole ith row as *working*, and moves its head back to position (i, j). Then it locates the corresponding tape field in the right half of P, at position $(i, n + j)$. For doing so, a bouncing traversal as shown in Figure 5 is performed until the working row is reached during the final phase of this movement. Finally, $\mathcal{A}$ checks whether $\pi(P(i, j)) \neq \pi(P(i, n + j))$. □

It currently remains open whether $\mathcal{L}(\mathsf{2DSA})$ is closed under the operations of row and column closure.

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press



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

FIGURE 5. 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

In this and the next sections we compare the 2SA and the 2DSA to various other models of two-dimensional automata from the literature. First we show that the four-way finite automaton (4FA) of Blum and Hewitt [2] can be simulated by the deterministic Sgraffito automaton.

Theorem 5.1. $\mathcal{L}(\mathsf{4FA}) \subseteq \mathcal{L}(\mathsf{2DSA})$.

Proof. Let $\mathcal{A} = (Q, \Sigma, q_0, q_a, q_r, \delta)$ be a 4FA, and let $P \in \Sigma^{*,*}$ be an input picture. With $\mathcal{A}$ and P we associate a directed graph $G = (V, E)$ as follows:

- $V = \{\, (i,j,q) \mid 1 \leq i \leq \mathrm{rows}(\widehat{P}), 1 \leq j \leq \mathrm{cols}(\widehat{P}), q \in Q \,\}$, that is, for each position (i,j) of $\widehat{P}$ and each state $q \in Q$, we have a vertex corresponding to the triple (i,j,q),
- $E = \{\, ((i_1,j_1,q_1),(i_2,j_2,q_2)) \mid (q_2,d) \in \delta(q_1, \widehat{P}(i_1,j_1)),$ and (i_2,j_2) is reached from (i_1,j_1) by a move according to $d \,\}$, that is, G contains an edge $(i_1,j_1,q_1) \to (i_2,j_2,q_2)$ iff $\mathcal{A}$ can move from position (i_1,j_1) to position (i_2,j_2) and change from state q_1 to q_2 on seeing the symbol $\widehat{P}(i_1,j_1)$.

Now $\mathcal{A}$ accepts on input P iff there is a vertex of the form (i,j,q_a) that is reachable from $(2,2,q_0)$ in G.

To check the latter property, it suffices to perform a depth-first search on G (see, *e.g.*, [4]). We present a corresponding procedure that labels nodes and edges that are visited. At each moment, each vertex is in one of two possible states – *unexplored* or *explored*, and each edge is in one of three possible states – *unexplored*, *discovery edge*, or *back edge*. All vertices and edges are initially in the unexplored state. Here is our procedure:

1: $v := (2,2,q_0)$,
2: label v as *explored*,
3: **if** v represents an accepting configuration **then**
4: **ACCEPT**

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

5: **end if**,
6: **if** there is an unexplored edge $e = (v, w)$ **then**
7: **if** w is unexplored **then**
8: label e as *discovery edge*, move to w, set $v := w$,
9: **goto** 5
10: **else**
11: label e as *back edge*,
12: **goto** 5
13: **end if**
14: **else if** there is an incoming discovery edge (u, v) **then**
15: move to u, set $v := u$,
16: **goto** 5
17: **end if**,
18: **REJECT**

The label of a vertex (i, j, q) and the labels of its outgoing edges are all recorded in the tape field storing $\widehat{P}(i, j)$. The only exceptions are the vertices on the border, as the corresponding cells cannot be rewritten. The labels corresponding to such a cell are stored in the nearest cell that belongs to the inner part P of $\widehat{P}$. As the number of outgoing edges of each vertex is bounded from above by the number $4 \cdot |Q|$, there are only a constant number of visits to each cell and a constant amount of information to be stored in each cell. It follows that the above algorithm can be realized by a 2DSA. □

Actually, this result can be strengthened by showing that it is even possible to simulate four-way alternating automata (4AFA) [16] by 2DSAs. A *four-way alternating automaton* is given by a 6-tuple $\mathcal{A} = (Q_\exists, Q_\forall, \Sigma, \delta, q_0, Q_F)$, where the set of states is split into $Q_\exists$ (*existential states*) and $Q_\forall$ (*universal states*). The set of final states Q_F is a subset of $Q_\exists \cup Q_\forall$, and the transition relation δ assigns a finite subset of $(Q_\exists \cup Q_\forall) \times \mathcal{H}$ to each pair of the form $(q, a) \in (Q_\exists \cup Q_\forall) \times (\Sigma \cup \mathcal{S})$. Recall that $\mathcal{H} = \{\mathrm{R}, \mathrm{L}, \mathrm{D}, \mathrm{U}, \mathrm{Z}\}$ is the set of all possible head movements.

Theorem 5.2. $\mathcal{L}(\mathsf{4AFA}) \subsetneq \mathcal{L}(\mathsf{2DSA})$.

Proof. Let $\mathcal{A} = (Q_\exists, Q_\forall, \Sigma, \delta, q_0, Q_F)$ be a 4AFA, and let $P \in \Sigma^{m,n}$ be an input picture. As in the previous proof, we define a directed graph $G = (V, E)$ representing all possible transitions of $\mathcal{A}$ as follows:

- $V = \{1, \ldots, m\} \times \{1, \ldots, n\} \times (Q_\exists \cup Q_\forall)$;
- $((i_1, j_1, q_1), (i_2, j_2, q_2))$ is an edge in E iff $(q_2, d) \in \delta(q_1, P(i_1, j_1))$, where (i_2, j_2) is the coordinate reached from (i_1, j_1) by moving into the direction given by d.

Special care is required for transitions that visit the borders of $\widehat{P}$. In these cases we represent the composition of two consecutive transitions by a single edge (the head must not finish on the border).

Let $\rho : V \to \{0, 1\}$ be the function that expresses the result (0 – reject, 1 – accept) of the computation of $\mathcal{A}$ over P if started at position (i, j) in state q, that

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

is, $\rho(i,j,q) = 1$ if the computation of $\mathcal{A}$ over P that starts in state q at position (i,j) accepts, and $\rho(i,j,q) = 0$, if the corresponding computation does not accept. The values of ρ can be computed recurrently by the following rules:

(1) if $q \in Q_F$, then $\rho(i,j,q) = 1$ for all admissible values of i and j;
(2) if $q \in Q_\exists \setminus Q_F$, then $\rho(i,j,q) = 1$ iff there is an edge from (i,j,q) to some (i',j',q') such that $\rho(i',j',q') = 1$;
(3) if $q \in Q_\forall \setminus Q_F$, then $\rho(i,j,q) = 1$ iff for each edge from (i,j,q) to some (i',j',q'), $\rho(i',j',q') = 1$.

Obviously, $\mathcal{A}$ accepts on input P iff $\rho(1,1,q_0) = 1$. We design a computation that computes ρ for each vertex in V, applying the above rules. The values of ρ are initially set to 1 only for the vertices in the set V_I that is defined as follows: $v = (i,j,q) \in V_I$ iff $q \in Q_F$ or $q \in Q_\forall$ and there is no outgoing edge from v. Then a depth-first search is performed on the reversion of G. It is started at the vertices in V_I. Moreover, it continues through outgoing edges of a vertex v only if $\rho(v)$ has been set to 1. As in the previous proof, a 2DSA can be designed for this task. It follows that $\mathcal{L}$(4AFA) is contained in $\mathcal{L}$(2DSA).

Since $\mathcal{L}$(4AFA) is not closed under complement [16], while $\mathcal{L}$(2DSA) is by Theorem 4.3, it follows that the inclusion above is proper. □

The simulations used in the proofs above are based on representing a computation of a 4FA or a 4AFA on a given input picture as a directed graph and traversing this graph by a 2DSA in the depth-first manner (DFS). The general principles shared by these constructions can be stated as follows.

