A Survey on Picture-Walking Automata

Jarkko Kari and Ville Salo

University of Turku

Abstract. Picture walking automata were introduced by M. Blum and C. Hewitt in 1967 as a generalization of one-dimensional two-way finite automata to recognize pictures, or two-dimensional words. Several variants have been investigated since then, including deterministic, nondeterministic and alternating transition rules; four-, three- and two-way movements; single- and multi-headed variants; automata that must stay inside the input picture, or that may move outside. We survey results that compare the recognition power of different variants, consider their basic closure properties and study decidability questions.

Key words: Picture-walking automata, 2-dimensional automata, picture languages

1 Introduction

Informally, a picture is a matrix over a finite alphabet, and a picture language is a set of matrices over the same alphabet. A picture-walking automaton is a finite state automaton moving on the cells of the given picture according to a local rule, accepting if it reaches a final state [1].

The theory of picture languages is a branch of formal language theory which studies natural picture language families and connections between them. Most of this theory has concentrated on classes having to do with 'finite state', in an effort to find a natural counterpart for the one-dimensional regular languages. The classes obtained from picture-walking automata are usually not considered to be a very natural counterpart due to their rather weak closure properties, but the recognizable picture languages take this place instead. In fact, many interesting proofs about picture-walking automata have more of a 'navigational' than a language-theoretic feel to them.

In the first section, we deal with very basic and natural questions. We only consider Boolean closure properties and the lattice of inclusions between the four basic automata classes we define: DFA, NFA, UFA and AFA, that is, deterministic, non-deterministic, universally quantifying and alternating finite state automata, respectively. We present a slightly more refined view to these questions by considering unary rectangles and non-unary square pictures separately with the aim of clarifying the strengths and weaknesses of each approach. In each case we obtain a slightly different set of non-closure properties, which taken together give a rather complete set of results for the class of all pictures.

The second section introduces a very modest-looking change to the definition of an automaton: allowing it to exit the picture. We will see that this change is relevant for AFA, but not for the other classes, although this is not altogether straightforward to prove. In the third and fourth sections, we introduce much greater changes to the definition. In the third section, we restrict the directions in which the automaton is allowed to move. We give a brief introduction to the results known for these classes. In particular, for three-way automata, we give a complete set of Boolean closure properties and complete comparison results between all three- and four-way automata classes, mostly without proofs. In the fourth section we give our automata a finite set of markers they can move around the picture. In Section 5, we review decidability results, and recall connections between 2-dimensional DFA and Minsky machines.

Finally, we mention that a survey on two-dimensional picture-walking automata theory already exists [2], although it's main focus is not on finite state automata but on general Turing machines. Since few papers on picture-walking automata have been published after [2], it is still mostly up-to-date, except for some open problems which have since been solved (many of which are presented here). There is also a good survey of general two-dimensional language theory in the Handbook of Formal Languages [3].

2 Definitions

A *picture* is the two-dimensional analog of a finite word: a not necessarily square matrix over a finite alphabet. We may usually assume a binary alphabet $\Sigma = \{0, 1\}$. We write Σ_*^* for the full language of all matrices over Σ, and define a *picture language* as a subset of a full language. A class of picture languages, usually called a *picture class*, is a collection of picture languages.

For $p \in \Sigma_*^*$ we write $p[(i, j)] = p_{i,j}$ for the contents of the cell in position (i, j) in p, with the usual matrix indexing. We also draw p as a matrix, and thus, for instance, the cell with index $(1, 1)$ is considered the top left cell and the first axis is the vertical one, ascending downward. If (i, j) is not a cell of p, we define $p[(i, j)]$ to be a special symbol # (that is, in practise we index pictures by $\mathbb{Z}^2$). We write $\mathrm{dom}(p)$ for the set of indexes of (non-#) cells in p, called the *domain* of p, and $\mathrm{edom}(p)$ for the cells of $\mathrm{dom}(p)$ and all their neighbors. The width and height of a picture p are denoted by $|p|$ and $\overline{p}$, respectively. The set $\mathrm{edom}(p) - \mathrm{dom}(p)$ is called the *border* of p.

Let us start by defining the main picture-walking automata considered in this survey, and their corresponding picture classes. Our automata have a single head, and they walk on the positions of the picture according to a local rule, accepting if they reach a final state. Existential and universal states can be used to make (perfect) guesses and to check multiple local properties 'simultaneously', respectively.

Definition 1. *An alternating finite state automaton (an AFA) A is a tuple* $(Q, \Sigma, E, U, I, F, \delta)$ *where Q is the set of* states *partitioned into E and U, the*

sets of universal *and* existential *states. The sets* $I, F \subset Q$ *are called the sets of* initial *and* final *states. The function* δ *is called the* transition function *or the* local rule *and it has type* $\delta : Q \times \Sigma \to 2^{Q \times \{(1,0),(-1,0),(0,1),(0,-1)\}}$.

If $Q = E$ *(and thus* $U = \emptyset$*), the automaton is said to be* non-deterministic *(an NFA), and if* $Q = U$ *and* $|I| = 1$*, it is said to be* universally quantifying *(a UFA). If all images of* δ *are singletons or empty sets and* $|I| = 1$*, the automaton is called* deterministic *(a DFA). We use the variable XFA to state things for all the automata classes simultaneously.*

Definition 2. *Let* $A = (Q, \Sigma, E, U, I, F, \delta)$ *be an AFA. An instantaneous description (an ID) of* A *on a picture* $p \in \Sigma_*^*$ *is a pair* $(q, x) \in Q \times edom(p)$*. An ID* $l_2 = (q_2, x_2)$ *is a successor of another ID* $l_1 = (q_1, x_1)$ *if* $(q_2, x_2 - x_1) \in \delta(q_1, p[x_1])$*. We then write* $l_1 \to l_2$*. For a vertex-labeled tree* r*, we write* $a_1 \to a_2$ *if* a_2 *is a child of* a_1 *in* r*, and we write* $l(a)$ *for the label of node* $a \in r$*.*

Now, an accepting run of A *on* p *is a finite tree* r *labeled with ID's such that*

- *the root of* r *has its label in* $I \times \{(1,1)\}$*.*
- *if* $a_1 \in r$ *is not a leaf and* $l(a_1) \in E \times edom(p)$*, then*

$$\exists a_2 \in r : a_1 \to a_2 \wedge l(a_1) \to l(a_2).$$

- *if* $a_1 \in r$ *is not a leaf and* $l_1 = l(a_1) \in U \times edom(p)$*, then*

$$\forall l_2 : l_1 \to l_2 \implies \exists a_2 \in r : a_1 \to a_2 \wedge l(a_2) = l_2.$$

- *the leaves of* r *have labels in* $F \times edom(p)$*.*

We write $\mathcal{L}(A)$ *for the set of pictures* p *over* Σ *for which there exists an accepting run of* A*.*

The restriction $|I| = 1$ is important for UFA, since otherwise I provides the automaton with existential quantification. For DFA, NFA and UFA, we may leave E and U out of the definition, and for DFA, we take δ to have the type $\delta : Q \times \Sigma \to Q \times \{(1,0),(-1,0),(0,1),(0,-1)\}$, with the obvious meaning.

Note that an accepting run for an NFA is just a sequence of ID's. For UFA, an accepting run can look complicated, but we may – in an obvious way – define non-accepting runs for them, that is, possibly infinite sequences of ID's that prove no accepting run exists. This is possible since *all* choices of transitions must lead to acceptance or a picture is rejected by a UFA. Of course, a non-accepting run exists if and only if the picture is not accepted.

Note that our automata cannot sense if the border of the picture is next to them, but must step on it and read the special symbol # in order to obtain this information. We will also need 1-dimensional automata. These are always two-way automata, that is, they move both left and right. The left and right ends of the input word are again observed by reading a bordering #.

For each automata class XFA, the picture class corresponding to XFA will also be called XFA. That is, asking whether there exists an XFA for a picture

language L is equivalent to asking whether $L \in$ XFA. This should not cause confusion.

We use the naming scheme of [4] and [5] for the XFA classes. DFA and NFA were defined already in [1] in 1967 and were simply called 'automata', while the most used names for DFA and NFA seem to have been '2-DA' and '2-NA', respectively. UFA and AFA were discussed at least in [6], as a subset of Turing machines, and were given the names '2-UFA' and '2-AFA' in [7]. Slightly confusingly, we will later write 2XFA to refer to *two-way* automata instead of two-dimensional automata. Two- and three-way restrictions of automata have often been denoted by having 'TR' and 'TW' somewhere in the name of the class, respectively. We do not consider probabilistic automata in this article: see [8] for a incomparability result between probabilistic automata and AFA.

The square pictures and the unary pictures are natural subclasses of the set of all pictures (also called the general pictures). Some negative results (results of the type $L \notin$ CLS for some picture class CLS) are easier to prove in the square world (content results) and some are easier to prove in the unary world (shape results).

Definition 3. *For each automata class XFA, we write XFA_s for the languages of square pictures accepted by some automaton in XFA. We write XFA_u for the unary languages of the XFA automata. Complements of languages in XFA_s and XFA_u are usually taken with respect to the set of all square pictures and all unary pictures, respectively.*

The theory of picture languages has not been primarily concerned with these kinds of automata, but a different model of computation known as recognizability. We will not discuss the recognizable picture languages (REC) in length, but we do show some connections between the two worlds, and obtain some results for picture-walking automata from non-closure properties of REC. For more information on REC, see [3] or [5].

