ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume 21, Number 3, 2018, 238–248

Picture Scanning Automata and Group Actions on Pictures

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

[1]Fachbereich 4 – Abteilung Informatikwissenschaften, CIRT, Universität Trier, D-54286 Trier, Germany
[2]Department of Mathematics, Madras Christian College,
Chennai - 600059, India
Email: fernau@informatik.uni-trier.de,
paramasivan@informatik.uni-trier.de, dgthomasmcc@yahoo.com

Abstract We are systematically discussing finite automata working on rectangular-shaped arrays (i. e., pictures), reading them with different scanning strategies. We show that all 32 different variants only describe two different classes of array languages. Our main proof tool is using the dihedral group D_4.

Key-words: Finite automata; formal picture processing; symmetries.

1. Introduction

Syntactic considerations of digital images have a tradition of about five decades. They should (somehow) reflect methods applied to picture processing. Here, we continue 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, as well as by scanning line by line following the same direction each time. We consider finite automata that work the described ways, extending previous studies [1].

This shows tighter connections between finite automata working on pictures and so-called array grammars of different types; for overviews on this topic, we refer to [2], [3]. Notice that our pictures are always of a rectangular shape.

Our main contributions can be summarized as follows: (a) We show that an amazing variety of picture scanning devices basically only describe two different array language families. (b) This result is obtained by making use of connections to the theory of dihedral groups. In fact, the idea of using symmetry groups for obtaining picture grammars can be traced back throughout the history of formal picture processing, see [4], [5], but somehow restricted to pictures themselves

rather than picture languages or devices for describing pictures. Some parts of this paper (in particular, all results concerning closure properties and inclusion relations) already appeared in a conference proceedings [6]. The current presentation is more crisp and can be seen also as an extension of [1]. Suggestions for future research convey mostly new ideas.

2. General Definitions

A *picture over alphabet* 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. 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 (non-empty) pictures over Σ. Every subset $L \subseteq \Sigma^{++}$ is a *picture language*.

2.1. Binary Operations on Pictures

Let $W := [a_{i,j}]_{m,n}$ and $W' := [b_{i,j}]_{m',n'}$ be two pictures over Σ. The *column concatenation* of W and W', denoted by $W \oplus W'$, is undefined if $m \neq m'$ and is otherwise obtained by writing W to the left of W', yielding the picture

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

The *row concatenation* of W and W', denoted by $W \ominus W'$, is undefined if $n \neq n'$ and is otherwise obtained by writing W above of W'. The row and column catenation operations can be also viewed as operations on languages. Also, we can define n-fold iterations (powers) of column catenation as W^n and n-fold iterations (powers) of row catenation as W_n. Accordingly, Σ_m^n is understood, as well as $\Sigma_m^+ = \bigcup_{n \geq 1} \Sigma_m^n$ and similarly Σ_+^n. In this sense, $\Sigma^{++} = \Sigma_+^+$.

2.2. Unary Operations and Connections to Group Theory

As pictures are (also) geometrical objects, several further unary operations can be introduced [7]: *quarter-turn* (rotate clockwise by $90°$) Q, *half-turn* (rotate by $180°$) H, *anti-quarter-turn* (rotate anti-clockwise by $90°$ (or rotate clockwise by $270°$) Q^{-1}, *transpose* T (reflection along the main diagonal), *anti-transpose* T' (reflection along the anti-diagonal), R_h (reflection along a horizontal (base) line), R_v(reflection along a vertical line). Together with the *identity* I, these (now eight) operators form a non-commutative group (with respect to composition), the well-known dihedral group D_4 (we refer to [8]); see Table 1.

In Table 1, $\circ$ is the function composition. So, if $f : X \to Y$ and $g : Y \to Z$ are two functions, then $g \circ f : X \to Z$ is defined by $(g \circ f)(x) = g(f(x))$ for all $x \in X$. One example showing how Table 1 works is shown below for $W := [a_{i,j}]_{m,n}$.

$$(T \circ R_v)(W) = T(R_v(W)) = Q^{-1}(W).$$

$\circ$	I	Q^{-1}	H	Q	R_v	R_h	T	T'
I	I	Q^{-1}	H	Q	R_v	R_h	T	T'
Q^{-1}	Q^{-1}	H	Q	I	T	T'	R_h	R_v
H	H	Q	I	Q^{-1}	R_h	R_v	T'	T
Q	Q	I	Q^{-1}	H	T'	T	R_v	R_h
R_v	R_v	T'	R_h	T	I	H	Q	Q^{-1}
R_h	R_h	T	R_v	T'	H	I	Q^{-1}	Q
T	T	R_v	T'	R_h	Q^{-1}	Q	I	H
T'	T'	R_h	T	R_v	Q	Q^{-1}	H	I

