ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **18**, Number 1, 2015, 3–17

Networks of Polarized Evolutionary Picture Processors

Ştefan POPESCU

Faculty of Mathematics and Computer Science, University of Bucharest
E-mail: popescustefanbdv@gmail.com

Abstract. We define a new bio-inspired computational model for deciding 2-dimensional languages similar to those presented in [2] and [3]. The novel factor being the communication protocol, which is based on the polarity associated with the symbols on the picture frame. This model can be viewed as a 2-dimensional extension of the one presented in [1].

We compare the class of languages accepted by these networks to other classes of 2-dimensional languages, namely the classes of *Local Languages* and *Tiling Recognizable Languages* [12].

We show that these networks can accept the complement of any local picture language and some picture languages that are not recognizable by tiling systems. We also show that these networks can recognize any input that contains a given sub-picture, provided that the sub-picture is made up of at most 3 rows or 3 columns.

Key-words: rectangular picture, picture processor, network of evolutionary picture processors, local picture language, recognizable picture language, bio-inspired computation.

1. Introduction

Picture languages have been introduced in literature as a two-dimensional extension of traditional string languages. A picture is seen as a two-dimensional array of elements from a finite set of symbols, called an alphabet. They can be considered as formal models for image processing or pattern recognition. Many tools were used to generalize the properties from strings to 2-dimensional languages as well as to classify picture languages: regular expressions [10], grammars [6, 11], automata [8]. One of the main focuses in this area is to define models for picture recognizability. A widely

used formal model for the recognition of picture languages, based on projection of local properties, is the tiling system introduced in [5].

We set our attention on two widely studied classes of picture languages, namely local languages and tile-recognizable languages.

In this paper we define a new computational model for rectangular picture languages, based on the work started in [2] and [3], which in turn is a 2D generalization of models for string languages, such as [9]. Mechanisms inspired from cell biology were considered: networks of interconnected processors which are able to perform just one type of point operation: row/column deletion or symbol substitution. These nodes are assigned with communication protocols defined by some very simple context conditions. More simply said, each node processor acts on the local data in accordance with some predefined rules. This data is then filtered and passed over the network according to some protocols and processor adjacency. This filtering process may require the satisfaction of some conditions imposed by the sending and receiving processors. All the nodes simultaneously send their data to and receive data from the nodes they are connected to.

The mechanism described here can also be viewed as a generalization of the work done on string languages in [1] to rectangular pictures. More exactly, the processors remain essentially the same as in [2] regarding the operations they are able to perform, but the communication protocol is fundamentally changed.
This change in protocol has proven to be interesting from an engineering point of view [4]. We stress that this model is purely a theoretical concept, inspired by biological phenomena.

2. Basic Definitions and Notations

We assume the reader is familiar with the basic notations and properties for one dimensional words (string languages). For a more complete description on these subjects the reader is referred to [7]. The basic notations for 2-dimensional languages are meant to be an extension of string languages.

Let V be a set of symbols, called an alphabet. A *picture*, or two dimensional word, can be seen as a two dimensional array (matrix) over V. The set of all pictures over V is denoted by V_*^*.

Let π be a picture. The minimal alphabet over which π is written is denoted by $alph(\pi)$. The pair (m, n) is said to be the size of the picture π if π has m rows and n columns. $l_1(\pi)$ will represent the number of rows of the picture π while $l_2(\pi)$ will denote the number of columns of π. The empty picture, denoted by ϵ, is the class of all pictures of size(m, n) with $mn = 0$. The set of pictures with m rows will be denoted as V_m^* and the set of pictures with n columns will be V_*^n. The set of pictures with m rows and n columns will be V_m^n. The element from the picture π situated at the intersection of the i^{th} row and j^{th} column will be represented as $\pi(i, j)$.

We briefly remind the notions of *local languages* and *tile recognizable languages*. For a more formal definition the reader is referenced to [6].

We first define the class of locally recognizable picture languages: A set of tiles Θ is a finite set of square pictures of size 2. We say a picture π is *locally accepted* by Θ if every of its sub-pictures of size $(2,2)$ can be found in Θ. A language is recognized by the set Θ if every of its pictures is locally accepted by Θ.

A tiling system τ is defined as a 4-tuple (Θ, V, W, δ) where Θ is a finite set of square pictures of size $(2,2)$, V and W are the picture alphabet and the tile alphabet, respectively and δ is a mapping function between the two alphabets - the picture alphabet and the tiling alphabet. Informally, we can say that a picture language is recognized by τ if for each of its pictures, the set of sub-pictures of size 2x2 can be obtained by projecting tiles from Θ using δ. The class of local languages and the class of tile recognizable languages shall be denoted by $L(LOC)$ and $L(REC)$, respectively.

Let V be an alphabet, an operation of the form $a \to b$, with $a, b \in V \cup \{\epsilon\}$ is called an *evolutionary rule*. This rule is:

- a *substitution* rule if neither a nor b are the empty symbol ϵ. The set of all substitutions over V shall be denoted with Sub_V.
- a *deletion* rule if $b = \epsilon$. The set of all deletions over V shall be denoted with Del_V.

