Networks of Evolutionary Picture Processors

Paolo Bottoni[a], Anna Labella[a], Victor Mitrana[b,c], Jose M. Sempere[c]

[a]Department of Computer Science, "Sapienza" University of Rome
Via Salaria 113, 00198 Rome, Italy
[b]Faculty of Mathematics, University of Bucharest
Str. Academiei 14, 70109 Bucharest, Romania
[c]Department of Information Systems and Computation
Technical University of Valencia,
Camino de Vera s/n. 46022 Valencia, Spain

Abstract

We extend the study of accepting networks of evolutionary processors to rectangular pictures by introducing networks of evolutionary picture processors. Two ways of accepting pictures are considered: *weak acceptance*, when at least one output node is nonempty, and *strong acceptance*, when all output nodes are nonempty. Every language weakly accepted by a network can be strongly accepted by another network. The closure properties of these devices under some common operations on picture languages are briefly investigated. We show that networks of evolutionary picture processors can weakly accept the complement of any local language, as well as languages that are not recognizable. The problem of pattern matching in pictures is then considered in the framework of networks of evolutionary picture processors. A partial solution to this problem is given for the weak acceptance case and a similar result is discussed for the strong acceptance. Some open problems are finally discussed.

Key words: picture languages, networks of evolutionary processors, two-dimensional pattern matching

Email addresses: bottoni@di.uniroma1.it (**Paolo Bottoni**), labella@di.uniroma1.it (**Anna Labella**), mitrana@fmi.unibuc.ro (**Victor Mitrana**), jsempere@dsic.upv.es (**Jose M. Sempere**)

1. Introduction

Picture languages defined by different mechanisms have been studied extensively in the literature. Two-dimensional matrix and array models describing pictures that are rectangular arrays of symbols have been proposed in [21, 22, 27, 24]. On the other hand, models defining pictures that are connected arrays but not necessarily rectangular have been proposed as early as 70's [18] and a hierarchy of these grammars was considered in [26]. Classes of grammars for picture generation, again not necessarily rectangular, have been proposed in [3, 16, 23]. A new model of recognizable picture languages, extending to two dimensions the characterization of the one-dimensional recognizable languages in terms of alphabetic morphisms of local languages, has been introduced in [7]. Similarly to the string case, characterizations of recognizable picture series were proposed, see, e.g., [2, 13]. An early survey on automata recognizing rectangular pictures languages is [9], a bit more recent one considering different mechanisms defining picture languages, not necessarily rectangular, is [18] and an even more recent and concise one is [6]. Rather unexpected connections between different types of picture languages and logics were reported in [8, 14].

This work tries to carry over to rectangular pictures the investigation started in [4] and [10] and continued in a series of papers; the reader may consult the early survey [12]. In these papers a mechanism inspired from cell biology was considered, namely networks of evolutionary processors, i.e. networks whose nodes are very simple processors able to perform just one type of point mutation (insertion, deletion or substitution of a symbol). These nodes are endowed with filters defined by some very simple context conditions. In a more general view, each node processor acts on the local data in accordance with some predefined rules. Local data is then transmitted over the network following a given protocol. Only data which can pass a filtering process can be communicated. This filtering process may require to satisfy some conditions imposed by the sending processor, by the receiving processor or by both of them. All the nodes simultaneously send their data to and receive data from the nodes which they are connected to.

The approach proposed here is very different from the one considered in [15] and continued in [5], where networks for *generating* 2D and 3D languages are investigated. In this paper, we consider networks of evolutionary picture processors acting on rectangular pictures as *acceptors*. Each node is either a row/column substitution node or a row/column deletion node. The action

of each node on the data it contains is precisely defined. For instance, if a node is a row substitution node, then it can substitute a letter by another letter in either the topmost or the last or an arbitrary row. Moreover, if there are more occurrences of the letter that is to be substituted in the row on which the substitution rule acts, then each such occurrence is substituted in different copies of that picture. An implicit assumption is that arbitrarily many copies of every picture are available. A similar informal explanation concerns the column substitution and deletion nodes, respectively.

Although this computational process is not exactly an evolutionary process in the Darwinian sense, the rewriting operations performed in the nodes might be interpreted as a 2D generalization of gene mutations in chromosomes and the filtering process viewed as a selection process. Recombination is missing but it was asserted that evolutionary and functional relationships between genes can be captured by taking only local mutations into consideration [20]. We would like to stress from the very beginning that the evolutionary processor we propose here is just a mathematical object and the biological hints mentioned above are intended to explain in an informal way how some biological phenomena are *sources of inspiration* for our model.

The paper is structured as follows: in the next section we present the formal definitions of the concepts forming the computational model of networks of picture processors; then we discuss a few preliminary results useful for the rest of the paper. We show that every language weakly accepted by a network can be strongly accepted by another network and prove a few closure properties of these devices. We explain the composition of two or more networks by two examples. The fourth section presents a brief comparison of the computational power of accepting networks of evolutionary picture processors with that of the models defining recognizable and local picture languages. We show that these networks can weakly accept the complement of any local language and languages that are not recognizable. Afterwards we consider the problem of pattern matching in pictures and propose a partial solution based on networks of evolutionary picture processors with weak acceptance. The same strategy is then applied to networks of evolutionary picture processors with strong acceptance. Finally, we briefly discuss some open problems.

2. Basic Definitions

For basic terminology and notations concerning the theory of one-dimensional languages the reader is referred to [19]. The definitions and notations concerning two-dimensional languages are taken from [6].

The set of natural numbers from 1 to n is denoted by $[n]$. The cardinality of a finite set A is denoted by $card(A)$. Let V be an alphabet, V^* the set of one-dimensional strings over V and ε the empty string. A *picture* (or a two-dimensional string) over the alphabet V is a two-dimensional array of elements from V. We denote the set of all pictures over the alphabet V by V_*^*, while the empty picture will be still denoted by ε. A two-dimensional language over V is a subset of V_*^*. The minimal alphabet containing all symbols appearing in a picture π is denoted by $alph(\pi)$. Let π be a picture in V_*^*; we denote the number of rows and the number of columns of π by $\overline{\pi}$ and $|\pi|$, respectively. The pair $(\overline{\pi}, |\pi|)$ is called *the size* of the picture π. The size of the empty picture ε is obviously $(0, 0)$. The set of all pictures over V of size (m, n), where $m, n \geq 1$, is denoted by V_m^n. The symbol placed at the intersection of the ith row with the jth column of the picture π, is denoted by $\pi(i, j)$. The row picture of size $(1, n)$ containing occurrences of the symbol a only is denoted by a_1^n. Similarly the column picture of size $(m, 1)$ containing occurrences of the symbol a only is denoted by a_m^1.

We recall informally the row and column concatenation operations between pictures. For a formal definition the reader is referred to [9] or [6]. The row concatenation of two pictures π of size (m, n) and ρ of size (m', n') is denoted by ® and is defined only if $n = n'$. The picture $\pi ® \rho$ is obtained by adding the picture ρ after the last row of π. Analogously one defines the column concatenation denoted by ©. We now define four new operations, in some sense the inverse operations of the row and column concatenation. Let π and ρ be two pictures of size (m, n) and (m', n'), respectively. We define

- The column right-quotient of π with ρ: $\pi /_{\rightarrow} \rho = \theta$ iff $\pi = \theta © \rho$.
- The column left-quotient of π with ρ: $\pi /_{\leftarrow} \rho = \theta$ iff $\pi = \rho © \theta$.
- The row down-quotient of π with ρ to the right: $\pi /_{\downarrow} \rho = \theta$ iff $\pi = \theta ® \rho$.
- The column up-quotient of π with ρ: $\pi /_{\uparrow} \rho = \theta$ iff $\pi = \rho ® \theta$.

Let V be an alphabet; a rule of the form $a \rightarrow b(X)$, with $a, b \in V \cup \{\varepsilon\}$ and $X \in \{-, |\}$ is called an *evolutionary rule*. For any rule $a \rightarrow b(X)$, X

indicates which component of a picture (row if $X = -$ or column if $X = |$) the rule is applied to. We say that a rule $a \to b(X)$ is a *substitution rule* if both a and b are not ε, is a *deletion rule* if $a \neq \varepsilon$, $b = \varepsilon$, and is an *insertion rule* if $a = \varepsilon$, $b \neq \varepsilon$. In this paper we shall ignore insertion rules because we want to process every given picture in a space bounded by the size of that picture. We denote by $RSub_V = \{a \to b(-) \mid a, b \in V\}$ and $RDel_V = \{a \to \varepsilon(-) \mid a \in V\}$. The sets $CSub_V$ and $CDel_V$ are defined analogously.