Table 1.: Composition table of unary operators

Let $\mathcal{O} = \{I, Q^{-1}, H, Q, R_v, R_h, T, T'\}$ be the set of these 8 unary operators comprising D_4. The operators in D_4 are usually partitioned into the four rotations (including the identity) $\{I, Q^{-1}, H, Q\}$, which form the cyclic subgroup $\mathbb{Z}_4$ of D_4, and four reflections $\{R_v, R_h, T, T'\}$. The four operations $\{I, H, R_v, R_h\}$ form another interesting subgroup, the Klein group K_4. These operations can be also applied (picture-wise) to picture languages and (language-wise) to families of picture languages. It is interesting to add the fact that one single rotation Q generates all rotations (as a subgroup of D_4) and that all of D_4 are generated by one rotation Q and one reflection T.

2.3. Combining Operators

We now consider combining unary and binary operations, as collected in Table 2.

	$W_1 \ominus W_2$	$W_1 \oplus W_2$
I	$I(W_1) \ominus I(W_2)$	$I(W_1) \oplus I(W_2)$
Q	$Q(W_2) \oplus Q(W_1)$	$Q(W_1) \ominus Q(W_2)$
Q^{-1}	$Q^{-1}(W_1) \oplus Q^{-1}(W_2)$	$Q^{-1}(W_2) \ominus Q^{-1}(W_1)$
H	$H(W_2) \ominus H(W_1)$	$H(W_2) \oplus H(W_1)$
T	$T(W_1) \oplus T(W_2)$	$T(W_1) \ominus T(W_2)$
T'	$T'(W_2) \oplus T'(W_1)$	$T'(W_2) \ominus T'(W_1)$
R_v	$R_v(W_1) \ominus R_v(W_2)$	$R_v(W_2) \oplus R_v(W_1)$
R_h	$R_h(W_2) \ominus R_h(W_1)$	$R_h(W_1) \oplus R_h(W_2)$

Table 2.: Table of Operators

3. General Boustrophedon Finite Automata

We now recall from [6] the definition of general boustrophedon finite automaton, a new, parameterized automaton model for picture processing.

A *general boustrophedon finite automaton*, or GBFA for short, can be specified as an 8-tuple $M = (Q, \Sigma, R, s, F, \#, \square, D)$, where Q is a finite set of states, partitioned into Q_f and Q_b, Σ is

an input alphabet, and $R \subseteq Q \times (\Sigma \cup \{\#\}) \times Q$ is a finite set of rules. A rule $(q, a, p) \in R$ is usually written as $qa \to p$. We impose some additional restrictions. If $q \in Q_f$ and $a \in \Sigma$, then $qa \to p \in R$ is only possible if $p \in Q_f$. Such transitions are also called *forward transitions* and collected within R_f. Similarly, if $q \in Q_b$ and $a \in \Sigma$, $qa \to p \in R$ is only possible if $p \in Q_b$ (*backward transitions*, collected in R_b). Finally, *border transitions* (collected in $R_\#$) are of the form $q\# \to p$ with $q \in Q_f$ iff $p \in Q_b$. Namely, the special symbol $\# \notin \Sigma$ indicates the border of the rectangular picture that is processed, $s \in Q_f$ is the initial state, F is the set of final states, and $D \in \mathcal{D}$ indicates the move directions. Here,

$$\mathcal{D} = \left\{ \begin{pmatrix} s\to & \downarrow \\ \downarrow & \leftarrow \end{pmatrix}, \begin{pmatrix} s\downarrow & \to \\ \to & \uparrow \end{pmatrix}, \begin{pmatrix} \downarrow & \leftarrow s \\ \to & \downarrow \end{pmatrix}, \begin{pmatrix} \leftarrow & \downarrow s \\ \uparrow & \leftarrow \end{pmatrix}, \begin{pmatrix} \to & \downarrow \\ s\uparrow & \to \end{pmatrix}, \begin{pmatrix} \uparrow & \leftarrow \\ s\to & \uparrow \end{pmatrix}, \begin{pmatrix} \downarrow & \leftarrow \\ \leftarrow & \uparrow s \end{pmatrix}, \begin{pmatrix} \to & \uparrow \\ \uparrow & \leftarrow s \end{pmatrix} \right\}$$

We are now going to discuss the notions of configurations, valid configurations and an according configuration transition to formalize the work of GBFAs, based on snapshots of their work.