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

- if $\sigma \equiv a \to b \in Sub_V$, then $\sigma^{\leftarrow}(\pi)$ is the set of all pictures that are obtained from π where an occurrence of symbol a is replaced by b in the leftmost column of π. We note that the action σ is applied to *any* occurrence of a in the leftmost column of π in different copies of π. If there is no occurrence of a in the leftmost column, then the result will be the unchanged picture π.
  - In the same way we define $\sigma^{\rightarrow}(\pi)$, $\sigma^{\uparrow}(\pi)$, $\sigma^{\downarrow}(\pi)$, as the set of all pictures obtained by applying σ to the rightmost column, the top row and to the bottom row, respectively.
- if $\sigma \equiv a \to b \in Del_V$, then $\sigma^{\leftarrow}(\pi)$ is the set of all pictures that are obtained from π by deleting the leftmost column of π, if and only if this column contains at least one occurrence of a.
  - In the same way we define $\sigma^{\rightarrow}(\pi)$, $\sigma^{\uparrow}(\pi)$, $\sigma^{\downarrow}(\pi)$, as the set of all pictures obtained by applying σ to the rightmost column, the top row and to the bottom row respectively.

Example 1. *Let* $\pi = \begin{array}{|c|c|c|c|}\hline a & b & a & b \\ \hline b & c & a & c \\ \hline c & b & a & b \\ \hline \end{array}$ *and the following evolutionary rules:*

$\sigma_1 \equiv a \to c \in Sub_V$ *and* $\sigma_2 \equiv a \to \epsilon \in Del_V$.

Then $\sigma_1^{\uparrow}(\pi) = \left\{ \begin{array}{|c|c|c|c|}\hline \boldsymbol{c} & b & a & b \\ \hline b & c & a & c \\ \hline c & b & a & b \\ \hline \end{array}, \begin{array}{|c|c|c|c|}\hline a & b & \boldsymbol{c} & b \\ \hline b & c & a & c \\ \hline c & b & a & b \\ \hline \end{array} \right\}$,

$$\sigma_2^{\uparrow}(\pi) = \left\{ \begin{array}{|c|c|c|c|} \hline b & c & a & c \\ \hline c & b & a & b \\ \hline \end{array} \right\},\ \sigma_1^{\rightarrow}(\pi) = \sigma_2^{\rightarrow}(\pi) = \left\{ \begin{array}{|c|c|c|c|} \hline a & b & a & b \\ \hline b & c & a & c \\ \hline c & b & a & b \\ \hline \end{array} \right\}$$

Applying $\sigma_1^{\uparrow}$ on the picture π yields two results because there are two occurrences of a that can be substituted on the first row. In the case of $\sigma_2^{\uparrow}$, the top row is deleted because it contains at least one a. Both $\sigma_1^{\rightarrow}(\pi)$ and $\sigma_2^{\rightarrow}(\pi)$ have the same outcome, the unmodified picture π, because there is no occurrence of a on the rightmost column.

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

Definition 1. We call the *valence of a symbol* of V as being the resulting value of a function $\varphi : V \to \mathbb{Z}$ that associates with every symbol from V an integer value.

Definition 2. The *polarity of a picture* is denoted by the function $\Phi : V_*^* \to \{-, 0, +\}^4$ which associates a symbol from $\{-, 0, +\}$ to the top and bottom rows, the leftmost and rightmost column. We define the polarity of a picture π of size (m, n) as:

$$\begin{aligned} \Phi(\pi) =& (sign(\sum_{i=1}^{n} \varphi(\pi(1, i))), sign(\sum_{i=1}^{m} \varphi(\pi(i, n))), \\ & sign(\sum_{i=1}^{n} \varphi(\pi(m, i))), sign(\sum_{i=1}^{m} \varphi(\pi(i, 1)))). \end{aligned}$$

Informally we say that the polarity of a row or a column as being the *sign* of the sum of the valences of every symbol in that row or column. The polarity of a picture can be seen as a 4-tuple over the set $\{-, 0, +\}$ and represents the polarities, in this order, of the first row, the last column, the last row and the first column.

Example 2. *Let $V = \{a, b, c, d\}$, with the following symbol valences: $\varphi(a) = 1$; $\varphi(b) = -2$ and $\varphi(c) = \varphi(d) = 0$.*

For $\pi = \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline a & d & c & c \\ \hline b & a & a & a \\ \hline c & b & c & d \\ \hline \end{array}$ *the picture polarity of π is $\Phi(\pi) = (-, +, -, 0)$.*

3. Accepting Networks of Polarized Evolutionary Picture Processors

A ***polarized evolutionary picture processor*** over V is a 3-tuple $(M, \mathcal{AP}, \alpha)$ where:

- M represents the set of evolutionary rules associated with the processor such that either $M \subseteq Sub_V$ or $M \subseteq Del_V$.
- $\mathcal{AP} \in \{-, 0, +\}^4$ is a 4-tuple representing the picture polarity compatible with the processor.
- $\alpha \in \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$ denotes the action mode of the rules of pictures associated with the processor.