Definition 4. *A Wang tile is an element of C^4 for some set of colors C containing a special element #. A set T of Wang tiles with the same C defines a picture language over T by taking the pictures where neighboring colors match, and #'s occur (only) facing the border of the picture. Such a language is called a* local picture language. *A* recognizable picture language *is the image of a local picture language through a symbol-to-symbol projection. We denote the class of recognizable picture languages by REC.*

We may also think of an REC grammar (tileset and projection) as 'accepting' languages instead of generating them, by taking a picture p, and assigning 'states' (elements of T) on top of the cells of p, which agree in their colors with the neighboring states. When picture-walking automata are involved, we will, however, avoid the term 'state' in this context, and simply say tiles are assigned on top of the cells.

3 Basic results on 4-way automata

As is often the case in mathematics, the main results in the theory of picture-walking automata tend to be of one of two types: 'positive' or 'negative'. A positive result says a class is a subset of another, or that a language belongs to a class, while a negative result tells us a class is a *proper* subset of another, or that a language *cannot* be accepted by some type of machine.

Many negative results are known for both XFA_u and XFA_s, and taken together, one could say the natural questions for XFA have mostly been answered. However, the intersection of these classes is not understood at all. We repeat the following two conjectures, which were made in [9] in 2004, and are still unanswered.

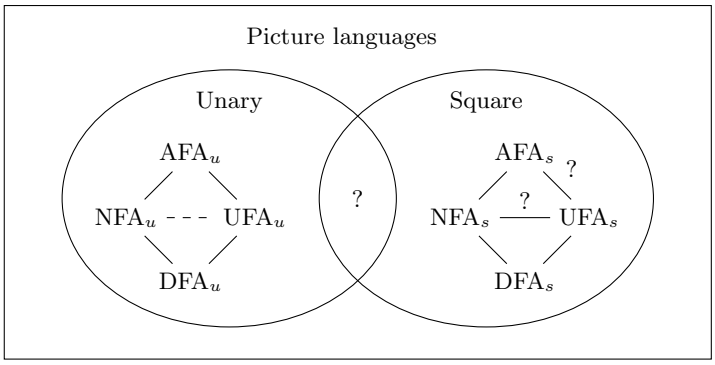
Conjecture 1. The language of unary squares with prime side length are not in DFA.

Conjecture 2. DFA is a proper subset of NFA when restricted to unary squares.

As we mentioned, by varying either shape *or* content, we can build the beginning of a (proper) polynomial hierarchy and prove basic non-closure properties for all the automata classes involved. We first show positive results for the class of all picture languages and then devote a subsection for negative results in both the case of varying shape with unary alphabet and varying content with square shape. In the unary case, a proper diamond of inclusions is shown to exist between the four classes. As for the square case, proper inclusions are known to hold between DFA_s, NFA_s and AFA_s, but little is known about UFA_s. In each case a slightly different set of Boolean closure properties is obtained.

Figure 1 summarizes the known results. All inclusions drawn are proper, and dashed lines signify incomparable classes. On the right side, UFA_s is known to be between DFA_s and AFA_s, but not between DFA_s and NFA_s.

Fig. 1. A diagram of inclusions of the automata classes in the unary and square case.



In the following sections, we will see that DFA is closed under complementation, while NFA, UFA and AFA are not. We prove non-closure under complement separately for the unary and square cases for NFA and UFA. Non-closure in the case of all pictures can easily be inferred from either proof. The case of complementing AFA_u is unknown, but we prove non-closure for AFA_s, from which the general case of AFA easily follows. Figure 2 summarizes the closure properties known for the different types of automata, where we write XFA_x for XFA, XFA_u and XFA_s.

Fig. 2. Refined table of Boolean closure properties of the XFA classes.

	DFA_x	NFA	NFA_u	NFA_s	UFA_x	AFA	AFA_u	AFA_s
$\neg$	Yes	No	No	?	No	No	?	No
$\cap$	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
$\cup$	Yes	Yes	Yes	Yes	?	Yes	Yes	Yes

Note that for general pictures, all Boolean closure properties except for the closure of UFA under union are known. In [5], a negative answer is conjectured.

Conjecture 3. UFA is not closed under union.

3.1 Positive results

Obviously, DFA $\subset$ NFA $\subset$ AFA and DFA $\subset$ UFA $\subset$ AFA with not necessarily proper inclusions. It is also clear that all the classes are closed under rotation around the center of the picture, vertical and horizontal flips and matrix transpose. We can also prove natural Boolean closure properties for these classes with simple constructions outlined below.

The following observation has been used in numerous articles, including [7, 4, 5]. The proof is a direct modification of [10].

Theorem 1. *For every DFA A, there is a DFA A' with $\mathcal{L}(A') = \mathcal{L}(A)$ such that A' halts on every input.* □

Corollary 1. *DFA is closed under complementation, that is, DFA = co-DFA.* □

Theorem 2. *All the XFA classes are closed under intersection and DFA, NFA and AFA are closed under union.*

Proof. Since an accepting computation is necessarily halting in all branches of the accepting run, intersection can be implemented for any machine by simply testing inclusion in the languages in question one by one. As for union, NFA and AFA can use an existential state, and for a halting DFA (guaranteed by Theorem 1), union can be implemented like intersection. □

3.2 Negative results for varying shape and unary alphabet

The shape-based approach is more recent than the content-based one, and gives better results for connections between classes. It is based on 1-dimensional automata. The tools we need are a reduction from 2D to 1D (Lemma 1) and a lemma connecting the number of states in a 1-dimensional automata, and the period and threshold of its language (Lemma 2). After this, a single concrete language is enough to separate all the four automata classes. We also obtain non-closure under complementation for NFA_u and UFA_u.

Let A be an XFA running on unary input, with language $L = \mathcal{L}(A)$. For each height h, we may, in a natural way, associate a 1-dimensional XFA A_h accepting the unary language of the corresponding widths. Furthermore, A_h has $O(h)$ states where the invisible constant only depends on A.

Lemma 1. *[4] Let A be an AFA with k_u universal states and k_e existential states recognizing the unary picture language $L \subset \{1\}^*_*$. Then, for each h, the language $L_h = \{1^{|p|} \mid p \in L, \underline{p} = h\}$ is recognized by a one-dimensional two-way AFA A_h with $k_u(h+2)$ universal states and $k_e(h+2)$ existential states.*

Proof. Let $A = (Q, \{1\}, E, U, I, F, \delta)$ and $X = [0, h+1]$. Then

$$A_h = (Q \times X, \{1\}, E \times X, U \times X, I \times \{1\}, F \times X, \delta_h)$$

where δ_h simulates the local rule of A, interpreting the location of A_h as the horizontal location of A, and the X component of the state of A_h as the vertical location of A. It is clear that A_h correctly recognizes L_h. □

Of course, each A_h accepts a unary regular language, and thus an eventually periodic language. Since the A_h have linearly many states with respect to h, it will be useful to prove a lemma which tells us something about the connection between the number of states in a (one-dimensional) automaton, and the period and threshold of the unary language it accepts.

Lemma 2. *[4, 5] Let A be a 1D (two-way) AFA with k states over a unary alphabet with language $L \subset 1^*$. Let $n > k$. Then*

- *if A is an NFA, $1^n \in L \implies 1^{n+k!} \in L$.*
- *if A is a UFA, $1^n \in L \impliedby 1^{n+k!} \in L$.*

In particular, if A is a DFA, an equivalence holds, since if the transition relation is a function, we may change between universal and existential states without changing the language.

Proof. First, assume all states are existential, and let $1^n \in L$. Consider a subsequence s of an accepting run r for 1^n, which visits border symbols # at s_1 and $s_{|s|}$ or possibly ends with the automaton halting in the middle of the word (we think of the automaton as being on the left border just before starting its run). We will translate such 'partial runs' to corresponding partial runs for the longer input $1^{n+k!}$, which we then glue together to obtain $1^{n+k!} \in L$.

If a partial run moves from the left border back to the left border, the same partial run can be used for the longer input, and similarly for the right border. If a partial run ends in the middle of the word, it can also directly be translated for the longer word.

If s moves from the left border to the right border, we note that there must be a repetition of states such that the automaton moved some number of steps $0 < l \leq k$ to the right in between. But l divides $k!$, so we may repeat this partial run an additional $\frac{k!}{l}$ times to obtain a corresponding partial run for the longer input. A similar claim holds for partial runs from right to left.

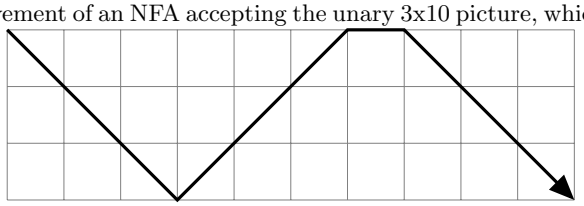
The claim for UFA follows by considering non-accepting runs instead. The fact that these runs can be infinite does not lead to complications. □

We will now separate NFA and UFA by using the duality between existential and universal quantification found in Lemma 2. For this, we will use the following 'billiard ball language' from [4].

Definition 5. *The billiard ball language $L_{billiard}$ is defined as*

$$L_{billiard} = \{p \in \{1\}_*^* \mid |p| \in \langle \underline{p}, \overline{p} + 1 \rangle\} = \{m\overline{p} + n(\underline{p} + 1) \mid m, n \in (\mathbb{N} \cup 0)\}$$

Fig. 3. The movement of an NFA accepting the unary 3x10 picture, which is in $L_{billiard}$.