Let $\square$ be a new symbol indicating an *erased* position and let $\Sigma_{\#,\square} := \Sigma \cup \{\#, \square\}$. Then $C_M := Q \times (\Sigma^{++}_{\#,\square} \cap \{\#\}^+ \ominus (\{\#\}_+ \oplus (\Sigma \cup \{\square\})^{++} \oplus \{\#\}_+) \ominus \{\#\}^+) \times \{f, b\}$ is the set of configurations of M. Hence, the first and last columns and the first and last rows are completely filled with $\#$, and these are the only positions that contain $\#$.

The *initial configuration* is determined by the input array $A \in \Sigma^{++}$. More precisely, if A has m rows and n columns, then

$$\big(s, \#^{n+2} \ominus (\#_m \oplus A \oplus \#_m) \ominus \#^{n+2}, f\big)$$

shows the according initial configuration $c_{init}(A)$. Similarly, a *final configuration* is then given by

$$\big(q_f, \#^{n+2} \ominus (\#_m \oplus \square^n_m \oplus \#_m) \ominus \#^{n+2}, d\big)$$

for some $q_f \in F$ and $d \in \{b, f\}$.

The processing of the automaton is then crucially depending on $D \in \mathcal{D}$. The arrow that appears together with s indicates the direction of the forward processing of the first, third, fifth etc. line. For instance, the first listed direction contains $s \to$, determining that the odd-numbered rows of the input array are scanned left to right. Similarly, the second listed direction contains $s \downarrow$, determining that the odd-numbered columns of the input array are scanned top to bottom. When the automaton encounters a border symbol, it processes the next line in the reversed way (backward processing). This is also indicated in the little pictures that describe $D \in \mathcal{D}$. For instance, in the first case, the $\downarrow$ in the first row indicates that after hitting the border, the automaton moves downwards, processing (as indicated by the $\leftarrow$ in the last row) now from right to left, until the border is hit again, which means to move downwards one more row, as indicated by the $\downarrow$ in the last row. The other $D \in \mathcal{D}$ can be interpreted in a similar fashion, as explained below. Let us now formalize this description. Notice that an odd-numbered row of the input array corresponds to an even-numbered row if we consider the input array bordered by a $\#$-layer according to our definition of a configuration.

- If (p, A, f) and (q, A', f) are two configurations such that A and A' are identical but for one position (i, j), $2 \le i \le m + 1$, $2 \le j \le n + 1$, where $A'[i, j] = \square$ while $A[i, j] \in \Sigma$ and $A[i, j - 1] \in \{\#, \square\}$, then $(p, A, f) \vdash_M (q, A', f)$ if $pA[i, j] \to q \in R_f$. Moreover, i is even.

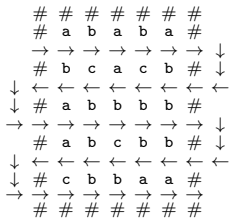
- Conversely, if (p, A, b) and (q, A', b) are two configurations such that A and A' are identical but for one position (i, j), $2 \le i \le m + 1$, $2 \le j \le n + 1$, where $A'[i, j] = \square$ while

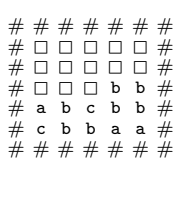
$A[i,j] \in \Sigma$ and $A[i,j+1] \in \{\#, \square\}$, then $(p,A,b) \vdash_M (q,A',b)$ if $pA[i,j] \to q \in R_b$. Moreover, i is odd.

- If (p,A,f) and (q,A,b) are two configurations, then $(p,A,f) \vdash_M (q,A,b)$ or $(p,A,b) \vdash_M (q,A,f)$ if $p\# \to q \in R_\#$.

The reflexive transitive closure of the relation $\vdash_M$ is denoted by $\vdash_M^*$. $A \in \Sigma^{++}$ is accepted by a GBFA M with direction $D_{BFA} := \begin{pmatrix} s\rightarrow & \downarrow \\ \downarrow & \leftarrow \end{pmatrix}$ if $c_{init}(A) \vdash_M^* c$ such that c is a final configuration.

The following illustrates how such a GBFA scans some input picture and how a picture of a valid configuration looks like; it can be seen that the sequence of $\square$ only indicates how far the input has been processed:





The representation on the right-hand side of the previous picture contains all information necessary to describe a configuration apart from the state.

The other modes are defined here by applying transformations according to the following table.