Let $G = (V,E)$ be a directed graph that satisfies the following conditions:

(1) $V \subseteq \{1,\ldots,m\} \times \{1,\ldots,n\} \times U$ for some integers m, n and a finite set U.
(2) For every edge $((i_1,j_1,u_1),(i_2,j_2,u_2))$ in E, $|i_1 - i_2| + |j_1 - j_2| \leq 1$.

The graph G can be represented in a picture P of size $m \times n$, where the field at a position (i,j) records the vertices of the form (i,j,u) $(u \in U)$ in V and the outgoing edges of these vertices. Since the edges only go to the vertices represented in the field itself and in its neighbouring fields, it is only necessary to represent $\mathcal{O}(|U|)$ many vertices and edges in each tape field.

Assume that a 2DSA has created a representation of G and moved its head back to the initial position. To traverse G, it assigns a status to vertices as well as to edges. Initially, each vertex has status *unexplored*. When visiting a vertex v during DFS for the first time, its status is changed to *open*, and when DFS backtracks to v, since the whole subtree rooted at v has been searched, then its status is set to *explored*. Analogously, each edge e has initially the status *unexplored*. This changes when e is being traversed. If it leads to an *unexplored* vertex, its status is set to *discovery*, otherwise its status is set to *back*. The edges with status *discovery* will form DFS-trees at the end of the search. The search is now implemented as follows:

(1) If there is no vertex with status *unexplored* at the field scanned, go to Step (3).

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

(2) While there is a vertex v with status *unexplored* (possibly fulfilling some additional requirement), mark it as *open* and start DFS, which will return to v at the end.
(3) If not all the fields have been scanned, move the head to the next field in the row or to the first field of the next row when the right border is reached. Continue with Step (1).

In our constructions above, U corresponds to the set of states of an automaton to be simulated. The requirement mentioned in Step (2) may select, for example, source nodes only (vertices without incoming edges). The whole process visits and rewrites each tape field $\mathcal{O}(|U|)$ many times. Hence, it can be realized by a 2DSA.

For example, a simple application of DFS would be to check whether the black pixels within a black and white picture form just a single connected component. Actually, this can already be checked by a deterministic one-marker four-way automaton (see Sect. 6).

We continue by giving another application of this technique. The *sudoku-deterministically recognizable picture languages* (SDREC) were introduced by Borchert and Reinhardt in [3]. They are defined *via* domino tiling systems [6] and the sudoku-deterministic process. A domino tiling system $\mathcal{T} = (\Sigma, \Gamma, \Delta, \pi)$ is specified by a set of dominoes Δ (that is, pictures over $\Gamma \cup \mathcal{S}$ of sizes 1×2 and 2×1) and a projection $\pi : \Gamma \to \Sigma$. These systems can be used to define picture languages in REC that are projections of local languages whose tiles all match dominoes in Δ.

The sudoku-deterministic process follows a different approach. Given a picture $P \in \Sigma^{m,n}$, it is initialized by the picture S_P of the same size in which each position (i,j) stores the set $S_P(i,j) := \pi^{-1}(P(i,j)) \in 2^\Gamma$. These preimages are then reduced iteratively in the same manner as a Sudoku-puzzle is solved. In one step, all those symbols that do not conform locally to the set of allowed dominoes are discarded from a given position. Formally, for $S, S' \in (2^\Gamma)^{m,n}$, S is reduced to S' if

$$S'(i,j) = \{\, x \in S(i,j) \mid \exists y_1, y_2, y_3, y_4 \in \Gamma \cup \mathcal{S} : y_1 \in \widehat{S}(i+1,j), y_2 \in \widehat{S}(i-1,j),$$
$$y_3 \in \widehat{S}(i,j+1), y_4 \in \widehat{S}(i,j-1), \text{and } \boxed{x}\boxed{y_1}, \boxed{y_2}\boxed{x}, \begin{array}{|c|}\hline x \\ \hline y_3 \\ \hline \end{array}, \begin{array}{|c|}\hline y_4 \\ \hline x \\ \hline \end{array} \in \Delta \,\}$$

for all positions (i,j). The accepted picture language is the set of all pictures P for which there exists a way to transform the initialized picture S_P in finitely many steps into a picture in which every position consists of exactly one element and which cannot be transformed further.

Theorem 5.3. SDREC $\subseteq \mathcal{L}$(2DSA).

Proof. Let L be a language in SDREC, and let $\mathcal{T} = (\Sigma, \Gamma, \Delta, \pi)$ be a domino tiling system for L. To design a 2DSA $\mathcal{A}$ accepting L, we benefit from the known fact that the result of a sudoku-deterministic process does not depend on the order in

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

which particular symbols are discarded. Let $P \in \Sigma^{m,n}$ be an input picture. Then $\mathcal{A}$ simulates the sudoku-deterministic process as follows:

(1) For each pair (i,j), $\mathcal{A}$ initializes $S(i,j)$ by the set $S(i,j) := \pi^{-1}(P(i,j))$.
(2) $\mathcal{A}$ moves its head row by row. Every visited position (i,j) is marked as 'reached'. $\mathcal{A}$ checks whether some elements in $S(i,j)$ can be discarded by the sudoku-deterministic process – this is decided based on the content of the neighbouring fields.
(3) If at least one element is discarded, then $\mathcal{A}$ searches all those neighbouring fields that have already been marked as 'reached'. It checks whether further symbols can be discarded at these fields, and then it repeats this process for the corresponding neighbours. Actually, this process can be realized by an extended version of the DFS algorithm. $\mathcal{A}$ visits neighbouring fields marked as 'reached' using the depth-first principle, but when returning from some field (i',j') to its predecessor (i,j) in the depth-first tree, $\mathcal{A}$ checks again whether some element at this field must be discarded. If so, then the corresponding elements are discarded, and the DFS algorithm is restarted at this field, again visiting the (already visited) neighbours of the field (i,j) again. After the extended DFS finishes, $\mathcal{A}$ returns to the last field marked as 'reached', and it continues with the row by row movement.
(4) When the simulation has been completed, then $\mathcal{A}$ revisits all fields in a sequential manner and accepts iff $|S(i,j)| = 1$ for all positions (i,j).

The simulation guarantees that all symbols that are discarded by the sudoku-deterministic process are also discarded by $\mathcal{A}$. Finally, let us determine how many times a field f is visited during Steps (2) and (3). When it is visited for the first time, then it is being marked as 'reached'. When the DFS is launched at f, then f is visited at most $8 \cdot |\Gamma|$ many times, as f has at most 4 neighbours, and each time a symbol is being discarded from one of these neighbours, two more visits to f may become necessary. And finally, f is visited at most $4 \cdot |\Gamma|$ times when the DFS is launched at other fields – a DFS contributes a visit to f only when a symbol is discarded from a neighbouring field, and this happens in any field at most $|\Gamma|$ many times. Hence, $\mathcal{A}$ visits every field only a bounded number of times. □

Borchert and Reinhardt proved that $\mathcal{L}(\mathsf{4AFA}) \subseteq \mathsf{SDREC}$. This means that Theorem 5.2 is a consequence of Theorem 5.3. However, the direct proof of Theorem 5.2 we presented above is simpler, and it nicely illustrates the power of the 2DSA. In Section 6 we will present still another application of the DFS implementation by 2DSAs, but first we want to relate the (deterministic) Sgraffito automata to the classes REC and DREC mentioned before.

Theorem 5.4. $\mathsf{REC} \subseteq \mathcal{L}(\mathsf{2SA})$, *and* $\mathsf{DREC} \subseteq \mathcal{L}(\mathsf{2DSA})$.

