Accepting Networks of Evolutionary Picture Processors

Paolo Bottoni

Anna Labella

Victor Mitrana

Abstract. We extend the study of networks of evolutionary processors accepting words to a similar model, processing rectangular pictures. To this aim, we introduce accepting networks of evolutionary picture processors and investigate their computational power. We show that these networks can accept the complement of any local picture language as well as picture languages that are not recognizable. Some open problems regarding decidability issues and closure properties are finally discussed.

Keywords: Rectangular picture, local picture language, recognizable picture language, evolutionary picture processor, network of evolutionary picture processors.

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 [11, 12, 16, 14]. On the other hand, models defining pictures that are connected arrays but not necessarily rectangular have been proposed as early as in the 70's in [8] and a hierarchy of these grammars was considered in [15, 13]. A related class of grammars for picture generation, again not necessarily rectangular, has been proposed in [7]. 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 [3]. An early survey on automata recognizing rectangular pictures languages is [4], a more recent one considering different mechanisms defining picture languages, not necessarily rectangular, is [8] and an even more recent and concise one is [2].

This work tries to carry over to rectangular pictures the investigation started in [1] and [5] and continued in a series of papers; the reader may consult the early survey [6]. In these papers a mechanism inspired from cell biology was considered, namely accepting 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 they are connected to.

In this paper, we consider networks of evolutionary picture processors acting as acceptors on rectangular pictures. 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, according to the action of the rules associated with that node. Moreover, if there are more occurrences of the letter that is to be substituted in the row on which the substitution rule acts, then an instance of the picture for each such occurrence will be produced such that a different occurrence of the letter is substituted by the rule in every copy of the given picture. An implicit assumption is that arbitrarily many copies of every picture are available. A similar informal explanation concerns column substitution and deletion nodes.

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 [10]. 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 accepting networks of picture processors; then we discuss a

brief comparison of its expressive power with respect to two well-known classes of rectangular picture languages, namely, local and recognizable picture languages [3].

2. Preliminaries

The basic terminology and notations concerning two-dimensional languages are taken from [2].

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_*^*; 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 set of all pictures over V with m rows is denoted by V_m^* while the set of all pictures over V with n columns is denoted by V_*^n. Consequently, 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 [4] or [2]. 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 adjoining the picture ρ under 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 partial operations:

- 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: $\pi/_{\downarrow}\rho = \theta$ iff $\pi = \theta Ⓡ \rho$.
- The row up-quotient of π with ρ: $\pi/_{\uparrow}\rho = \theta$ iff $\pi = \rho Ⓡ \theta$.

Let V be an alphabet; a rule of the form $a \rightarrow b$, with $a, b \in V \cup \{\varepsilon\}$ is called an *evolutionary rule*. We say that a rule $a \rightarrow b$ is: a) a *substitution rule* if neither of a and b is ε; b) a *deletion rule* if $a \neq \varepsilon$, $b = \varepsilon$; c) 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. Let $Sub_V = \{a \rightarrow b \mid a, b \in V\}$ and $Del_V = \{a \rightarrow \varepsilon \mid a \in V\}$. Given a rule σ as above and a picture $\pi \in V_m^n$, we define the following *actions* of σ on π:

- If $\sigma \equiv a \rightarrow b \in Sub_V$, then

  – If the first column of π contains an occurrence of a, then $\sigma^{\leftarrow}(\pi)$ is the set of all pictures π' such that the following conditions are satisfied:

  (i) there exists $1 \leq i \leq m$ such that $\pi(i, 1) = a$ and $\pi'(i, 1) = b$,
  (ii) $\pi'(j, l) = \pi(j, l)$ for all $(j, l) \in ([m] \times [n]) \setminus \{(i, 1)\}$.

– If this column does not contain any occurrence of a, then $\sigma^{\leftarrow}(\pi) = \{\pi\}$.

Informally, $\sigma^{\leftarrow}(\pi)$ is the set of all pictures that can be obtained from π by replacing an occurrence of a by b in the first (leftmost) column of π. Note that σ is applied to all occurrences of the letter a in the leftmost column of π in different copies of the picture π.

In an analogous way, we define $\sigma^{\rightarrow}(\pi)$, $\sigma^{\uparrow}(\pi)$, $\sigma^{\downarrow}(\pi)$, $\sigma^{+}(\pi)$, as the set of all pictures obtained by applying σ to the rightmost column, to the first row, to the last row, and to any column/row of π.