D =	$\begin{pmatrix} s\downarrow & \rightarrow \\ \rightarrow & \uparrow \end{pmatrix}$	$\begin{pmatrix} \downarrow & \leftarrow s \\ \rightarrow & \downarrow \end{pmatrix}$	$\begin{pmatrix} \leftarrow & \downarrow s \\ \uparrow & \leftarrow \end{pmatrix}$	$\begin{pmatrix} \rightarrow & \downarrow \\ s\uparrow & \rightarrow \end{pmatrix}$	$\begin{pmatrix} \uparrow & \leftarrow \\ s\rightarrow & \uparrow \end{pmatrix}$	$\begin{pmatrix} \downarrow & \leftarrow \\ \leftarrow & \uparrow s \end{pmatrix}$	$\begin{pmatrix} \rightarrow & \uparrow \\ \uparrow & \leftarrow s \end{pmatrix}$
$f_D(A)$ =	$T(A)$	$R_v(A)$	$Q^{-1}(A)$	$Q(A)$	$R_h(A)$	$T'(A)$	$H(A)$

A is accepted by a GBFA M with a different direction D if $f_D(A)$ is accepted by the GBFA M_{BFA} that coincides with M in every detail except for the direction, which is now D_{BFA}. This means, for instance, in the case of $D = \begin{pmatrix} s\downarrow & \rightarrow \\ \rightarrow & \uparrow \end{pmatrix}$, that instead of scanning the input array A column by column, the first column top-down, the second bottom-up, and so forth, we rather transpose A and then scan the transposed array row by row, the first row left-right, the second right-left, and so forth.

The GBFA M is deterministic, or a GBDFA for short, if for each $p \in Q$ and $a \in \Sigma \cup \{\#\}$, there is at most one $q \in Q$ with $pa \to q \in R$. So, we define language classes like $\mathcal{L}_D(\mathrm{GBFA})$ of picture languages accepted by GBFAs working with direction D, as well as $\mathcal{L}(\mathrm{GBFA}) := \bigcup_{D \in \mathcal{D}} \mathcal{L}_D(\mathrm{GBFA})$, sometimes adding D for 'deterministic'.

4. Families of Picture Languages

We are now studying the question if the eight language families that we can obtain in these ways are really different from each other or not. Also, the situation of $\mathcal{L}(\mathrm{GBFA})$ needs to be investigated, as well as the role of determinism. We will see that the group-theoretic excursion from above will simplify our reasoning a lot. From the definitions themselves, we can immediately derive the following characterization.

Theorem 1. *The class $\mathcal{L}_{D_{BFA}}(\mathrm{GBFA})$ coincides with the following classes of picture languages:* $T\left(\mathcal{L}_{\left(\begin{smallmatrix} s\downarrow & \rightarrow \\ \rightarrow & \uparrow \end{smallmatrix}\right)}(\mathrm{GBFA})\right)$, $R_v\left(\mathcal{L}_{\left(\begin{smallmatrix} \downarrow & \leftarrow s \\ \rightarrow & \downarrow \end{smallmatrix}\right)}(\mathrm{GBFA})\right)$,

$Q^{-1}\left(\mathcal{L}_{\binom{\leftarrow\ \downarrow s}{\uparrow\ \leftarrow}}(\mathrm{GBFA})\right)$, $Q\left(\mathcal{L}_{\binom{\rightarrow\ \downarrow}{s\uparrow\ \rightarrow}}(\mathrm{GBFA})\right)$, $R_h\left(\mathcal{L}_{\binom{\uparrow\ \leftarrow}{s\rightarrow\ \uparrow}}(\mathrm{GBFA})\right)$, $T'\left(\mathcal{L}_{\binom{\downarrow\ \leftarrow}{\leftarrow\ \uparrow s}}(\mathrm{GBFA})\right)$, *and* $H\left(\mathcal{L}_{\binom{\rightarrow\ \uparrow}{\uparrow\ \leftarrow s}}(\mathrm{GBFA})\right)$.

Due to the connections to group theory described above, we can easily infer from the previous theorem characterizations of the seven other classes, referring back to $\mathcal{L}_{D_{BFA}}(\mathrm{GBFA})$, for instance, $\mathcal{L}_{\binom{s\downarrow\ \rightarrow}{\rightarrow\ \uparrow}}(\mathrm{GBFA}) = T\left(\mathcal{L}_{D_{BFA}}(\mathrm{GBFA})\right)$. We refrain from presenting a list of similar characterizations in the following; see [6].

In [1], *boustrophedon finite automata* have been introduced that basically work as GBFAs do when working in mode D_{BFA}, apart from the fact that we have integrated in our definition of GBFA that the automaton 'knows' if it processes an even- or odd-numbered row. In other words, they are aware of the direction of their movement (left to right or right to left). However, as the basic model is finite automata, it is not difficult to show BFAs can be turned into one that is direction-aware, by counting the number of reads of #-symbols modulo two. Hence, we can easily profit from results in [1]. We give two examples of such results now.

Theorem 2. *For each direction mode D, we know:* $\mathcal{L}_D(\mathrm{GBFA}) = \mathcal{L}_D(\mathrm{GBDFA})$.