Proof. Let L be a language in REC. Then there exists a 2OTA $\mathcal{A}_1$ that accepts L. We can easily construct a 2SA $\mathcal{A}_2$ that simulates $\mathcal{A}_1$. This 2SA traverses its given input picture row by row, retrieves the information required to simulate a transition at each cell, and represents the state obtained in the corresponding tape

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

field. It nondeterministically branches whenever $\mathcal{A}_1$ 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 as shown in [1]. □

Actually the inclusions above are proper. In order to prove this, we consider the language L_{perm} which is not an element of REC (see Prop. 4.4).

Lemma 5.5. *L_{perm} is accepted by a* 2DSA.

Proof. We construct a 2DSA $\mathcal{A}$ that accepts L_{perm}. It starts by checking whether the given input picture P is of size $(n, 2n)$ for some $n \geq 1$. In the negative, it halts and rejects, while in the affirmative it marks the nth column. Next, it verifies whether the left half Q_1 and the right half Q_2 of P both represent permutations, that is, whether each of their rows and columns contains exactly one occurrence of the symbol ■. This is done by traversing P first row by row, and then again column by column.

If this condition is not satisfied, then $\mathcal{A}$ halts and rejects; otherwise it enters the second stage in which it compares Q_1 and Q_2 row by row. This in turn is done as follows. Consider row i for some i such that $1 \leq i \leq n$. First, all cells in this row are marked so that $\mathcal{A}$ will detect that it hits this row when moving across the current picture. Then the leftmost occurrence of the symbol ■ is located in this row. Assume that this symbol is in column j for some j such that $1 \leq j \leq n$. Now, $\mathcal{A}$ moves its head to the top of this column (that is, to coordinate $(1, j)$). Next, the field at coordinate $(1, n+j)$ is located using the bouncing style given by Figure 5a. Finally, the position $(i, n+j)$ is reached by moving the head down and stopping in row i, which is recognized because of the aforementioned markings. If at this position an occurrence of the symbol ■ is found, then this means that the content of row i of Q_1 is identical to the content of row i of Q_2. In this case, all markings are removed from row i, and this process is repeated at row $i+1$.

It remains to show that $\mathcal{A}$ visits each field of the picture P only constantly many times. During the first stage above, only constantly many traversals through P are performed. During the second stage, each iteration works in a unique row and column. Especially the uniqueness of the column $j \in \{1, \ldots, n\}$ ensures that different paths are always used to locate the tape field in the right half. Thus, only a constant number of visits to each field are required. It follows that the process described can indeed be realized by a 2DSA. □

From Proposition 4.4, Corollary 4.5, Theorem 5.4, and Lemma 5.5 we obtain the following results.

Corollary 5.6.

(a) $\mathsf{REC} \subsetneq \mathcal{L}(\mathsf{2SA})$.
(b) *The language classes* $\mathcal{L}(\mathsf{2DSA})$ *and* REC *are incomparable.*

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

In Section 9 we will strenghten this result by showing that it remains valid even in the setting of unary picture languages, that is, for picture languages over the unary alphabet $\Sigma = \{\square\}$.

6. Comparing the 2SA to one-marker automata

Together with the 4FA, Blum and Hewitt also introduced the *marker automaton*, which is a four-way finite automaton with the additional capability of using a bounded number of *markers*. In fact, Blum and Hewitt considered two types of markers – *physical* and *abstract* markers. A physical marker can only be moved, that is, placed or removed, when the head of the automaton is at the corresponding position, while an abstract marker disappears from its previous position whenever it is being placed anew at the current position of the head. However, Blum and Hewitt proved that k physical markers can be simulated by k abstract markers and vice versa. Accordingly, we concentrate on physical markers, that is, to move a marker, it is necessary that the head first moves over the marker and then transfers the marker by a head movement to a neighbouring position. By $2\mathsf{M}_1\mathsf{A}$ we denote the class of four-way finite automata that are allowed to use a single marker, and the deterministic $2\mathsf{M}_1\mathsf{A}$ are denoted as $2\mathsf{DM}_1\mathsf{A}$. Note that $2\mathsf{M}_1\mathsf{As}$ behave on the boundaries of boundary pictures in just the same way as 2SAs. Our next result states that $2\mathsf{DM}_1\mathsf{As}$ can be simulated by 2DSAs.

Theorem 6.1. $\mathcal{L}(2\mathsf{DM}_1\mathsf{A}) \subseteq \mathcal{L}(2\mathsf{DSA})$.

Proof. Let $\mathcal{M}$ be a $2\mathsf{DM}_1\mathsf{A}$, let Q be its set of states, let $q_0 \in Q$ be its initial state, and let P be an input picture of size $m \times n$. W.l.o.g. we can assume that $\mathcal{M}$ is allowed to halt (and accept or reject) only when it scans the marker and when it does not move its head. In all other cases, it always performs an instruction.

We now construct a 2DSA $\mathcal{A}$ that accepts the language $L(\mathcal{M})$. The key idea of this construction is to design $\mathcal{A}$ in such a way that it detects all situations in which $\mathcal{M}$ leaves the marker, moves across some tape fields (without the marker), and returns back to the marker. We write $(i,j,q) \circlearrowleft_{\mathcal{M}} q'$, when there is such a traversal of $\mathcal{M}$ starting at position (i,j) in state q and ending at the same position in state q'. We define a directed graph $G_M = (V_M, E_M)$, where $V_M = \{1,\ldots,m\} \times \{1,\ldots,n\} \times Q$. We represent $(i,j,q) \circlearrowleft_{\mathcal{M}} q'$ in G_M by the edge $((i,j,q),(i,j,q'))$. Moreover, for every configuration in which $\mathcal{M}$ scans the marker and performs a transition that either moves the marker or does not move the head at all, we add an edge representing this transition to E_M.

Suppose G_M has been created. Since $\mathcal{M}$ is deterministic, the outgoing degree of each vertex is at most one. $\mathcal{A}$ can easily detect whether $\mathcal{M}$ accepts or not. It starts at $(1,1,q_0)$ and follows the path given by outgoing edges. The path either ends in some (i,j,q) or it closes a loop onto itself (this can be detected by marking the vertices traversed). $\mathcal{A}$ accepts in the former case if $q \in Q_F$.

To construct G_M, we define an auxiliary graph $G = (V,E)$ as follows. The vertices are tuples of the form (i,j,q,r), where (i,j) is a position, $q \in Q$, and

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

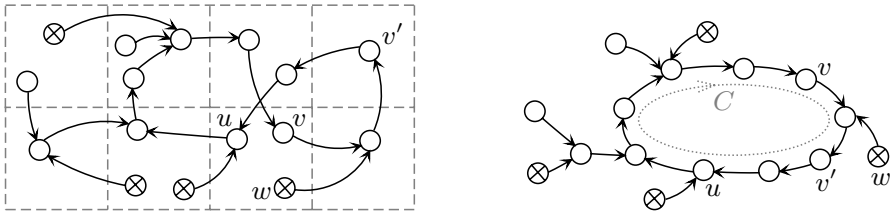


FIGURE 6. An example of a component of the graph G, displayed over a portion of the tape (a), and also as an isomorphic graph without the tape (b). The vertices ∘ and ⊗ are from V_1 and V_2, respectively. If the DFS on G^R is started at v and it reaches w, the stack in the corresponding field contains u and v, where u is the topmost element, and thus, the directed path from w to u in G is detected. If the DFS is started at v' and reaches in two steps w, the same path is found again, since u is contained in the walk formed of the path from w to v' and the cycle C.

$r \in \{0,1\}$. The bit r indicates the presence of the marker (0 – not present, 1 – present) at the time a transition is performed over the field (i,j) from state q. We define V as the union of two subsets:

(1) V_1 contains each vertex (i,j,q,r) where $r = 0$;
(2) V_2 contains $(i,j,q,1)$ iff $\mathcal{M}$ in state q scanning the tape field (i,j) with the marker makes a transition moving to a neighbouring field without moving the marker.

