Accepting Networks of Non-inserting Evolutionary Processors

Jürgen Dassow[1] and Victor Mitrana[2,⋆]

[1] Faculty of Computer Science, University of Magdeburg
P.O.Box 4120, 39016 Magdeburg, Germany
dassow@iws.cs.uni-magdeburg.de
[2] Faculty of Mathematics, University of Bucharest
Str. Academiei 14, 70109 Bucharest, Romania
Department of Information Systems and Computation
Technical University of Valencia,
Camino de Vera s/n. 46022 Valencia, Spain
mitrana@fmi.unibuc.ro

Abstract. In this paper we consider four variants of accepting networks of evolutionary processors with in-place computations, that is the length of every word in every node at any step in the computation is bounded by the length of the input word. These devices are called here accepting networks of non-inserting evolutionary processors (ANNIEP shortly). The variants differ in two respects: filters that are used to control the exchange of information, i.e., we use random context conditions and regular languages as filters, and the way of accepting the input word, i.e., at least one output node or all output nodes are nonempty at some moment in the computation. The computational power of these devices is investigated. In the case of filters defined by regular languages, both variants lead to the class of context-sensitive languages. If random context conditions are used for defining filters, all linear context-free languages and some non-semilinear (even over the one-letter alphabet) can be accepted with both variants. Moreover, some closure properties of the classes of languages ANNIEPs with random context filters are also given.

1 Introduction

The origin of networks of evolutionary processors (NEP for short) is a basic architecture for parallel and distributed symbolic processing, related to the Connection Machine [9] as well as the Logic Flow paradigm [7], which consists of several processors, each of them being placed in a node of a virtual complete graph, which are able to handle data associated with the respective node. All the nodes send simultaneously their data and the receiving nodes handle also simultaneously all the arriving messages, according to some strategies, see, e.g., [8,9]. Similar ideas may be met in other bio-inspired models like *membrane systems* [16], *evolutionary systems* [4], or models from Distributed Computing area

⋆ Work supported by the Alexander von Humboldt Foundation.

C. Priami et al. (Eds.): Trans. on Comput. Syst. Biol. XI, LNBI 5750, pp. 187–199, 2009.

like *parallel communicating grammar systems* [15], *networks of parallel language processors* [3].

In a series of papers (see [14] for an early survey) one considers that each node may be viewed as a cell having genetic information encoded in DNA sequences which may evolve by local evolutionary events, that is point mutations. Each node is specialized just for one of these evolutionary operations. Furthermore, the data in each node are organized in the form of multisets of words (each word appears in an arbitrarily large number of copies), and all the copies are processed in parallel such that all the possible events that can take place do actually take place. Obviously, the computational process just described is not exactly an evolutionary process in the Darwinian sense. But the rewriting operations we have considered might be interpreted as mutations and the filtering process might be 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 [17].

In [13] one presents a characterization of the complexity class **NP** based on accepting networks of evolutionary processors (ANEP for short). This characterization is extended in [12] to **PSPACE** and **P**. The work [10] discusses how ANEPs can be considered as problem solvers. In [11], one shows that every recursively enumerable language can be accepted by an ANEP with 24 nodes. More precisely, one proposes a method for constructing, for every **NP**-language, an ANEP of size 24 deciding that language in polynomial time. While the number of nodes of this ANEP does not depend on the language, the other parameters of the network (rules, symbols, filters) depend on it.

From a computational point of view it is of interest to consider ANEPs with in-place computations, that is the length of every word in every node at any step in the computation is bounded by the length of the input word. This is our main reason to consider here some variants of networks of evolutionary processors without insertion nodes, called here accepting networks of non-inserting evolutionary processors, ANNIEP shortly. The differences between the variants of ANNIEPs consist in the filters and in the way of accepting the input word.

Besides accepting networks of evolutionary processors, generating networks of such processors have been investigated (see [2], [5], [14]). In the paper [6], the generative power of networks where only two types of point mutations are allowed for the nodes have been investigated. In case of non-inserting processors one only gets the set of all finite languages. This paper presents the counterpart for accepting networks, where the situation is completely different.

We study the computational power of accepting networks of non-inserting processors. In the case of filters defined by regular languages, both variants of accepting lead to the same class of languages, namely the class of context-sensitive languages. If random context conditions are used for defining filters, all linear context-free languages and some non-semilinear (even over the one-letter alphabet) can be accepted with both variants. Therefore the power of accepting networks is much greater than that of generating networks (both with non-inserting processors). Moreover, some closure properties of the classes of

languages accepted by ANNIEPs with filters defined by random context conditions are also discussed.

2 Some Notations and Definitions

Throughout the paper we assume that the reader is familiar with the basic notions of the theory of formal languages. We here only recall some notation and notions as they are used in the paper.

An *alphabet* is a finite and nonempty set of symbols. The cardinality of a finite set A is written $card(A)$. Any sequence of symbols from an alphabet V is called *word* over V. The set of all words over V is denoted by V^* and the empty word is denoted by ε. A language over V is a subset of V^*.