That is, for each height h, the widths must be some linear combination of h and $h + 1$. It is easy to make an NFA accepting this language by having it move to the right diagonally, bouncing off the walls as in Figure 3. Just as easily, we can make a UFA accepting its (unary) complement. Using the lemmas we proved, we can also prove the converse claims: a UFA cannot accept $L_{billiard}$ and an NFA cannot accept its complement:

Lemma 3. *$L_{billiard} \notin UFA_u$, $L^c_{billiard} \notin NFA_u$*

Proof. We will only prove $L_{billiard} \notin \text{UFA}_u$, the proof for NFA_u is symmetric. So assume on the contrary that A is a UFA with k states accepting $L_{billiard}$, and for each h let A_h with $\mathcal{L}(A_h) = L_h$ be the 1-dimensional UFA given by Lemma 1.

By looking at the definition of $L_{billiard}$, it is not hard to see that for each h, $h^2 - h - 1 = h(h-2) + (h-1)$ is not in L_h, but $h^2 - h$ is. Let h be large enough that $h^2 - h - 1 > k(h+2)$. Here, $k(h+2) = k_h$ is the number of states of A_h, and $1^{h^2-h-1} \notin L_h$, so by Lemma 2, also $1^{h^2-h-1+mk_h!} \notin L_h$ for all m. But obviously L_h contains every word longer than some threshold t, a contradiction. □

Corollary 2. *$NFA_u \bowtie UFA_u$.* □

Corollary 3.

$$DFA_u \subsetneq NFA_u \subsetneq AFA_u$$

and

$$DFA_u \subsetneq UFA_u \subsetneq AFA_u$$

□

Thus, we have built the diamond on the left in Figure 1. As a side product, we obtained that NFA and UFA are not closed under complement, completing the unary part of Figure 2.

Corollary 4. *NFA_u and UFA_u are not closed under complement.* □

3.3 Negative results for varying content and square shape

The separation of DFA and NFA on binary squares was done already in [1], and it is based on a pigeonhole argument.

Definition 6. *The language L_{center} consists of square pictures p of odd side length over $\{0,1\}$ containing a 1 in the middle cell.*

Theorem 3. *L_{center} is in $NFA_s - DFA_s$.*

Proof. It is easy to see that L_{center} is in NFA_s: An NFA for it follows the main diagonal southeast, and can turn northeast at any 1 it sees. It accepts if it reaches the northeast corner. (Checking that the picture has square shape is trivial.)

Now, consider a hypothetical DFA $A = (Q, \Sigma, I, F, \delta)$ for it. We may assume A always leaves the domain of the picture before accepting it, and that it always halts by Theorem 1. To each binary $n \times n$ picture p we can then associate a function f_p characterizing the behavior of A inside the picture p. The function f_p has the type

$$f_p : Q \times E(p) \to Q \times (\text{edom}(p) - \text{dom}(p)),$$

where $E(p)$ is the set of cells of p with a neighbor outside $\text{dom}(p)$, and $f_p((q,x)) = (q',x')$ if from the ID (q,x), A eventually leaves the domain of p in ID (q',x') (this is well-defined by the assumptions we made).

For some constant C depending only on A, we have that for any n there are less than Cn elements in $Q \times E(p)$, and there are less than Cn elements in $Q \times (\text{edom}(p) - \text{dom}(p))$. Therefore, there are less than Cn^{Cn} different functions f_p for a fixed side length n. Of course, there are 2^{n^2} pictures p with side length n.

By taking n large enough that $Cn^{Cn} < 2^{n^2}$, we thus obtain two pictures of side length n with $p[x] \neq q[x]$ for some x, but $f_p = f_q$. Now we can construct two larger pictures p' and q' that agree everywhere except at a subpicture where p' has a translated copy of p and q' has q such that the position x is moved to the middle of the larger pictures. One of these pictures is in L_{center} and the other is not, but A will obviously accept p' if and only if it accepts q', a contradiction. □

By Theorem 1, we then have the following.

Corollary 5. *$DFA_s \subsetneq NFA_s$.* □

The proofs involving universal states use non-closure properties of REC. Of course, we will first need to prove some connections between the automata classes and REC. Theorem 5 and its natural corollaries first appeared in [7], and independently in [4] where also Theorem 4 and its natural corollaries were proven. We will only scetch these proofs, more details can be found in [7] and [5]. The proofs are for general pictures.

Theorem 4. *$NFA \subset REC$.*

Proof. For an NFA, we construct a set of Wang tiles whose valid tilings draw accepting runs of the NFA on top of the picture. This is done by carrying the set of states the NFA can reach at each tile. The top left corner must contain an initial state, and some tile must contain a final state. The problem is we need to make sure a final state can only occur if there is a path to it from the top left corner. To achieve this, every state contained in a tile (except the initial state at the top left corner) will have a single predecessor, and a final state is the predecessor of no state. Then no loops containing a final state can occur, so a chain of successor links ending in a final state must have started at the initial state at the top left cell – and thus represents a valid loopless computation of the NFA. □

Theorem 5. *co-$AFA \subset REC$.*

Proof. For this, we construct a set of Wang tiles whose valid tilings depict failed computations. Again, tiles contain a set of states, and a tile contains a state if there is no accepting computation from that state. The top left tile must contain all the initial states, a universal state must have at least one successor in the neighboring tiles (at least one choice of action leads to a failing computation), and an existential state must have all its successors in neighboring tiles (every possible choice of action leads to a failing computation). No final states may occur. It is then easy to believe that a tiling exists if and only if the automaton doesn't accept the input picture. □

Next, we will need a concrete language. Let $L_{acyclic}$ be the language of (not necessarily planar) acyclic graphs. Any representation of graphs where the graphs are somehow 'drawn' on the (square) picture will do. We will assume a node can be contained in every cell, an edge can go through multiple cells, edges can cross each other freely, and a large (but fixed) number of them can move through a single cell.

We note that $L_{acyclic}$ is in UFA_s, since an automaton can use universal states to check that every possible run along the forward edges of the graph leads to a dead end. However, a standard pigeonhole argument shows it is not in REC [4]: Split pictures of side length n in two, with a vertical border in the middle, containing n nodes. On both sides, implement the same total order, connecting

each node to its immediate successor. For large enough n, there are more partial orders than there are possible tilings of the middle column in tilings accepting the pictures representing the partial orders. Therefore, some two pictures can swap their right sides, necessarily resulting in a picture with a valid tiling, but which contains a cycle.

By the previous paragraph, UFA_s cannot be a subset of NFA_s, or it would also be a subset of REC, which is contradicted by $L_{acyclic} \in \text{UFA}_s - \text{REC}$. We thus obtain the partial diamond on the right side in Figure 1.

Theorem 6.

$$DFA_s \subsetneq NFA_s \subsetneq AFA_s,$$

$$DFA_s \subsetneq UFA_s$$

and

$$UFA_s \not\subset NFA_s$$

Theorem 7. *UFA_s and AFA_s are not closed under complement.*

Proof. Otherwise,

$$\text{UFA}_s = \text{co-UFA}_s \subset \text{REC},$$

which is a contradiction. The case of AFA_s is proved similarly. □

Using the connection with REC, it is also easy to show certain non-closure properties for all of the classes XFA simultaneously. We outline the idea for an operation that inherently works on general pictures instead of squares: concatenation. Consider the following lemma, which directly follows from Theorem 5.

Lemma 4. *If f is an operation on languages, $L_1, \ldots, L_k \in DFA$ but $f(L_1, \ldots, L_k)^c \notin REC$, then none of the classes XFA are closed under the operation f.* □

This implies that we can prove certain non-closure properties working completely within REC – and REC is very easy to work with!

Theorem 8. *None of the classes XFA are closed under concatenation.*

Proof. We prove the claim for horizontal concatenation. Consider the language $L_{l=r} = \{p \in \Sigma_*^* \mid p[*,1] = p[*,|p|]\}$, where $\Sigma = \{a, b, c\}$. Clearly, $L_{l=r}$ is in DFA, and therefore in all of the classes. Consider the language $L = L_{l=r}L_{l=r}L_{l=r}$. If one of the classes XFA were closed under concatenation, then L would be in this class. We show that L is not even in the class AFA. If it were, then L^c would be in co-AFA, and therefore also in REC by Theorem 5. We will show, however, that L^c is outside REC, proving the claim. Suppose, on the contrary, that L^c were in REC.

The language L^c is the the language of pictures p such that

$$\forall i, j \text{ such that } 3 \leq i < j \leq |p| - 2 :$$
$$p[*,i] = p[*,j] \implies p[*,1] \neq p[*,i-1] \vee p[*,j+1] \neq p[*,|p|]\,'$$

where $p[*,k]$ means the kth column of p. It looks kind of hard to work with, so we use the many well-known closure properties of REC to simplify it: First we restrict to the subset of pictures with columns alternatingly over $\{c\}$ and $\{a,b\}$ and with additional borders over $\{c\}$. In other words, the rows will be in the regular language given by the expression $c(c(a+b))^*cc$. This restriction is obtained by intersecting with a recognizable language. On this subset of L^c, the constraint simply forbids equal columns over $\{a,b\}$. We then erase the columns over $\{c\}$ using further closure properties to obtain that the simpler language

$$L_{c\neq c} = \{p \mid \nexists i \neq j : p[*,i] = p[*,j]\}$$

is in REC. But it is proven in [5] that $L_{c\neq c}$ is in fact not in REC, a contradiction. See [5] for the details. □

4 Moving outside the picture

In this section, we think of pictures as being embedded on the plane $\mathbb{Z}^2$ in the position indicated by the indexes. Thus, extending the indexing of matrices, the plane will be drawn with the first axis being the vertical one, and the second the horizontal one, with the first coordinate increasing as we move down, and the second as we move to the right.