Proof. By the previous reasoning, [1] yields that $\mathcal{L}_D(\mathrm{GBFA}) = \mathcal{L}_D(\mathrm{GBDFA})$ is true for $D = D_{BFA}$. As indicated above, we have characterizations of $\mathcal{L}_D(\mathrm{GBFA})$ in terms of $\mathcal{L}_{D_{BFA}}(\mathrm{GBFA})$ by applying appropriate unary operators. These characterizations are also valid for the corresponding deterministic classes. □

Theorem 3. *For each direction mode D, we know:* $\mathcal{L}_D(\mathrm{GBFA})$ *is closed under the Klein group operations I, R_v, R_h, H.*

Proof. By [1, Theorem 27], $\mathcal{L}_{D_{BFA}}(\mathrm{GBFA})$ is closed under R_v and R_h. By Table 1, it is closed under H, as well. As every class $\mathcal{L}_D(\mathrm{GBFA})$ can be characterized in terms of $\mathcal{L}_{D_{BFA}}(\mathrm{GBFA})$ by applying appropriate unary operators, the claim follows. □

This allows us to conclude that four of our classes coincide.

Corollary 4. $\mathcal{L}_D(\mathrm{GBFA}) = R_v(\mathcal{L}_D(\mathrm{GBFA})) = R_h(\mathcal{L}_D(\mathrm{GBFA})) = H(\mathcal{L}_D(\mathrm{GBFA}))$ *for any direction mode D.*

In particular, this corollary is true for $D = D_{BFA}$. But, how many picture language families can we describe with our formalism?

Already back in the 1970s, regular matrix languages were introduced and studied in a sequence of papers, mainly by groups of Indian authors; see [7], [9]–[11]. Following [1], we call this picture language class $\mathcal{L}(\mathrm{RMG})$.

Theorem 5. $\mathcal{L}(\mathrm{RMG}) = T(\mathcal{L}_{D_{BFA}}(\mathrm{GBFA})) = Q(\mathcal{L}_{D_{BFA}}(\mathrm{GBFA})) = Q^{-1}(\mathcal{L}_{D_{BFA}}(\mathrm{GBFA})) = T'(\mathcal{L}_{D_{BFA}}(\mathrm{GBFA}))$.

Proof. By [1, Theorem 26], $\mathcal{L}(\mathrm{RMG}) = T(\mathcal{L}_{D_{BFA}}(\mathrm{GBFA}))$. As $T \circ R_h = Q$, $T \circ R_v = Q^{-1}$, $T \circ H = T'$ and by Theorem 3, the remaining claims follow. □

We can now return to the question we posed above and present the following answer.

Theorem 6. *For each direction mode D, we have either $\mathcal{L}_D(\mathrm{GBFA}) = \mathcal{L}_{D_{BFA}}(\mathrm{GBFA})$ or $\mathcal{L}_D(\mathrm{GBFA}) = T(\mathcal{L}_{D_{BFA}}(\mathrm{GBFA}))$. Moreover, both possible picture language classes are indeed different and properly included in $\mathcal{L}(\mathrm{GBFA})$.*

Proof. As each possible picture language class that we consider can be written as $O(\mathcal{L}_{D_{BFA}}(\mathrm{GBFA}))$ for one of the unary operators O that we consider, Corollary 4 and Theorem 5 together show that at most two different picture language families exist. The separating examples from [1, Lemmas 45 and 46] prove that these are two different language families, also separating them from their union, which is $\mathcal{L}(\mathrm{GBFA})$. □

Corollary 7. $\forall O \in \mathcal{O} : O(\mathcal{L}(\mathrm{GBFA})) = \mathcal{L}(\mathrm{GBFA})$.

A useful auxiliary model introduced in [1] was so-called *returning finite automata* (RFA). These automata formally look just like BFA, but when they process a picture row-by-row, they always start processing at the left end of each row. We are going to generalize their work in the following, again by introducing working modes for them.

5. Generalized Returning Finite Automata

Now, a pair of directions like $D = (s\rightarrow\downarrow)$ is sufficient, indicating that an input array is always processed row by row, top down, where each row is scanned from left to right; moreover, the procedure is (here) started at the upper left corner of the array, as indicated by the position of s. The processing mode just describes coincide with that of RFAs from [1]. Leaving out definitorial details for now, which are (formally) coinciding with GBFAs, we arrive at language families like $\mathcal{L}_D(\mathrm{GRFA})$. Again, we have deterministic variants. There are (again) eight natural processing modes D:

$$(s\rightarrow\downarrow),\ (s\downarrow\rightarrow),\ (\downarrow\leftarrow s),\ (\leftarrow\downarrow s),\ (s\rightarrow\uparrow),\ (s\uparrow\rightarrow),\ (\uparrow\leftarrow s),\ (\leftarrow\uparrow s).$$

These can be naturally partitioned into the row-first modes $\mathcal{D}_{row-f} = \{(s\rightarrow\downarrow),\ (\downarrow\leftarrow s),\ (s\rightarrow\uparrow),\ (\uparrow\leftarrow s)\}$ and the four other column-first modes in $\mathcal{D}_{col-f}$.