The length of a word x is denoted by $|x|$ while $alph(x)$ denotes the (with respect to inclusion) minimal alphabet W such that $x \in W^*$. A morphism $h : V^* \longrightarrow U^*$ is said to be *literal* if $|h(a)| = 1$ for all $a \in V$; it is *weak literal* if $|h(a)| \leq 1$ for all $a \in V$. In other words a (weak) literal morphism is called (weak) coding.

Let V be an alphabet. We say that a rule $a \to b$, with $a, b \in V \cup \{\varepsilon\}$ is a *substitution rule* if both a and b are not ε; it is a *deletion rule* if $a \neq \varepsilon$ and $b = \varepsilon$. The set of all substitution and deletion rules over an alphabet V are denoted by Sub_V and Del_V, respectively. Given a rule σ as above and a word $w \in V^*$, we define the following *actions* of σ on w:

- If $\sigma \equiv a \to b \in Sub_V$, then $\sigma^*(w) = \begin{cases} \{ubv : \exists u, v \in V^* \ (w = uav)\}, \\ \{w\}, \text{ otherwise} \end{cases}$

$$\sigma^r(w) = \begin{cases} \{ub : \ w = ua\}, \\ \{w\}, \text{ otherwise} \end{cases} \qquad \sigma^l(w) = \begin{cases} \{bv : \ w = av\}, \\ \{w\}, \text{ otherwise} \end{cases}$$

- If $\sigma \equiv a \to \varepsilon \in Del_V$, then $\sigma^*(w) = \begin{cases} \{uv : \exists u, v \in V^* \ (w = uav)\}, \\ \{w\}, \text{ otherwise} \end{cases}$

$$\sigma^r(w) = \begin{cases} \{u : \ w = ua\}, \\ \{w\}, \text{ otherwise} \end{cases} \qquad \sigma^l(w) = \begin{cases} \{v : \ w = av\}, \\ \{w\}, \text{ otherwise} \end{cases}$$

The action $\alpha \in \{*, l, r\}$ expresses the way of applying a substitution or deletion rule to a word, namely at any position ($\alpha = *$), in the left ($\alpha = l$), or in the right ($\alpha = r$) end of the word, respectively. For every rule σ, any action $\alpha \in \{*, l, r\}$, and any $L \subseteq V^*$, we define the *α-action of σ on L* by $\sigma^\alpha(L) = \bigcup_{w \in L} \sigma^\alpha(w)$. Given a finite set of rules M, we define the *α-action of M* on the word w and the language L by:

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

respectively.

If $\theta V^* \longrightarrow \{0,1\}$ is a predicate and $L \subseteq V^*$, we write:

$$\theta(L) = L \cap \theta^{-1}(1).$$

We are interested in some special predicates. For two disjoint subsets P and F of an alphabet V, a regular set R over V, and a word x over V, we define the predicates

$$\begin{aligned}
\theta^{s,P,F}(x) &= 1 \text{ if and only if } P \subseteq alph(x) \text{ and } F \cap alph(x) = \emptyset,\\
\theta^{w,P,F}(x) &= 1 \text{ if and only if } alph(x) \cap P \neq \emptyset \text{ and } F \cap alph(x) = \emptyset,\\
\theta^{R}(x) &= 1 \text{ if and only if } x \in R.
\end{aligned}$$

The first two predicates are based on *random context conditions* defined by the two sets P (*permitting contexts/symbols*) and F (*forbidding contexts/symbols*). Informally, the first condition requires (s stands for strong) that all permitting symbols are and no forbidding symbol is present in x, while the second (w stands for weak) is a weaker variant such that at least one permitting symbol appears in x but still no forbidding symbol is present in x. We call these two predicates random context predicates. The third predicate asks for membership in a regular set, and is called a regular predicate.

A *non-inserting evolutionary processor over* V is a tuple (M, φ, ψ), where:

- M is a set of either substitution or deletion rules over the alphabet V; formally, $M \subseteq Sub_V$ 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" in one evolutionary operation, only.
- φ is the *input predicate*, while ψ is the *output predicate* of the processor. Informally, these two predicates work as filters. A word w can enter or leave the processor, if it satisfies the predicate φ or ψ, respectively.

We are interested in two types of processors, random context non-inserting evolutionary processor over V (or short rcNIEP$_V$) and regular non-inserting evolutionary processor over V (or short regNIEP$_V$). These processors are defined by the requirement that,

- for an rcNIEP$_V$, both predicates are of the form $\theta^{s,P,F}$ or of the form $\theta^{w,P,F}$ for certain subsets P and F of V,
- for an regNIEP$_V$, both predicates are of the form θ^R for some regular set $R \subseteq V^*$.

We want to stress from the very beginning that the evolutionary processor we discuss here is a mathematical object only and the biological hints presented in the introduction are intended to explain in an informal way how some biological phenomena are *sources of inspiration* for our mathematical computing model. We denote the set of non-inserting evolutionary processors over V by $NIEP_V$.