We say a picture π and a processor are *compatible* if and only if the picture's polarity, $\Phi(\pi)$, is identical to the $\mathcal{AP}$ element associated with the processor. We will denote the set of all polarized picture processors over the alphabet V by EPP_V.

An ***accepting network of polarized evolutionary picture processors*** (AN-PEPP for short) is a tuple

$$\Gamma = (V, U, \varphi, G, \mathcal{N}, In, Out)$$

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 X_G as the set of nodes and E_G as the set of edges. G will be called the network's *underlying graph*;
- $\varphi : U \rightarrow \mathbb{Z}$ - gives the valence of each symbol;
- $\mathcal{N}$ is a mapping $\mathcal{N} : X_G \rightarrow EPP_V$, which associates each node x of G to a picture processor $(M_x, \mathcal{AP}_x, \alpha_x)$;
- In, Out are the *input node* and the *output node*, respectively.

The ***configuration of a network*** Γ is a mapping function $C : X_G \rightarrow 2^{U_*^*}$ which associates a finite set of pictures to every node of the graph. A configuration can also be viewed as the set of pictures present in any node at any given time. The initial configuration of a network that takes as input a picture π is defined as $C_0^{(\pi)}(x_I) = \pi$ and $C_0^{(\pi)}(x) = \emptyset$ for all the other nodes. A configuration change can happen either by an *evolutionary step* or a *communication step*. We define a configuration change:

- via an *evolutionary step* as:

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

- via a *communication step* as:

$$C'(x) = (C(x) \backslash \{\pi \in C(x) | \Phi(\pi) \neq \mathcal{AP}_x\}) \cup \bigcup_{\{x,y\} \in E_G} (\{\pi \in C(y) | \Phi(\pi) = \mathcal{AP}_x\}).$$

We denote that C' is obtained from C via an evolutionary step by $C \Rightarrow C'$. When C' is obtained from C via a communication step we will write $C \vdash C'$. Let Γ be a ANPEPP, the computation of Γ that takes as input a picture $\pi \in V_*^*$ is a sequence of configurations $C_0^\pi, C_1^\pi, C_2^\pi, ...$ where C_0^π is the initial configuration of the network, then $C_{2i}^\pi \Rightarrow C_{2i+1}^\pi$ and $C_{2i+1}^\pi \vdash C_{2i+2}^\pi$. The configurations in the series are changed by alternative steps and each of them is uniquely determined by the previous one, except for the initial configuration, which was defined above.

The process is defined as follows: Each processors has a set of pictures associated with it. During an evolutionary step, the pictures associated with the processor are subject to the processor's associated rules, yielding a new set of pictures. During the communication step, each processor ejects all pictures, keeping one copy of each picture that is compatible with that processor and accepts from neighboring processors all pictures that are compatible with it.

The computational process ends and the input is said to be *accepted* if a picture enters the *Out* node.

We will refer to the class of languages that can be accepted by such a network as *L(ANPEPP)*.

4. Comparison with other Picture Language Classes

Theorem 1. $L(ANPEPP) \cap (L(REC) \backslash L(LOC)) \neq \emptyset$.

Proof. We claim that the language of squares over a one letter alphabet, formally described below, is recognized by an *ANPEPP*:

$$L = \{\pi \in \{a\}_*^* \mid l_1(\pi) = l_2(\pi)\}$$

As shown in [6], $L \in REC \backslash LOC$. We will now construct a network $\Gamma_1 = (V, U, G, \mathcal{N}, In, Out)$ that accepts L:

The input alphabet is $V = \{a\}$, the network's alphabet is $W = \{a, a^+, a^-\}$. The valence of each symbol and the node descriptions are given in the tables below. The network's underlying graph is defined by the adjacency list from Fig. 1.

Node	$\mathcal{AP}$	M	α	*Adjacency List*
In	$(0,0,0,0)$	$a \to a^+$	$\uparrow$	N_1, Del_1, Del_2, OUT
N_1	$(+,0,0,+)$	$a \to a^-$	$\downarrow$	In, N_2
N_2	$(+,0,-,0)$	$a^+ \to \epsilon$	$\uparrow$	N_1, Del_1, Del_2
Del_1	$(0,0,-,-)$	$a^- \to \epsilon$	$\leftarrow$	In, N_2
Del_2	$(0,0,-,-)$	$a^- \to \epsilon$	$\leftarrow$	In, N_2
Out	$(+,+,+,+)$	$\emptyset$	$-$	In

Fig. 1. Processor description for Γ_1.

We remark that a picture π is a square if and only if, after deleting a row and a column, the resulting picture will also be a square, provided the initial picture had at least two rows and columns. The basic square picture is the picture made up of one

element. The main reasoning behind our network design is as follows: we repeatedly delete the top row and left-most column until a picture of size $(1, 1)$ is obtained. If no such picture is obtained, the input is rejected. The computation will eventually end, there being a limited number of row/column deletions that can take place.