Example 8. *The set of all arrays over $\{\mathtt{a},\mathtt{b}\}$ that has exactly one row completely filled with* `b`*'s and* `a`*'s everywhere else is accepted by a GRFA M as shown in Figure 1. This automaton may work in any direction mode from $\mathcal{D}_{row-f}$.*

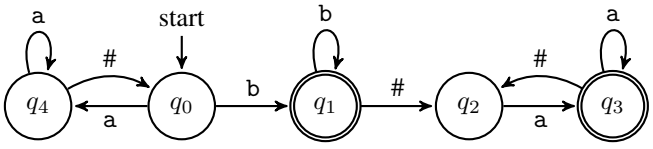


Figure 1.: GRFA M that accepts the language in Example 8

As with GBFAs, we can alternatively describe the work of a GRFA working in mode D by first performing a unary operation on the image and then processing the image in the mode $D_{RFA} = (s\rightarrow\downarrow)$ that corresponds to RFAs as introduced in [1]. Therefore, we obtain the following characterizations for these modes.

Theorem 9. *The class* $\mathcal{L}_{D_{RFA}}(\mathrm{GRFA})$ *coincides with the following classes:* $T\left(\mathcal{L}_{(\,s\downarrow\,\rightarrow\,)}(\mathrm{GRFA})\right)$, $R_v\left(\mathcal{L}_{(\downarrow\,\leftarrow s\,)}(\mathrm{GRFA})\right)$, $Q^{-1}\left(\mathcal{L}_{(\leftarrow\,\downarrow s\,)}(\mathrm{GRFA})\right)$, $Q\left(\mathcal{L}_{(\,s\uparrow\,\rightarrow\,)}(\mathrm{GRFA})\right)$, $R_h\left(\mathcal{L}_{(\,s\rightarrow\,\uparrow\,)}(\mathrm{GRFA})\right)$, $T'\left(\mathcal{L}_{(\leftarrow\,\uparrow s\,)}(\mathrm{GRFA})\right)$, *and* $H\left(\mathcal{L}_{(\uparrow\,\leftarrow s\,)}(\mathrm{GRFA})\right)$.

As we have shown in [1] that BFAs and RFAs describe the same class of picture languages, we can immediately deduce the following classification result of the potentially eight language classes that we just introduced, based on what we derived previously for GBFAs, as well as based on the previous theorem.

Corollary 10. *1. For each* $D \in \mathcal{D}_{row-f}$, $\mathcal{L}_D(\mathrm{GRFA}) = \mathcal{L}_D(\mathrm{GRDFA}) = \mathcal{L}(\mathrm{BFA})$.
2. For each $D \in \mathcal{D}_{col-f}$, $\mathcal{L}_D(\mathrm{GRFA}) = \mathcal{L}_D(\mathrm{GRDFA}) = \mathcal{L}(\mathrm{RMG})$.

In fact, the proof of $\mathcal{L}(\mathrm{RMG}) = T(\mathcal{L}(\mathrm{BFA}))$ that we presented in [1] consisted of two steps: (a) $\mathcal{L}(\mathrm{RMG}) = T(\mathcal{L}(\mathrm{RFA}))$ and (b) $\mathcal{L}(\mathrm{RFA}) = \mathcal{L}(\mathrm{BFA})$.

6. Catenation Operations

By [1, Theorem 59] and the results of the previous section, we can immediately derive:

Corollary 11. $\forall L_1, L_2 \in \mathcal{L}_{D_{RFA}}(\mathrm{GRFA}), L_1 \ominus L_2 \in \mathcal{L}_{D_{RFA}}(\mathrm{GRFA})$.

Corollary 12. $\forall L_1, L_2 \in Q(\mathcal{L}_{D_{RFA}}(\mathrm{GRFA})), L_1 \oplus L_2 \in Q(\mathcal{L}_{D_{RFA}}(\mathrm{GRFA}))$.