Given a rule σ as above and a picture $\pi \in V_m^n$, we define the following *actions* of σ on π:

• If $\sigma \equiv a \to b(|) \in CSub_V$, then

$$\sigma^{\leftarrow}(\pi) = \begin{cases} \{\pi' \in V_m^n \mid \exists i \in [m]\ (\pi(i,1) = a\ \&\ \pi'(i,1) = b), \pi'(k,1) = \pi(k,1), \\ k \in [m] \setminus \{i\}, \pi'(j,l) = \pi(j,l), \forall (j,l) \in [m] \times ([n] \setminus \{1\})\} \\ \\ \{\pi\}, \text{ if the first column of } \pi \text{ does not contain any occurrence} \\ \text{ of the letter } a. \end{cases}$$

$$\sigma^{\rightarrow}(\pi) = \begin{cases} \{\pi' \in V_m^n \mid \exists i \in [m]\ (\pi(i,n) = a\ \&\ \pi'(i,n) = b), \pi'(k,n) = \pi(k,n), \\ k \in [m] \setminus \{i\}, \pi'(j,l) = \pi(j,l), \forall (j,l) \in [m] \times [n-1]\} \\ \\ \{\pi\}, \text{ if the last column of } \pi \text{ does not contain any occurrence} \\ \text{ of the letter } a. \end{cases}$$

$$\sigma^{*}(\pi) = \begin{cases} \{\pi' \in V_m^n \mid \exists (i,j) \in [m] \times [n] \text{ such that } \pi(i,j) = a \text{ and} \\ \pi'(i,j) = b, \pi'(k,l) = \pi(k,l), \forall (k,l) \in ([m] \times [n]) \setminus \{(i,j)\}\} \\ \\ \{\pi\}, \text{ if no column of } \pi \text{ contains any occurrence of the letter } a. \end{cases}$$

Note that a rule as above is applied to all occurrences of the letter a either in the first or in the last or in any column of π, respectively, in different copies of the picture π. Analogously, we define:

• If $\sigma \equiv a \to b(-) \in RSub_V$, then

$$\sigma^{\uparrow}(\pi) \quad = \quad \begin{cases} \{\pi' \in V_m^n \mid \exists i \in [n] (\pi(1,i) = a\ \&\ \pi'(1,i) = b), \pi'(1,k) = \pi(1,k), \\ \forall k \in [n] \setminus \{i\}, \pi'(j,l) = \pi(j,l), \forall (j,l) \in ([m] \setminus \{1\}) \times [n]\} \\ \\ \{\pi\}, \text{ if the first row of } \pi \text{ does not contain any occurrence} \\ \text{ of the letter } a. \end{cases}$$

$$\sigma^{\downarrow}(\pi) \;=\; \begin{cases} \{\pi' \in V_m^n \mid \exists i \in [n](\pi(m,i) = a \;\&\; \pi'(m,i) = b), \pi'(m,k) = \pi(m,k), \\ \forall k \in [n] \setminus \{i\}, \pi'(j,l) = \pi(j,l), \forall (j,l) \in [m-1] \times [n]\} \\ \\ \{\pi\}, \text{ if the last row of } \pi \text{ does not contain any occurrence} \\ \text{of the letter } a. \end{cases}$$

$\sigma^*(\pi) = \rho^*(\pi)$, where $\rho \equiv a \to b(|) \in CSub_V$.

• If $\sigma \equiv a \to \varepsilon(|) \in CDel_V$, then

$$\sigma^{\leftarrow}(\pi) \;=\; \begin{cases} \pi/_{\leftarrow}\rho, \text{ where } \rho \text{ is the leftmost column of } \pi, \text{ if the leftmost} \\ \text{column of } \pi \text{ does contain at least one occurrence of the letter } a \\ \\ \pi, \text{ if the leftmost column of } \pi \text{ does not contain any occurrence} \\ \text{of the letter } a. \end{cases}$$

$$\sigma^{\rightarrow}(\pi) \;=\; \begin{cases} \pi/_{\rightarrow}\rho, \text{ where } \rho \text{ is the rightmost column of } \pi, \text{ if the rightmost} \\ \text{column of } \pi \text{ does contain at least one occurrence of the letter } a \\ \\ \pi, \text{ if the rightmost column of } \pi \text{ does not contain any occurrence} \\ \text{of the letter } a. \end{cases}$$

$$\sigma^{*}(\pi) \;=\; \begin{cases} \{\pi_1 \copyright \pi_2 \mid \pi = \pi_1 \copyright \rho \copyright \pi_2, \text{ for some } \pi_1, \pi_2 \in V_*^* \text{ and } \rho \text{ is a} \\ \text{column of } \pi_1 \text{ that contains an occurrence of the letter } a\} \\ \\ \{\pi\}, \text{ if } \pi \text{ does not contain any occurrence of the letter } a. \end{cases}$$

In an analogous way we define:

• If $\sigma \equiv a \to \varepsilon(-) \in RDel_V$, then

$$\sigma^{\uparrow}(\pi) \;=\; \begin{cases} \pi/_{\uparrow}\rho, \text{ where } \rho \text{ is the first row of } \pi, \text{ if the first row} \\ \text{of } \pi \text{ does contain at least one occurrence of the letter } a \\ \\ \pi, \text{ if the first row of } \pi \text{ does not contain any occurrence} \\ \text{of the letter } a. \end{cases}$$

$$\sigma^{\downarrow}(\pi) \;=\; \begin{cases} \pi/_{\downarrow}\rho, \text{ where } \rho \text{ is the last row of } \pi, \text{ if the last row} \\ \text{of } \pi \text{ does contain at least one occurrence of the letter } a \\ \\ \pi, \text{ if the last row of } \pi \text{ does not contain any occurrence} \\ \text{of the letter } a. \end{cases}$$

$$\sigma^*(\pi) \quad = \quad \begin{cases} \{\pi_1 ® \pi_2 \mid \pi = \pi_1 ® \rho ® \pi_2, \text{ for some } \pi_1, \pi_2 \in V_*^* \text{ and } \rho \text{ is a} \\ \text{row of } \pi_1 \text{ that contains an occurrence of the letter } a\} \\ \\ \{\pi\}, \text{ if } \pi \text{ does not contain any occurrence of the letter } a. \end{cases}$$

For every rule σ, action $\alpha \in \{*, \leftarrow, \rightarrow, \uparrow, \downarrow\}$, and $L \subseteq V_*^*$, we define the *α-action of σ on L* by $\sigma^\alpha(L) = \bigcup_{\pi \in L} \sigma^\alpha(\pi)$. Given a finite set of rules M, we define the *α-action of M* on the picture π and the language L by:

$$M^\alpha(\pi) = \bigcup_{\sigma \in M} \sigma^\alpha(\pi) \quad \text{and} \quad M^\alpha(L) = \bigcup_{\pi \in L} M^\alpha(\pi),$$

respectively. In what follows, we shall refer to the rewriting operations defined above as *evolutionary picture operations* since they may be viewed as the 2-dimensional linguistic formulations of local gene mutations. For two disjoint subsets P and F of an alphabet V and a picture π over V, we define the following two predicates which will define later two types of filters:

$$\begin{array}{lcrcl} rc_s(\pi; P, F) & \equiv & P \subseteq alph(\pi) & \wedge & F \cap alph(\pi) = \emptyset \\ rc_w(\pi; P, F) & \equiv & alph(\pi) \cap P \neq \emptyset & \wedge & F \cap alph(\pi) = \emptyset. \end{array}$$

The construction of these predicates is based on *context conditions* defined by the two sets P (*permitting contexts/symbols*) and F (*forbidding contexts/symbols*). Informally, both conditions requires that no forbidding symbol is present in π; furthermore the first condition requires all permitting symbols to appear in π, while the second one requires that at least one permitting symbol appears in π. It is plain to see that the first condition is stronger than the second one.

For every picture language $L \subseteq V_*^*$ and $\beta \in \{s, w\}$, we define:

$$rc_\beta(L, P, F) = \{\pi \in L \mid rc_\beta(\pi; P, F) = \texttt{true}\}.$$

An *evolutionary picture processor over V* is a 5-tuple (M, PI, FI, PO, FO), where:

– Either $(M \subseteq CSub_V)$ or $(M \subseteq RSub_V)$ or $(M \subseteq CDel_V)$ or $(M \subseteq RDel_V)$. The set M represents the set of evolutionary rules of the processor. As one can see, a processor is "specialized" into one type of evolutionary operation, only.