Due to our assumption on computations of $\mathcal{M}$ above, each configuration given by $v = (i,j,q,r) \in V$ induces one transition. Let the transition move the head to a position (i',j') and let it change the state to q'. Then, this transition is represented by adding the edge $((i,j,q,r),(i',j',q',0))$ to E. Each vertex in V_2 is thus a source (with no incoming edges). Since each vertex has one outgoing edge, each component of G has the form depicted in Figure 6(b): there is one directed cycle and possibly several trees rooted in vertices of this cycle.

If there is a directed path in G from $v_1 = (i,j,q,1) \in V$ to $v_2 = (i,j,q',0) \in V$, and if this path does not contain any other vertex of the form $(i,j,q'',0)$, then $(i,j,q) \circlearrowleft_{\mathcal{M}} q'$. Note that the indicator bit in v_2 is 0 even though following the path the head of $\mathcal{M}$ will return to the marker. This is intensional for technical reasons – v_2 is the last element of the path considered, but it can be an inner element of other paths which start in different configurations. We want to represent both these situations just by a single vertex of G. Conversely, if $(i,j,q) \circlearrowleft_{\mathcal{M}} q'$, then G necessary contains a directed path with the aforementioned properties.

Let C be the cycle of a component and let v be one of its vertices. Consider the DFS on G^R, the reversion of G, starting at v, and let $w = (i,j,q,1) \in V_2$. Assume that the DFS detects a directed path from v to w in G^R, *i.e.*, a directed

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

path from w to v in G (all the vertices of the path have currently the status *open*), see Figure 6(a). Take the walk in G created by prolonging the path by adding the cycle C after v. Now, if $(i,j,q) \circlearrowleft_{\mathcal{M}} q'$, then $(i,j,q',0)$ is a vertex contained in this walk. This observation helps us to design an algorithm for $\mathcal{A}$ for finding all these situations. We use a stack $T(i,j)$ in each tape field (i,j) to record the order of vertices (from the tape field (i,j)) that appear in the walks described. This information is important to guarantee that no directed path in G that passes through the starting tape field multiple times will be used to produce an edge in G_M. This algorithm proceeds as follows:

(1) Choose a vertex in G, follow its outgoing edges (and mark them) until a directed cycle C is detected. Assume this happens at a vertex v. Mark v.
(2) Traverse back through C until v is encountered again. During this process, push every vertex visited at a tape field (i,j) onto $T(i,j)$.
(3) Starting at v, perform a DFS on G^R. Whenever a vertex $u = (i,j,q,r)$ is visited and its status changes to *open*, push it onto $T(i,j)$. When the status is changed to *explored*, pop the vertex from $T(i,j)$. Whenever a vertex $w = (i,j,q,1) \in V_2$ is reached, check the topmost element in $T(i,j)$ (before pushing w onto it). This is the only vertex in which the directed path in G starting at w ends, without passing more vertices at the same tape field. If the topmost element is $u = (i,j,q',0)$, then $(i,j,q) \circlearrowleft_{\mathcal{M}} q'$ (hence, add the induced edge to G_M).
(4) Return to Step (1) to find the next component of G.

The number of operations performed over each stack is bounded by a constant, since the corresponding field is visited constantly many times. Hence, all the states of a stack $T(i,j)$ can be represented in the tape field (i,j). It follows that the above computation can indeed be realized by the 2DSA $\mathcal{A}$. □

In order to prove that the inclusion in Theorem 6.1 is proper, we use a theorem of Hsia and Yeh [10]. Here a *two-way k-marker finite automaton* (working over strings) is denoted as $\mathsf{M_kA}$, while its deterministic variant is denoted as $\mathsf{DM_kA}$.

Proposition 6.2. [10] *Let $n \geq 1$, let L be a language over Σ that is accepted by an always halting $\mathsf{DM_nA}$, but not by any $\mathsf{DM_{n-1}A}$, and let $\rho(L,\$)$ be the following language over $\Sigma \cup \{\$\}$, where $\$ \notin \Sigma$:*

$$\rho(L,\$) = \{\, w_1\$w_2\$\dots\$w_m \mid m \geq 2,\ |\{\, i \mid w_i \in L \,\}| = |\{\, j \mid w_j \notin L \,\}| \,\}.$$

Then $\rho(L,\$)$ is accepted by a $\mathsf{DM_{n+1}A}$, but it is not accepted by any $\mathsf{DM_nA}$.

For a string $w \in \Sigma^*$ and a symbol $a \in \Sigma$, let $S(w,a)$ denote the square picture over Σ of size $|w| \times |w|$, where the first row contains the word w and all remaining fields contain the symbol a. Moreover, for a (string) language $L \subseteq \Sigma^*$, let $S(L,a)$ be the picture language defined as $S(L,a) = \{\, S(w,a) \mid w \in L \,\}$.

Lemma 6.3. *Let L be a string language over Σ, and let $a \in \Sigma$. If the picture language $S(L,a)$ is accepted by a $\mathsf{2DM_1A}$, then L is accepted by a $\mathsf{DM_4A}$.*

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

Proof. Let $\mathcal{M}_1$ be a $\mathsf{2DM_1A}$ accepting $S(L, a)$. We design a $\mathsf{DM_4A}$ $\mathcal{M}_2$ that simulates $\mathcal{M}_1$ as follows. Let the four markers of $\mathcal{M}_2$ be denoted as K_1, K_2, K_3, and K_4. A configuration of $\mathcal{M}_1$ is identified by the marker position (i_M, j_M), the head position (i_H, j_H), and the state of the finite control. $\mathcal{M}_2$ uses K_1 and K_2 to represent the marker position by placing K_1 on the i_Mth field of the tape and K_2 on the j_Mth field. Analogously, the head position is represented by K_3 and K_4, while the state of $\mathcal{M}_1$ is stored within the state of $\mathcal{M}_2$. Initially, all the markers K_i are placed at the leftmost tape field. $\mathcal{M}_2$ now simulates $\mathcal{M}_1$ step by step. The symbol scanned by $\mathcal{M}_1$ is retrieved from the tape iff its head is located in the first row (*i.e.*, K_3 is at the first tape field), otherwise $\mathcal{M}_1$ is known to scan the symbol a. Finally, $\mathcal{M}_2$ accepts, if and when $\mathcal{M}_1$ enters a final state. □

Based on Proposition 6.2 we now construct a string language that is not accepted by any $\mathsf{DM_4A}$. Let

$$L_1 = \{\, w \in \{0,1\}^* \mid |w|_0 = |w|_1 \,\},$$

which is easily seen to be accepted by an always halting $\mathsf{DM_2A}$, but which is not accepted by any $\mathsf{DM_1A}$, as one-marker finite automata only accept regular (string) languages [2]. Next we define L_2 to L_4 by taking

$$L_2 = \rho(L_1, \$_1), L_3 = \rho(L_2, \$_2), \text{ and } L_4 = \rho(L_3, \$_3),$$

where $\$_1$, $\$_2$, and $\$_3$ are auxiliary symbols used as separators. For each $1 \leq k \leq 4$, the language L_k is not accepted by any $\mathsf{DM_kA}$, and thus, L_4 is the language we are looking for. To illustrate these languages we give some examples:

$$\begin{aligned} 011\$_1 10\$_1 1001\$_1 00 &\in L_2, \\ u_1 = 01\$_1 1\$_2 101 &\in L_3, \\ u_2 = 01\$_1 1\$_2 100\$_1 01 &\notin L_3, \\ u_1\$_3 u_2 = 01\$_1 1\$_2 101\$_3 01\$_1 1\$_2 100\$_1 01 &\in L_4. \end{aligned}$$

Lemma 6.4. *The language $S(L_4, 0)$ is accepted by a* $\mathsf{2DSA}$.