So far, we have considered automata that are not allowed to exit the picture they are accepting. It is easy to see that a DFA does not gain any extra strength if it is allowed to exit a picture [11]. The corresponding question of whether NFA are strengthened if they are allowed to exit the picture was solved for pictures of height 1 in [11] in the negative. The question remained open for arbitrary shapes [11, 12, 4], until a negative answer was given recently in [5]. The solution is based on a theorem from [9] characterizing the languages of non-deterministic automata that are not allowed to enter the picture they are accepting, and we will outline the proof in Section 4.2. Also the case of AFA was solved in [5]. In this case, a simple example shows automata are in fact strengthened if they can exit the picture.

Definition 7. *FNFA is the class of picture languages accepted by NFA that are allowed to exit the pictures they are accepting. FAFA is the corresponding picture class for AFA.*

Formally, this just means redefining the set of ID's as $Q \times \mathbb{Z}^2$ instead of $Q \times \text{edom}(p)$. As already mentioned, we will also need the following 'dual' automata.

Definition 8. *ONFA is the class of unary picture languages accepted by NFA that cannot* enter *the picture they are accepting. ONFA can sense the border of the picture next to them, and they are started at* $(0,1)$*, that is, just above the top left cell.*

4.1 AFA is not FAFA

In the one-dimensional case, it is well-known that two-way alternating finite state automata accept exactly the regular languages [13]. On the other hand, even two-headed one-way deterministic automata clearly accept non-regular languages. We use the natural embedding of words into pictures by considering the sublanguages of $L_{words} = \{p \in \Sigma_*^* : \underline{\overline{p}} = 1\}$, and show that even restricted to such languages, alternating finite state automata become stronger if allowed to exit the domain of the picture. We denote by 2HAFA the class of picture languages accepted by 2-headed AFA that are not allowed to exit the picture.

Theorem 9. *AFA $\subsetneq$ FAFA. More precisely,*

$$\emptyset \neq (2HAFA - AFA) \cap L_{words} \subseteq FAFA - AFA.$$

Proof. Clearly AFA $\subseteq$ FAFA. To prove proper inclusion, we will simulate an arbitrary 2-headed AFA on strings, which are represented as $1 \times n$ pictures, using the space above the string to remember the distance between the two heads. This proves the claim, because a two-headed one-dimensional AFA can recognize, for instance, the non-regular language $\{a^n b^n \mid n \in \mathbb{N}\}$, while a one-headed AFA restricted to a string will only recognize (embeddings of) regular languages.

Given a one-dimensional language $L \subseteq \Sigma^*$ recognized by a 2-headed AFA A, we construct a 1-headed FAFA A' recognizing the picture language $L' = \{p \in \Sigma_*^* \mid \underline{\overline{p}} = 1, p[1,*] \in L\}$.

We may assume A's first head is always to the left of its second head. When the first head of the one-dimensional 2HAFA is at p_1 and its second head is at p_2, the head of A' hovers over the input string at $(1 + p_1 - p_2, p_1)$. The moves of A are directly translatable to moves of A', so the only problem is to read the contents of the cells under the two heads. But an AFA can do this by guessing the contents, and using a universal state to branch two heads that check that the guess was correct. A third head then continues the simulation. □

4.2 NFA is FNFA

We give a proof of this result based on 'landing sequences' of an FNFA, the information of which states and cells it can reach when it returns to the domain of the picture after exiting it. That is, we simulate the run of an FNFA by an NFA while it stays inside the picture, and when it leaves, we predict its landing without leaving the picture. When considering the behavior of an FNFA outside the domain of the picture, we only need to consider transition functions $\delta : Q \to 2^{Q \times \mathbb{Z}^2}$ where the $\mathbb{Z}^2$ part is the move of the automaton in the transition, which we can assume to be in $\{(0,1),(0,-1),(1,0),(-1,0)\}$. This is because an FNFA reads only unary input outside the picture. We give some further definitions to simplify the following discussion.

Definition 9. *A* run *of an FNFA A on unary input is a sequence $((q_i, x_i))_i$ of pairs (q, x) where q is a state of A and $x \in \mathbb{Z}^2$, such that $\forall i : \delta(q_i) \ni (q_{i+1}, x_{i+1} - x_i)$ where δ is the transition rule of A.*

We assume an FNFA never accepts outside the picture, since it can always navigate back to the picture in order to accept.

Definition 10. *A* state configuration *is a subset of $Q \times \mathbb{Z}^2$ (that is, a set of ID's).*

Definition 11. *Let A be an FNFA with states Q. Then, if X, Y, Z are state configurations, we define*

$$A(X, Y, Z) = \{z \in Z \mid \exists \text{ run } r \text{ of } A : r \in XY^*z\}$$

referred to as a set of landings *(which is also a state configuration). We also use the syntactic conventions that if a tuple is used in place of one of X, Y, Z, the tuple is enclosed in a singleton set, which is then used as the argument. If one of X, Y, Z is not a subset of $Q \times \mathbb{Z}^2$, but a subset of $\mathbb{Z}^2$, then its cartesian product with Q is used instead.*

The proof relies crucially on the following Landing Lemma, from [9], which is interesting in its own right. It says that, while moving one step downward, the possible moves to the left and to the right form an eventually periodic set. We call a run that ends right after moving one step downward from the initial position a *south landing run*.

In the proof, we use the following basic definitions and results from automata theory: A *semilinear subset* of a commutative monoid M is a finite union of *linear sets*, which are sets of the form $x + \mathbb{N}_0 x_1 + \cdots + \mathbb{N}_0 x_k$ for $x, x_i \in M$. The Parikh set corresponding to $L \subset \Sigma^*$ is

$$\{x \in \mathbb{N}_0^\Sigma \mid \exists w \in L : \forall a \in \Sigma : |w|_a = x_a\}.$$

Lemma 5 (Landing Lemma). *For all $s, f \in Q$, the set*

$$A((s, (-1, 0)), \{(y, x) \in \mathbb{Z}^2 \mid y < 0\}, \{f\} \times (\{0\} \times \mathbb{Z})),$$

considered as a $\mathbb{Z}$-indexed sequence, is eventually periodic in both directions.

Proof. Let $s \in \{0, 1\}^\mathbb{Z}$ be the corresponding binary sequence containing a 1 in the positions A can reach in state f. To show s_i is eventually periodic in both directions, we construct a PDA accepting the language L_{moves} of words

$$w \in \{(1, 0), (-1, 0), (0, 1), (0, -1)\}^*$$

such that some south landing run r of A has exactly this sequence as its sequence of moves.

The PDA accepts when the stack becomes empty. It originally has one symbol Z on the stack, representing the fact that the NFA starts at height 1. The finite

control makes state transitions just as the NFA would, always reading the current move from the input word, rejecting the word if a different move is read. A Z' is pushed on top of the stack whenever the NFA moves up, and a Z' or Z is popped whenever it moves down. However, the automaton may only pop a Z if it is about to enter state f. The stack is not changed when the NFA moves horizontally. It is clear that such a PDA accepts exactly L_{moves}, since when the stack become empty, the NFA being simulated would have entered state f, and would have, altogether, moved one step down from its initial position.

Because this language is context-free, its Parikh set L_p is a semilinear subset of $\mathbb{N}_0^4$ [14] and thus also a semilinear subset of $\mathbb{Z}^4$. But then, by well-known closure properties of semilinear sets,

$$\{r - l \mid (d, u, r, l) \in L_p\}$$

is semilinear in $\mathbb{Z}$, where r and l are the amounts of right and left moves, respectively. But this is exactly the set $\{i \mid s_i = 1\}$, and a semilinear subset of $\mathbb{Z}$ must be eventually periodic in both directions. □

Of course, symmetric claims hold for landings in other directions than down. The problem of whether NFA = FNFA essentially boils down to proving such a lemma for landings on the border of a rectangle instead of a straight line.

Definition 12. *The* fundamental threshold *and* fundamental period *of an ONFA A are the smallest possible t and q such that its landing sequences in all directions from any state s to any state f all have t as a threshold and q as a period in both directions. More precisely, we let q be the smallest period, and t is then chosen to be the smallest threshold for this particular q.*

The proof we present is based on the following characterization of the class ONFA. This result is also interesting in its own right, and was also proven in [9].

Definition 13. *An eventually periodic subset of $\mathbb{N}^2$ is a set X such that there exist t, q such that*

$$\forall w > t : (h, w) \in X \iff (h, w + q) \in X$$

and

$$\forall h > t : (h, w) \in X \iff (h + q, w) \in X.$$

We state two characterizations of the eventually periodic sets without proof, see [5] or [9].

Lemma 6. *A subset X of $\mathbb{N}^2$ is eventually periodic if and only if there exist t, q such that*

$$\forall w > t : (h, w) \in X \implies (h, w + q) \in X$$

and

$$\forall h > t : (h, w) \in X \implies (h + q, w) \in X.$$

□

Lemma 7. *A subset X of $\mathbb{N}^2$ is eventually periodic if and only if the language $\{0^m 1^n \mid (m,n) \in X\}$ is a regular language.* □

Theorem 10. *The shapes of languages of ONFA are exactly the eventually periodic sets.*

Proof. It is not hard to see that ONFA can accept all the eventually periodic sets using Lemma 7. For the other direction, we will use Lemma 6. We consider a language L' in ONFA accepted by some automaton A, and let $L = \{(\underline{p}, |p|) \mid p \in L'\}$, which uniquely determines the language. We prove both the widths and heights of pictures in the language can be pumped, that is, there is a threshold t and a period q such that

$$\forall (h,w) \in L : h > t \implies (h+q, w) \in L$$

$$\forall (h,w) \in L : w > t \implies (h, w+q) \in L$$

It is enough to find such t and q that work for widths, since a symmetric argument will work for heights, and we can use the larger of the thresholds and a least common period of the periods to find the threshold and the period of the whole language.

