Scanning Pictures The Boustrophedon Way

Henning Fernau[1], Meenakshi Paramasivan[1], Markus L. Schmid[1]
and D. Gnanaraj Thomas[2]

[1] Fachbereich 4 – Abteilung Informatik, Universität Trier, D-54286 Trier, Germany,
{Fernau,Paramasivan,MSchmid}@uni-trier.de
[2] Department of Mathematics, Madras Christian College, Chennai - 600059, India,
dgthomasmcc@yahoo.com

Abstract. We are introducing and discussing finite automata working on rectangular-shaped arrays (i. e., pictures) in a boustrophedon reading mode. We derive several combinatorial, algebraic and decidability results for the corresponding class of picture languages.

1 Introduction

Syntactic considerations of digital images have a tradition of about five decades. They should (somehow) reflect methods applied to picture processing. However, one of the basic methods of scanning pictures in practice have not been thoroughly investigated from a more theoretical point of view: that of using space-filling curves. Here, we start such an investigation with what can be considered as the most simple way of defining space-filling curves: scanning line after line of an image, alternating the direction of movement every time when the image boundary is encountered (more information on the use of space-filling curves in connection with image processing or picture languages can be found in [11,13,17,20]).

We consider finite automata that work this way. We show that they are (essentially) equivalent to regular matrix languages as introduced in a sequence of papers of Rani Siromoney and her co-authors already in the early 1970s. Possibly surprisingly enough, we also present quite a number of new results for this class of picture languages, including a discussion of natural decidability questions lacking so far.

2 Our Model and Some Examples

General Definitions. In this section, we briefly recall the standard definitions and notations regarding one- and two-dimensional words and languages.

Let $\mathbb{N} := \{1, 2, 3, \ldots\}$ and let $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$. For a finite alphabet Σ, a *string* or *word* (*over* Σ) is a finite sequence of symbols from Σ, and ε stands for the *empty string*. The notation Σ^+ denotes the set of all nonempty strings over Σ, and $\Sigma^* := \Sigma^+ \cup \{\varepsilon\}$. For the *concatenation* of two strings w_1, w_2 we write $w_1 \cdot w_2$ or simply $w_1 w_2$. We say that a string $v \in \Sigma^*$ is a *factor* of a string $w \in \Sigma^*$ if

there are $u_1, u_2 \in \Sigma^*$ such that $w = u_1 \cdot v \cdot u_2$. If u_1 or u_2 is the empty string, then v is a *prefix* (or a *suffix*, respectively) of w. The notation $|w|$ stands for the length of a string w.

A *two-dimensional word* (also called *picture*, *matrix* or *array*) *over* Σ is a tuple

$$W := ((a_{1,1}, a_{1,2}, \ldots, a_{1,n}), (a_{2,1}, a_{2,2}, \ldots, a_{2,n}), \ldots, (a_{m,1}, a_{m,2}, \ldots, a_{m,n}))\,,$$

where $m, n \in \mathbb{N}$ and, for every i, $1 \leq i \leq m$, and j, $1 \leq j \leq n$, $a_{i,j} \in \Sigma$. We define the *number of columns* (or *width*) and *number of rows* (or *height*) of W by $|W|_c := n$ and $|W|_r := m$, respectively. The *empty picture* is denoted by λ, i. e., $|\lambda|_c = |\lambda|_r = 0$. For the sake of convenience, we also denote W by $[a_{i,j}]_{m,n}$ or by a matrix in a more pictorial form. If we want to refer to the j^{th} symbol in row i of the picture W, then we use $W[i, j] = a_{i,j}$. By Σ^{++}, we denote the set of all nonempty pictures over Σ, and $\Sigma^{**} := \Sigma^{++} \cup \{\lambda\}$. Every subset $L \subseteq \Sigma^{**}$ is a *picture language*.

Let $W := [a_{i,j}]_{m,n}$ and $W' := [a'_{i,j}]_{m',n'}$ be two non-empty pictures over Σ. The *column concatenation* of W and W', denoted by $W \oplus W'$, is undefined if $m \neq m'$ and is the picture

$$\begin{array}{cccccccc} a_{1,1} & a_{1,2} & \cdots & a_{1,n} & b_{1,1} & b_{1,2} & \cdots & b_{1,n'} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} & b_{2,1} & b_{2,2} & \cdots & b_{2,n'} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} & b_{m',1} & b_{m',2} & \cdots & b_{m',n'} \end{array}$$

otherwise. The *row concatenation* of W and W', denoted by $W \ominus W'$, is undefined if $n \neq n'$ and is the picture

$$\begin{array}{cccc} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \\ b_{1,1} & b_{1,2} & \cdots & b_{1,n'} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n'} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m',1} & b_{m',2} & \cdots & b_{m',n'} \end{array}$$

otherwise. In order to denote that, e. g., $U \ominus V$ is undefined, we also write $U \ominus V =$ `undef`.