– $PI, FI \subseteq V$ are the *input* sets of permitting/forbidding symbols (contexts) of the processor, while $PO, FO \subseteq V$ are the *output* sets of permitting/forbidding symbols of the processor (with $PI \cap FI = \emptyset$ and $PO \cap FO = \emptyset$).

We denote the set of evolutionary picture processors over V by EPP_V. As we stated in the Introduction, the evolutionary processor described here is just a mathematical concept similar to that of an evolutionary algorithm, both being inspired from the Darwinian evolution.

An *accepting network of evolutionary picture processors* (ANEPP for short) is a 8-tuple $\Gamma = (V, U, G, N, \alpha, \beta, x_I, Out)$, where:

- V and U are the input and network alphabet, respectively, $V \subseteq U$.
- $G = (X_G, E_G)$ is an undirected graph without loops with the set of vertices X_G and the set of edges E_G. G is called the *underlying graph* of the network.
- $N : X_G \longrightarrow EPP_V$ is a mapping which associates with each node $x \in X_G$ the evolutionary processor $N(x) = (M_x, PI_x, FI_x, PO_x, FO_x)$.
- $\alpha : X_G \longrightarrow \{*, \leftarrow, \rightarrow, \uparrow, \downarrow\}$; $\alpha(x)$ gives the action mode of the rules of node x on the pictures existing in that node.
- $\beta : X_G \longrightarrow \{s, w\}$ defines the type of the *input/output filters* of a node. More precisely, for every node, $x \in X_G$, the following filters are defined:

$$\begin{aligned} \text{input filter:} \quad & \rho_x(\cdot) = rc_{\beta(x)}(\cdot; PI_x, FI_x), \\ \text{output filter:} \quad & \tau_x(\cdot) = rc_{\beta(x)}(\cdot; PO_x, FO_x). \end{aligned}$$

  That is, $\rho_x(\pi)$ (resp. $\tau_x(\pi)$) indicates whether or not the picture π can pass the input (resp. output) filter of x. More generally, $\rho_x(L)$ (resp. $\tau_x(L)$) is the set of pictures of L that can pass the input (resp. output) filter of x.

- $x_I \in X_G$ is the *input* node and $Out \subset X_G$ is the set of *output* nodes of Γ.

We say that $card(X_G)$ is the size of Γ. A *configuration* of an ANEPP Γ as above is a mapping $C : X_G \longrightarrow 2^{U^*_*}$ which associates a finite set of pictures with every node of the graph. A configuration may be understood as the sets

of pictures which are present in any node at a given moment. Given a picture $\pi \in V_*^*$, the initial configuration of Γ on π is defined by $C_0^{(\pi)}(x_I) = \{\pi\}$ and $C_0^{(\pi)}(x) = \emptyset$ for all $x \in X_G - \{x_I\}$.

A configuration can change via either an *evolutionary step* or a *communication step*. When changing via an evolutionary step, each component $C(x)$ of the configuration C is changed in accordance with the set of evolutionary rules M_x associated with the node x and the way of applying these rules $\alpha(x)$. Formally, we say that the configuration C' is obtained in *one evolutionary step* from the configuration C, written as $C \Longrightarrow C'$, iff

$$C'(x) = M_x^{\alpha(x)}(C(x)) \text{ for all } x \in X_G.$$

When changing via a communication step, each node processor $x \in X_G$ sends one copy of each picture it has, which is able to pass the output filter of x, to all the node processors connected to x and receives all the pictures sent by any node processor connected with x provided that they can pass its input filter.

Formally, we say that the configuration C' is obtained in *one communication step* from configuration C, written as $C \vdash C'$, iff

$$C'(x) = (C(x) - \tau_x(C(x))) \ \cup \bigcup_{\{x,y\} \in E_G} (\tau_y(C(y)) \cap \rho_x(C(y))) \text{ for all } x \in X_G.$$

Note that pictures that cannot pass the output filter of a node remain in that node and can be further modified in the subsequent evolutionary steps, while pictures that can pass the output filter of a node are expelled. Further, all the expelled pictures that cannot pass the input filter of any node are lost.

Let Γ be an ANEPP, the computation of Γ on an input picture $\pi \in V_*^*$ is a sequence of configurations $C_0^{(\pi)}, C_1^{(\pi)}, C_2^{(\pi)}, \ldots$, where $C_0^{(\pi)}$ is the initial configuration of Γ on π, $C_{2i}^{(\pi)} \Longrightarrow C_{2i+1}^{(\pi)}$ and $C_{2i+1}^{(\pi)} \vdash C_{2i+2}^{(\pi)}$, $\forall i \geq 0$. Note that configurations are changed by alternative steps. By the previous definitions, each configuration $C_i^{(\pi)}$ is uniquely determined by $C_{i-1}^{(\pi)}$. A computation *halts*, and is said to be *weak (strong) halting*, if one of the following two conditions holds:

(i) There exists a configuration in which the set of pictures existing in at least one output node (all output nodes) is non-empty. In this case, the computation is said to be a weak (strong) *accepting computation*.

(ii) There exist two identical configurations obtained either in consecutive evolutionary steps or in consecutive communication steps.

The *picture language weakly (strongly) accepted* by Γ is

$$L_{wa(sa)}(\Gamma) = \{\pi \in V_*^* \mid \text{ the computation of } \Gamma \text{ on } \pi \text{ is a weak (strong) accepting one}\}.$$

In network theory, some types of underlying graphs are common like *rings, stars, grids*, etc. Networks of evolutionary strings processors, seen as language generating or accepting devices, having underlying graphs of these special forms have been considered in several papers, see, e.g., [12] for an early survey. We focus here on *complete* ANEPPs i.e., ANEPPs having a complete underlying graph, so that we can replace the graph G in the definition of an ANEPP by the set of its nodes.

3. Preliminary Results

The following two notions will be very useful in the sequel. If h is a one-to-one mapping from U to W and $\Gamma = (V, U, \chi, N, \alpha, \beta, x_I, Out)$ is an ANEPP, then we denote by Γ_h the ANEPP $\Gamma_h = (h(V), h(U), \chi, h(N), \alpha, \beta, x_I, Out)$, where by $h(N)$ we mean $h(N)(x) = (h(M_x), h(PI_x), h(FI_x), h(PO_x), h(FO_x))$ for every $x \in \chi$, provided that $N(x) = (M_x, PI_x, FI_x, PO_x, FO_x)$. Further, $h(a \to b(X)) = h(a) \to h(b)(X)$ for any evolutionary rule $a \to b(X)$. Now, given two ANEPPs $\Gamma_i = (V_i, U_i, \chi_i, N_i, \alpha_i, \beta_i, x_I^i, Out_i)$, $i = 1, 2$, $\chi_1 \cap \chi_2 = \emptyset$, we denote by $\Gamma_1 \sqcup \Gamma_2 = (V_1, U_1 \cup U_2, \chi_1 \cup \chi_2, N, \alpha, \beta, x_I^1, Out_2)$ the *composition* of the two ANEPPs Γ_1 and Γ_2, where $\circ\,|_{\chi_i} = \circ_i$ for all $\circ \in \{N, \alpha, \beta\}$ and $i = 1, 2$.
We first establish a useful relationship between the two classes $\mathcal{L}_{wa}(ANEPP)$ and $\mathcal{L}_{sa}(ANEPP)$. As it was expected, we have

Theorem 1. $\mathcal{L}_{wa}(ANEPP) \subseteq \mathcal{L}_{sa}(ANEPP)$.

Proof. Actually, we prove a bit more general result, namely that for every ANEPP Γ there exists an ANEPP Γ' with one output node only and $L_{wa}(\Gamma) = L_{wa}(\Gamma') = L_{sa}(\Gamma')$. W.l.o.g. we assume that the set of rules in every output node of Γ is empty and that all its filter types are strong. Indeed, if the filter type of one node is a weak one with P its input set of permitting symbols, then this node can be replaced by $2^{card(P)} - 1$ output nodes, each

of them having a strong filter type where the input sets of permitting and forbidding symbols are a nonempty subset of P and the empty set, respectively. Further on, the output set of permitting and forbidding symbols of every such node is $\{Z\}$ and the empty set, respectively, where Z is a new symbol. Now, in order to get Γ', we add one more node to Γ, which is the unique output node of Γ'. This node can receive only pictures containing the new symbol Z. We now associate with each output node of Γ a set of substitution rules formed by one substitution only, namely $X \rightarrow Z(-)$, where X is an arbitrary symbol from the input set of permitting symbols of that node applied in the $*$ mode. □

We now present a preliminary result concerning the closure properties of the classes $\mathcal{L}_{wa}(ANEPP)$ and $\mathcal{L}_{sa}(ANEPP)$.