The network acts as follows: The initial picture is located in the *In* node, while all other nodes are empty.

$s \in U$	$\varphi(s)$
a	0
a^-	-1
a^+	1

Fig. 2. Valence function for Γ_1.

Here, via an evolutionary step, an a symbol is substituted for an a^+ on the first row. If the picture was made up of only one element, then the resulting picture, having a polarity compatible with the *Out* node, will migrate to the output node and the process will halt by accepting the input. If the picture was made up of only one row but multiple columns, then after the substitution step, the bottom row will be associated with a positive polarity. In this case, there is no node compatible with such a polarity, and the input will be rejected. The same thing is true if the input picture is made up of only one column and several rows. The resulting picture will have a positive polarity associated with its rightmost column, and the input will be rejected.

In the case in which the input picture is made up of multiple rows and columns we will proceed to delete the top row and the leftmost column. In the input node, after the substitution takes place, if the substituted symbol on the first row is also positioned on the leftmost column, then the picture will migrate to node N_1. All other possible results are ejected from the network.

In N_1, a symbol on the bottom row is substituted with an a^-. All other copies where the substitution can take place, besides on the bottom-left corner, are rejected because they will have a positive polarity associated with the leftmost column and a negative polarity associated with the bottom row. The only picture that will not be ejected must have an a^+ on its upper-left corner, and an a^- on its lower-left corner - all other positions being occupied with the symbol a. This picture will be sent to node N_2.

The picture that reaches N_2, as we saw, is made up of at least two rows and two column. In N_2 the top row is deleted. The resulting picture, provided that it now has at least two rows, has a neutral top row polarity, and will migrate to Del_1. If the resulting picture, after the deletion in N_2, is made up of only one row, then it will migrate to node Del_2.

Let us treat the latter case first: In Del_2 the picture is basically a string. Here, the leftmost symbol is deleted. The result will migrate back to the input node. Being made up of a single row, the In node will either send it to the *Out* node or eject it from the network.

In the case that the picture is sent to the Del_1, the leftmost column is deleted. The result is sent back to the In node and the process resumes - each time a row and column being deleted. □

Theorem 2. $L(ANPEPP)\backslash L(REC) \neq \emptyset$.

Proof. We consider the following language:

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

As shown in [2], $L \notin REC$, provided that $card(V) \geq 2$. L is the set of all pictures with at least 2 columns, an even number of rows and have the two middle rows identical.

We construct a network $\Gamma_2 = (V, U, G, \mathcal{N}, In, Out)$ that accepts L. The construction is similar to those described in [2, 3]. The network can be divided into two parts. The first sub-network takes the picture as input and repeatedly deletes the top and bottom rows. If the input picture π has an even number of rows, eventually the sub-network will generate the picture made up of the two middle rows of π. The second part of the network checks if a picture is made up of exactly two identical rows.

We construct the network as follows:

- $V = \{a_i | i \in \{1,...,n\}, n \geq 2\}$;
- $U = V \cup \{a_i^+, a_i^- | i \in \{1,...,n\}\} \cup \{s^+, s^-\}$;
- The valences for the symbols from U are given in Fig. 3;
- The tings and rules of each processor are given in Fig. 4.

At first, a picture π is give as input to the node In. Here a substitution takes place on the top row: either an a_i symbol is substituted for another with a positive polarity, or an a_i symbol is replaced by an s^-. Let us first discuss the latter case.

After the evolutionary step, the resulting picture is sent to node M if the substitution took place on the upper-right corner of the picture. Any picture with an s^- symbol in any other position is removed from the network. In node M, a symbol on the bottom row is substituted for a s^+.

$x \in W$	$\varphi(x)$
a_i	0
a_i^+	$+i$
a_i^-	$-i$
s^+	1
s^-	-1

Fig. 3. Valence function for Γ_2.

The only picture from the resulting set that will remain in the network is the one in which the substitution took place on its lower-right corner. The new picture travels to node D_1. Here, the top row is deleted. If the initial picture had at least 2 rows,

the resulting picture will have a neutral polarity associated with its top row, and will migrate to D_2. Here, the bottom row is also deleted, and the result is sent back to In. As it can be easily seen, the picture that returns to the input node is the initial picture without its top and bottom rows. If the input picture has an even number of rows, the picture made up of its two middle rows will eventually return to the input node.