Example 1. Let

$$W_1 := \begin{smallmatrix} \mathtt{a}&\mathtt{b}&\mathtt{a} \\ \mathtt{b}&\mathtt{c}&\mathtt{a} \\ \mathtt{a}&\mathtt{b}&\mathtt{b} \end{smallmatrix}, W_2 := \begin{smallmatrix} \mathtt{b}&\mathtt{c} \\ \mathtt{b}&\mathtt{a} \\ \mathtt{c}&\mathtt{a} \end{smallmatrix}, W_3 := \begin{smallmatrix} \mathtt{a}&\mathtt{b}&\mathtt{c} \\ \mathtt{c}&\mathtt{b}&\mathtt{b} \end{smallmatrix} \text{ and } W_4 := \begin{smallmatrix} \mathtt{a}&\mathtt{a} \\ \mathtt{a}&\mathtt{b} \end{smallmatrix}.$$

Then $W_1 \ominus W_2 = W_1 \oplus W_3 =$ `undef`, but

$$W_1 \oplus W_2 = \begin{smallmatrix} \mathtt{a}&\mathtt{b}&\mathtt{a}&\mathtt{b}&\mathtt{c} \\ \mathtt{b}&\mathtt{c}&\mathtt{a}&\mathtt{b}&\mathtt{a} \\ \mathtt{a}&\mathtt{b}&\mathtt{b}&\mathtt{c}&\mathtt{a} \end{smallmatrix} \text{ and } W_1 \ominus W_3 = \begin{smallmatrix} \mathtt{a}&\mathtt{b}&\mathtt{a} \\ \mathtt{b}&\mathtt{c}&\mathtt{a} \\ \mathtt{a}&\mathtt{b}&\mathtt{b} \\ \mathtt{a}&\mathtt{b}&\mathtt{c} \\ \mathtt{c}&\mathtt{b}&\mathtt{b} \end{smallmatrix}.$$

For a picture W and $k, k' \in \mathbb{N}$, by W^k we denote the k-fold column-concatenation of W, by W_k we denote the k-fold row-concatenation of W, and $W^k_{k'} = (W^k)_{k'}$.

Boustrophedon Finite Automaton. A *boustrophedon finite automaton*, or BFA for short, can be specified as a quintuple $M = (Q, \Sigma, R, s, F)$, where Q is a finite set of states, Σ is an input alphabet, $R \subseteq Q \times (\Sigma^* \cup \{\#\}) \times Q$ is a finite set of rules. The special symbol $\# \notin \Sigma$ indicates the border of the rectangular picture that is processed, $s \in Q$ is the initial state, F is the set of final states.

Let $\square$ be a new symbol indicating a blank position and let $\Sigma_+ := \Sigma \cup \{\#, \square\}$. Then $C_M := Q \times \Sigma_+^{**} \times \mathbb{N}$ is the set of configurations of M.

A configuration $(p, A, m) \in C_M$ is *valid* if $1 \leq m \leq |A|_r$ and, for every i, $1 \leq i \leq m-1$, the ith row equals $\# \, \square^{|A|_c - 2} \, \#$, for every j, $m+1 \leq j \leq |A|_r$, the jth row equals $\#w\#$, $w \in \Sigma^{|A|_c - 2}$, and, for some n, $0 \leq n \leq |A|_c - 2$, $w \in \Sigma^{|A|_c - n - 2}$, the mth row equals $\# \, \square^n \, w\#$, if m is odd and $\#w \, \square^n \, \#$, if m is even.

If (p, A, m') and (q, A', m') are two valid configurations such that A and A' are identical but for one position (i, j), where $A'[i, j] = \square$ while $A[i, j] \in \Sigma$, then $(p, A, m') \vdash_M (q, A', m')$ if $pA[i, j] \to q \in R$. If (p, A, m') and $(q, A, m' + 1)$ are two valid configurations, then $(p, A, m') \vdash_M (q, A, m' + 1)$ if the m'th row contains only $\#$ and $\square$ symbols, and if $p\# \to q \in R$. The reflexive transitive closure of the relation $\vdash_M$ is denoted by $\vdash_M^*$.

The BFA M is deterministic, or a BDFA for short, if for all $p \in Q$ and $a \in \Sigma \cup \{\#\}$, there is at most one $q \in Q$ with $pa \to q \in R$.

The language accepted by M is then the set of all $m \times n$ pictures A over Σ such that

$$(s, \#_m \oplus A \oplus \#_m, 1) \vdash_M^* (f, \#_m \oplus \square_m^n \oplus \#_m, m)$$

for some $f \in F$ (note that the automaton works on a picture with a first and last column of only $\#$ symbols, but only the part in between these border columns is accepted). In other words, the computation starts with scanning the left uppermost corner of the picture and then working through the picture row-by-row, as the ox turns, i.e., the boustrophedon way, until the last entry of the last row is scanned. The following illustrates how a BFA scans some input picture and how a picture of a possible configuration looks like:

```
  # a b a b a #
  → → → → → → → ↓
  # b c a c b # ↓
↓ ← ← ← ← ← ← ← ←
↓ # a b b b b #
→ → → → → → → → ↓
  # a b c b b # ↓
↓ ← ← ← ← ← ← ← ←
↓ # c b b a a #
→ → → → → → → →
```

```
# □ □ □ □ □ #
# □ □ □ □ □ #
# □ □ □ b b #
# a b c b b #
# c b b a a #
```

Notice that since rules of the form $p\# \to q$ need not be present in R, in some natural sense the classical regular string languages are a special case of BFA languages.

Example 2. The *set of tokens* L of all sizes and of all proportions is accepted by the BFA $M = (Q, \Sigma, R, s, F)$, where $Q = \{s, s_1, s_2, s_3, s_4\}$, $\Sigma = \{\mathtt{x}, .\}$,
$R = \{(s\mathtt{x}, s_1), (s_1\cdot, s_1), (s_1\#, s_2), (s_1\#, s_4), (s_2\cdot, s_2), (s_2\mathtt{x}, s_3), (s_3\#, s), (s_3\#, s_4), (s_4\mathtt{x}, s_4)\}$, and $F = \{s_4\}$.