Proof. We describe a $\mathsf{2DSA}$ $\mathcal{A}$ for $S(L_4, 0)$. Let P be an input picture over $\Sigma = \{0, 1, \$_1, \$_2, \$_3\}$. $\mathcal{A}$ can easily check whether P is a square picture in which all rows except the first one only contain the symbol 0.

Let R be the first row of P, and let x be a working symbol. The idea of the computation consists in iteratively evaluating membership of substrings in R in the languages L_i. For this the following algorithm is used, starting with $i = 1$:

(1) Check whether the consecutive substrings u from $\{0, 1, x\}^*$ in R (delimited by $\$_1$, $\$_2$, $\$_3$ or by the left or right sentinel, respectively) contain the same number of 0's and 1's. For each substring u, write the result into its leftmost field (1 – yes, 0 – no), and replace the content of the other fields of u by x.
(2) If $i < 4$, rewrite every occurrence of the symbol $\$_i$ in R into x, increase i by 1, and return to Step (1).

Figure 7(a) illustrates the stages of this iteration on input $u_1\$_3 u_2$.

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

0	1	$\$_1$	1	$\$_2$	1	0	1	$\$_3$	0	1	$\$_1$	1	$\$_2$	1	0	0	$\$_1$	0	1
1	x	x	0	$\$_2$	0	x	x	$\$_3$	1	x	x	0	$\$_2$	0	x	x	x	1	x
1	x	x	x	x	0	x	x	$\$_3$	1	x	x	x	x	1	x	x	x	x	x
1	x	x	x	x	x	x	x	x	0	x	x	x	x	x	x	x	x	x	x
1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

(a)

1	0	0	1	0	0	0	1	1	0
x	x	0	1	0	0	0	1	1	0
x	x	x	x	0	0	0	1	1	0
x	x	x	x	x	0	0	x	1	0
x	x	x	x	x	x	0	x	x	0

(b)

FIGURE 7. (a) The first row is the input, the other rows show the result after each of the four iterations. (b) A comparison of the number of 0's and 1's in the string 1001000110. Each partial product is written into a new row. The last row indicates that there are more 0's than 1's.

It remains to show how $\mathcal{A}$ makes the evaluation in Step (1). Consider a general situation when R stores a string over $\Sigma \cup \{x\}$. $\mathcal{A}$ detects the consecutive blocks of symbols in $\{0, 1, x\}$ by moving its head from left to right across R. Assume that $R' \subseteq R$ stores such a complete block u. Then $\mathcal{A}$ uses the space below this block to compare the numbers of 0's and 1's. First, it copies u to the row below. Then it scans this copy, replacing one occurrence of the symbol 0 and one occurrence of the symbol 1 by x. This subroutine is repeated until there is no occurrence of 0 or of 1 left. At this point $\mathcal{A}$ detects whether the numbers match or not. An evaluation of the string 1001000110 is demonstrated in Figure 7(b).

Since there are four iterations in the main algorithm, and since, in each of them, each tape field is visited only constantly many times, the above algorithm can be realized by the 2DSA $\mathcal{A}$. □

From the choice of L_4 and from Lemma 6.3 we see that $S(L_4, 0)$ is not accepted by any $2DM_1A$. This yields the following improvement of Theorem 6.1.

Corollary 6.5. $\mathcal{L}(2DM_1A) \subsetneq \mathcal{L}(2DSA)$.

So far we have seen that the 2DSA is rather expressive, as it can simulate quite a few other deterministic types of two-dimensional automata. In the next section, however, we present a proper upper bound for the expressive power of the 2DSA.

7. Comparing the 2DSA to the forgetting automaton

The *two-dimensional deterministic forgetting automaton* (2DFA) was introduced by Jiřička and Král in [15]. It is a bounded two-dimensional Turing machine that

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

is allowed to rewrite by only using a special symbol @. In comparison to the 2DSA, there is no bound on the number of visits to any tape field. Here we study the question of whether the larger number of possible rewrites is an advantage for the 2DSA over the 2DFA. We will see, however, that this is not the case. On the contrary, we show that the 2DFA is more powerful than the 2DSA.

Jiřička and Král described a technique of how a 2DFA can store information on its tape while still remembering its original content. Let $P \in \Sigma^{*,*}$ be an input, let M be the set of tape fields storing P, and let $B \subseteq M$ be a rectangular subarea of M. Divide B into two parts, say H and G, where H consists of the $|\Sigma|$ first fields of B, and $G = B \setminus H$. For $a \in \Sigma$, let $G(a)$ denote the subset of those fields of G that initially contain the symbol a. Choose $a \in \Sigma$ such that $|G(a)| \geq |G|/|\Sigma|$. The idea is to use $G(a)$ to record information by erasing some of its fields, while memorizing which a has been chosen in order to be able to reconstruct the original content. Suppose a is the ith symbol of Σ. Erase the ith field of H to indicate this, and represent the erased symbol in unary using the first $|\Sigma|$ fields of $G(a)$. All the other fields of $G(a)$ can each store one bit of information: it is either erased or not erased. Hence, the total number of available bits is

$$|G(a)| - |\Sigma| \geq \frac{|B| - |\Sigma|}{|\Sigma|} - |\Sigma|.$$

Theorem 7.1. $\mathcal{L}(\text{2DSA}) \subsetneq \mathcal{L}(\text{2DFA})$.

Proof. We first show how a 2DSA $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Q_F, \mu)$ can be simulated by a 2DFA $\mathcal{F}$. Let P be an input picture, let M be the set of fields storing P, and let n be an integer, dependent on $\mathcal{A}$, whose value will be determined later.

We split M into blocks of size $n \times n$, except for those blocks neighbouring the right border or the bottom – their size could be up to $2n - 1$. If the height or the width of P is smaller than n, then the blocks will be only as high or as wide as the picture. In case that both, the height and the width of P are smaller than n, $\mathcal{F}$ can decide whether to accept or to reject P simply by table look-up. Figure 8 summarizes the non-trivial variants.

Note that $\mathcal{F}$ can detect the borders between blocks by counting modulo n for both, the number of movements in the vertical and the horizontal directions.

The construction is now done as follows. $\mathcal{F}$ simulates $\mathcal{A}$ in the top-left block. All symbols in the block are scanned and memorized in states first, then $\mathcal{F}$ determines whether $\mathcal{A}$ finishes its computation inside the block or whether it enters one of its neighbouring blocks, say block B. In the latter case $\mathcal{F}$ writes to B, in which state and in which position B was entered. Then it again simulates $\mathcal{A}$ within this block and so on. The information on entry points to a block (ordered by time) is sufficient to reconstruct its content. There are at most $8n$ positions on the border of a block, thus a position can be encoded in binary using $\lceil \log_2 8n \rceil$ bits, while a state is encoded using $\lceil \log_2 |Q| \rceil$ bits. Any (border) field in M is entered at most $|\Gamma|$ times. This implies that $\mathcal{O}(n \log n)$ bits are needed. We have seen above that the capacity of each block is $\Theta(n^2)$ bits. Thus, a suitable value of n fulfilling the

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

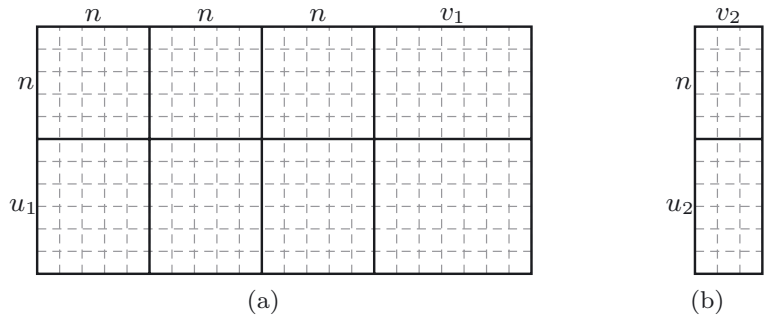


FIGURE 8. Decomposition of M into blocks of fields, when (a) both dimensions of M are at least n, or (b) the width v_2 is smaller than n.

memory requirement can be found. When the height or the width of P is only $m < n$, then all blocks are entered across the borders of length m, which means that only $\mathcal{O}(m \log m)$ bits are needed, while the capacity of the block is $\mathcal{O}(mn)$. Thus, also in this case a corresponding value for n exists.