Node	M	$\mathcal{AP}$	α	*Adjacency List*
In	$\{a_i \to a_i^+, a_i \to s^- \mid i \in \{1,...,n\}\}$	$(0,0,0,0)$	$\uparrow$	M, D_2, N_1
M	$\{a_i \to s^+ \mid i \in \{1,...,n\}\}$	$(-,-,0,0)$	$\downarrow$	In, D_1
D_1	$\{s^- \to \epsilon\}$	$(-,0,+,0)$	$\uparrow$	M, D_2
D_2	$\{s^+ \to \epsilon\}$	$(0,+,+,0)$	$\downarrow$	In, D_1
N_1	$\{a_i \to a_i^- \mid i \in \{1,...,n\}\}$	$(+,0,0,+)$	$\downarrow$	In, Del_1, N_2
Del_1	$\{a_i^+ \to \epsilon \mid i \in \{1,...,n\}\}$	$(+,0,-,0)$	$\leftarrow$	N_1, N_2
N_2	$\{a_i \to a_i^+ \mid i \in \{1,...,n\}\}$	$(0,0,0,0)$	$\uparrow$	N_1, N_3, Del_1
N_3	$\{a_i \to a_i^- \mid i \in \{1,...,n\}\}$	$(+,+,0,+)$	$\downarrow$	N_2, Del_2
Del_2	$\{a_i^+ \to \epsilon \mid i \in \{1,...,n\}\}$	$(+,0,-,0)$	$\uparrow$	N_2, Out
Out	$\emptyset$	$(-,-,-,-)$	$-$	N_2

Fig. 4. Processor settings for Γ_2.

In the case of the pictures that, after the evolutionary step in the In node, contain a a_i^+ symbol - they migrate to N_1 if and only if the substitution was made on the upper-left corner. All other pictures are ejected from the network. The main purpose of this second part of the network is to check if the picture in In is made up of exactly two identical rows. This is done by repeatedly verifying that the symbol on the top left corner is identical to the one on the bottom left corner and then deleting the leftmost column. As a last step, it checks if the picture was made up of exactly two rows.

In N_1 an a_i is substituted by an a_i^-, giving the bottom row an overall negative polarity. The resulting pictures will be ejected from N_1 while Del_1 will accept the one picture where the a_i^- symbol is placed on the lower-left corner. Moreover, the a_i^+ on the top left corner and the a_i^- have to be complementary one to another. More exactly, both a_i^+ and a_i^- have to originate from the same a_i symbol.

In Del_1, the left-most column is deleted. The picture is now made-up of only neutrally polarized symbols, and travels to N_2. Here, like in In, an a_i symbol is substituted for an a_i^+ symbol on the top row. The pictures where the substitution takes place other than on the top-left corner, are rejected. The resulting pictures can migrate towards two nodes. If the picture has at least 2 columns it will migrate back to N_1 and the cycle will repeat. If the picture is made up of exactly one column (the first and last column are the same one), the picture will migrate to N_3. A picture only reaches N_3 if the first row is identical to the last row, except for the right-most two elements (top-right and bottom-right corners). What remains to be checked is that the two aforementioned corners contain the same symbol, and that the picture, when it first entered N_1 was made up of exactly two rows. In N_3, the only element on the bottom row is substituted for for its negatively polarized counterpart. The resulting

picture will go to Del_2 if the top element and bottom element originally used have the same symbol.

At this point we have successfully checked if the first two rows, from the original picture, are identical. The final step is to check is the picture is made up exactly two rows.

In Del_2 the top element is deleted. If the resulting picture is made up of only one element, it migrates to the output node, and the input is accepted.

Because of the row/deletion operations, and the communication protocol, the computation is guarantied to halt in a limited number of steps. □

Theorem 3. *The complement of any local language can be accepted by an ANPEPP.*

Proof. Again we follow the idea from [2, 3]. We consider the local language over the alphabet V determined by the set of tiles T. Let T^c be the complementary set of tiles, namely $T^c = V_2^2 \backslash T$. We will construct an *ANPEPP* Γ_3 that takes as input a picture $\pi \in V_*^*$, and accepts it if and only if $B_{2,2}(\pi) \cap T^c \neq \emptyset$.
Γ_3 is structured as follows:

- $V = \{a_i \mid i \in \{1, ..., n\}\}$;
- $W = \{a_i, a_i^{-1}, a_i^{+1}, a_i^{+2} \mid i \in \{1, ..., n\}\}$;
- The processor descriptions along with the adjacencies can be found in Fig. 5;
- The valence of the symbols can be found in Fig. 6.