A sample token of L accepted by M is

We now recall the notion of two-dimensional right-linear grammars (2RLG) as given in [7]. The original definition of a 2RLG (under the name of a regular matrix grammar (RMG)) and the properties of the corresponding class of picture languages called RML can be found in [8,14,15,18].

Definition 1. *A* two-dimensional right-linear grammar (2RLG) *is defined by a* 7*-tuple* $G = (V_h, V_v, \Sigma_I, \Sigma, S, R_h, R_v)$*, where:*

- V_h *is a finite set of* horizontal nonterminals*;*
- V_v *is a finite set of* vertical nonterminals*;*
- $\Sigma_I \subseteq V_v$ *is a finite set of* intermediates*;*
- Σ *is a finite set of* terminals*;*
- $S \in V_h$ *is a* starting symbol*;*
- R_h *is a finite set of* horizontal rules *of the form* $V \to AV'$ *or* $V \to A$*, where* V*,* $V' \in V_h$ *and* $A \in \Sigma_I$*;*
- R_v *is a finite set of* vertical rules *of the form* $W \to aW'$ *or* $W \to a$*, where* $W, W' \in V_v$ *and* $a \in \Sigma$*.*

There are two phases of derivation of the 2RLG. In the first phase, a horizontal string of intermediate symbols is generated by means of the string grammar $G_h = (V_h, \Sigma_I, S, R_h)$, denoted by $H(G)$. In the second phase, treating each intermediate as a start symbol, the vertical generation of the actual picture is done in parallel, by applying a finite set of right-linear rules R_v. In order to produce a rectangular-shaped picture, the rules of R_v must be applied in parallel; also this means that the rules of the form $V_i \to a_i$ are all applied in every column simultaneously to finish the picture with generating its last row. These grammars make sure that the columns can grow only in downward direction.

We note that our model is closely connected with 2RLG, as we will show more precisely in the following. The formalization is closer to our model than the original one due to Siromoney and her co-authors.

3 Characterization Results

Clearly, BFAs are a special form of 4-NFA (4-way nondeterministic finite automata, see [7]), and it is known that for these 2-dimensional automata, the deterministic variant is weaker regarding its descriptive capacity compared to the nondeterministic one. Hence, and also because of the practical relevance of the deterministic model, the following result is interesting.

Theorem 1. *BDFAs and BFAs describe the same class of picture languages.*

Proof. Apply the well-known subset construction. This works out as our BFAs are syntactically the same as classical finite automata, only the interpretation of their processsing is different. □

Next, we examine the question whether the boustrophedon processing mode of our automata is essential. To this end, let us consider yet another interpretation of finite automata, this time termed *returning finite automata*, or RFA for short. Syntactically, they are identical to BFA, so they can be again described by a quintuple $M = (Q, \Sigma, R, s, F)$. However, they always process rows from left to right. Formally, this means that we can carry over all parts of the definition of BFA apart from the notion of a valid configuration, which needs to be slightly modified. Now, a configuration $(p, A, m) \in C_M$ is *valid* if $1 \leq m \leq |A|_r$ and, for every i, $1 \leq i \leq m-1$, the ith row equals $\# \, \square^{|A|_c-2} \, \#$, for every j, $m+1 \leq j \leq |A|_r$, the jth row equals $\#w\#$, $w \in \Sigma^{|A|_c-2}$, and, for some n, $0 \leq n \leq |A|_c - 2$, $w \in \Sigma^{|A|_c-n-2}$, the mth row equals $\# \, \square^n \, w\#$.

Theorem 2. *BFAs and RFAs describe the same class of picture languages.*

Proof. We first show how an RFA can simulate a BFA. The basic idea can be summarised as follows. On the first row, which is scanned from left to right by both automata, the RFA simulates the BFA one to one. Assume that the BFA, while moving on to the second row, changes into a state q, scans the row from right to left and enters a state p when the beginning of this row is reached. In order to simulate this behaviour, the RFA stores its current state q in the finite state control and guesses the state p. It then scans the second row from left to right (starting in state p) by applying the transitions of the BFA in reverse direction. When the end of the row is reached, the computation only proceeds if the RFA is in state q. This procedure is then repeated.

More formally, the states of the BFA are triples (p, q, r), where p is the actual state, q is the state that should be reached after finishing a row and r indicates wheter or not a left to right scan or a right to left scan is performed (this is necessary in order to lock or unlock the possibility of taking transitions in reverse direction). The formal definition is as follows.

Let $M = (Q, \Sigma, R, s, F)$ be some BFA. Then, define the equivalent RFA $M' = (Q', \Sigma, R', s', F')$ as follows: $Q' = Q \times Q \times \{1, 2\} \cup \{s'\}$,

$$\begin{aligned} R' = &\{(p, r, 1)a \to (q, r, 1) \mid pa \to q \in R, r \in Q\} \\ &\cup \{(q, r, 2)a \to (p, r, 2) \mid pa \to q \in R, r \in Q\} \\ &\cup \{(p, p, 1)\# \to (r, q, 2) \mid p\# \to q \in R, r \in Q\} \\ &\cup \{(p, p, 2)\# \to (r, q, 1) \mid p\# \to q \in R, r \in Q\} \\ &\cup \{s'a \to (p, r, 1) \mid sa \to p \in R, r \in Q\}\,. \end{aligned}$$

Moreover, $F' = \{(r, r, 1) \mid r \in F\} \cup \{(p, r, 2) \mid r \in F, p \in Q\}$. The formal (induction) proof of the correctness of the construction is left to the reader.