Let q' and t' be the fundamental period and fundamental threshold of A. We will prove that $q = (|Q|t')!q'$ and $t = (|Q|+1)t'$, work as pumping constants for L. We start by naming some areas of the plane. We define the top left ray, the top right ray, the bottom left ray and the bottom right ray as the sets of cells

$$\{(0, -n) \mid n \in \mathbb{N}_0\},$$

$$\{(0, |p| + n + 1) \mid n \in \mathbb{N}_0\},$$

$$\{(\underline{p} + 1, -n) \mid n \in \mathbb{N}_0\},$$

and

$$\{(\underline{p} + 1, |p| + n + 1) \mid n \in \mathbb{N}_0\},$$

respectively. Note that the top and bottom rays do not coincide even if the picture has height 1.

Now consider a run r accepting the picture of shape (h,w), where $w > t$. We split the run into partial runs $r^1, \ldots, r^k$ between the points of intersection with one of the four rays, that is, we partition the run into semiopen intervals by cutting the run at the points where a ray is crossed (we may assume A accepts on a ray).

These partial runs are transformed into a run of the automaton on the larger picture of shape $(h, w+q)$. There is an obvious way to identify the rays of the small picture with the rays of the large picture, and each partial run is transformed into a run that starts and ends at the corresponding cell in the corresponding ray. It is then clear that by gluing together the partial runs obtained for the larger picture, a valid accepting run for the larger picture is obtained, proving $(h, w+q) \in L$.

For partial runs between two cells both in left rays, or two cells both in right rays, it is obvious how to find a corresponding partial run for the larger picture. Now, all we need to do is apply the Landing Lemma 5 to transform the partial runs that move between the two sides of the small picture to such runs on the large picture. For this, consider such a partial run $r^j = u$ on the small picture. By symmetry, we may assume u starts just above a cell of the top left ray, and ends on a cell of the top right ray. We split u further into subpartial runs u^j exactly as we did for r, but this time at the positions where the automaton senses that it is next to the picture. Let J the sequence of indices of u where the automaton senses p.

We now have two cases to consider:

- One of the subpartial runs u^j moves the automaton to the right more than t' cells.
- All subpartial runs u^j move the automaton at most t' to the right.

In the first case, we are done, since the automaton's fundamental threshold has been exceeded, so the subpartial run u^j can be made mq' cells longer to the right, for any $m \in \mathbb{N}$. In particular, it can be made $q = (|Q|t')!q'$ cells longer, proving the claim in this case.

In the latter case, we note that there must be at least than w/t' of these subpartial runs, where we chose $w > t = (|Q| + 1)t'$, and thus $|J| > |Q|$. Then, we may take an ascending subsequence K of J such that

- the sequence of positions of $(u_i)_{i \in K}$ is ascending to the right.
- between each two indices of K the automaton has moved at most t' cells to the right.
- $|K| > |Q|$.

Now note that there must be a repetition of states in $(u_i)_{i \in K}$, say at $a, b \in K, a < b$, and the distance d between u_a and u_b satisfies $1 \leq d \leq |Q|t'$. Then the partial run of length d between u_a and u_b can be repeated $\frac{q}{d} = \frac{(|Q|t')!q'}{d}$ times to obtain a partial run for the picture $(h, w + q)$.

This means that in both cases, we were able to make the partial run work for a picture q wider, as long as the picture had width at least t, and thus t and q as we chose them are a threshold and period for pumping the widths of pictures in L. □

Let us now give more notation for parts of a picture, and the space around it. Given a picture p, we call the cells (i, j) such that $j < 1$ the west half-plane $H_w(p)$, and similarly we get the east, north and south half-planes $H_e(p)$, $H_n(p)$ and $H_s(p)$. These cover all the positions outside the picture. In what follows, we use a slightly different definition of rays: the top left west ray $R_{tlw}(p)$ is the set $\{(1, x) : x < 1\}$ and similarly we define $R_{tln}(p)$, $R_{trn}(p)$, $R_{tre}(p)$, $R_{bre}(p)$, $R_{brs}(p)$, $R_{bls}(p)$ and $R_{blw}(p)$. The west edge of p is the set $E_w(p)$ of cells on its domain with a left neighbor outside its domain. Similarly we obtain $E_n(p)$, $E_e(p)$ and $E_s(p)$. We define the edge of p as $E(p) = E_w(p) \cup E_n(p) \cup E_e(p) \cup E_s(p)$.

We define the west line of p as $L_w(p) = R_{tln}(p) \cup E_w(p) \cup R_{bls}(p)$, and similarly we get $L_n(p)$, $L_e(p)$ and $L_s(p)$. We define the outside of p as $O(p) = H_w(p) \cup H_e(p) \cup H_n(p) \cup H_s(p)$. Finally, we define $C_m(p)$ as the subset of $E(p)$ of cells at most distance m away from some corner of p.

Given a run of an FNFA starting from just outside the edge of a picture and ending at an edge, staying outside the picture during the run, we call the unique half-plane on which it starts the *initial half-plane*. The part of the run before exiting the initial half-plane for the first time is called the *initial segment* of the run. Similarly, we define the *final half-plane* and the *final segment* of the run. Note that the initial and final segments can partially overlap, or be equivalent if the initial half-plane is never exited.

In what follows, we will use the terms 'landing' and 'computing a state configuration' somewhat informally. Implicitly, one of the following two definitions will then apply.

Definition 14. *Let Q be a finite set of states. For any $Z \subset \mathbb{Z}^2$ and FNFA A with unique initial state s, the* set of landings of A from x onto Z *is defined as the state configuration c on Z given by*

$$A((s, x), Q \times O(Z), Q \times Z).$$

Let A be an FNFA with state set $Q' \supset Q$ and with a unique initial state $s \in Q' - Q$. We say A computes the state configuration c over Q from x within Z *if c is the state configuration*

$$A((s, x), (Q' - Q) \times Z, Q \times Z).$$

Now, we can prove the main theorem of this section: NFA = FNFA. First, let us compute the sub-landing sequences 'near the corners' of pictures, assuming the FNFA leaves the picture at one of its corners.

Lemma 8. *Let A be an FNFA with states Q. Then, for every $i \in Q$ and $m > 0$ there exists an NFA A' with states $Q' \supset Q$ such that*

$$\forall p : A((i, (0, 1)), O(p), C(p, m)) = A'((s', (1, 1)), dom(p), Q \times C(p, m))$$

Proof. $C(p, m)$ is a finite set, so A' simply needs to guess one of the landing possibilities $(s, x) \in Q \times C(p, m)$ and check if it's a possible landing of A. If it is, A' finds x and enters s on it. But it's easy to check if (s, x) is a landing of A: There exists an ONFA B that simulates A until it tries to enter the picture, and accepts if it would've entered the correct cell x in the correct state s. The set $\{(h, w) \mid \exists p \in \mathcal{L}(L(B)) : (\overline{p}, |p|) = (h, w)\}$ is eventually periodic, so A' can easily check if p belongs to $\mathcal{L}(B)$. Since $p \in \mathcal{L}(B)$ if and only if (s, x) is a landing of A, we are done. □

Using the previous lemma, we can now characterize *all* landings of runs that start near a corner.

Lemma 9. *Let A be an FNFA with states Q. Then, for every $i \in Q$ there exists an NFA A' with states $Q' \supset Q$ such that*

$$\forall p : A((i,(0,1)), O(p), E(p)) = A'((s',(1,1)), dom(p), Q \times E(p))$$

Proof. Let t and q be the fundamental threshold and period of A, respectively, and assume $(s,x) \in Z = A((i,(0,1)), O(p), E(p))$. First assume $x \in E_w(p)$. Then

$$x \notin C(p, t+q+1) \implies (s, x+(q,0)) \in Z$$

by the Landing Lemma 5, since A's run with landing (s,x) must have entered the west half-plane for the last time at some point, and we may thus pump the final segment of the run before landing, by q steps in either direction. By repeating the argument, (s,y) is also in Z for some $y \in C(p,t+q+1) - C(p,t+1)$.

Conversely, if (s,y) is a landing of A onto $C(p,t+q+1) - C(p,t+1)$, on the west side, then also all other $(s, y+(mq,0))$ such that $y+(mq,0) \in E(p) - C(p,t+1)$ are landings of A, again by the Landing Lemma 5, in particular if y is obtained from pumping x onto $C(p,t+q+1) - C(p,t+1)$, then the landing (s,x) is found by pumping y the same distance in the other direction.

Now, for landings of A on $C(p,t+q+1)$, we apply Lemma 8. As for other landings, consider again landings on the west side of p. For all landings of A onto (s,x), where $x \in C(p,t+q+1) - C(p,t+1)$, we let A' move q steps up or down as many times as it likes before entering s, while staying inside $E(p) - C(p,t+1)$. By the argument of the first paragraph, we obtain all landings of A on E_w this way. The other edges are handled similarly. □

Finally, the result is obtained by handling runs that never exit the initial half-plane separately, and then using Lemma 9 for those that exit it.

Theorem 11. *NFA = FNFA.*

Proof. Given FNFA A, we construct an NFA A' that has states $Q \cup Q'$ where Q are the states of A and Q' are helper states used in the simulation. The NFA A' uses only states of Q when inside p, and has the same transition function as A when restricted to these states. Of course, this means A might try to exit p during its accepting run of p. When it tries to exit p, x being the first cell outside p it would've entered, it instead enters a special search state in Q' that computes the landing sequence of A' from x onto the edges of the picture inside $\mathrm{dom}(p)$. We assume A never accepts a picture p while outside $\mathrm{dom}(p)$.