- If $\sigma \equiv a \rightarrow \varepsilon \in Del_V$, then

  – $\sigma^{\leftarrow}(\pi)$ is the picture obtained from π by deleting the leftmost column of π, provided that this column contains at least one occurrence of a. More formally,

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

  Analogously, $\sigma^{\rightarrow}(\pi)$, $\sigma^{\uparrow}(\pi)$, and $\sigma^{\downarrow}(\pi)$ is the picture obtained from π by applying σ to the rightmost column, to the first row, and to the last row of π, respectively. Furthermore,

  – $\sigma^{|}(\pi)$ ($\sigma^{-}(\pi)$) is the set of pictures obtained from π by deleting an arbitrary column (row) containing an occurrence of a from π. If more than one column (row) of π contains a, then each such column (row) is removed from different copies of π. If π does not contain any occurrence of a, then $\sigma^{|}(\pi) = \sigma^{-}(\pi) = \{\pi\}$. More formally,

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

For every rule σ, symbol $\alpha \in \{\leftarrow, \rightarrow, \uparrow, \downarrow, |, -, +\}$, and $L \subseteq V_*^*$, we define the *α-action of σ on L* by $\sigma^{\alpha}(L) = \bigcup_{\pi \in L} \sigma^{\alpha}(\pi)$. Note that $+$ is defined only for substitution rules, while $|$ and $-$ are defined only for deletion rules. Given a finite set of rules M, we define the *α-action of M* on the picture π and the language L by:

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

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 consider the following two predicates which we will later use to define two types of filters:

$$rc_s(\pi; P, F) \equiv (P \subseteq alph(\pi)) \wedge (F \cap alph(\pi) = \emptyset)$$

$$rc_w(\pi; P, F) \equiv (alph(\pi) \cap P \neq \emptyset) \wedge (F \cap alph(\pi) = \emptyset).$$

The construction of these predicates is based on *random context* conditions defined by the two sets P (*permitting contexts/symbols*) and F (*forbidding contexts/symbols*). Informally, both conditions require 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, provided that P is not empty.

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 $M \subseteq Sub_V$ or $M \subseteq Del_V$. 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.

An *accepting network of evolutionary picture processors* (ANEPP for short) is a 8-tuple

$$\Gamma = (V, U, G, \mathcal{N}, \alpha, \beta, x_I, Out),$$

where:

- V and U are the input and network alphabet, respectively, with $V \subseteq U$.
- $G = (X_G, E_G)$ is an undirected graph without loops with set of vertices X_G and set of edges E_G. G is called the *underlying graph* of the network. Although in network theory, several types of graphs are common like *complete, rings, stars, grids*, we focus here on complete underlying graphs, so that we can implicitly define the graph G by specifying the set of its nodes.
- $\mathcal{N}$ is a mapping which associates with each node $x \in X_G$ the picture processor $\mathcal{N}(x) = (M_x, PI_x, FI_x, PO_x, FO_x)$.
- $\alpha : X_G \longrightarrow \{\leftarrow, \rightarrow, \uparrow, \downarrow, |, -, +\}$; $\alpha(x)$ 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:

  $$\text{input filter: } \mu_x(\cdot) = rc_{\beta(x)}(\cdot; PI_x, FI_x),$$
  $$\text{output filter: } \tau_x(\cdot) = rc_{\beta(x)}(\cdot; PO_x, FO_x).$$

  That is, $\mu_x(\pi)$ (resp. $\tau_x(\pi)$) indicates whether or not the picture π can pass the input (resp. output) filter of x. More generally, $\mu_x(L)$ (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 $C : X_G \longrightarrow 2_f^{U_*^*}$ 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 $x \in X_G \setminus \{x_I\}$.

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 for all $x \in X_G$, the following holds:

$$C'(x) = (C(x) \setminus \tau_x(C(x))) \cup \bigcup_{\{x,y\} \in E_G} (\tau_y(C(y)) \cap \mu_x(C(y))).$$

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, then 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 as above *weakly (strongly) accepts* π, if there exists a configuration in which the set of pictures existing in at least one output node (all output nodes) is non-empty. The *picture language weakly (strongly) accepted* by Γ, denoted by $L_{wa}(\Gamma)$ ($L_{sa}(\Gamma)$), is the set of all input pictures π such that the computation of Γ on π weakly (strongly) accepts π.