The converse direction can be seen in a similar way. □

We can likewise define finite automata that read all rows in a right-to-left fashion. A similar construction as in the previous theorem shows (again) that this model is equivalent to BFAs. This also shows that the direction of the rotation mentioned in the next theorem does not matter.

Theorem 3. *A picture language can be described by a BFA if and only if its image, rotated by 90 degrees, is in RML.*

Proof. We provide two simulations to show the claim.

Let $G = (V_h, V_v, \Sigma_I, \Sigma, S, R_h, R_v)$, be a 2-dimensional right-linear grammar. The rotation can be interpreted as $H(G)$ describing the leftmost column of the picture, while the second phase of G then means to generate all rows, starting from the intermediate string from $H(G)$. We are going to construct an equivalent RFA $M = (Q, \Sigma, R, s, F)$, which is sufficient for giving a BFA thanks to Theorem 2. Let $Q = (V_h \cup \{f\}) \times (V_v \cup \{f\}) \cup \{s\}$, where $f \notin V_h \cup V_v$, and $F = \{(f, f)\}$. Let R contain the following rules:

- $sa \to (S', A')$, if $S \to AS' \in R_h$ and $A \to aA' \in R_v$,
- $sa \to (f, A')$, if $S \to A \in R_h$ and $A \to aA' \in R_v$,
- $(X, A)a \to (X, A')$, if $X \in V_h \cup \{f\}$ and $A \to aA' \in R_v$,
- $(X, A)a \to (X, f)$, if $A \to a \in R_v$, $X \in V_h$,
- $(X, f)\# \to (X', A)$, if $X \to AX' \in R_h$,
- $(X, f)\# \to (f, A)$, if $X \to A \in R_h$,
- $(f, A)a \to (f, f)$, if $A \to a \in R_v$.

The idea of the construction is that the generation of columns of G is performed in the second component of the state pairs, whereas the first component corresponds to the generation of the axiom (i. e., the first row of the pictures generated by G). The crucial difference is that the first symbol of the axiom (which in the case of RFA is the first column instead of the first row) is generated and then the first row is generated before the second letter of the axiom is generated in the next row. Hence, the two phases of the picture construction of G is dovetailed.

The converse is seen as follows. Let $M = (Q, \Sigma, R, s, F)$ be some RFA. We construct an equivalent 2-dimensional right-linear grammar $G = (V_h, V_v, \Sigma_I, \Sigma, S, R_h, R_v)$ (generating the rotated picture) with $V_h = Q \cup \{S\}$, $\Sigma_I = Q \times Q$, and rules

- $S \to (s, r)r \in R_h$ for all $r \in Q$,
- $q \to (q, r)r \in R_h$ for all $q, r \in Q$,
- $q \to \varepsilon$ for all $q \in Q$,
- $(p, r) \to (q, r)a \in R_v$ for all $pa \to q \in R$, $r \in Q$,
- $(p, q) \to \varepsilon \in R_v$ for all $p\# \to q \in R$.

The astute reader will have noticed that we took the freedom to incorporate erasing productions for convenience, but these can be avoided by using standard formal language constructions. This concludes the proof. □

Due to Theorem 3, we can inherit several properties for the class of picture languages described by BFAs. For instance, the class is not closed under rotation by 90 degrees, also known as quarter turns, see [15]. On the positive side, RML (and hence BFA picture languages) are closed under Boolean operations. More precisely, it was shown in [15] that RML (and hence BFA picture languages) are closed under union. We supplement this by the following two results.

Theorem 4. *BFA picture languages are closed under complementation.*

Proof. First, let us recall from Theorem 1 that BDFA and BFA describe the same class of picture languages. Let $M = (Q, \Sigma, R, s, F)$ be some BDFA. Then we can construct a BDFA $\overline{M}$ by state complementation, i. e., $\overline{M} = (Q, \Sigma, R, s, Q - F)$. On some input picture A, M reaches the same state as $\overline{M}$ and, furthermore, since both M and $\overline{M}$ are deterministic, there exists exactly one state $q \in Q$ that can be reached by M and $\overline{M}$ on input A. This directly implies that $A \in L(M)$ if and only if $A \notin L(\overline{M})$; thus, $L(\overline{M}) = \overline{L(M)}$. Hence BFA picture languages are closed under complementation. □

Notice that the previous theorem has become easy because we have a deterministic model for BFAs, in contrast to what has been established for RML before. De Morgan's law now immediately yields:

Corollary 1. *BFA picture languages are closed under intersection.*

Conversely, the results we derive in the following for BFAs can be immediately read as results for RML, as well.