Theorem 2. *1. The class $\mathcal{L}_{wa}(ANEPP)$ is closed under rotation, boolean union, projection, inverse projection.*
2. The class $\mathcal{L}_{sa}(ANEPP)$ is closed under rotation, boolean intersection, projection, inverse projection.

Proof. 1. The closure under rotation is immediate. For union, we give an informal proof that can be easily formalized by the reader. Let Γ_1 and Γ_2 be two ANEPPs; we construct a new ANEPP Γ that contains three subnetworks. In the input node of the first subnetwork, an arbitrary symbol of the input picture is substituted by either its primed copy or its barred copy. All pictures containing a primed symbol are received by a specific node while those containing a barred symbol are received by another specific node. All symbols of the pictures arrived in these two nodes are replaced by their primed and barred copies, respectively. When this process is finished, each of the two nodes contains only one picture. The picture containing primed symbols only is given as an input picture to the subnetwork formed from Γ_1 suitably modified. The other picture is processed analogously by the subnetwork formed from Γ_2 suitably modified. The set of output nodes of Γ is the union of the sets of output nodes of the modified Γ_1 and Γ_2. Clearly, $L_{wa}(\Gamma) = L_{wa}(\Gamma_1) \cup L_{wa}(\Gamma_2)$.

If $h : V \longrightarrow U$ is a projection and Γ is an ANEPP with input alphabet V, then let Γ' be the ANEPP with input alphabet U formed by two subnetworks as follows. In the input node of the first subnetwork, each symbol b of the input picture is substituted by a symbol a' such that a' is a copy of $a \in V$ that does not appear in $V \cup U$ and $h(a) = b$. When all symbols of the input

picture are substituted, all the obtained pictures will be sent to the input node of the subnetwork formed from Γ suitably modified. It is plain to see that $h(L_{wa}(\Gamma)) = L_{wa}(\Gamma')$. The construction for the closure under inverse projection is pretty similar and left to the reader.

2. The closure under intersection, projection and inverse projection follows similarly to the previous case. Note the fundamental role played by the strong acceptance in the case of intersection. □

We continue the series of preliminary results with one simple example which lays the basis for further results.

Example 1. *Let L be the set of all pictures $\pi \in V_2^*$ with two identical rows over the alphabet V. The language L can be formally described as*

$$L = \{\pi \in V_2^m \mid \pi(1,i) = \pi(2,i),\ i \in [m],\ m \geq 1\}.$$

L can be weakly (strongly) accepted by the following complete ANEPP with $3 \cdot card(V)+3$ nodes, namely x_I, x_a, x'_a, x''_a, $a \in V$, x_{del}, the working alphabet $U = V \cup \{X_a, Y_a \mid a \in V\} \cup \{X, Y\}$, and one output node only, namely x_O:

$$x_I : \begin{cases} M = \emptyset, \\ PI = V, FI = U \setminus V, \\ PO = V, FO = \emptyset, \\ \alpha = \uparrow,\ \beta = w, \end{cases} \qquad x_a : \begin{cases} M = \{a \to X_a(-)\}, \\ PI = V, FI = U \setminus V, \\ PO = \{X_a\}, FO = \emptyset, \\ \alpha = \uparrow,\ \beta = w, \end{cases}$$

$$x_{del} : \begin{cases} M = \{X_a \to \varepsilon(|) \mid a \in V\}, \\ PI = \{Y_a \mid a \in V\}, FI = \emptyset, \\ PO = V, FO = U \setminus V, \\ \alpha = \leftarrow,\ \beta = w, \end{cases} \qquad x'_a : \begin{cases} M = \{a \to Y_a(-)\}, \\ PI = \{X_a\}, FI = \{Y_b \mid b \in V\}, \\ PO = \{Y_a\}, FO = \emptyset, \\ \alpha = \downarrow,\ \beta = s, \end{cases}$$

$$x_O : \begin{cases} M = \emptyset, \\ PI = \{X, Y\}, \\ FI = U \setminus \{X, Y\}, \\ PO = \emptyset, FO = U, \\ \alpha = *,\ \beta = s. \end{cases} \qquad x''_a : \begin{cases} M = \{X_a \to X(|), Y_a \to Y(|)\}, \\ PI = \{X_a, Y_a\}, \\ FI = U \setminus \{X_a, Y_a\}, \\ PO = \{X, Y\}, FO = \emptyset, \\ \alpha = \rightarrow,\ \beta = s, \end{cases}$$

Let us follow a computation of this network on an input picture π. In the input node no action is done on this picture in the first computation step, but a copy of this picture is sent simultaneously to all nodes x_a, $a \in V$ in the next communication step. We now follow what happens with this picture in the node x_a for some a. Here an occurrence of a of the first row is replaced by X_a and all pictures are sent out. They can be received by x'_a only, where

an occurrence of a of the last row is replaced by Y_a. All pictures going out from all nodes x'_a, $a \in V$, arrive in x_{del}. They all remain here forever except for those having the leftmost column starting with X_a and ending with Y_a, for some $a \in V$. They are sent out after their leftmost column is removed. A copy of each of them will enter every node x_a, $a \in V$, and the process resumes. The computation either continues until a column picture starting with X_a and ending with Y_a, for some $a \in V$ is obtained in x'_a, or halts without accepting the input picture. If such a column picture is obtained in x'_a, for some $a \in V$, then it enters x''_a where X_a and Y_a are replaced by X and Y, respectively. The new column picture is sent out by x''_a but it is lost unless its length is two, in which case it enters x_O and the input picture is accepted. By these explanations, it follows that every input picture with a different number of rows than two cannot be accepted. □

Clearly, the language of all pictures of size $(n, 2)$, $n \geq 1$, over a given alphabet V, where the two columns are identical can also be accepted by an ANEPP. The network from Example 1 can be extended to decide the language of all pictures (of any size) having two identical rows. The role of this example is to show how to ANEPPs can be combined in order to form a new ANEPP.

Example 2. *Let L be the set of all pictures $\pi \in V_*^*$ with two identical rows over the alphabet V. The language L can be formally described as*

$$L = \{\pi \in V_n^m \mid \exists i, j \in N, 1 \leq i \neq j \leq n \, (\pi(i,k) = \pi(j,k)), \; k \in [m], n, m \geq 1\}.$$

In what follows we assume that the same alphabet V is used in Examples 1 and 2. First, we construct the ANEPP $\Gamma_1 = (V, U_1, \chi_1, N_1, \alpha_1, \beta_1, y_I, \{\bar{y}'_a \mid a \in V\})$ of size $4card(V) + 2$ with the working alphabet $U_1 = V \cup \{a', a'', \bar{a} \mid a \in V\}$, and the nodes defined by:

$$y_I : \begin{cases} M = \{a \to a'(|) \mid a \in V\}, \\ PI = \emptyset, FI = U_1, \\ PO = \{a' \mid a \in V\}, FO = \emptyset, \\ \alpha_1 = \leftarrow, \; \beta_1 = w, \end{cases} \qquad y' : \begin{cases} M = \{a \to a''(|) \mid a \in V\}, \\ PI = \{a' \mid a \in V\}, \\ FI = U_1 \setminus (V \cup \{a' \mid a \in V\}), \\ PO = \{a'' \mid a \in V\}, FO = \emptyset, \\ \alpha_1 = \leftarrow, \; \beta_1 = w, \end{cases}$$

$$y_a(a \in V) : \begin{cases} M = \{b \to \varepsilon(-) \mid b \in V\}, \\ PI = \{a', a''\}, \\ FI = U_1 \setminus (V \cup \{a', a''\}), \\ PO = \emptyset, FO = \emptyset, \\ \alpha_1 = *, \; \beta_1 = s, \end{cases}$$

$$y'_a(a \in V): \begin{cases} M = \{b \to \varepsilon(-) \mid b \in V\}, \\ PI = \{a', a''\}, \\ FI = U_1 \setminus (V \cup \{a', a''\}), \\ PO = \emptyset, FO = \emptyset, \\ \alpha_1 = *,\ \beta_1 = s, \end{cases}$$

$$\bar{y}_a\ (a \in V): \begin{cases} M = \{a' \to \bar{a}(-)\} \cup \{b \to \bar{b}(-) \mid b \in V\}, \\ PI = \{a', a''\}, FI = U_1 \setminus (V \cup \{a', a''\}), \\ PO = \emptyset, FO = \{a'\}, \\ \alpha_1 = \uparrow,\ \beta_1 = s, \end{cases}$$

$$\bar{y}'_a\ (a \in V): \begin{cases} M = \{a'' \to \bar{a}(-)\} \cup \{b \to \bar{b}(-) \mid b \in V\}, \\ PI = \{\bar{a}, a''\}, FI = U_1 \setminus (V \cup \{\bar{a}, a''\}), \\ PO = \emptyset, FO = U_1 \setminus \{\bar{b} \mid b \in V\}, \\ \alpha_1 = \downarrow,\ \beta_1 = s, \end{cases}$$