The following two notions will be very useful in the sequel. We recall that all considered ANEPPs have a complete underlying graph, hence we may replace the graph G by its set of vertices denoted by χ. If h is a one-to-one mapping from U to W and $\Gamma = (V, U, \chi, \mathcal{N}, \alpha, \beta, x_I, Out)$ is an ANEPP, then we denote by Γ_h the ANEPP $\Gamma_h = (h(V), h(U), \chi, h(\mathcal{N}), \alpha, \beta, x_I, Out)$, where by $h(\mathcal{N})$ we mean $h(\mathcal{N})(x) = (h(M_x), h(PI_x), h(FI_x), h(PO_x), h(FO_x))$ for every $x \in \chi$, provided that $\mathcal{N}(x) = (M_x, PI_x, FI_x, PO_x, FO_x)$. Further, $h(a \to b) = h(a) \to h(b)$ for any evolutionary rule $a \to b$. Now, given two ANEPPs $\Gamma_i = (V_i, U_i, \chi_i, \mathcal{N}_i, \alpha_i, \beta_i, x_I^i, Out_i)$, $i = 1, 2$, $\chi_1 \cap \chi_2 = \emptyset$, we denote by $\Gamma_1 \sqcup \Gamma_2 = (V_1, U_1 \cup U_2, \chi_1 \cup \chi_2, \mathcal{N}, \alpha, \beta, x_I^1, Out_2)$, where $\circ\,|_{\chi_i} = \circ_i$ for all $\circ \in \{\mathcal{N}, \alpha, \beta\}$ and $i = 1, 2$.

3. 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 [3]. Before starting this investigation, we establish a 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 an arbitrary non-empty subset of P and the empty set, respectively. Further on, the output sets of permitting and forbidding symbols of every such node are $\{Z\}$ 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 those 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 $X \to Z$, where X is an arbitrary symbol from the input set of permitting symbols of that node. The action mode of all these rules is $+$. □

We start with one simple example which lays the basis for further results.

Example 1. Let L be the set of all pictures $\pi \in V_2^*$ consisting of 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 complete ANEPP given in Table 1, 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:

Node	M	PI	PO	α
		FI	FO	β
x_I	$\emptyset$	V	V	$\uparrow$
		$U \setminus V$	$\emptyset$	w
x_a	$\{a \to X_a\}$	V	$\{X_a\}$	$\uparrow$
		$U \setminus V$	$\emptyset$	w
x'_a	$\{a \to Y_a\}$	$\{X_a\}$	$\{Y_a\}$	$\downarrow$
		$\{Y_b \mid b \in V\}$	$\emptyset$	s
x''_a	$\{X_a \to X, Y_a \to Y\}$	$\{X_a, Y_a\}$	$\{X, Y\}$	$\rightarrow$
		$U \setminus \{X_a, Y_a\}$	$\emptyset$	s
x_{del}	$\{X_a \to \varepsilon \mid a \in V\}$	$\{Y_a \mid a \in V\}$	V	$\leftarrow$
		$\emptyset$	$U \setminus V$	w
x_O	$\emptyset$	$\{X, Y\}$	$\emptyset$	$+$
		$U \setminus \{X, Y\}$	U	s

Table 1.

Let us follow a computation of this network on a generic 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 in 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 single 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 number of rows other 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 accept the language of all pictures (of any size) having two identical rows. The role of the next example is to show how two 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

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

In what follows we assume that the same alphabet V is used in Examples 1 and 2. First, we construct the ANEPP $\Gamma_1 = (V, U_1, \chi_1, \mathcal{N}_1, \alpha_1, \beta_1, y_I, \{\bar{y}'_a \mid a \in V\})$ of size $4 \cdot card(V) + 2$ with the working alphabet $U_1 = V \cup \{a', a'', \bar{a} \mid a \in V\}$, and the nodes defined as shown in Table 2:

Node	M	PI	PO	α_1
		FI	FO	β_1
y_I	$\{a \to a' \mid a \in V\}$	$\emptyset$	$\{a' \mid a \in V\}$	$\leftarrow$
		U_1	$\emptyset$	w
y'	$\{a \to a'' \mid a \in V\}$	$\{a' \mid a \in V\}$	$\{a'' \mid a \in V\}$	$\leftarrow$
		$\{a'', \bar{a} \mid a \in V\}$	$\emptyset$	w
y_a $a \in V$	$\{b \to \varepsilon \mid b \in V\}$	$\{a', a''\}$	$\emptyset$	$-$
		$U_1 \setminus (V \cup \{a', a''\})$	$\emptyset$	s
y'_a $a \in V$	$\{b \to \varepsilon \mid b \in V\}$	$\{a', a''\}$	$\emptyset$	$-$
		$U \setminus \{X_a, Y_a\}$	$\emptyset$	s
$\bar{y}_a$ $a \in V$	$\{a' \to \bar{a}\} \cup \{b \to \bar{b} \mid b \in V\}$	$\{a', a''\}$	$\emptyset$	$\uparrow$
		$U_1 \setminus (V \cup \{a', a''\})$	$\{a'\}$	s
$\bar{y}'_a$ $a \in V$	$\{a'' \to \bar{a}\} \cup \{b \to \bar{b} \mid b \in V\}$	$\{\bar{a}, a''\}$	$\emptyset$	$\downarrow$
		$U_1 \setminus (V \cup \{\bar{a}, a''\})$	$V \cup \{b', b'' \mid b \in V\}$	s

Table 2.

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 $\Gamma = (V, U, \chi, \mathcal{N}, \alpha, \beta, x_I, x_O)$ 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 be 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. We recall that a picture language L is said to be local if there exists a finite set of pictures of size $(2, 2)$ which contains the set of $(2, 2)$ sub-pictures of every picture of L. Furthermore, every recognizable picture language is the projection of a local picture language.

Now we can state:

Theorem 2. $\mathcal{L}_{wa}(ANEPP) \setminus \mathcal{L}(REC) \neq \emptyset$.

Proof:
We 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 $card(V) \geq 2$. A bit more informally, 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. We now formally show that this language is not recognizable. 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. 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. 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 non-empty as well. Therefore, there are two pictures $\zeta \in L'(n_1, m)$ and $\theta \in L'(n_2, m)$, 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 ζ equals the stripe rectangle of size $(2, m)$ consisting of the n_2-th and $n_2 + 1$-th rows in θ. Consequently, both pictures obtained from ζ and θ by interchanging with each other their first halves are in L'. However, the projection by h of any of these pictures is not in L, a contradiction.

In the last part of the proof, we give some informal hints on how this language can be weakly accepted by an ANEPP consisting of two subnetworks: the first one alternatively deletes the first and the last row of the input picture until it is reduced to only two rows. Now these pictures are given as inputs to the subnetwork from Example 1. A formal construction of the first subnetwork, that is the network which processes the input pictures until they are sent to the input node of the network from Example 1 suitably modified, is presented in the sequel. It consists of six nodes that are defined as follows (Table 3):

Node	M	PI	PO	α
		FI	FO	β
x_I	$\{a \to X \mid a \in V\} \cup$	$\emptyset$	$\{X\} \cup \{a' \mid a \in V\}$	$\uparrow$
	$\{a \to a' \mid a \in V\}$	$\{X, Y\} \cup \{a' \mid a \in V\}$	$\emptyset$	w
x_1	$\{a \to Y \mid a \in V\}$	$\{X\}$	$\{Y\}$	$\downarrow$
		$\{Y\} \cup \{a' \mid a \in V\}$	$\emptyset$	w
x_2	$\{X \to \varepsilon\}$	$\{X, Y\}$	$\{Y\}$	$\uparrow$
		$\{a' \mid a \in V\}$	$\{X\}$	s
x_3	$\{Y \to \varepsilon\}$	$\{Y\}$	$\emptyset$	$\downarrow$
		$\{X\} \cup \{a' \mid a \in V\}$	$\{X, Y\}$	s
x_4	$\{a \to a' \mid a \in V\}$	$\{a' \mid a \in V\}$	$\{a' \mid a \in V\}$	$\uparrow$
		$\{X, Y\}$	$\emptyset$	w
x_5	$\{a \to a' \mid a \in V\}$	$\{a' \mid a \in V\}$	$\{a' \mid a \in V\}$	$\downarrow$
		$\{X, Y\}$	$\emptyset$	w

Table 3.