4 Pumping and Interchange Lemmas

Since in the pictures of an RML, the first row as well as the columns are generated by regular grammars, there are two ways to apply the pumping lemma for regular languages: we can pump the first row, which results in repetitions of a column-factor of the picture, or we can pump each column individually, which will only lead to a rectangular shaped picture if the pumping exponents are, in a sense, well-chosen. Hence, we can conclude a horizontal and a vertical pumping lemma for RML (see [9]) and, due to Theorem 3, these pumping lemmas carry over to BFA languages:

Lemma 1. *Let M be a BFA. Then there exists an $n \in \mathbb{N}$, such that, for every $W \in L(M)$ with $|W|_r \geq n$, $W = X \ominus Y \ominus Z$, $|X \ominus Y|_r \leq n$, $|Y|_r \geq 1$ and, for every $k \geq 0$, $X \ominus Y_k \ominus Z \in L(M)$.*

Lemma 2. *Let M be a BFA and let $W \in L(M)$ with $|W|_r = m$. Then there exist $n, r_1, r_2, \ldots, r_m \in \mathbb{N}$, such that, for every $W \in L(M)$ with $|W|_c \geq n$,*

$$W = (x_1 \oplus y_1 \oplus z_1) \ominus (x_2 \oplus y_2 \oplus z_2) \ominus \ldots \ominus (x_m \oplus y_m \oplus z_m),$$

$|x_i \ominus y_i|_c \leq n$, $|y_i|_c \geq 1$, $1 \leq i \leq m$, *and, for every* $k \geq 1$,

$$W = (x_1 \oplus y_1^{(k\,t_1)} \oplus z_1) \ominus (x_2 \oplus y_2^{(k\,t_2)} \oplus z_2) \ominus \ldots \ominus (x_m \oplus y_m^{(k\,t_m)} \oplus z_m) \in L(M)\,,$$

where $t_i = \frac{lcm(r_1,r_2,\ldots,r_m)}{r_i}$, $1 \leq i \leq m$.

Lemma 1 is straightforward and in order to see that Lemma 2 holds, it is sufficient to note that n is the maximum of all the pumping lemma constants for the individual rows (recall that each row is generated by an individual regular grammar) and the r_i are the lengths of the factors that are pumped. Obviously, not every way of pumping the rows results in a rectangular shaped picture, so we can only pump by multiples of the t_i.

While the vertical pumping lemma has the nice property that a whole row-factor can be pumped, in the horizontal pumping lemma we can only pump factors of each individual row, that are independent from each other. As a result, this lemma does not guarantee the possibility of pumping by 0, i. e., removing a factor, which, for classical regular languages, often constitutes a particularly elegant way of showing the non-regularity of a language.

However, it can be shown that also for BFA there exists a horizontal pumping lemma that pumps whole column-factors (which then also translates into a vertical pumping lemma for RML that pumps whole row-factors).

Lemma 3. *Let* M *be a BFA and let* $m \in \mathbb{N}$. *Then there exists an* $n \in \mathbb{N}$, *such that, for every* $W \in L(M)$ *with* $|W|_r \leq m$ *and* $|W|_c \geq n$, $W = X \oplus Y \oplus Z$, $|X \oplus Y|_c \leq n$, $|Y|_c \geq 1$ *and, for every* $k \geq 0$, $X \oplus Y^k \oplus Z \in L(M)$.

Proof. Let Q be the set of states of M, let q_0 be the start state and let $n = |Q|^m + 1$. Furthermore, let $W \in L(M)$ with $|W|_r = m$ (the case $|W|_r < m$ can be handled analogously) and $|W|_c = n' \geq n$. Since M accepts W, there is an accepting computation $(p_1, W_1, m_1) \vdash_M^* (p_k, W_k, m_k)$ for W, i. e., $(p_1, W_1, m_1) = (s, \#_m \oplus A \oplus \#_m, 1)$ and $(p_k, W_k, m_k) = (f, \#_m \oplus \square_m^{n'} \oplus \#_m, m)$. We can now consider the extended configurations (p_i, W_i', m_i), where W_i' is like W_i with the only difference that each $\square$ symbol is replaced by the state that has been entered by producing this occurrence of $\square$. Since W has m rows, the maximum number of different columns in W_k' is $|Q|^m$ and since W has at least $n = |Q|^m + 1$ columns, we can conclude that $W_k' = X' \oplus \alpha' \oplus Y' \oplus \alpha' \oplus Z'$, where $|\alpha'|_c = 1$. Furthermore, $|X' \oplus \alpha' \oplus Y'|_c \leq n$ and $|\alpha' \oplus Y'|_c \geq 1$. Now let $W = X \oplus \alpha \oplus Y \oplus \alpha \oplus Z$, where $|X|_c = |X'|_c$, $|Y|_c = |Y'|_c$ and $|Z|_c = |Z'|_c$. By definition of BFA, for every $i \geq 0$, M accepts $X \oplus (\alpha \oplus Y)^i \oplus \alpha \oplus Z$. □

We wish to point out that in a similar way, we can also prove a row and a column interchange lemma (the only difference is that the number n has to be chosen large enough to enforce repeating pairs of states):

Lemma 4. *Let* M *be a BFA. Then there exists an* $n \in \mathbb{N}$, *such that, for every* $W \in L(M)$ *with* $|W|_r \geq n$, *there exists a factorisation* $W = V_1 \ominus X \ominus V_2 \ominus Y \ominus V_3$, $|X|_c \geq 1$, $|Y|_c \geq 1$, *such that* $V_1 \ominus Y \ominus V_2 \ominus X \ominus V_3 \in L(M)$.

Lemma 5. *Let M be a BFA and let $m \in \mathbb{N}$. Then there exists an $n \in \mathbb{N}$, such that, for every $W \in L(M)$ with $|W|_r \leq m$ and $|W|_c \geq n$, there exists a factorisation $W = V_1 \oplus X \oplus V_2 \oplus Y \oplus V_3$, $|X|_c \geq 1$, $|Y|_c \geq 1$, such that $V_1 \oplus Y \oplus V_2 \oplus X \oplus V_3 \in L(M)$.*

5 Complexity Results

Only few complexity results have been obtained so far for RML. The only reference that we could find was an unpublished manuscript of Dassow [2] that also merely classified the decidability versus undecidability status of several decision problems. Here, we give the exact complexity status of the basic decidability questions for RML, formulated in terms of BFA.