If we can accomplish this, then it should be clear that the languages of A and A' are the same. We do not give a formal construction of A', but an informal algorithm for obtaining such an automaton from the behavior of A. So assume A exits p during a run, and let x be the first cell outside p it sees. We may assume x is on the west side of p, since the other cases are symmetric. First, we note that

$$\begin{gathered} A((s,x),O,E) = A((s,x),H_w,E_w) \cup \\ A(A((s,x),H_w,R_{tln}),O,E) \cup \\ A(A((s,x),H_w,R_{bls}),O,E) \end{gathered}$$

where we write X instead of $X(p)$ for brevity. We explain how A' can compute each of the three parts within $\mathrm{dom}(p)$, in which case the pointwise union is also computable. By symmetry, it's enough to handle $A((s,x),H_w,E_w)$ and $A(A((s,x),H_w,R_{tln}),O,E)$.

Case 1: Computing $A((s,x),H_w,E_w)$ inside $\mathrm{dom}(p)$.

Consider the landing sequence s^1 of A from (s,x) upwards, that is, s^1_i is the set of landing states on the cell $x+(-i,1)$. By the Landing Lemma 5, this sequence s^1 is eventually periodic. Therefore A' can compute this sequence inside E_w, since the sequence does not depend on the picture. This concludes the first case.

Case 2: Computing $A(A((s,x),H_w,R_{tln}),O,E)$ inside $\mathrm{dom}(p)$.

Let $x=(i,j)$. Then,

$$A((s,x),H_w,R_{tln}) = s^1_{[i+1,\ldots)} = s^2,$$

where s^1 is as in Case 1. Let B_{s^2} be an FNFA with initial state s'' that computes this sequence onto R_{tln}, and then simulates A. That is, B_{s^2} has states $Q \cup Q''$, with Q'' and Q disjoint, and

$$B_{s^2}((s'',(0,1)),Q''\times R_{tln},Q\times R_{tln}) = A((s,x),H_w,R_{tln})$$

for the unique initial state s'' of B_{s^2}, and B_{s^2} has the same transition function as A, when restricted to states in Q, which implies

$$B_{s^2}((s'',(0,1)),O,E) = A(A((s,x),H_w,R_{tln}),O,E).$$

By Lemma 9, there is an inner NFA with the same landings as B_{s^2}. Therefore, A' can move up the side of p, determine the sequence s^2, and depending on this sequence, compute the landing sequence of B_{s^2} onto E inside $\mathrm{dom}(p)$. This is possible, because the sequence s^2 is a final segment of the eventually periodic sequence s^1, and thus one of a finite amount of possibilities. This concludes the second case.

□

The techniques presented here do not generalize for 3- or more-dimensional pictures (with the obvious definitions). We end this section with the following conjecture from [5].

Conjecture 4. NFA = FNFA on three-dimensional pictures.

5 Restricting the directions of movement

We will now turn to automata for which some directions of movement are forbidden. We define the 2XFA as XFA that cannot use the directions up and left

(recall that our automata are started from the top left corner). The classes 3XFA are defined by forbidding only upward moves. Although results will still be used from both the unary and the square worlds, we will only give results for general pictures in this section. We will first collect results from the literature to give the Boolean closure properties for each 3XFA. We then compare the three-way classes with each other and the four-way classes and give a strong connection between 2AFA and a deterministic version of REC.

Analysis of three-way automata is somewhat simpler than that of four-way automata. Interestingly, now 3AFA can be shown to be closed under complement [7], and so can 3DFA [15], while 3NFA and 3UFA still lack this property [4] (proof below). Now our proofs for the other Boolean closure properties break down, and in fact the class 3DFA is not closed under either intersection or union, and 3NFA is not closed under intersection [16]. Also the question for 3UFA corresponding to Conjecture 3 has a positive answer – 3UFA is not closed under union [6]. Of course, 3NFA is closed under union, 3UFA is closed under intersection and 3AFA is still closed under both operations. Thus, we obtain the rather natural situation of Figure 4.

Fig. 4. Boolean closure properties of the 3XFA classes.

	3DFA	3NFA	3UFA	3AFA
$\neg$	Yes	No	No	Yes
$\cap$	No	No	Yes	Yes
$\cup$	No	Yes	No	Yes

It is clear that 2XFA $\subset$ 3XFA $\subset$ XFA for all the XFA classes. From the considerations of Section 3.2, we obtain that the second inclusion is always proper, and a stronger separation of NFA and UFA:

Theorem 12. *The sets* $3NFA - UFA$ *and* $3UFA - NFA$ *are nonempty and neither of 3NFA and 3UFA is closed under complement.*

Proof. The 90° rotation of the billiard ball language from Definition 5 is in the first set, and the complement of this language is in the other one. The non-closures under complement follow similarly. □

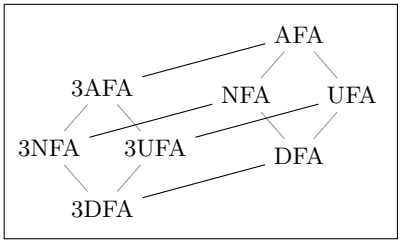
The theorem implies that a four-way automaton cannot recognize the languages of three-way automata of a 'stronger' type. In fact, no non-trivial inclusions exist between four-way and three-way automata classes, as is implied by the following result, which is proven in [17] in a stronger form.

Theorem 13. *The set* $DFA - 3AFA$ *is nonempty.* □

So by combining results of [17] with the results obtained by using the billiard ball language from [4], we have obtained a complete picture of the relations

between three- and four-way automata classes, on general pictures. We depict this in Figure 5, where all inclusions are proper, all inclusions between classes are given by diagram chasing, and all other pairs of classes are incomparable.

Fig. 5. Diagram of inclusions for three-way and four-way automata classes.



In Section 3.3, we showed some connections between REC and the XFA classes (namely Theorems 4 and 5). It seems to be unknown whether co-AFA is actually equal to REC [7], although this seems unlikely. However, for 2-way automata, we have a natural connection between the two worlds: 2AFA is equal to a deterministic version of REC [18]. However, due to bad luck, the way the classes are usually defined we have to rotate one of them by 180° to make them coincide.

Definition 15. *DREC is the subfamily of REC obtained by using north-west deterministic tiles, that is, tilesets such that no two Wang tiles in the set may share both their north and west colors.*

With the interpretation that REC accepts a language by assigning states on the cells of a picture, this means when assigning states of a DREC grammar, knowing the north and west neighbors uniquely determines the state of the current cell. We next prove the following [18]:

Theorem 14. *$2AFA^R = DREC$.*

We split the proof into two inclusions. Given a picture p and a position y, we define *top left rectangle at y* to be the rectangle between $(1,1)$ and y. Given such a rectangle, we define its north child and west child as the rectangles that start from $(1,1)$ and end at $y-(1,0)$ and $y-(0,1)$, respectively. Note that 2AFA^R is of course just the class of languages accepted by 2-way AFA that start at the bottom right corner and can only move up and to the left.

Lemma 10. *$2AFA^R \subseteq DREC$.*

Proof. We construct a DREC grammar G simulating the given 2AFA^R A. At each tile of G, the subset of states of A is held from which there is an accepting

computation of A for the top left rectangle at the position. At the top left corner, the correct subset of states of A is enforced, based on the symbol at that position. It should be obvious how to inductively find the correct state sets elsewhere, north-west deterministically, based on how the acceptance of an AFA is defined. □

Lemma 11. *$DREC \subseteq 2AFA^R$.*

Proof. Given a DREC grammar G, we find a 2AFA^R A with the same language. The automaton A has the states $T \times S$ where T is the tileset of G, and S is a small set of helper states needed in the construction (which we leave unspecified). Let us distinquish a state $s \in S$. When A is started in state (q, s) in cell x, it accepts if and only if there is a consistent assignment of DREC states onto the cells of the top left rectangle at x such that the tile q is used as the state of cell x.

The top left corner can easily be detected by a 2AFA^R. In the top left corner, A simply checks whether such a consistent assignment (of one tile) exists using a look-up table. In the general case, A guesses the north and west neighbors q_1 and q_2 of q with colors matching those of q. It then recursively checks these that the north and west children can be consistently colored in such a way that the corresponding q_i is used in the bottom right corner.

The algorithm works because in the recursion, in addition to consistent placements of states for the subpictures starting from the north and west neighbors existing, they must also have the same tiles in the overlapping zone due to the determinism of DREC.

Then, the initial states of A simply start such searches from all tiles $q \in T$. □

6 Markers

In this section, we only consider deterministic automata, and work on general pictures. However, we give our automata a finite set of markers, which the automaton can carry around, drop on the cells of the picture, and later lift up again. The main results we present are from [1], although we will prove slightly stronger claims. There are many ways to formalize this idea, but most of these can be shown equivalent [1]. We choose the definition of 'physical markers', which need to be lifted before reusing, and which can be stacked on top of each other.

Of course, once k markers are given to our automata, we will obtain a hierarchy of languages based on how many markers are needed for a DFA to accept them. This hierarchy is proven infinite in [1] with a diagonalization argument. We will investigate only the beginning of this hierarchy, using concrete and natural picture languages.

The language L_{center} shows us DFA is already strengthened by the addition of one marker, since we can use the marker to implement essentially the same algorithm we used to prove $L_{center} \in$ NFA. Denoting the class of picture languages accepted by DFA with n markers by nmDFA, we then have the following.