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 this picture with each other 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. □

The last result of this section could imply the previous result provided that there exists a local language whose complement is not recognizable. We are not aware of such a result though it is known that $\mathcal{L}(REC)$ is not closed under complement. In any case, the previous result provides a new language that is not recognizable, therefore it might be useful from this point of view too.

Theorem 3. The complement of every local language can be weakly accepted by an ANEPP.

Proof:
For a better understanding, we first give an informal argument followed by the complete formal proof. 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.

Formally, let L be a local language over the alphabet V defined by the set F of $(2,2)$-tiles. We consider the set F^c of all $(2,2)$-tiles over V that do not belong to F. 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 in Table 4; x_I and x_O are the input and output node, respectively.

Node	M	PI	PO	α_1
		FI	FO	β_1
x_I	$\{a \to a_i\}$	$\{a, b, c, d\}$	$\{a_i\}$	$\leftarrow$
		$\{a_i, b_i, c_i, d_i\}$	$\emptyset$	w
x_1	$\{c \to c_i\}$	$\{a_i\}$	$\{c_i\}$	$\downarrow$
		$\{b_i, c_i, d_i\}$	$\emptyset$	s
x_2	$\{b \to b_i\}$	$\{a_i, c_i\}$	$\{b_i\}$	$\uparrow$
		$\{b_i, d_i\}$	$\emptyset$	s
x_3	$\{d \to d_i\}$	$\{a_i, b_i, c_i\}$	$\{d_i\}$	$\rightarrow$
		$\{d_i\}$	$\emptyset$	s
x_4	$\{a_i \to \varepsilon\}$	$\{a_i, b_i, c_i, d_i\}$	$\{b_i, d_i\}$	$\leftarrow$
		V	$\{a_i, c_i\}$	s
x_O	$\emptyset$	$\{b_i, d_i\}$	$\{b_i, d_i\}$	$+$
		V	$\emptyset$	s

Table 4.

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, $1 \leq i \leq n$. □

4. Closure Properties

Closure properties of the families of picture languages defined here under some common operations on picture languages appear to be of interest. We mention here the following result, where rotations are intended as 90 rotations clockwise or counterclockwise:

Theorem 4. 1. The class $\mathcal{L}_{wa}(ANEPP)$ is closed under union, rotation, vertical and horizontal reflection, projection, inverse projection.
2. The class $\mathcal{L}_{sa}(ANEPP)$ is closed under intersection, rotation, vertical and horizontal reflection, projection, inverse projection.

Proof:
1. For union, we give an informal proof that can be easily formalized by the reader. Let Γ_1 and Γ_2 be to 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, $L_{wa}(\Gamma) = L_{wa}(\Gamma_1) \cup L_{wa}(\Gamma_2)$.

The closure under rotation is immediate as soon as we modify the row rules into column rules and vice versa, and the action mode accordingly. The closure under vertical and horizontal reflection can be proved analogously.

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 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 strong acceptance in the case of intersection. □

The closure of $\mathcal{L}_{wa}(ANEPP)$ under intersection as well as that of $\mathcal{L}_{sa}(ANEPP)$ under union are left open. The closure of the two classes under row and column concatenation has a similar status.

5. Final Remarks

We finish this work with a very short discussion on some problems left open here besides those concerning the closure properties which have been mentioned above. The first natural problem regards the equality between 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 $\mathcal{L}(REC)$.

Along the same lines, a future direction may be the comparison with other other array language families like those considered in [11-16].

It is rather plain that the membership problem is decidable for both classes $\mathcal{L}_{wa}(ANEPP)$ and $\mathcal{L}_{sa}(ANEPP)$. What other problems are still decidable?

References

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

[2] Giammarresi, D., Restivo, A.: Two-dimensional languages, in [9], 215–267.

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

[4] Inoue, I., Takanami, I.: A survey of two-dimensional automata theory, in *Proc. 5th Int. Meeting of Young Computer Scientists*, Lecture Notes in Computer Science **381**, 1990, 72–91.

[5] Margenstern, M., Mitrana, V., Perez-Jimenez, M.: Accepting hybrid networks of evolutionary systems, in *DNA Based Computers 10* Lecture Notes in Computer Science **3384**, 2005, 235–246.

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

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

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

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

[10] Sankoff, D. 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.

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

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

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

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

[15] Wang, P.S.: Hierarchical structure and complexities of parallel isometric patterns, *IEEE Trans. PAM I*, **5**, 1975, 92–99.

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