We will only look into classical formal language questions, which are:

- Universal membership: Given a B(D)FA M and a picture A (as input of some algorithm), is A accepted by M?
- Non-emptiness: Given a B(D)FA M, is there some picture A accepted by M?
- Inequivalence: Given two B(D)FAs M_1 and M_2, do both automata accept the same set of pictures?

Also, we shortly discuss the issue of minimization in the context of BDFAs. Our complexity considerations will be concerned with standard complexity classes, like (N)L, i.e., (non-)deterministic logarithmic space, (N)P, i.e., (non-) deterministic polynomial time, and NPSPACE (which is equal to PSPACE), i.e., polynomial space. All reductions that we sketch are implementable in deterministic logarithmic space.

Theorem 5. *The universal membership problem for BFA is NL-complete*

Proof. As universal membership is NL-hard for NFAs (working on strings), NL-hardness is clear. As a configuration of a BFA can be specified (basically) by the state and a pointer into the input picture, membership in NL is obvious for this problem, as well. □

By a similar argument applied to deterministic devices, we conclude:

Corollary 2. *The universal membership problem for BDFA is L-complete*

We now turn to the problem of deciding whether or not a given BFA accepts a non-empty language.

Proposition 1. *The non-emptiness problem for BFA languages is in PSPACE.*

Proof. Dassow [2] shows that the emptiness problem for RML is decidable. His proof actually shows (together with our constructions given above) that the BFA non-emptiness problem is solvable using polynomial space. □

In fact, we can give an alternative argument based on the pumping lemmas that we derived. Namely, these lemmas show that any n-state BFA M either accepts a picture with at most n rows and at most n^n columns, or it does not accept any picture at all. We can use this result by computing, for $k = 1$ to n^n, whether or not a word (i.e., a picture with one row) of length k (i.e., with k columns) can be processed starting in state p and ending in state q. Notice that we can store this information in a binary matrix with $n \times n$ many entries, and we can update this matrix by matrix multiplication. As the maximum counter content $k = n^n$ can be stored in polynomial space, the sketched algorithm either finds a way to accept some picture of at most n rows and at most n^n columns that is accepted by M, or it can finally be sure that the picture language accepted by M is empty.

Notice that in case the number of rows is a fixed constant, then this argument shows that the BFA-non-emptiness problem is in NP. However, the hardness proof would then fail, as the problem that we reduce from would then become solvable in logarithmic space.

Theorem 6. *The non-emptiness problem for BFA languages is NP-hard.*

Proof. We reduce from the well-known NP-complete intersection emptiness problem for finite automata with a one-letter input alphabet, see [10]. The idea is to run each of the input automata $A_1, \ldots, A_m$ in one line. We can assume that all state alphabets are disjoint. All transition rules of each A_i are also transition rules of the BFA M we are going to construct. For each final state f_i of A_i, we introduce a rule $f_i\# \to s_{i+1}$, where s_{i+1} is the initial state of A_{i+1}. The set F_m of final states of A_m is the set of final states of M. The intersection $L(A_1) \cap \cdots \cap L(A_m)$ is non-empty if and only if some word a^k is accepted by all these automata, which means that the picture a_m^k is accepted by M. □

We leave the question whether or not non-emptiness of BFAs is in NP open.

Theorem 7. *The inequivalence problem for BFA languages is in PSPACE (but NP-hard).*

Proof. The hardness immediately transfers from Theorem 6. As the BFA picture languages are closed under Boolean operations (and the constructions can be carried out in polynomial time), given two BFAs M_1 and M_2, we can construct a BFA M such that the picture language accepted by M is the symmetric difference of the picture languages of M_1 and M_2; hence, M_1 and M_2 are equivalent if and only if the picture language of M is empty. □

Let us finally comment shortly about minimization. Here, the question is, given a BDFA A, to find a BDFA A^* that has as few states as possible but describes the same pictures as A does. Notice that this problem can be solved in polynomial time for DFAs accepting words. It might be tempting to use this well-known algorithm and apply it to a given BDFA A. In fact, this would result, in general, in a smaller automaton A' that is also picture-equivalent to A. However, in general A^* and A' can differ significantly.

Let us explain the problems with this approach with a simple example. Consider the (string) language

$$L = \{a^{385}\}^+ \# \{a^{385}\}^+ \# \{a^{385}\}^+ .$$

The smallest DFA accepting this language has 1159 states. The pictures that are accepted by this automaton (as a BDFA) are pictures of three rows, where each row has a number of a's that is a multiple of 385. However, as the length of rows have to synchronize, it is sufficient to make sure that at least one row has the right length, so that

$$L' = \{a^{385}\}^+ \# \{a\}^+ \# \{a\}^+$$

would be another string language, whose minimal state deterministic finite automaton has only 388 states, that accepts the same picture language. As $385 = 5 * 7 * 11$, we can even see that the minimal DFA for

$$L'' = \{a^5\}^+ \# \{a^7\}^+ \# \{a^{11}\}^+ ,$$

which has 26 states, again accepts the same picture language when interpreted as a BDFA. Still, it remains unclear to us if this is the minimal deterministic automaton for the picture language in question. Even more, we do not see a general efficient methodology how to obtain minimal-state BDFAs. The only method that we can propose is brute-force, cycling through all smaller BDFAs and then testing for equivalence. This can be easily implemented in polynomial space, so that we can conclude (in terms of a decision problem).