Finally, it is known that 2DFAs working over strings accept the deterministic context-free languages [14]. This proves that the above inclusion is proper, as 2DSAs only accept the regular string languages. □

8. Comparing the nondeterministic models

In Section 6 we have seen that the deterministic Sgraffito automaton, 2DSA, is strictly more expressive than the deterministic one-marker automaton, $2DM_1A$. Here we study the question of whether a corresponding result also holds for the nondeterministic Sgraffito automaton. As we will see, however, this is not the case. In fact, we will see that the language class $\mathcal{L}(2M_1A)$ is not contained in the class $\mathcal{L}(2SA)$.

In order to derive this result, we make use of the picture languages $\hat{L}_{\text{copy}}$ defined at the end of Section 3 and its sublanguage L_{cols}, which contains all those pictures $U \oplus C \oplus U$ in which each column of U contains exactly one black pixel ■ (see Fig. 9). Already in [13] Ito *et al.* noticed that the language $\hat{L}_{\text{copy}}$ is accepted by a $2M_1A$. Here we give our own proof, as we use the same strategy to accept L_{cols} by a $2DM_1A$.

Lemma 8.1. *$\hat{L}_{\text{copy}}$ is accepted by a $2M_1A$.*

Proof. A $2M_1A$ $\mathcal{M}$ can accept $\hat{L}_{\text{copy}}$ as follows. Let P be an input picture. First, $\mathcal{M}$ checks whether P is of size $n \times (2n + 1)$: it starts at the top-left corner, and repeatedly moves its head two fields to the right and one field down. Next it locates the topmost field of the central column which is detected by $\mathcal{M}$ when

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

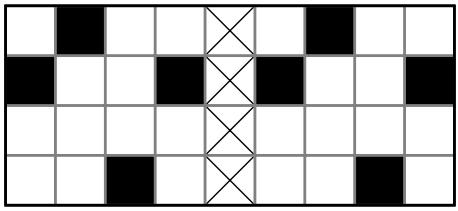


FIGURE 9. A picture of size 4×9 from L_{cols}.

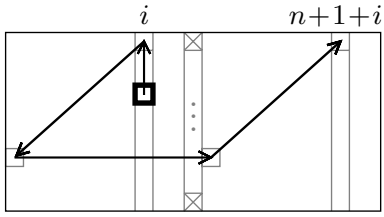


FIGURE 10. Finding the corresponding column in the second half of a picture of size $n \times (2n+1)$. The walk is a composition of vertical, horizontal and diagonal moves.

moving diagonally from the bottom-right corner, and then it checks whether the central column only contains the symbol ⊠, and whether the other columns do not contain this symbol, which is done by scanning P column by column.

If all these tests succeed, then P is of the form $U \oplus C \oplus V$, where U and V are square pictures over $\{\square, \blacksquare\}$ and C is a column of ⊠'s. Next $\mathcal{M}$ moves its head to the top-left corner. The marker is still at this position as it has not yet been moved. Now the following algorithm is used to check whether $U = V$:

(1) Let the marker be placed at a position (i, j). Memorize the symbol $U(i, j)$.
(2) Leaving the marker at this position, use the movement depicted in Figure 10 to reach position $(1, n+1+i)$ (*i.e.*, the topmost field of column i of V).
(3) Move downwards until the border is detected. During this movement, whenever $V(r, j) = U(i, j)$ for a row r, then there is the nondeterministic choice of moving leftwards along row r. If the marker is reached, then $r = i$ and $U(i, j) = V(i, j)$, otherwise this branch of the computation fails.
(4) If the marker is reached in Step (3), then the marker is moved to the neighbouring field in the same row. If the symbol ⊠ is scanned, then the marker is moved to the leftmost field of the next row. If there is no next field of U, accept, otherwise continue with Step (1).

It follows that $\mathcal{M}$ accepts the language $\hat{L}_{\text{copy}}$. □

Using the same strategy a $\mathsf{2DM_1A}$ can be designed for the language L_{cols}. Observe that for accepting this language, the nondeterministic choices in Step (3) are

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

not needed, as each column only contains a single black pixel. Thus, we get the following result.

Lemma 8.2. *L_{cols} is accepted by a* $\mathsf{2DM_1A}$.

Based on Lemma 3.7 we have seen in Corollary 3.9 that $\hat{L}_{\text{copy}}$ is not accepted by any $\mathsf{2SA}$. An analogous counting argument applies to every language L in REC. The number of different placements of tiles along the border between two columns is $2^{\mathcal{O}(n)}$, which restricts the number of pictures in L of size $n \times 2n+1$ by $2^{\mathcal{O}(n)}$ [6]. As L_{cols} contains $n^n \in 2^{\mathcal{O}(n \log n)}$ pictures of size $n \times (2n+1)$, this implies that $L_{\text{cols}} \notin \mathsf{REC}$. Together with Lemma 8.1 and Lemma 8.2 this yields the following results.

Theorem 8.3. *$\mathcal{L}(\mathsf{2M_1A})$ is not contained in $\mathcal{L}(\mathsf{2SA})$, and $\mathcal{L}(\mathsf{2DM_1A})$ is not contained in* REC.

Thus, while $\mathcal{L}(\mathsf{2DM_1A})$ is a proper subclass of $\mathcal{L}(\mathsf{2DSA})$ (Cor. 6.5), Theorem 8.3 shows that $\mathcal{L}(\mathsf{2M_1A})$ is not contained in $\mathcal{L}(\mathsf{2SA})$. Does the converse inclusion hold? Below we relate this question to the problem of whether the nondeterministic space complexity class NL (nondeterministic logarithmic space) is properly contained in the time complexity class NP (nondeterministic polynomial time). For doing so, we first extend the simulation presented in Lemma 6.3 to nondeterministic marker automata and to square pictures for which the number of rows and columns is polynomial in the length of the input string. For $w \in \Sigma^*$, $a \in \Sigma$ and $k \in \mathbb{N}$, we denote by $S(w,a,k)$ the picture over Σ of size $|w|^k \times |w|^k$ that contains w in the first row, starting at column one. The remaining fields all contain the symbol a. For a string language $L \subseteq \Sigma^*$, let $S(L,a,k) = \{\, S(w,a,k) \mid w \in L \,\}$.

Lemma 8.4. *Let L be a string language over Σ, let $a \in \Sigma$, and $k \in \mathbb{N}$. If the picture language $S(L,a,k)$ is accepted by a* $\mathsf{2M_1A}$, *then L is accepted by a* $\mathsf{M_{4k}A}$.

Proof. In comparison to the construction in the proof of Lemma 6.3, it is now necessary to represent the vertical and horizontal positions of the head and the marker, respectively, in the range from 1 to $|w|^k$. This can be done using k markers for each value, in this way representing each such value in the numeral system with base $|w|$. Hence, in total $4k$ markers are sufficient. □

The additional space provided by the picture language $S(L,a,k)$ now allows us to simulate a polynomial time-bounded one-dimensional Turing machine for L by a $\mathsf{2DSA}$.