The informal idea is the following. In the nodes y_I and y' two symbols, say a and b (possibly the same) on the leftmost column are replaced by a' and b'', respectively. If $a \neq b$, then no other pictures can be obtained. If $a = b$, then by means of the nodes y_a and y'_a, some rows are deleted from the pictures. Only those pictures in which all rows except the rows starting with a' and a'' are deleted can still be active for the rest of the computation. Furthermore, these pictures must have the first row starting with a' and the second one starting with a''. We follow what happens with these pictures as soon as they arrive in $\bar{y}_a$, for some $a \in V$. Here some symbols from the first row are transformed into their barred copies, including a'. Then, some symbols on the second row are transformed into their barred copies in $\bar{y}'_a$. A picture cannot go out from $\bar{y}'_a$, for any $a \in V$, unless all its symbols were substituted by barred copies. Therefore, for a picture to go out from $\bar{y}'_a$, it must have only barred symbols on its first row when leaving $\bar{y}_a$.

We now consider the ANEPP $\Gamma = (V, U, \chi, N, \alpha, \beta, x_I, x_O)$ from Example 1 and the one-to-one mapping $h : U \longrightarrow \{\bar{a} \mid a \in V\} \cup (U \setminus V)$ defined by $h(a) = \bar{a}$, $a \in V$, and $h(b) = b$, $b \in U \setminus V$. Let Γ_2 the ANEPP obtained from Γ_h by replacing $h(U)$ with $U_1 \cup U$ wherever $h(U)$ appears in the definition of parameters of Γ_h. We claim that $\Gamma_1 \sqcup \Gamma_2$ weakly accepts L. Indeed, the subnetwork Γ_2 can start to work when it receives pictures having barred symbols only. These pictures can be obtained from the nodes $\bar{y}'_a$, $a \in V$. By the above explanations, they are pictures with only two rows that are barred copies of two rows randomly selected from the input picture. □

In what follows, instead of giving all the details of how two networks are merged, as in Example 2, we simply say that the pictures processed by the

network Γ_1 are given as inputs to the network Γ_2 suitably modified.

4. Comparison With Other Devices

In this section we compare the classes $\mathcal{L}_{wa}(ANEPP)$ and $\mathcal{L}_{sa}(ANEPP)$ of picture languages weakly and strongly accepted by ANEPPs, respectively, with $\mathcal{L}(LOC)$ and $\mathcal{L}(REC)$ denoting the classes of local and recognizable picture languages, respectively [7].

Theorem 3. $\mathcal{L}_{wa}(ANEPP) \setminus \mathcal{L}(REC) \neq \emptyset$.

Proof. We first claim that the following language

$$L = \{\pi \in V_{2n}^m \mid n, m \geq 1, (\pi(n, i) = \pi(n + 1, i)), \forall i \in [m]\}$$

is not recognizable, provided that $card(V) \geq 2$. Clearly, L consists of all pictures that can be written in the form $\pi_1 ® \pi_2$, where π_1, π_2 are pictures of the same size and the last row of π_1 is equal to the first row of π_2. Assume that L is recognizable and let $L = h(L')$, where h is a projection from some alphabet U to V and $L' \subseteq U_*^*$ is a local language. For two positive integers n, m, let $L(n, m)$ be the subset of L formed by all pictures that can be written in the form $\pi_1 ® \pi_2$ with π_1, π_2 as above but satisfying also the following two conditions:

- both π_1 and π_2 are of size (n, m);
- neither π_1 nor π_2 contains two consecutive identical rows.

Therefore, there exists a subset $L'(n, m)$ of L' such that $L(n, m) = h(L'(n, m))$ for all n, m. Let m be fixed; as every set $L(n, m)$ is not empty for all values of n, it follows that all sets $L'(n, m)$ are nonempty as well.

Therefore, there are two pictures $\rho \in L'(n_1, m)$ and $\tau \in L'(n_2, m)$, with $n_1 \neq n_2$ such that the stripe rectangle of size $(2, m)$ consisting of the n_1-th and (n_1+1)-th rows in ρ equals the stripe rectangle of size $(2, m)$ consisting of the n_2-th and (n_2+1)-th rows in τ. Consequently, both pictures obtained from ρ and τ by interchanging their first halves with each other are in L'. However, the projection by h of any of these pictures is not in L, a contradiction.

We now prove that the language

$$L = \{\pi \in V_{2n}^m \mid n, m \geq 1, \pi(n, i) = \pi(n + 1, i), \forall i \in [m]\}$$

is in $\mathcal{L}_{wa}(ANEPP)$ for any alphabet V. We give only the description of the network processing the input pictures until they are sent to the input

node of the network from Example 1 suitably modified. The six nodes of this network are defined as follows:

$$x_I \ : \ \begin{cases} M = \{a \to X(-) \mid a \in V\} \cup \{a \to a'(-) \mid a \in V\}, \\ PI = \emptyset, FI = \{X, Y\} \cup \{a' \mid a \in V\}, \\ PO = \{X\} \cup \{a' \mid a \in V\}, FO = \emptyset, \\ \alpha = \uparrow, \ \beta = w, \end{cases}$$

$$x_1 \ : \ \begin{cases} M = \{a \to Y(-) \mid a \in V\}, \\ PI = \{X\}, FI = \{Y\} \cup \{a' \mid a \in V\}, \\ PO = \{Y\}, FO = \emptyset, \\ \alpha = \downarrow, \ \beta = w, \end{cases}$$

$$x_2 \ : \ \begin{cases} M = \{X \to \varepsilon(-)\}, \\ PI = \{X, Y\}, FI = \{a' \mid a \in V\}, \\ PO = \{Y\}, FO = \{X\}, \\ \alpha = \uparrow, \ \beta = s, \end{cases}$$

$$x_3 \ : \ \begin{cases} M = \{Y \to \varepsilon(-)\}, \\ PI = \{Y\}, FI = \{X\} \cup \{a' \mid a \in V\}, \\ PO = \emptyset, FO = \{X, Y\}, \\ \alpha = \downarrow, \ \beta = s, \end{cases}$$

$$x_4 \ : \ \begin{cases} M = \{a \to a'(-) \mid a \in V\}, \\ PI = \{a' \mid a \in V\}, FI = \{X, Y\}, \\ PO = \{a' \mid a \in V\}, FO = \emptyset, \\ \alpha = \uparrow, \ \beta = w. \end{cases}$$

$$x_5 \ : \ \begin{cases} M = \{a \to a'(-) \mid a \in V\}, \\ PI = \{a' \mid a \in V\}, FI = \{X, Y\}, \\ PO = \{a' \mid a \in V\}, FO = \emptyset, \\ \alpha = \downarrow, \ \beta = w. \end{cases}$$

The working mode of this network is rather simple. In the input node the first row of the picture is marked either for deletion (if a symbol of the first row was replaced by X) or for the checking phase. If the first row was marked for deletion, the picture goes to the node x_1 where the last row is marked for deletion. Then these two rows are deleted in the nodes x_2 and x_3, and the process resumes in the input node x_I. Let us now see what happens with a picture marked for the checking phase in the input node. This picture enters nodes x_4 and x_5 which exchange with each other this picture until all symbols

on the first and last row are replaced by the primed copies of the original symbols. Now, this picture is sent to the input node of the subnetwork from Example 1 suitably modified. As this node cannot accept pictures containing other symbols than primed ones, it follows that the pictures able to enter this node have exactly two rows. This concludes the proof. □

We do not know whether the inclusion $\mathcal{L}(REC) \subset \mathcal{L}_{wa}(ANEPP)$ holds, however a large part of $\mathcal{L}(REC)$ is included in $\mathcal{L}_{wa}(ANEPP)$ as the next result states. We recall that the complement of any local language is recognizable [7].