$Node$	M	$\mathcal{AP}$	α	Adjacency List
In	$\emptyset$	$(0,0,0,0)$	$-$	$Del_\uparrow, Del_\downarrow, Del_\leftarrow, Del_\rightarrow, Z_i^1$
$Del_\uparrow$	$\{s \to \epsilon \mid s \in V\}$	$(0,0,0,0)$	$\uparrow$	In
$Del_\downarrow$	$\{s \to \epsilon \mid s \in V\}$	$(0,0,0,0)$	$\downarrow$	In
$Del_\leftarrow$	$\{s \to \epsilon \mid s \in V\}$	$(0,0,0,0)$	$\leftarrow$	In
$Del_\rightarrow$	$\{s \to \epsilon \mid s \in V\}$	$(0,0,0,0)$	$\rightarrow$	In
Z_i^1	$a_i \to a_i^{-1}$	$(0,0,0,0)$	$\uparrow$	In, Z_i^2
Z_i^2	$d_i \to d_i^{-1}$	$(-,0,0,-)$	$\downarrow$	Z_i^1, Z_i^3
Z_i^3	$b_i \to b_i^{+2}$	$(-,-,-,-)$	$\uparrow$	Z_i^2, Z_i^4
Z_i^4	$c_i \to c_i^{+1}$	$(+,+,-,-)$	$\downarrow$	Z_i^3, Z_i^5
Z_i^5	$a_i^{-1} \to \epsilon$	$(+,+,0,0)$	$\leftarrow$	Z_i^4, Z_i^6
Z_i^6	$b_i^{+2} \to \epsilon$	$(+,+,-,+)$	$\uparrow$	Z_i^5, Out
Out	$\emptyset$	$(-,-,-,-)$	$-$	Z_i^6

Fig. 5. Processor settings for Γ_3 - a network that accepts the complement of any language in *LOC*.

The network is centered around the main idea that given an input picture π and a set of tiles set T, we construct $T^c = \{B_1, B_2, ..., B_n\}$ as the complementary set of tiles and check if there is any sub-picture of size $(2, 2)$ of π that can be found in T^c.

We will denote that each block element B_i is of the form $\begin{array}{|c|c|}\hline a_i & b_i \\ \hline c_i & d_i \\ \hline \end{array}$.

As can be observed, there is a constant flow of information between the In node and the set of deletion nodes. Initially, a picture π will be given as input. The In node has no rules associated with it, so after a communication step, a copy of the unmodified π will enter each of the four deletion nodes. Each node, deletes a certain row or column, then sends the result back to In. This constant flow assures us every possible sub-picture of π will enter, at some moment, the input node.

The second part of the Γ_3 can be viewed as n disjoint subnetworks that work in parallel. Each of the n subnetworks is made up of the following set of nodes $\{Z_i^1, Z_i^2, Z_i^3, Z_i^4, Z_i^5, Z_i^6\}$, $1 \leq i \leq n$, and checks weather a picture of size $(2,2)$ that was passed from the In node is identical to a certain block B_i.

$x \in W$	$\varphi(x)$
a_i	0
a_i^{+1}	1
a_i^{-1}	-1
a_i^{+2}	2

Fig. 6. Valence function for Γ_3.

As we saw, the input picture π and every of its sub-pictures will, at some point be in the input node. After a communication step, copies of every such picture will be sent to all Z_i^1 nodes. We will further discuss the step-by-step process for only one of the sub-networks, namely the sub-network that checks if the picture received is identical to the block B_i. The general idea is as follows: the nodes Z_i^1 through Z_i^4 checks if the picture's four corners are the same as B_i's corners, while nodes Z_i^5 and Z_i^6 check if the picture is of size $(2,2)$. A more detailed step-by-step explanation is as follows:

In the Z_i^1 node, the top left corner is marked with a negative polarity if and only if symbol on that position is the same symbol as in B_i. Any substitution that takes place on the top row in any other position will result in a picture that will be ejected from the network. The resulting picture migrates to Z_i^2 where a similar process takes place: a symbol is marked on the bottom right corner if that symbol is the symbol that can be found on B_i's bottom right corner. The resulting picture, having a negative polarity on all four edges, migrates to Z_i^3. Here, the top right element is marked, and given a valence of $+2$, if and only if it is the same as the one on the top right corner of B_i. Any substitution made on another position will result in the picture being ejected. The new picture is sent to Z_i^4. Here the bottom left symbol is marked with a positive polarity if it is the same as the one on the same position in B_i. As a result, the picture that migrates to Z_i^5 originally had the same symbols on all 4 corners as B_i.

It remains to check that the picture in Z_i^5 is of size $(2,2)$. We first check that the picture is made up of two columns: in Z_i^5, the left-most column is deleted. The resulting picture migrates to Z_i^6 if and only if it is now made up of only one column. In

any other case, the new picture will have an overall neutral polarity on its left column, and is sent out from the network. The resulting single column, is now checked if it is made up of two rows. This is done by deleting the top-most symbol. If the resulting picture is of size $(1,1)$ then its only element is negatively polarized and the picture migrates to the output node. □

Proposition 1. *Let $\pi, \pi' \in V_*^*$, with $l_2(\pi') = 3$. We can construct an ANPEPP that can decide if $\pi' \in B_{*,*}(\pi)$. More exactly, we can construct an ANPEPP that checks if π' is a sub-picture of π.*

Proof. We will construct an *ANPEPP* Γ_4 that takes as input a picture π and accepts it only if a predefined picture $\pi' = \begin{array}{|c|c|c|}\hline a_1 & b_1 & c_1 \\ \hline ... & ... & ... \\ \hline a_n & b_n & c_n \\ \hline\end{array}$ is a sub-picture of π. Let $V = \{s_1, ..., s_m\}$ be the picture alphabet and $W = V \cup \{s^+, s^- \mid s \in V\}$ be the network's alphabet. The valence for each symbol is given in Fig. 7, while the processor settings are given in Fig. 8. The network's underlying graph is shown in Fig. 9.