Lemma 8.5. *Let L be a string language over Σ, and let $a \in \Sigma$. If $L \in \mathsf{P}$, then there is an integer $k \in \mathbb{N}$ such that the picture language $S(L,a,k)$ is accepted by some* $\mathsf{2DSA}$.

Proof. Let $\mathcal{T}$ be a deterministic Turing machine that accepts the language $L \subseteq \Sigma^*$ in time $t(n) \in \mathcal{O}(n^k)$. We can assume that the tape of $\mathcal{T}$ is infinite only to the

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

right. Moreover, we can assume that $t(n) \leq n^k$ for all $n \geq n_0$, where n_0 is a suitable integer.

We design a 2DSA $\mathcal{A}$ for accepting the picture language $S(L, a, k)$ as follows. Let P be an input picture. First, $\mathcal{A}$ verifies whether row one of P contains a word of the form wa^r for some $w \in \Sigma^*$ and $r \geq 0$, and whether all other rows only contain the symbol a. Secondly, it checks whether P is of the correct size $n^k \times n^k$, where $n = |w|$. Polynomials are recognizable functions [6], and so this check can be performed. If $|w| < n_0$, then $\mathcal{A}$ decides whether $\mathcal{T}$ accepts w simply by table-lookup. Otherwise, $\mathcal{A}$ simulates $\mathcal{T}$ on input w. For every $1 \leq i \leq n^k$, the ith configuration of the computation of $\mathcal{T}$ on input w is written into the ith row of P. There is sufficient space to represent all these configurations. Acceptance is then decided based on the last of these configurations. □

From this technical result we can now draw the following conclusion.

Theorem 8.6. *If* $\mathcal{L}(\mathsf{2DSA}) \subseteq \mathcal{L}(\mathsf{2M_1A})$*, then the complexity classes* NL *and* P *coincide.*

Proof. Let L_1 be a problem from the complexity class P. For some input symbol a and a suitable integer k, the picture language $L_2 = S(L_1, a, k)$ is accepted by a 2DSA (Lem. 8.5). Hence, if $\mathcal{L}(\mathsf{2DSA}) \subseteq \mathcal{L}(\mathsf{2M_1A})$, then L_2 is accepted by some $\mathsf{2M_1A}$. Thus, Lemma 8.4 implies that L_1 is accepted by an $\mathsf{M_{4k}A}$, which in turn yields that L_1 belongs to the complexity class NL. Thus, NL = P follows. □

Actually, Lemma 8.5 easily extends to the nondeterministic case.

Lemma 8.7. *Let L be a string language over Σ, and let $a \in \Sigma$. If $L \in$ NP, then there is an integer $k \in \mathbb{N}$ such that the picture language $S(L, a, k)$ is accepted by some* 2SA.

As above this yields the following consequence.

Theorem 8.8. *If* $\mathcal{L}(\mathsf{2SA}) \subseteq \mathcal{L}(\mathsf{2M_1A})$*, then the complexity classes* NL *and* NP *coincide.*

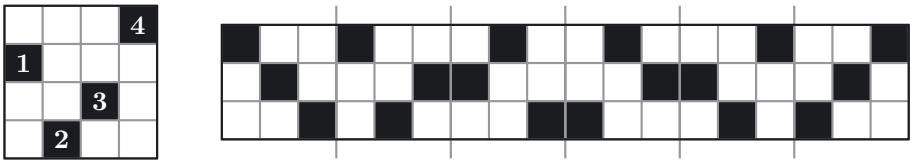
As it is widely believed that NL is a proper subclass of P, we obtain the following conjecture:

Conjecture: The classes of picture languages $\mathcal{L}(\mathsf{2DSA})$ and $\mathcal{L}(\mathsf{2SA})$ are incomparable to the language class $\mathcal{L}(\mathsf{2M_1A})$ with respect to inclusion.

9. Recognition of unary picture languages

Giammaresi and Restivo studied the problem of which functions can be represented by recognizable picture languages. Let $\Sigma = \{\Box\}$ denote a one-letter alphabet. A function $f : \mathbb{N} \to \mathbb{N}$ is called *representable* if the language $L(f) = \{ \Box^{n, f(n)} \mid n \in \mathbb{N} \}$ belongs to REC. A representable function cannot grow faster than an exponential function [6]. Here we show that using 2DSA, functions can be represented that grow faster than any exponential function.

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press



(a) (b)

FIGURE 11. (a) The permutation $(4, 1, 3, 2)$ represented as a black and white picture. (b) Representations of all permutations on $\{1, 2, 3\}$ written side by side.

Proposition 9.1. *The language* $L_1 = \{ \square^{n,n!} \mid n \in \mathbb{N} \}$ *is accepted by a* 2DSA.

Proof. We first describe a 2DSA $\mathcal{A}$ that accepts pictures of size $n \times (n \cdot n!)$ over Σ. Let $\mathcal{P}(m)$ be the set of all permutations on $M = \{1, \ldots, m\}$. Each element of $\mathcal{P}(m)$ can be represented as a sequence $(a_1, \ldots, a_m)$, where the a_i's are the different numbers from M. For two permutations $a = (a_1, \ldots, a_m)$ and $b = (b_1, \ldots, b_m)$, we write $a < b$ iff there is an index j such that $a_j < b_j$ and $a_i = b_i$ for all $i = 1, \ldots, j-1$. We encode a permutation $(a_1, \ldots, a_m)$ by a square picture of size $m \times m$ over $\{\square, \blacksquare\}$ as follows: for all i, row i is composed of white pixels except at position a_i, which contains a black pixel. An example is given in Figure 11a.

Given an input picture $P \in \Sigma^{m,n}$, $\mathcal{A}$ tries to write the sequence of all permutations of M in ascending order into this picture (see Fig. 11b). The critical part of this design is a procedure that computes the next permutation from a given permutation $(a_1, \ldots, a_m)$. This is realized by the following algorithm:

1. find the greatest index j such that $a_{j-1} < a_j$;
2. among $a_j, \ldots, a_m$, find the smallest element a_k that is larger than a_{j-1};
3. switch elements a_{j-1} and a_k;
4. reverse the order of the elements $a_j, \ldots, a_m$.

This algorithm can be adopted to work over a picture R representing $(a_1, \ldots, a_m)$. W.l.o.g. we can assume that $\mathcal{A}$ can also see the symbols in the neighbouring fields. It traverses R row by row from the bottom to the top, finds j and marks the jth row. After that it traverses R from the left to the right column by column, starting at column $a_{j-1}+1$, and finds k. The situation is illustrated in Figure 12(a). A swap of elements a_{j-1} and a_k follows, see the general pattern in Figure 12(b). It suffices to mark the rows and the columns in which the corresponding black pixels are located, to erase the pixels, and to write them at the new positions (note that the markers are always discarded when they are no longer needed). The whole process only visits the related marked fields. In the next step the order of the elements $a_j, \ldots, a_m$ is reversed. This is done by swapping a_j with a_m, a_{j+1} with a_{m-1}, and so forth.

Since each row and each column of R only contains a single black pixel, the total number of visits of each field during the swaps is constant. Finally, the new

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

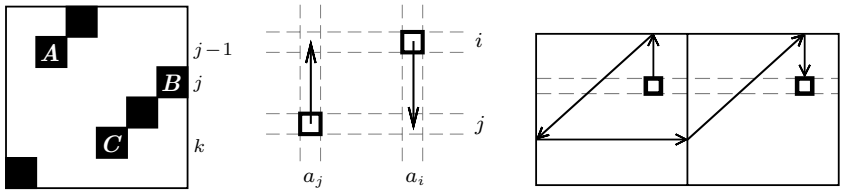


FIGURE 12. (a) Finding the indices j and k. B is the lowest one which is positioned to the right of the black pixel in the previous row (which is A). C has to lie in rows below (and including) B. Its column is the first one to the right of A's column that contains a black pixel below row $j-1$. (b) The swap of the ith and jth element of a permutation. (c) Copy one element of a permutation to the next block. The sequence consists of vertical, horizontal and diagonal movements. The related row is marked.

permutation is copied to the next block in P, applying the movements shown in Figure 12(c) to transfer each black pixel. Again, a constant number of visits is paid to each field after copying all the black pixels, thanks to the uniqueness of the rows and columns.