Theorem 4. *The complement of every local language can be weakly accepted by an ANEPP.*

Proof. We first give an informal argument such that the formal proof can be understood easily. The argument starts with the observation that one can construct a network that weakly accepts only a fixed picture of size $(2, 2)$. Now, if L is a local language over the alphabet V defined by the set F of $(2, 2)$-tiles, then we consider the set F^c of all $(2, 2)$-tiles over V that do not belong to F. The rough idea of the network weakly accepting the complement of L is the following one. First, one constructs a set of completely disjoint networks each one accepting exactly one picture from F^c. Then another network cuts an arbitrary $(2, 2)$-tile from the input picture and sends it to all these networks suitably modified.

We now give the formal proof. Assume that F^c has the tiles $t_1, t_2, \ldots t_n$ for some $n \geq 1$. We first define a network Γ_i that accepts exactly the singleton picture language $\{t_i\}$ for some $1 \leq i \leq n$; for sake of simplicity we assume that $t_i = \begin{matrix} a & b \\ c & d \end{matrix}$. The nodes of this network are described below; x_I and x_O are the input and output node, respectively.

$$x_I : \begin{cases} M = \{a \to a_i(|)\}, \\ PI = \{a, b, c, d\}, \\ FI = \{a_i, b_i, c_i, d_i\}, \\ PO = \{a_i\}, FO = \emptyset, \\ \alpha = \leftarrow,\ \beta = w, \end{cases} \qquad x_1 : \begin{cases} M = \{c \to c_i(-)\}, \\ PI = \{a_i\}, FI = \{b_i, c_i, d_i\}, \\ PO = \{c_i\}, FO = \emptyset, \\ \alpha = \downarrow,\ \beta = s, \end{cases}$$

$$x_2 : \begin{cases} M = \{b \to b_i(-)\}, \\ PI = \{a_i, c_i\}, FI = \{b_i, d_i\}, \\ PO = \{b_i\}, FO = \emptyset, \\ \alpha = \uparrow,\ \beta = s, \end{cases} \qquad x_3 : \begin{cases} M = \{d \to d_i(|)\}, \\ PI = \{a_i, b_i, c_i\}, FI = \{d_i\}, \\ PO = \{d_i\}, FO = \emptyset, \\ \alpha = \rightarrow,\ \beta = s, \end{cases}$$

$$x_4 : \begin{cases} M = \{a_i \to \varepsilon(|)\}, \\ PI = \{a_i, b_i, c_i, d_i\}, FI = V, \\ PO = \{b_i, d_i\}, FO = \{a_i, c_i\}, \\ \alpha = \leftarrow,\ \beta = s. \end{cases} \qquad x_O : \begin{cases} M = \emptyset, \\ PI = \{b_i, d_i\}, FI = V, \\ PO = \{b_i, d_i\}, FO = \emptyset, \\ \alpha = *,\ \beta = s, \end{cases}$$

One can rather easily see that only pictures of size $(2, 2)$ might be eventually accepted. We modify the filters of each network Γ_i, $1 \leq i \leq n$, such that as soon as a picture enters a node of some network Γ_i, it is processed only in Γ_i until the computation halts. The network weakly accepting the complement of L contains all networks Γ_i, $1 \leq i \leq n$, modified as above, as subnetworks and has the set of output nodes formed by the output nodes of all Γ_i, $1 \leq i \leq n$. It has four further nodes, two for deleting rows and two for deleting columns, in the aim of preparing input pictures for the subnetworks Γ_i, $1 \leq i \leq n$. □

5. Solving Picture Matching With ANEPPs

As one can see in the proof of the previous theorem, it is possible to construct an ANEPP that weakly accepts the singleton language formed by a given picture of size $(2, 2)$. A natural problem is to find a pattern (a given picture) in a given picture. This problem is widely known as the two-dimensional pattern matching problem. It is largely motivated by different aspects in low-level image processing [17]. The more general problem of picture matching (it is not obligatory for the picture to be a two-dimensional array) is widely known in Pattern Recognition field and is connected with Image Analysis and Artificial Vision [11, 28].

For the rest of this paper we consider only ANEPPs that halt on every input. The previous theorem shows that the two-dimensional pattern

matching problem can be solved by ANEPPs with weak acceptance (or the problem is weakly decided by ANEPP) provided that the pattern is of size $(2,2)$. Can this result be extended to patterns of arbitrary size? We show that it can be extended to patterns of size (i,n) and (n,i) for any $1 \leq i \leq 3$ and $n \geq 1$. However, the general problem of pattern matching in a given picture remains open. In order to prove that the pattern matching in a picture with patterns of size (i,n) (or (n,i)) for all $1 \leq i \leq 3$ and $n \geq 1$ can be weakly decided by ANEPPs, it suffices to construct a network able to weakly accept the singleton language formed by a given picture of size (i,n) (or (n,i)) for all $1 \leq i \leq 3$ and $n \geq 1$. Indeed, if such a network, say Γ, can be constructed, then given an arbitrary picture one can construct a network that extract all its subpictures which are given as inputs to the subnetwork Γ suitably modified. If at least one of these subpictures matches the pattern, then the input picture is accepted.

Theorem 5. *Let π be a picture of size (i,n) for some $1 \leq i \leq 3$ and $n \geq 1$. The language $\{\pi\}$ can be weakly accepted by an ANEPP.*

Proof. Actually, we only prove the most difficult case, namely $i = 3$, the proofs of the other cases that can be easily deduced from this one are left to the reader.

Let V be the alphabet of π; the nodes of this network are described below, where x_I and x_O are the input and output node, respectively. Note that $U = V \cup \{a^{(i)}, a(i), a_{(i)} \mid a \in V, 1 \leq i \leq n\}$.

$$x_I \ : \begin{cases} M = \{\pi(1,1) \to \pi(1,1)^{(1)}(-)\}, \\ PI = V, FI = U \setminus V, \\ PO = \{\pi(1,1)^{(1)}\}, FO = \emptyset, \\ \alpha = \uparrow,\ \beta = w, \end{cases}$$

$$x^{(i)} \ : \begin{cases} M = \{\pi(1,i) \to \pi(1,i)^{(i)}(-)\}, \\ PI = \{\pi(1,i-1)^{(i-1)}\}, FI = U \setminus (V \cup \{\pi(1,i-1)^{(i-1)}\}), \\ PO = \{\pi(1,i)^{(i)}\}, FO = \emptyset, \\ \alpha = \uparrow,\ \beta = s, \qquad 2 \leq i \leq n \end{cases}$$

$$x_{(i)} \ : \begin{cases} M = \{\pi(3,i) \to \pi(3,i)_{(i)}(-)\}, \\ PI = \{\pi(1,i)^{(i)}, \pi(1,i+1)^{(i+1)}\}, \\ FI = U \setminus (V \cup \{\pi(1,i)^{(i)}, \pi(1,i+1)^{(i+1)}\}), \\ PO = \{\pi(3,i)_{(i)}\}, FO = \emptyset, \\ \alpha = \downarrow,\ \beta = s, \qquad 1 \leq i \leq n-1 \end{cases}$$

$$
x(i) \;:\; \begin{cases} M = \{\pi(2,i) \to \pi(2,i)(i)(|)\}, \\ PI = \{\pi(1,i)^{(i)}, \pi(3,i)_{(i)}\}, FI = \emptyset, \\ PO = \{\pi(2,i)(i)\}, FO = \emptyset, \\ \alpha = \leftarrow,\ \beta = s, \end{cases} \qquad 1 \le i \le n
$$

$$
x_{del}^{(i)} \;:\; \begin{cases} M = \{\pi(1,i)^{(i)} \to \varepsilon(|)\}, \\ PI = \{\pi(2,i)(i)\}, FI = \emptyset, \\ PO = \emptyset, FO = \{a^{(i)}, a_{(i)}, a(i) \mid a \in V\}, \\ \alpha = \leftarrow,\ \beta = s, \end{cases} \qquad 1 \le i \le n-1
$$

$$
x_{(n)} \;:\; \begin{cases} M = \{\pi(3,n) \to \pi(3,n)_{(n)}(-)\}, \\ PI = \{\pi(1,n)^{(n)}\}, FI = U \setminus (V \cup \{\pi(1,n)^{(n)}\}), \\ PO = \{\pi(3,n)_{(n)}\}, FO = \emptyset, \\ \alpha = \downarrow,\ \beta = s, \end{cases}
$$

$$
x_O \;:\; \begin{cases} M = \emptyset, \\ PI = \{\pi(1,n)^{(n)}, \pi(2,n)(n), \pi(3,n)_{(n)}\}, \\ FI = U \setminus \{\pi(1,n)^{(n)}, \pi(2,n)(n), \pi(3,n)_{(n)}\}, \\ PO = \emptyset, FO = \emptyset, \\ \alpha = *,\ \beta = s, \end{cases}
$$