$x \in W$	$\varphi(x)$
$\{s \mid s \in V\}$	0
$\{s^+ \mid s \in V\}$	$+1$
$\{s^- \mid s \in V\}$	-3

Fig. 7. Valence function for Γ_4.

Node	M	$\mathcal{AP}$	α
In	$\emptyset$	$(0,0,0,0)$	$-$
Del_a	$\{s \to \epsilon \mid s \in V\}$	$(0,0,0,0)$	$\uparrow$
Del_b	$\{s \to \epsilon \mid s \in V\}$	$(0,0,0,0)$	$\rightarrow$
Del_c	$\{s \to \epsilon \mid s \in V\}$	$(0,0,0,0)$	$\downarrow$
Del_d	$\{s \to \epsilon \mid s \in V\}$	$(0,0,0,0)$	$\leftarrow$
Z_1^i	$a_i \to a_i^+$	$(0,0,0,0)$	$\uparrow$
Z_2^i	$c_i \to c_i^+$	$(+,0,0,+)$	$\uparrow$
Z_3^i	$b_i \to b_i^-$	$(+,+,0,+)$	$\uparrow$
Del^i	$b_i \to \epsilon$	$(-,+,0,+)$	$\uparrow$
Z_1^n	$a_n \to a_n^+$	$(0,0,0,0)$	$\uparrow$
Z_2^n	$c_n \to c_n^+$	$(+,0,+,+)$	$\uparrow$
Z_3^n	$b_n \to b_n^-$	$(+,+,+,+)$	$\uparrow$
Del_1	$a_n \to \epsilon$	$(-,+,-,+)$	$\leftarrow$
Del_2	$c_n \to \epsilon$	$(-,+,-,-)$	$\rightarrow$
Out	$\emptyset$	$(-,-,-,-)$	$-$

Fig. 8. Processor settings for Γ_4.

The main idea is similar to the one in the previous proof. We use a set of deletion nodes to break down the input picture into all possible sub-pictures. These sub-pictures will be eventually past down to the second part of the network which is

formed by n sets of nodes, connected in series one after another, each set denoted by $S_i = \{Z_1^i, Z_2^i, Z_3^i\}$. Every sub-network S_i checks weather the i^{th} row of the given block is a possible match with the corresponding row of π'. We say possible match, because we do not check that they are exactly the same in a latter part of the network.

We will give a short description of how this second part of the network works, for a given block B, generated by the deletion nodes from the input π. In Z_1^1 the top left symbol is marked with a positive polarity if it is the same as the left-most symbol on the first row of π'.

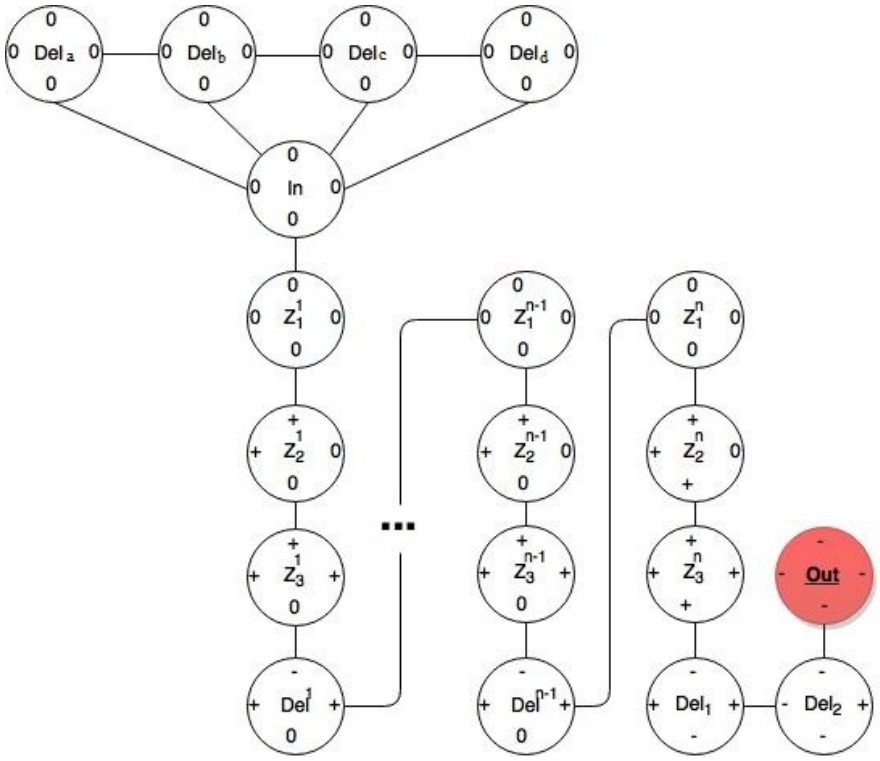

Fig. 9. The structure of Γ_4.