Proposition 2. *The question to determine whether a given BFA is minimal-state can be solved in PSPACE.*

Notice that we could state this (even) for nondeterministic devices, but as indicated above, we do not know anything better for deterministic ones, either.

6 Possible Applications to Character Recognition

Character recognition has always been the testbed application for picture processing methods. We refer to [4,12] and the literature quoted therein. In this regard, we are now going to discuss the recognition of some classes of characters, also (sometimes) showing the limitations of our approach, making use of the pumping lemmas that we have shown above.

For example, consider the set K of all L tokens of all sizes with fixed proportion i.e., the ratio between the two arms of L being 1. The first three members of K are as follows:

```
x .      x . .      x . . .
x x ,    x . . ,    x . . . .
         x x x      x x x x
```

We claim that K is not accepted by any BFA. Suppose there exists a BFA to accept K. Then by Lemma 1 there exists an $n \in \mathbb{N}$, such that, for every $W \in K$

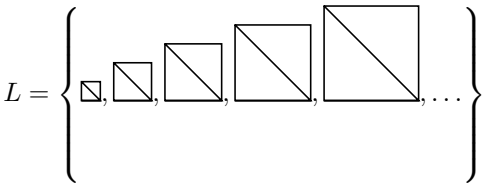
with $|W|_r \geq n$, $W = X \ominus Y \ominus Z$, $|X \ominus Y|_r \leq n$, $|Y|_r \geq 1$ and, for every $k \geq 0$, $X \ominus Y_k \ominus Z \in K$. But, unfortunately, for many values of k we get L tokens with unequal arms which are not members of K which gives a contradiction to our assumption.

On the other hand, as pointed out by Example 2, if we do not require the ratio between the two arms to be fixed, then the corresponding set of pictures can be recognised by a BFA. Similarly, the characters A (if given in the form $\square\!\!\!\!\!\!\square$), E, F, H, I, P (if given in the form $\square\!\!\!\!\!\!\square$), T, U (if given in the form $\square\!\!\!\!\!\!\square$) can be recognised by BFA, if we do not require fixed proportions. In particular, this means that $\square\!\!\!\!\!\!\square$, $\square\!\!\!\!\!\!\square$, $\square\!\!\!\!\!\!\square$, $\square\!\!\!\!\!\!\square$, $\square\!\!\!\!\!\!\square$, $\square\!\!\!\!\!\!\square$ are valid characters as well. Note that the character I plays a special role: this set of characters can only be recognised by a BFA if it is given in the form $\{\cdot_n^{k_1} \oplus \mathtt{x}_n \oplus \cdot_n^{k_2} \mid k_1 \leq k, k_2, n \in \mathbb{N}\}$, for some fixed constant $k \in \mathbb{N}$ (i. e., a BFA is not able to recognise the set of all vertical lines).

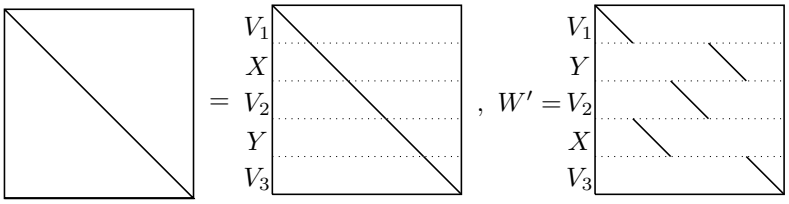
However, if we insist on fixed proportions, then it can be easily shown that the character classes mentioned above cannot be recognised by BFAs. For example, if the length of an arm of a character (or the distance between two parallel arms) is only allowed to grow in proportion to the length of another arm, then the vertical or horizontal pumping lemma shows that this class of characters cannot be recognised by a BFA.

More generally, even single diagonal lines cannot be detected by BFA, which excludes several classes of characters from the class of BFA languages, e. g., A, K, M, N, X. We shall prove this claim more formally, by applying the vertical interchange lemma.

Let L be the set of pictures of diagonal lines from the upper-left corner to the lower-right corner, i. e.,

$L =$  .

If L can be recognised by a BFA, then, according to Lemma 4, there is a picture $W \in L$ with $W = V_1 \ominus X \ominus V_2 \ominus Y \ominus V_3$, $|X|_c \geq 1$, $|Y|_c \geq 1$, and $W' = V_1 \ominus Y \ominus V_2 \ominus X \ominus V_3 \in L(M)$. The following illustrates how this leads to a contradiction:

$W =$ 

.

We chose L to only contain square pictures with a diagonal connecting the upper-left corner with the lower-right one for presentational reasons. In the same

way, it can be shown that the set of single continuous diagonal lines cannot be recognised by BFA.

7 Discussions

Scanning pictures line by line is for sure not a new invention in image processing. We have tried to derive a formal model that does mirror this strategy. On the one hand, we have shown that this formal model is pretty stable, as it has various characterizations, and it is even linked to RML, one of the earliest formal models of picture processing. On the other hand, there are quite some natural operations under which we would hope such a model to be closed, as, for example, quarter turns.

There are more powerful models than ours that have been proposed for picture processing, like 4-way NFAs or OTAs, see [7]. These have better closure properties, but also much weaker decidability results; for instance, the emptiness problem for such devices is undecidable.