We analyze the computation of this network on an input picture μ of size (k,m) for some $k,m \ge 1$. We first consider the case $k = 3$ and $m = n$. In x_I an occurrence of $\pi(1,1)$ on the first row of μ is replaced by $\pi(1,1)^{(1)}$ and all pictures are sent out. These pictures can be received only by either $x^{(2)}$, provided that $n \ge 2$, or $x_{(n)}$, provided that $n = 1$. We assume that $n \ge 2$ and continue the computation in $x^{(2)}$. Here an occurrence of $\pi(1,2)$ on the first row of all pictures is replaced by $\pi(1,2)^{(2)}$. All pictures having replaced an occurrence of $\pi(1,2)$ by $\pi(1,2)^{(2)}$ can leave $x^{(2)}$ and enter $x_{(1)}$ where an occurrence of $\pi(3,1)$ on the last row is replaced by $\pi(3,1)_{(1)}$. Now all pictures arrive in $x(1)$ where an occurrence of $\pi(2,1)$ on the leftmost column is replaced by $\pi(2,1)(1)$. Note that if a picture does not have an occurrence of the symbol that is to be replaced in any of the nodes $x^{(2)}$, $x_{(1)}$, and $x(1)$, then it remains forever in that node.

Pictures going out from $x(1)$ can enter $x_{del}^{(1)}$ only, where the leftmost column is deleted provided that $\pi(1,1)^{(1)}$ is situated on that column. The second condition to continue the computation is that $\pi(3,1)_{(1)}$ is also situated on the column which is to be deleted in $x_{del}^{(1)}$. Therefore, the first column of μ

must be $\begin{matrix}\pi(1,1)\\ \ldots\ldots\\ \pi(2,1)\\ \ldots\ldots\\ \pi(3,1)\end{matrix}$. Now the process described above resumes for all pictures going out from $x_{del}^{(1)}$, as all these pictures contain $\pi(1,2)^{(2)}$ on their first row. Inductively, for every $1 \leq i \leq n-2$ every picture that has just gone out from $x_{del}^{(i)}$ must contain $\pi(1,i+1)^{(i+1)}$ on its first row. Further on, it must follow the following itinerary through the network: $x^{(i+2)}$, $x_{(i+1)}$, $x(i+1)$, $x_{del}^{(i+1)}$.

We now analyze the case when the symbol on the first row of a picture going out from $x_{del}^{(n-1)}$ is $\pi(1,n)^{(n)}$. This picture enters $x_{(n)}$ only, where an occurrence of $\pi(3,n)$ is replaced by $\pi(3,n)_{(n)}$ and then enters $x(n)$ where an occurrence of $\pi(2,n)$ is replaced by $\pi(2,n)(n)$. Now, if the picture is $\begin{matrix}\pi(1,n)^{(n)}\\ \pi(2,n)(n)\\ \pi(3,n)_{(n)}\end{matrix}$, then it enters x_O, otherwise it is lost.

By these explanations we infer the followings:

- If μ is of size $(3,n)$, then it is accepted if and only if $\mu = \pi$.
- If $m < n$, then the computation on μ will be eventually blocked after at most $m-1$ column deletions.
- If $m > n$, then the computation on μ will be eventually blocked after at most $n-1$ column deletions.
- If $k < 3$, then the computation on μ is blocked after the first column deletion.
- If $k > 3$, then the computation on μ will be eventually blocked after at most $n-1$ column deletions.

In conclusion, the network accepts only one picture, namely π. □

This result, together with the fact that the class $\mathcal{L}_{wa}(ANEPP)$ is closed under boolean union (see Theorem 2), allows the following statement.

Theorem 6. *Given a finite set F of patterns of size (i,n) and (n,i) for all $1 \leq i \leq 3$ and $n \geq 1$, the pattern matching problem with patterns from F can be weakly decided by ANEPPs.*

Various algorithms exist for the exact two-dimensional matching problem. The fastest algorithms for finding a rectangular picture pattern in a given picture of size (n, m) run in $\mathcal{O}(n \times m)$ time, see, e.g., [1, 29]. It is rather easy to note that an ANEPP which weakly decides whether a pattern of size (i, p), $1 \leq i \leq 3$, appears in a given picture of size (n, m) does this in $\mathcal{O}(n + m)$ computational (evolutionary and communication) steps. On the other hand, the space complexity of the algorithm proposed in [29] is $\mathcal{O}(n \times m)$, while the number of pictures moving through the network is pretty large. We recall that some biological phenomena are sources of inspiration for our model. In this context, it is considered to be biologically feasible to have sufficiently many identical copies of a molecule. By techniques of genetic engineering, in a polynomial number of lab operations one can get an exponential number of identical molecules.

As we have seen, the general problem of weakly deciding whether a given pattern appears in a picture remained open. Could the problem be strongly decided? Unfortunately, we do not have a complete answer for this case either. However, similarly to the previous case we can state:

Theorem 7. *Let π be a picture of size (n, m) for some $n, m \geq 1$. The language $\{\pi\}$ can be strongly accepted by an ANEPP.*

Proof. By Theorem 5, there exists an ANEPP Γ that weakly accepts exactly $\{\mu\}$ for a given picture μ of size $(1, m)$. Moreover, Γ has one output node only. Let Γ_i, $1 \leq i \leq n$, be the ANEPP with one output node which weakly accepts the language $\{\pi_i\}$, where π_i is the row picture formed by the i^{th} row of π.

The idea of the proof is to extract all rows π_i, for $1 \leq i \leq n$, from π and give them as inputs to the corresponding subnetworks Γ_i suitably modified. All these subnetworks work with mutually disjoint alphabets. To this aim, we need to design a subnetwork that extracts the picture π_i, for some $1 \leq i \leq n$, and transforms it such that only Γ_i suitably modified can compute on it.

The input for all the subnetworks extracting pictures π_i is provided by another subnetwork that transforms the first column of π from $\begin{matrix} a_1 \\ \dots \\ a_2 \\ \dots \\ a_n \end{matrix}$ into

$a_1(\sigma(1))$
$\ldots\ldots$
$a_2(\sigma(2))$ for some permutation $\sigma : \{1, 2, \ldots, n\} \longrightarrow \{1, 2, \ldots, n\}$. Note
$\ldots\ldots$
$a_n(\sigma(n))$
that this subnetwork cannot provide inputs for any Γ_i suitably modified, if the input picture π has less than n rows.