Theorem 15. *DFA* $\subsetneq$ 1*mDFA.* □

We will devote the main part of this section to separating DFA with two markers from those with just one, illustrating interesting programming techniques for DFA with markers, and an extended pigeonhole argument for DFA with one marker.

Definition 16. *For each* n, L_{nC} *is the language of pictures over* $\{0,1\}$ *containing exactly* n *connected components over* 1 *(connected components of the graph where the vertices are the cells containing a* 1*, and there are edges between adjacent cells). We write* $L_C = L_{1C}$*. For each* n*, we define* L_{neq} *as the subset of* L_{nC} *containing exactly* n *components which are all translations of a single component.*

In [1], in order to separate 1mDFA and 2mDFA, it is proven that there exists a language $L \in$ 2mDFA whose intersection with $L_{2\mathrm{C}}$ is exactly $L_{2\mathrm{eq}}$, but there is no such language in 1mDFA. We will instead prove the perhaps more interesting new result that $L_{2\mathrm{eq}} \in$ 2mDFA $-$ 1mDFA.

The language L_{C} has been of interest to many authors in both the case of picture-walking automata and recognizability. In [19], L_{C} was proved to be in REC, while [17] proved it to be outside 3AFA. In [1], it was shown that a DFA can accept L_{C} with one marker, and it was implicitly conjectured that this is not possible with a regular DFA.

Conjecture 5. [1] The connected patterns are not in DFA.

Note that a proof through REC will certainly not work, since it is easy to write an AFA for L_{C} [20]. Neither does a direct pigeonhole argument seem possible.

Let us explain the construction of [1] for L_{C} with a one marker DFA, since it also nicely illustrates some of the techniques we will need for showing $L_{2\mathrm{eq}} \in$ 2mDFA. First, we give some slightly informal terminology. By a local property P we mean a set of $(p, x, m_1, \ldots, m_k)$, where $x, m_i \in \mathrm{dom}(p)$, x representing the head of the automaton and m_i the positions of markers. We say P is an n-marker property, if a DFA with n unused markers started in cell x of picture p with some fixed k markers of its at $m_1, \ldots, m_k$ can check whether $(p, x, m_1, \ldots, m_k) \in P$, then returning to x carrying again the n markers.

So, let $p \in \Sigma_*^*$ be a binary picture. We say a column j of p is left separating if it contains a 1, there is a 1 in p somewhere to the left of column j, and

$$\nexists i : p[i,j] = 1 \wedge p[i,j-1] = 1.$$

That is, left separating columns contain 1's all of which are separated from all 1's to the left of them. Similarly, we define right separating columns. It is clear that the 1's of p are connected if and only if there are no left or right separating columns, and on each column, two 1's with only 0's between them are connected.

Theorem 16. $L_{1C} \in$ 1*mDFA.*

Proof. Checking that no column is left or right separating is easy. Next, the automaton reads the columns top-down, from left to right. At each 1 which is not the last in its column, we check that it is in the same component as the next 1 in the column. More precisely, we note that the local property 'The current cell x contains a 1 and the next 1 in this column (at y) is in the same component.' is a 1-marker property: the automaton drops the marker and then follows the edge of the area of 1's (using the classical labyrinth algorithm) until it either returns to x, or it finds a 1 under the marker at x with only 0's in between. Note that the latter is a 0-marker property and thus checkable during the search. In the first case, y is not in the same component, and we conclude $p \notin L_{\mathrm{C}}$. If the second case always occurs, we conclude $p \in L_{\mathrm{C}}$. □

We will first count that there are exactly two components, by extending the previous argument. This technique can be used to prove $L_{n\mathrm{C}} \in n\mathrm{mDFA}$, although, as in the case of one component, it is unknown whether this is optimal.

Theorem 17. *$L_{2C} \in 2mDFA$.*

Proof. We define the shoulder of p as the cell containing a 1 seen first during a columnwise top-down left to right search, and similarly we define the shoulder of a component of 1's in p. We define the top left component as the component of 1's that the shoulder of p belongs to. Note that being on the outer border of the top left component is a 1-marker property, since whether the current cell is the shoulder is a 0-marker property. For brevity, we call the top left component A.

We now start a columnwise top-down left to right search over the picture. At each column, we continue down without permanently dropping any markers, while staying in A. This is possible since we can check whether the first cell we see is in A (since it is necessarily on the outer border of its component), and we can check whether the component changes as we move down the column. If another component is not seen during the whole search, we conclude there is just one component, and thus $p \notin L_{2\mathrm{C}}$.

If a different component is seen at any time, we drop one of the markers. Note that it is then dropped at the outer border Y of some component B. The marker will not be moved again, so in effect, whether a cell is on the outer border of either A or B has turned into a 1-marker property. Just as importantly, if B is completely *within* A, whether a cell belongs to the edge of the corresponding hole of A is now also a 1-marker property. Let X be the set of cells on this edge, or the outer border of A if B is not within A. We obtain that belonging to X and belonging to Y are both one-marker properties.

After marking X and Y, we continue the search as previously, now essentially with a one marker automaton. When we enter the first component on a column at x, we immediately conclude $p \notin L_{2\mathrm{C}}$ if this x is not in A or B (since, again, x is at the outer border of *some* component). After either A or B has been entered, the automaton continues down the column while the component does not change. When it does change, we immediately reject if it does not go from X to Y or from Y to X. If there are exactly two components, only such transitions

happen, and if only such transitions happen, A and B are of course the only components. □

Next, let us compare these two components.

Theorem 18. *$L_{2eq} \in 2mDFA$.*

Proof. By the previous theorem, we can assume there are exactly two components. Again, let A be the top left component and B the other one. Now, B cannot be inside A or we can directly reject p. This is easy to check during the previous algorithm, and even easier to do directly. Knowing this, we first locate the shoulder of B: we do a columnwise search until either the first component seen is not A or the component changes after a cell of A. We keep the other marker – the B marker – here, and move the other marker to the shoulder of A. The two markers will stay exactly this vector v away from each other during the rest of the search.

Now, consider the components as sets of vectors and p as the set of indices where it contains a 1. We will check that $A + v \subset p$ and $B - v \subset p$, which clearly implies $A + v = B$. For the first inclusion, we do a columnwise search using both markers at once, and whenever the A-marker is in a cell of A, we check that the B-marker is on top of a 1. The second inclusion is done symmetrically.

In order to keep track of when the A-marker is on a cell of A, we start with the markers at the shoulders of the corresponding components. Both markers are moved down simultaneously while the A-marker stays within A, skipping 0's in the usual way, and we check the B-marker is on top of a 1 whenever the A-marker is. When the A-marker changes component, the head must have entered B. We then continue until the component changes again: we must have returned to A, and the search continues normally. □

Now, let us prove also the negative result – that two markers are in fact necessary for $L_{2\text{eq}}$.

Theorem 19. *$L_{2eq} \notin 1mDFA$*

Proof. Suppose A is a 1mDFA with $\mathcal{L}(A) = L_{2\text{eq}}$ and let A have k states. We assume A directly returns to the domain of the picture if it enters its border.

We will use a similar argument as the one in the proof of Theorem 3. For each picture p we define the function f_p mapping 'incoming ID's' to 'outgoing ID's', although this time we also allow 'accept', 'reject' and 'loop' in the codomain. (We could avoid this by repeating the argument of [10] for one-marker DFA.) The function f_p completely characterizes the behavior of A on p as long as the automaton enters the picture without carrying the marker. We say that two pictures p and q are A-equivalent if $f_p = f_q$.

The basic idea for applying the pigeonhole principle here is that while the marker stays on one side of the picture, the original partition into A-equivalent pictures applies to the other side, and the marker cannot be moved across the middle of the picture more than a linear amount of times without the automaton entering a cycle. Let us make this idea more precise.

Let n be fixed, and let X be the set of all pictures of size $n \times n$ containing exactly one connected component with 0's around it. To each element of X^2 we naturally associate a picture of shape $n \times 2n$ where the two elements of X are simply concatenated (note that such a picture is in L_{2C}). The left- and right-hand sides of these pictures p are called the LHS and the RHS. We will show that for large enough n, A will accept all pictures in $Y \times Y$ for some subset Y of X with more than one element, which is a contradiction.

We note that for some n-independent D, $|X| \geq 2^{Dn^2}$ for large enough n by using a suitable skeleton of 1's to allow rows on which bits can be chosen freely while keeping the pattern connected. Let P be the partition of X into A-equivalence classes, and again note that there are at most Cn^{Cn} components in P for some C depending only on A. We write X/P for a maximal subset Z of X containing only A-equivalent elements (tie-breaking in some natural way). Of course, $|Z| \geq \frac{|X|}{|P|}$.

Let us restrict our attention to $Y_0 = X/P$. Consider the pictures Y_0^2. The automaton cannot accept any of the pictures in this set without moving the marker on the right side, since the RHS are all A-equivalent. For some subset $Y_0' \subset Y_0$ of size at least $\frac{|Y_0|}{kn}$, A exits the LHS with the marker the first time the same way in all pictures of ${Y_0'}^2$. Note that the content of the RHS does not influence how the marker leaves the LHS, since all pictures in Y_0 are A-equivalent. If Y_0' has more than one element, A must eventually move the marker back to the LHS. Again, it does this the same way on some subset $Y_1 \subset Y_0'$ of size $\frac{|Y_0'|}{kn}$.