Proof. Consider $L_1, L_2 \in Q(\mathcal{L}_{D_{RFA}}(\mathrm{GRFA}))$, i.e., $Q(L_1), Q(L_2) \in \mathcal{L}_{D_{RFA}}(\mathrm{GRFA})$. By the previous corollary, $Q(L_2) \oplus Q(L_1) \in \mathcal{L}_{D_{RFA}}(\mathrm{GRFA})$. According to Table 2, $Q(L_2) \oplus Q(L_1) = Q(L_1 \ominus L_2)$. Hence, $L_1 \ominus L_2 \in Q(\mathcal{L}_{D_{RFA}}(\mathrm{GRFA}))$. □

Define a mapping $str : \Sigma_2^+ \rightarrow (\Sigma \cup \{\#\})^+, W \mapsto w$ where $w = w_1 \# w_2$, $|W|_c = n$, $n \geq 2$ and $|W|_r = 2$; moreover, $|w_1| = |w_2| = n$, so that $|w| = 2n + 1$.

Lemma 13. *If* $L \in \mathcal{L}_{D_{RFA}}(\mathrm{GRFA})$ *with* $L \subseteq \Sigma_2^+$, *then* $str(L)$ *is context-free.*

Proof. A pushdown automaton uses its finite control as the GRFA and simply checks if the two rows have equal length by using its pushdown store. □

Theorem 14. $\exists L_1, L_2 \in \mathcal{L}_{D_{RFA}}(\mathrm{GRFA}) : L_1 \oplus L_2 \notin \mathcal{L}_{D_{RFA}}(\mathrm{GRFA})$.

Proof. Let $L = \left\{ \begin{smallmatrix} 1\ 0^\ell \\ 1\ 0^\ell \end{smallmatrix} : \ell \geq 1 \right\}$. Clearly, $L \in \mathcal{L}_{D_{RFA}}(\mathrm{GRFA})$. However, $str(L \oplus L)$ is a variant of the crossing dependency language known to be not context-free. By Lemma 13, $L \oplus L \notin \mathcal{L}_{D_{RFA}}(\mathrm{GRFA})$. □

Corollary 15. $\exists L_1, L_2 \in Q(\mathcal{L}_{D_{RFA}}(\mathrm{GRFA})) : L_1 \ominus L_2 \notin Q(\mathcal{L}_{D_{RFA}}(\mathrm{GRFA}))$.

Theorem 16. $\mathcal{L}(\mathrm{GRFA})$ *is closed neither under column catenation nor under row catenation.*

Proof. Consider L from the proof of Theorem 14. By Corollary 11, $L' = L \ominus \{0\}_+^+ \ominus \{1\}^+ \ominus \{0\}_+^+ \in \mathcal{L}_{D_{RFA}}(\mathrm{GRFA})$. If $L' \oplus L' \in \mathcal{L}_{D_{RFA}}(\mathrm{GRFA})$, then also $L \oplus L \in \mathcal{L}_{D_{RFA}}(\mathrm{GRFA})$, contradicting the reasoning from Theorem 14. Interchange arguments as applied in [1, Lemma 45] easily show that $L' \oplus L' \notin Q(\mathcal{L}_{D_{RFA}}(\mathrm{GRFA}))$, intuitively because horizontal lines in arbitrary positions cannot be checked by finite automata working column by column. Hence, $L' \oplus L' \notin \mathcal{L}(\mathrm{GRFA})$. □

7. Conclusions and Future Research

In this paper, we showed how to use properties of the dihedral group D_4 and subgroups of it in order to obtain results on several families of picture language, with the idea of these groups acting on said languages.

We also made use of the various interconnections between picture language classes as obtained in [1, Theorems 20 and 26]. In this context, it would be interesting to know if the blow-ups that we incur in these constructions are avoidable. To make this point clearer, let us make these blow-ups explicit in the following.

- Each RFA A with q states can be converted into a regular matrix grammar describing $T(L(A))$ with $O(q \times q)$ many nonterminals.
- Each regular matrix grammar G with n nonterminals can be converted into an RFA describing $T(L(G))$ with $O(n \times n)$ many states.
- Each RFA with q states can be converted into an equivalent BFA with $O(q^3)$ many states.
- Each BFA with q states can be converted into an equivalent RFA with $O(q^3)$ many states.

Are these blow-ups unavoidable? Does this mean that the blow-up occurred by starting with a BFA A with q states, ending up in a regular matrix grammar describing $T(L(A))$ with $O(q^6)$ many nonterminals, is really the best-possible construction? A similar blow-up occurs in the reverse direction according to our constructions. Is this unavoidable?

From a practical perspective, the deterministic automata variants are even more important. Here, it is important (yet sad) to note that their minimization problem is NP-hard, see [1, Cor. 75 and 77]. This makes it also interesting to observe that our conversions between BFAs and RFAs destroy determinism. Can this be avoided?

A possible further direction of research could be to integrate these models into pattern recognition algorithms. As exhibited by Flasiński in [12], this would necessitate the development of Grammatical Inference algorithms. In this context, it appears that we have to overcome the following technical problem: mostly, learners converge to canonical hypotheses like minimum-state automata and hence, efficient learners also comprise efficient state minimization algorithms. As these do not exist, as mentioned above, we might want to find other canonical objects. Defining analogues to, say, universal automata and then proving learnability results as done, for instance, in [13], is again an interesting topic for future research. In fact, relatively little has been done on the learnability of picture languages, even when using the idea of patterns [14], [15].