The subnetwork that extracts and prepares the input for Γ_i suitably modified is defined below. As one can easily see, only the input picture with exactly n rows and σ being the identical permutation will eventually reach the output node and then enter the input node of Γ_i. The input node of this network is denoted x_I while x_O denotes the output node. Here U denotes the alphabet

$$U = V \cup \{a^{(i)}, a(r)^{(i)}, a(r) \mid a \in V, 1 \leq i \leq n, 1 \leq r \leq n\}.$$

$$x_I \ : \begin{cases} M = \{a(r) \rightarrow a(r)^{(i)}(|) \mid a \in V, 1 \leq r \leq n\}, \\ PI = \{a(n) \mid a \in V\}, FI = U \setminus (V \cup \{a(t) \mid a \in V, 1 \leq t \leq n\}), \\ PO = U, FO = \{a(t) \mid a \in V, 1 \leq t \leq n\}, \\ \alpha = \leftarrow,\ \beta = w, \end{cases}$$

$$\overline{x}_j \ : \begin{cases} M = \{\pi(j,1)(j)^{(i)} \rightarrow \varepsilon(-)\}, \\ PI = \{\pi(j,1)(j)^{(i)}, \pi(j+1,1)(j+1)^{(i)}, \ldots, \pi(n,1)(n)^{(i)}\}, \\ FI = U \setminus (V \cup PI), \\ PO = \{\pi(j+1,1)(j+1)^{(i)}, \ldots, \pi(n,1)(n)^{(i)}\}, \\ FO = U \setminus (V \cup PO), \\ \alpha = \uparrow,\ \beta = s, \qquad 1 \leq j \leq i-1 \end{cases}$$

$$\underline{x}_k \ : \begin{cases} M = \{\pi(k,1)(k)^{(i)} \rightarrow \varepsilon(-)\}, \\ PI = \{\pi(i,1)(i)^{(i)}, \pi(i+1,1)(i+1)^{(i)}, \ldots, \pi(k,1)(k)^{(i)}\}, \\ FI = U \setminus (V \cup PI), \\ PO = \{\pi(i,1)(i)^{(i)}, \pi(i+1,1)(i+1)^{(i)}, \ldots, \pi(k-1,1)(k-1)^{(i)}\}, \\ FO = U \setminus (V \cup PO), \\ \alpha = \downarrow,\ \beta = s, \qquad i+1 \leq k \leq n \end{cases}$$

$$x_O \ : \begin{cases} M = \{\pi(i,1)(i)^{(i)} \rightarrow \pi(i,1)^{(i)}(|)\} \cup \{a \rightarrow a^{(i)}(|) \mid a \in V\}, \\ PI = \{\pi(i,1)(i)^{(i)}\}, FI = U \setminus (V \cup PI), \\ PO = \emptyset, FO = U \setminus \{a^{(i)} \mid a \in V\}, \\ \alpha = *,\ \beta = s, \end{cases}$$

Indeed, this subnetwork extracts exactly the subpicture formed by the i^{th} row of the input picture and changes every symbol a of this row picture into $a^{(i)}$. This row picture is ready now to be computed by Γ_i suitably modified. The set of output nodes of the whole network is formed by the output node of each Γ_i, $1 \leq i \leq n$, suitably modified. □

Unfortunately, this result cannot be further extended to a result similar to Theorem 6 by an argument analogous to that used in that theorem. Indeed, an analogous argument might lead to the fact that two subnetworks, say Γ_i and Γ_j, have their output nodes nonempty by computations on two row pictures that do not come from the same picture.

6. Further Work

We finish this work with a very short discussion on some problems left open here.

1. The first natural problem regards the equality of the classes $\mathcal{L}_{wa}(ANEPP)$ and $\mathcal{L}_{sa}(ANEPP)$. Another attractive problem, in our view, concerns the relationships between these two classes and the classes $\mathcal{L}(LOC)$ and $\mathcal{L}(REC)$.
2. Can the pattern matching problem with arbitrary pattern be weakly or strongly decided by ANEPPs?
3. It is rather plain to see that the membership problem is decidable for both classes $\mathcal{L}_{wa}(ANEPP)$ and $\mathcal{L}_{sa}(ANEPP)$. What other problems are also decidable?
4. Another important direction of research concerns the closure properties of the two classes $\mathcal{L}_{wa}(ANEPP)$ and $\mathcal{L}_{sa}(ANEPP)$. Theorem 2 presented in the third section is just a preliminary result in this area. Is any of the two classes closed under row and column concatenation?
5. A natural question is whether adding the row/column insertion operation to the operations considered so far would augment the power of the networks. As we do not have anymore the restriction on the space of the recognition process this might be the case. For example, these networks might be able to simulate array contextual grammars as defined in [25].

We believe that the problems mentioned above together with other properties of these classes deserve further investigation.

References

[1] A. Amir, G. Benson, M. Farach, Alphabet independent two dimensional matching, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, 1992, 59–68.

[2] S. Bozapalidis, A. Grammatikopoulou, Recognizable picture series, *J. of Automata, Languages and Combinatorics*, 10, 2-3(2005), 159–183.

[3] S. Bozapalidis, Picture deformation, *Acta Informatica*, 45, 1(2008), 1–31.

[4] E. Csuhaj-Varjú, C. Martín-Vide, V. Mitrana, Hybrid NEPs are computationally complete, *Acta Informatica*, 41, 4-5(2005), 257–272.

[5] K.S. Dersanambika, K.G. Subramanian, A. Roslin Sagaya Mary, 2D and 3D pictural networks of evolutionary processors, *Proc. 1st International Work-Conference on the Interplay between Natural and Artificial Computation* LNCS 3562 (2005), 92–101.

[6] D. Giammarresi, A. Restivo, Two-dimensional languages. In [19], 215–267.

[7] D. Giammarresi, A. Restivo, Recognizable picture languages, *Int. J. Pattern Recognition and Artificial Intelligence*, 6 (1992), 241–256.

[8] D. Giammaresi, A. Restivo, S. Seibert, W. Thomas, Monadic second-order logic over rectangular pictures and recognizability by tiling systems, *Infor. Comput.*, 125, 1(1996), 32–45.

[9] I. Inoue, I. Takanami, A survey of two-dimensional automata theory, *Proc. 5th Int. Meeting of Young Computer Scientists*, LNCS 381 (1990), 72-91.

[10] M. Margenstern, V. Mitrana, M. Perez-Jimenez, Accepting hybrid networks of evolutionary systems, *DNA Based Computers 10* LNCS 3384, Springer-Verlag, Berlin, 235–246, 2005.

[11] K. Marriott, B. E. Meyer, *Visual Language Theory*, Springer, 1998.

[12] C. Martín-Vide, V. Mitrana, Networks of evolutionary processors: results and perspectives, *Molecular Computational Models: Unconventional Approaches*, Idea Group Publishing, Hershey, 2005, 78-114.

[13] I. Maürer, Characterizations of recognizable picture series, *Theoretical Computer Science* 374 (2007), 214–228.

[14] I. Maürer, Weighted picture automata and weighted logics, *STACS 2006*, LNCS 3884, Springer Berlin, 2006, 313–324.

[15] V. Mitrana, K.G. Subramanian, M. Tătărâm, Pictural networks of evolutionary processors, *Romanian Journal of Information Science and Technology* 6(2003), 189–198.

[16] M. Nivat, A. Saoudi, K.G. Subramanian, R. Siromoney, V.R. Dare, Puzzle grammars and context-free array grammars, *Int. J. Pattern Recognition and Artificial Inteligence*, 5 (1991), 663–676.

[17] A. Rosenfeld, A.C. Kak, *Digital Picture Processing*, Academic Press, New York, 1982.

[18] A. Rosenfeld, R. Siromoney, Picture languages – a survey, *Languages of design*, 1 (1993), 229–245.

[19] G. Rozenberg, A. Salomaa (eds.), *Handbook of Formal Languages*, Springer–Verlag, Berlin, 1997.

[20] D. Sankoff et al., Gene order comparisons for phylogenetic inference:evolution of the mitochondrial genome, *Proceedings of the National Academy of Sciences of the United States of America* **89**, (1992), 6575–6579.

[21] G. Siromoney, R. Siromoney, K. Krithivasan, Abstract families of matrices and picture languages, *Computer Graphics and Image Processing*, 1 (1972), 284–307.

[22] G. Siromoney, R. Siromoney, K. Krithivasan, Picture languages with array rewriting rules, *Information and Control*, 22 (1973), 447–470.

[23] K.G. Subramanian, R. Siromoney, V.R. Dare, A. Saoudi, Basic puzzle languages, *Int. J. Pattern Recognition and Artificial Intelligence*, 9 (1995), 763–775.

[24] K.G. Subramanian, R. Siromoney, On array grammars and languages, *Cybernetics and Systems*, 18 (1987), 77–98.

[25] K.G. Subramanian, Do Long Van, P. Helen Chandra, Nghiem Do Quyen, Array grammars with contextual operations, *Fundamenta Informaticae*, 83, 4(2008), 411–428.

[26] P.S. Wang, Hierarchical structure and complexities of parallel isometric patterns, *IEEE Trans. PAM I*, 5 (1975), 92.

[27] P.S. Wang, Sequential/parallel matrix array languages, *Journal of Cybernetics*, 5 (1975), 19–36.

[28] P.S. Wang, H. Bunke (Eds.), *Handbook on Optical Character Recognition and Document Image Analysis*, World Scientific, 1996.

[29] R.F. Zhu, T. Takaoka, A technique for two-dimensional pattern matching, *Communications of the ACM*, 32, 9(1989), 1110–1120.