The whole computation is initialized by writing down the first permutation $(1, \ldots, m)$. If the area of P is exactly covered by all permutations, then its size is $m \times (m \cdot m!)$, and $\mathcal{A}$ accepts. A minor modification is needed to accept pictures of size $m \times m!$ – extend pictures representing permutations by one (dummy) row and column (that is, permutations on $M' = \{1, \ldots, m-1\}$ are generated). □

Since it is known that the number of different crossing sequences of a 2DSA between two neighbouring columns of height n is $2^{\mathcal{O}(n \log n)}$ (see the proof of Lem. 3.7), we have found a representable function with the fastest possible growth, which shows that the given bound is tight. Also we obtain the following consequence.

Corollary 9.2. $\mathsf{DREC} \cap \{\square\}^{*,*} \subsetneq \mathcal{L}(\mathsf{2DSA}) \cap \{\square\}^{*,*}$.

10. CONCLUSION

We have introduced a new model of a two-dimensional automaton, the Sgraffito automaton and its deterministic variant. While the Sgraffito automaton is strictly more expressive than the class REC of recognizable picture languages (Cor. 5.6), the class of picture languages it accepts shares many of the closure properties with REC. The deterministic Sgraffito automaton can implement a restricted depth-first search, and based on this observation we have shown that it

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

can simulate the 4FA, the 4AFA, SDREC, and the deterministic one-marker automaton. In addition, it can even accept a unary picture language that represents a function that grows faster than any exponential function.

However, many questions still remain open. The language class $\mathcal{L}(\mathsf{2DSA})$ is not closed under row and column concatenation, but it is open whether it is not closed under iterated row and column concatenation, either. While the deterministic Sgraffito automaton can simulate the deterministic one-marker automaton, a corresponding result does not hold for the nondeterministic case (Thm. 8.3), and it remains open whether nondeterministic one-marker automata are more expressive than (deterministic) Sgraffito automata. In fact, if the latter should be the case, then the complexity classes NL and P (or even NP) would coincide (Thms. 8.6 and 8.8). Also it remains to compare 2DSA and $\mathsf{2DM_1A}$ for unary picture languages, that is, picture languages over a one-letter alphabet. Are 2DSA still more expressive in this setting?

References

[1] M. Anselmo, D. Giammarresi and M. Madonia, From determinism to non-determinism in recognizable two-dimensional languages, in *Proc. of Developments in Language Theory, DLT 2007*. Vol. 4588 of *Lect. Notes Comput. Sci.* Edited by T. Harju, J. Karhumäki and A. Lepistö. Heidelberg. Springer (2007) 36–47.

[2] M. Blum and C. Hewitt, Automata on a 2-dimensional tape, in *Proc. of the 8th Annual Symposium on Switching and Automata Theory (SWAT 1967)*, FOCS '67, Washington, DC, USA. IEEE Computer Society (1967) 155–160.

[3] B. Borchert and K. Reinhardt, Deterministically and sudoku-deterministically recognizable picture languages, in *Proc. of LATA 2007*. Edited by R. Loos, S.Z. Fazekas and C. Martín-Vide. Report 35/07, Tarragona. Research Group on Mathematical Linguistics, Universitat Rovira i Virgili (2007) 175–186.

[4] S. Even and G. Even, Graph Algorithms. *Computer Software Engineering Series*. Cambridge University Press (2011).

[5] D. Giammarresi and A. Restivo, Recognizable picture languages. *Int. J. Pattern Recogn. Artif. Intell.* **6** (1992) 241–256.

[6] D. Giammarresi and A. Restivo, Two-dimensional languages. *Handbook of Formal Languages,* vol. 3. Edited by G. Rozenberg and A. Salomaa. Springer, New York, NY, USA (1997) 215–267.

[7] D. Giammarresi, A. Restivo, S. Seibert and W. Thomas, Monadic second order logic over rectangular pictures and recognizability by tiling systems. *Inf. Comput.* **125** (1996) 32–45.

[8] F.C. Hennie, One-tape, off-line Turing machine computations. *Inf. Control* **8** (1965) 553–578.

[9] J.E. Hopcroft and J.D. Ullman, Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Reading, MA (1979).

[10] P. Hsia and R.T. Yeh, Finite automata with markers, in *Proc. of ICALP*. Edited by M. Nivat. North-Holland, Amsterdam (1972) 443–451.

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

[12] K. Inoue and I. Takanami, A characterization of recognizable picture languages, in *Proc. of Parallel Image Analysis*. Vol. 654 of *Lect. Notes Comput. Sci.* Edited by A. Nakamura, M. Nivat, A. Saoudi, P.S.P. Wang and K. Inoue. Springer, Heidelberg (1993) 133–143.

[13] A. Inoue, K. Inoue, A. Ito, Y. Wang and T. Okazaki, A note on one-pebble two-dimensional Turing machines. *Information Sci.* **162** (2004) 295–314.

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press

[14] P. Jančar, F. Mráz and M. Plátek, Characterization of context-free languages by erasing automata, in *Proc. of MFCS 1992.* Vol. 629 of *Lect. Notes Comput. Sci.* Edited by I.M. Havel and V. Koubek. Springer, Heidelberg (1992) 307–314.
[15] P. Jiřička and J. Král, Deterministic forgetting planar automata are more powerful than non-deterministic finite-state planar automata. In *Developments in Language Theory*, Singapore. Edited by G. Rozenberg and W. Thomas. World Scientific (1999) 71–80.
[16] J. Kari and C. Moore, New results on alternating and non-deterministic two-dimensional finite-state automata, in *Proc. of STACS 2001.* Vol. 2010 of *Lect. Notes Comput. Sci.* Edited by A. Ferreira and H. Reichel. Springer, Heidelberg (2001) 396–406.
[17] K. Lindgren, C. Moore and M. Nordahl, Complexity of two-dimensional patterns. *J. Statist. Phys.* **91** (1998) 909–951.
[18] D. Průša, F. Mráz and F. Otto, Comparing two-dimensional one-marker automata to sgraffito automata, in *Proc. of CIAA 2013.* Vol. 7982 of *Lect. Notes Comput. Sci.* Edited by S. Konstantinidis. Springer, Heidelberg (2013) 268–279.
[19] D. Průša, Weight-reducing Hennie machines and their descriptional complexity. In *Proc. of LATA 2014.* Vol. 8370 of *Lect. Notes Comput. Sci.* Edited by A.-H. Dediu, C. Martín-Vide, J.-L. Sierra-Rodríguez and B. Truthe. Springer, Heidelberg (2014) 553–564.
[20] D. Průša and F. Mráz, Two-dimensional sgraffito automata, in *Proc. of DLT 2012.* Vol. 7410 of *Lect. Notes Comput. Sci.* Edited by H.C. Yen and O.H. Ibarra. Springer, Heidelberg (2012) 251–262.
[21] D. Průša, F Mráz and F. Otto, New results on deterministic sgraffito automata, in *Proc. of DLT 2013.* Vol. 7907 of *Lect. Notes Comput. Sci.* Edited by M.P. Béal and O. Carton. Springer, Heidelberg (2013) 409–419.
[22] A. Rosenfeld, Picture Languages – Formal Models of Picture Recognition. Academic Press, New York (1979).

Communicated by J. Kari.
Received December 20, 2013. Accepted May 28, 2014.

https://doi.org/10.1051/ita/2014023 Published online by Cambridge University Press