The resulting picture moves on to Z_2^1 which does a similar check for the top-right corner element. The result will migrate to Z_3^1 where a symbol is marked on the top row if it is identical to the middle element of the first row of π'. We notice that we didn't check a precise identity between B's rows or that if B is made up of four or more columns. We will make sure that B has exactly three columns later on. The result migrates to the Del^1 node, which deletes the top row, then sends the resulting picture to Z_1^2. Here the process is repeated again, this time checking if the top left symbol of the resulting block is the same as the element $\pi'(2,1)$. The process is repeated for the checking of the top $n-1$ rows of π'.

The process is essentially same until the block is eventually cut down to a single row. This single row has to be checked if it is the same as the bottom row of π' This is done similarly as before, going through nodes Z_1^n, Z_2^n, Z_3^n, but this time the polarity associated with the block's bottom row is also changed (the bottom and top rows are actually the same row). What remains is to check if the block B was made up of exactly three column. After the picture migrate through the nodes Del_1 and Del_2, the left-most column and the right-most column are deleted. If the remaining picture is made up of exactly one symbol, then B must have had exactly three columns, and therefore was identical to the sought picture π'. The resulting picture, of size $(1, 1)$, with a negative polarity, enters the output node, and the computation ends with acceptance. The computation will always end, because the number of sub-pictures of π is finite. □

By using the same principle we can deduce the following:

Proposition 2. *Given a picture π with $l_1(\pi) \leq 3$ or $l_2(\pi) \leq 3$ and $L = \{\pi' \mid \pi$ is a sub-picture of $\pi'\}$, then $L \in L(ANPEPP)$.*

5. Conclusions and Further Work

We have introduced a new computing model that can accept the complement of any local language and some languages that are not recognizable by tiling systems. The natural question that arises is how big is this new class of picture languages compared to the more traditional classes. Another topic of interest is represented by the closure properties of the families of languages generated by such networks. Although it is obvious that this class of languages is closed under rotation or mirror image, results regarding row/column concatenation or language complement would be of interest.

Acknowledgements. This work was supported by the strategic grant POSDRU/159/1.5/S/137750.

References

[1] ARROYO F., CANAVAL S. G., MITRANA V., POPESCU ŞT., *Networks of Polarized Evolutionary Processors Are Computationally Complete*, in *Language and Automata Theory and Applications*, Springer International Publishing, 2014, pp. 101–112.

[2] BOTTONI P., LABELLA A., MITRANA V., *Accepting Networks of Evolutionary Picture Processors*, Journal Fundamenta Informaticae - Formal Models-Computability, Complexity, Applications, Volume **131**, Issue 3-4, July 2014, pp. 337–349.

[3] BOTTONI P., LABELLA A., MANEA F., MITRANA V., SEMPERE J., *Networks of Evolutionary Picture Processors with Filtered Connections*, in *UC '09 Proceedings of the 8th International Conference on Unconventional Computation*, 2009, pp. 70–84.

[4] DÍAZ M.A., DE MINGO L.F., GÓMEZ BLAS N., CASTELLANOS J., *Implementation of massive parallel networks of evolutionary processors (MPNEP): 3-colorability problem*, in *Nature Inspired Cooperative Strategies for Optimization*, (NICSO 2007), Studies in Computational Intelligence 129, Springer, 2008, pp. 399—408.

[5] GIAMMARRESI D., RESTIVO A., *Recognizable picture languages*, Int. Journal Pattern Recognition and Artificial Intelligence, Vol. **6**, No. 2 & 3, 1992, pp. 241-–256.

[6] GIAMMARRESI D., RESTIVO A., *Two-dimensional languages*, in [12], pp. 215–267.

[7] HOPCROFT J. E., MOTWANI R., ULLMAN J. D., *Introduction to Automata Theory, Languages, and Computation* (3rd ed.). Pearson, 2013.

[8] INOUE K., TAKANAMI I., *A survey of two-dimensional automata theory*, in *Information Sciences* 55.1 (1991), pp. 99–121.

[9] MARGENSTERN M., MITRANA V., PÉREZ-JIMÉNEZ M. J., *Accepting Hybrid Networks of Evolutionary Processors*, in *Proc. 10th International Workshop on DNA Computing, DNA 10*, Lecture Notes in Computer Science 3384, Springer-Verlag, Berlin, 2004, pp. 235–246.

[10] MATZ O., *Regular expressions and context-free grammars for picture languages*, in *Proc. 14th STACS*, Lecture Notes in Computer Science 1200, 283–294, Springer-Verlag, 1997, pp. 283–294.

[11] ROSENFELD A., *Picture Languages – Formal Models of Picture Recognition*, Academic Press, New York, 1979.

[12] ROZENBERG G., SALOMAA A. (eds.), *Handbook of Formal Languages*, vol. I-III. Springer-Verlag, Berlin, 1997.