If Y_1 has more than one element, the marker eventually has to be moved on the right side, but now this can be done in only $kn - 1$ ways, since otherwise the automaton enters an infinite loop on all pictures of Y_1^2, which is impossible because $Y_1^2 \cap L_{2\text{eq}} \neq \emptyset$. We thus obtain sets Y_1' and Y_2, and similarly Y_i' and Y_{i+1} for all i until the size of one of these reaches 0 or 1. Note that $|Y_{i+1}| \geq \frac{|Y_i|}{(kn-i)^2}$. Since the automaton necessarily loops when i reaches kn, we have obtained $|X/P| \leq (kn)!^2$. But we can show this is false for large enough n, using a standard argument:

$$\frac{2^{Dn^2}}{Cn^{Cn}(kn)!^2} \leq \frac{|X/P|}{(kn)!^2} \leq 1 \iff$$

$$Dn^2 - Cn \log Cn - 2\log(kn)! \leq \log \frac{|X/P|}{(kn)!^2} \leq 0$$

which is clearly not the case, using String's approximation for $\log(kn)!$. This is a contradiction, and thus $L_{2\text{eq}} \notin$ 1mDFA. □

Corollary 6. *$1mDFA \subsetneq 2mDFA$.* □

7 Algorithmic questions

Picture walking automata admit efficient parsing of pictures. Pictures of a language can be recognized in linear time on the size $n = |p| \times \overline{p}$ of the picture p.

In the case of DFA, the linear time recognition can be done using logarithmic space: By Theorem 1 we can assume the DFA halts on every input picture so it is enough to run the DFA until it halts. This happens within $O(n)$ steps, and during the computation one stores the current state and position of the DFA using $O(\log n)$ space.

In the cases of NFA, UFA and AFA, linear space is enough. Consider the directed graph whose vertices are the instantaneous descriptions (q, x) of the automaton, and edges are given by the successor relation $l_1 \longrightarrow l_2$. For NFA, picture recognition amounts to finding a path from an initial ID to an accepting ID, which can be solved using standard depth-first search. In AFA, the vertices are classified as universal and existential. A simple linear-time algorithm executes a DFS search from the final ID's backwards, progressively marking new vertices that lead to acceptance. A universal vertex is marked only when all its followers become marked, while for existential vertices it is enough to have one follower marked. A picture is in the language iff an initial ID gets marked.

Another family of algorithmic questions concern the languages recognized by given automata. Here the situation is quite different, and undecidability usually prevails. The basic question is the emptiness problem: does the given automaton accept any pictures ? It was mentioned already in [1] that unary emptiness for DFA is undecidable because Minsky machines can be simulated. Recall that Minsky's two-counter machine without an input tape consists of a deterministic finite automaton and two counters, each storing one non-negative integer. The machine can detect when either counter is zero. The machine changes its state according to a deterministic transition rule. The new state only depends on the old state and on the results of the tests that check whether either counter is equal to zero. The transition rule also specifies whether to increment, decrement or keep unchanged the counters.

It is well known that two-counter machines can simulate Turing machines [21]. In particular, it is undecidable whether a given machine reaches a specified halting state h when started in its initial state i with both counters initialized to value 0.

A 2-counter machine can be interpreted as a DFA that operates on the (infinite) quadrant of the plane and has the same finite states as the 2-counter machine. The position of the DFA encodes the two counter values: Position (x, y) represents counter values x and y. Increments and decrements of the counters correspond to horizontal and vertical steps of the DFA on the plane.

Any computation of a 2-counter machine can therefore be simulated by a DFA inside a sufficiently large rectangle. The dimensions of the rectangle have to be at least as large as the largest counter values used during the accepting computation. Zero values of the counters can be recognized as these correspond to the machine stepping on the top or left boundary of the rectangle. If the DFA tries to step on the right or the bottom border, it enters an error state e that indicates that the rectangle was not large enough.

It is clear that the DFA outlined above accepts exactly the rectangles of sizes $w \times h$ where w and h are greater than the largest values in the two counters,

respectively, used during the accepting run of the Minsky machine. If the Minsky machine does not halt, the DFA does not accept any rectangle. Hence we have the following.

Theorem 20. *It is undecidable if a given unary DFA accepts any rectangles [1].*

□

In the restricted models the situation is more interesting, and various decidability questions have been investigated among three-way automata [22–24]. In [22], the unary emptiness was shown to be decidable among 3NFA, and in [24] the result was extended to arbitrary alphabets. It was also shown in [24] that unary emptiness is undecidable among 3AFA, and even among 2AFA.

Theorem 21. *The emptiness problem is*

- *decidable among 3NFA [22, 24],*
- *undecidable among unary 2AFA [24].*

□

Finally, we mention another application of the Minsky machine simulation by DFA, stating that the sizes of unary squares recognized by DFA can form a very sparse set [9].

Theorem 22. *Let $\{a_1, a_2, \ldots\}$ be any recursively enumerable set of positive integers. There exists a DFA that recognizes the language of unary squares of sizes $b_i \times b_i$ for $i = 1, 2, \ldots$ where $b_i > a_i$ for all $i = 1, 2, \ldots$.* □

The theorem is proved easily using the DFA simulation of a two-counter machine. First, an NFA is build where non-determinism is used to select the initial counter values for the Minsky machine. Such limited non-determinism can then be removed using the trick from [10]. See [9] for the details.

8 Conclusions

To conclude, we collect the open problems from the text into a single list.

Conjecture 1 *The language of unary squares with prime side length are not in DFA.*

Conjecture 2 *DFA is a proper subset of NFA when restricted to unary squares.*

Conjecture 3 *UFA is not closed under union.*

Conjecture 4 *NFA = FNFA on three-dimensional pictures.*

Conjecture 5 *The connected patterns are not in DFA.*

9 Acknowledgements

We would like to thank Ilkka Törmä for helpful remarks on marker automata, and for reviewing the section on them.

References

1. Blum, M., Hewitt, C.: Automata on a 2-dimensional tape. In: FOCS, IEEE (1967) 155–160
2. Inoue, K., Takanami, I.: A survey of two-dimensional automata theory. Inf. Sci. **55**(1-3) (1991) 99–121
3. Giammarresi, D., Restivo, A. In: Two-dimensional languages. Springer-Verlag New York, Inc., New York, NY, USA (1997) 215–267
4. Kari, J., Moore, C.: New results on alternating and non-deterministic two-dimensional finite-state automata. In Ferreira, A., Reichel, H., eds.: STACS. Volume 2010 of Lecture Notes in Computer Science., Springer (2001) 396–406
5. Salo, V.: Classes of picture languages defined by tiling systems, automata and closure properties. Master's thesis, University of Turku (2011)
6. Ito, A., Inoue, K., Takanami, I., Taniguchi, H.: Two-dimensional alternating turing machines with only universal states. Information and Control **55**(1-3) (1982) 193–221
7. Ibarra, O.H., Jiang, T., 0008, H.W.: Some results concerning 2-d on-line tessellation acceptors and 2-d alternating finite automata. In: MFCS. (1991) 221–230
8. Okazaki, T., Zhang, L., Inoue, K., Ito, A., 0002, Y.W.: A note on two-dimensional probabilistic finite automata. Inf. Sci. **110**(3-4) (1998) 303–314
9. Kari, J., Moore, C.: Rectangles and squares recognized by two-dimensional automata. In Karhumäki, J., Maurer, H.A., Paun, G., Rozenberg, G., eds.: Theory Is Forever. Volume 3113 of Lecture Notes in Computer Science., Springer (2004) 134–144
10. Sipser, M.: Halting space-bounded computations. In: FOCS, IEEE (1978) 73–74
11. Rosenfeld, A.: Picture Languages: Formal Models for Picture Recognition. Academic Press, Inc., Orlando, FL, USA (1979)
12. Lindgren, K., Moore, C., Nordahl, M.: Complexity of two-dimensional patterns. Journal of Statistical Physics **91** (1998) 909–951 10.1023/A:1023027932419.
13. Ladner, R.E., Lipton, R.J., Stockmeyer, L.J.: Alternating pushdown and stack automata. SIAM J. Comput. **13**(1) (1984) 135–155
14. Parikh, R.: On context-free languages. J. ACM **13**(4) (1966) 570–581
15. Szepietowski, A.: Some remarks on two-dimensional finite automata. Inf. Sci. **63**(1-2) (1992) 183–189
16. Inoue, K., Takanami, I.: Three-way tape-bounded two-dimensional turing machines. Inf. Sci. **17**(3) (1979) 195–220
17. Ito, A., Inoue, K., Takanami, I.: A note on three-way two-dimensional alternating turing machines. Inf. Sci. **45**(1) (1988) 1–22
18. Ito, A., Inoue, K., Takanami, I.: Deterministic two-dimensional on-line tessellation acceptors are equivalent to two-way two-dimensional alternating finite automata through 180-rotation. Theor. Comput. Sci. **66**(3) (1989) 273–287
19. Reinhardt, K.: On some recognizable picture-languages. In Brim, L., Gruska, J., Zlatuska, J., eds.: MFCS. Volume 1450 of Lecture Notes in Computer Science., Springer (1998) 760–770

20. Inoue, K., Takanami, I., Taniguchi, H.: Two-dimensional alternating turing machines. Theor. Comput. Sci. **27** (1983) 61–83
21. Minsky, M.L.: Computation: finite and infinite machines. Prentice-Hall, Inc. (1967)
22. Inoue, K., Takanami, I.: A note on decision problems for three-way two-dimensional finite automata. Inf. Process. Lett. **10**(4/5) (1980) 245–248
23. Kinber, E.B.: Three-way automata on rectangular tapes over a one-letter alphabet. Inf. Sci. **35**(1) (1985) 61–77
24. Petersen, H.: Some results concerning two-dimensional turing machines and finite automata. In Reichel, H., ed.: FCT. Volume 965 of Lecture Notes in Computer Science., Springer (1995) 374–382