However, OTAs are related to our model in the sense that they process a picture diagonal by diagonal, whereas our model process it row by row. The additional power seems to come from the fact that during a computation, OTAs label positions of the pictures by states and this labelling depends not only on the current symbol, but also on the state labels of the upper and left neighbours (i. e., OTAs are special versions of cellular automata). This means that information can be passed from top to bottom in every single column, whereas BFA can only accumulate information of a whole row. Notice that, when we remove this option from the way OTAs work, we arrive at a model that is possibly even closer to ours, the only difference being the way images are scanned. Clearly, diagonal scans can (now) detect diagonal lines, but now there is no way to detect vertical or horizontal ones, as would be the case for RML or BFA.

Conversely, we have seen that diagonals cannot be detected by neither RML nor BFA. Possibly, a more thorough study of different scanning schemes from the point of view of the (typical) classes of images that can be accepted would lead to new insights telling how images should be scanned by computers in practice.

It might be interesting to enhance the power of the automata that scan images. As we have seen, many interesting decidability questions are already pretty hard for finite automata; however, if we now extend our basic models, for example, from finite automata to weak forms of counter automata, we might be able to stay within the same complexity classes for these decision problems, while significantly increasing the usefulness of these models. For instance, we can use counters or linear-type grammars to recognize X-shaped letters (or diagonals).

The easiest way to deal with this might be to think about processing pictures with automata using two or more heads in a synchronized fashion, as already proposed in [5]. From a formal language point of view, using two heads, scanning row by row from left to right and right to left in parallel corresponds to even-linear languages as introduced in [1] and further generalized in [6,3,16,19].

References

1. V. Amar and G. Putzolu. On a family of linear grammars. *Information and Control (now Information and Computation)*, 7:283–291, 1964.
2. J. Dassow. Grammatical picture generation. Manuscript, available online, 2007.
3. H. Fernau. Even linear simple matrix languages: formal language properties and grammatical inference. *Theoretical Computer Science*, 289:425–489, 2002.
4. H. Fernau and R. Freund. Bounded parallelism in array grammars used for character recognition. In P. Perner, P. Wang, and A. Rosenfeld, editors, *Advances in Structural and Syntactical Pattern Recognition (Proceedings of the SSPR'96)*, volume 1121 of *LNCS*, pages 40–49. Springer, 1996.
5. H. Fernau, R. Freund, and M. Holzer. Character recognition with k-head finite array automata. In A. Amin et al., editors, *Proceedings of SSPR'98*, volume 1451 of *LNCS*, pages 282–291, 1998.
6. H. Fernau and J. M. Sempere. Permutations and control sets for learning non-regular language families. In A. L. Oliveira, editor, *Grammatical Inference: Algorithms and Applications, 5th International Colloquium ICGI 2000*, volume 1891 of *LNCS/LNAI*, pages 75–88. Springer, 2000.
7. D. Giammarresi and A. Restivo. Two-dimensional languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Volume III*, pages 215–267. Berlin: Springer, 1997.
8. K. Krithivasan and R. Siromoney. Array automata and operations on array languages. *International Journal of Computer Mathematics*, 4(A):3–40, 1974.
9. K. Krithivasan and R. Siromoney. Characterizations of regular and context-free matrices. *International Journal of Computer Mathematics*, 4(A):229–245, 1974.
10. K.-J. Lange and P. Rossmanith. The emptiness problem for intersections of regular languages. In I. M. Havel and V. Koubek, editors, *Mathematical Foundations of Computer Science 1992, 17th International Symposium, MFCS'92*, volume 629 of *LNCS*, pages 346–354. Springer, 1992.
11. R. Niedermeier, K. Reinhardt, and P. Sanders. Towards optimal locality in mesh-indexings. *Discrete Applied Mathematics*, 117:211–237, 2002.
12. Gh. Păun and A. Salomaa, editors. *Grammatical Models of Multi-Agent Systems*, chapter H. Fernau, R. Freund, and M. Holzer: Regulated array grammars of finite index, pages 157–181 (Part I) and 284–296 (Part II). London: Gordon and Breach, 1999.
13. H. Sagan. *Space-Filling Curves*. Springer, 1994.
14. G. Siromoney, R. Siromoney, and K. Krithivasan. Abstract families of matrices and picture languages. *Computer Graphics and Image Processing*, 1:284–307, 1972.
15. G. Siromoney, R. Siromoney, and K. Krithivasan. Picture languages with array rewriting rules. *Information and Control (now Information and Computation)*, 22(5):447–470, 1973.
16. R. Siromoney. On equal matrix languages. *Information and Control (now Information and Computation)*, 14:133–151, 1969.
17. R. Siromoney and K. G. Subramanian. Space-filling curves and infinite graphs. In H. Ehrig, M. Nagl, and G. Rozenberg, editors, *Graph grammars and their application to computer science*, volume 153 of *LNCS*, pages 380–391, 1983.
18. K. G. Subramanian, L. Revathi, and R. Siromoney. Siromoney array grammars and applications. *International Journal of Pattern Recognition and Artificial Intelligence*, 3:333–351, 1989.

19. Y. Takada. Learning even equal matrix languages based on control sets. In A. Nakamura et al., editors, *Parallel Image Analysis, ICPIA'92*, volume 652 of *LNCS*, pages 274–289. Springer, 1992.
20. I. H. Witten and B. Wyvill. On the generation and use of space-filling curves. *Software–Practice and Experience*, 13:519–525, 1983.