A further extension could be to work on automata on other types of picture structures, for instance, hexagonal ones or structures based on Cayley graphs; see [16], [17].

Acknowledgement

This paper resumes, simplifies and extends the group-theoretic considerations started in [6]. Support of the second and third author by a grant from DAAD and by some money from CIRT (Trier) and overhead money from a DFG project are gratefully acknowledged.

References

[1] H. Fernau, M. Paramasivan, M. L. Schmid, and D. G. Thomas, "Simple picture processing based on finite automata and regular grammars", *Journal of Computer and System Sciences*, vol. 95, pp. 232–258, 2018.

[2] K. Inoue and I. Takanami, "A survey of two-dimensional automata theory", *Information Sciences*, vol. 55, no. 1-3, pp. 99–121, 1991.

[3] J. Kari and V. Salo, "A survey on picture-walking automata", in *Algebraic Foundations in Computer Science - Essays Dedicated to Symeon Bozapalidis on the Occasion of His Retirement*, W. Kuich and G. Rahonis, Eds., ser. LNCS, vol. 7020, Springer, 2011, pp. 183–213.

[4] M. F. Dacey, "The syntax of a triangle and some other figures", *Pattern Recognition*, vol. 2, pp. 11–31, 1970.

[5] K. G. Subramanian, "A note on regular kolam array grammars generating right triangles", *Pattern Recognition*, vol. 11, no. 5-6, pp. 343–345, 1979.

[6] H. Fernau, M. Paramasivan, and D. G. Thomas, "Picture scanning automata", in *Computational Modeling of Objects Presented in Images. Fundamentals, Methods, and Applications - 5th International Symposium, CompIMAGE 2016*, R. P. Barneva, V. E. Brimkov, and J. M. R. S. Tavares, Eds., ser. LNCS, vol. 10149, Springer, 2017, pp. 132–147.

[7] G. Siromoney, R. Siromoney, and K. Krithivasan, "Picture languages with array rewriting rules", *Information and Control (now Information and Computation)*, vol. 22, no. 5, pp. 447–470, 1973.

[8] M. A. Armstrong, *Groups and Symmetry*. Springer-Verlag, 1988.

[9] K. Krithivasan and R. Siromoney, "Array automata and operations on array languages", *International Journal of Computer Mathematics*, vol. 4, no. A, pp. 3–40, 1974.

[10] G. Siromoney, R. Siromoney, and K. Krithivasan, "Abstract families of matrices and picture languages", *Computer Graphics and Image Processing*, vol. 1, pp. 284–307, 1972.

[11] K. G. Subramanian, L. Revathi, and R. Siromoney, "Siromoney array grammars and applications", *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 3, pp. 333–351, 1989.

[12] M. Flasiński, "Chapter 1.1; syntactic pattern recognition: Paradigm issues and open problems", in *Handbook of Pattern Recognition and Computer Vision, 5th Edition*, C. H. Chen, Ed., World Scientific, 2016, pp. 3–25.

[13] J. Björklund, H. Fernau, and A. Kasprzik, "Polynomial inference of universal automata from membership and equivalence queries", *Information and Computation*, vol. 246, pp. 3–19, 2016.

[14] H. Fernau, M. L. Schmid, and K. G. Subramanian, "Two-dimensional pattern languages", in *Fifth Workshop on Non-Classical Models for Automata and Applications, NCMA*, S. Bensch, F. Drewes, R. Freund, and F. Otto, Eds., ser. books@ocg.at, vol. 294, Österreichische Computer Gesellschaft, 2013, pp. 117–132.

[15] R. Siromoney, K. G. Subramanian, and L. Mathew, "Learning of pattern and picture languages", *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 6, no. 2&3, pp. 275–284, 1992.

[16] K. S. Dersanambika, K. Krithivasan, C. Martín-Vide, and K. G. Subramanian, "Hexagonal pattern languages", in *Combinatorial Image Analysis, 10th International Workshop, IWCIA*, R. Klette and J. D. Zunic, Eds., ser. LNCS, vol. 3322, Springer, 2004, pp. 52–64.

[17] R. Freund and M. Oswald, "Algebraic representation of regular array languages on Cayley graphs", in *Discrete Mathematics and Computer Science. In Memoriam Alexandru Mateescu (1952-2005)*, Gh. Păun, G. Rozenberg, and A. Salomaa, Eds., The Publishing House of the Romanian Academy, 2014, pp. 165–187.