An *accepting network of non-inserting evolutionary processors* (ANNIEP for short) is a 8-tuple $\Gamma = (V, U, G, N, \alpha, x_{In}, Out)$, where:

- V and U are the input and network alphabet, respectively, satisfying $V \subseteq U$.
- $G = (X_G, E_G)$ 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.
- $N : X_G \longrightarrow NIEP_V$ is a mapping which associates with each node $x \in X_G$ the evolutionary processor $N(x) = (M_x, \varphi_x, \psi_x)$.
- $\alpha : X_G \longrightarrow \{*, l, r\}$ is a mapping which associates with each node a type of action; $\alpha(x)$ gives the action mode of the rules of node x on the words existing in that node.
- $x_{In} \in X_G$ is the *input* node of Γ.
- $Out \subset X_G$ is the set of *output* nodes of Γ.

An ANNIEP is a random context ANNIEP or regular ANNIEP if all its non-inserting evolutionary processors are random context or regular non-inserting evolutionary processors, respectively.

We say that $card(X_G)$ is the size of Γ. A *configuration* of an ANNIEP Γ as above is a mapping $C : X_G \longrightarrow 2_f^{V^*}$ which associates a finite set of words with every node of the graph. A configuration may be understood as the sets of words which are present in any node (or in the associated prozessor) at a given moment. Given a word $z \in V^*$, the initial configuration of Γ on z is defined by $C_0^{(z)}(x_{In}) = \{z\}$ and $C_0^{(z)}(x) = \emptyset$ for all $x \in X_G \setminus \{x_{In}\}$.

A configuration can change either by an *evolutionary step* or by a *communication step*. When changing by 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 by a communication step, each node processor $x \in X_G$ sends one copy of each word it has, which is able to pass the output filter of x, to all the node processors connected to x and receives all the words 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

$$C'(x) = (C(x) - \psi_x(C(x))) \cup \bigcup_{\{x,y\} \in E_G} (\psi_y(C(y)) \cap \varphi_x(C(y))) \text{ for all } x \in X_G.$$

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

Let Γ be an ANNIEP, the computation of Γ on the input word $z \in V^*$ is a sequence of configurations $C_0^{(z)}, C_1^{(z)}, C_2^{(z)}, \ldots$, where $C_0^{(z)}$ is the initial configuration of Γ on z, $C_{2i}^{(z)} \Longrightarrow C_{2i+1}^{(z)}$ and $C_{2i+1}^{(z)} \vdash C_{2i+2}^{(z)}$, for all $i \geq 0$. Note that the configurations are changed by alternative steps. By the previous definitions, each

configuration $C_i^{(z)}$ is uniquely determined by the configuration $C_{i-1}^{(z)}$. A computation *halts* (and it 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 words 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 *language weakly (strongly) accepted* by Γ are defined as:

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

In the theory of networks some types of underlying graphs are common like *rings, stars, grids*, etc. Networks of evolutionary words processors, seen as language generating or accepting devices, with underlying graphs having these special forms have been considered in several papers, see, e.g., [14] for an early survey. We focus here on *complete* ANNIEPs i.e., ANNIEPs having a complete underlying graph. Therefore, in what follows we replace the graph G in the definition of an ANNIEP by the set of its nodes usually denoted by χ.

Moreover, we present an evolutionary network by its nodes x and the parameters corresponding to x, where instead of $\varphi^{\beta,PI_x,FI_x}$ and ψ^{β,PO_x,FO_x}, in case of random context processors, and instead of φ^{R_x} and $\varphi^{R'_x}$ for regular processors, we only mention $PI_x,\ FI_x,\ PO_x,\ FO_x, \beta$ and R_x, R'_x, β, respectively.

For $x \in \{wa, sa\}$ and $y \in \{rc, reg\}$, by $\mathcal{L}_x(yANNIEP)$ we denote the set of all languages which can be accepted by yANNIEPS.

The following two notions will be very useful in the sequel. If h is a one-to-one mapping from U to W and $\Gamma = (V, U, \chi, N, \alpha, x_{In}, Out)$ is an ANNIEP, then we denote by Γ_h the ANNIEP $\Gamma_h = (h(V), h(U), \chi, h(N), \alpha, x_{In}, Out)$, where by $h(N)$ we mean $h(N)(x) = (h(M_x), \varphi^{\beta,h(PI_x),h(FI_x)}, \psi^{\beta,h(PO_x),h(FO_x)})$ for every $x \in \chi$, provided that $N(x) = (M_x, \varphi^{\beta,PI_x,FI_x}, \psi^{\beta,PO_x,FO_x})$. Further, $h(a \to b) = h(a) \to h(b)$ for any evolutionary rule $a \to b$. Now, given two ANNIEPs $\Gamma_i = (V_i, U_i, \chi_i, N_i, \alpha_i, x_{In}^i, Out_i)$, $i = 1, 2$, $\chi_1 \cap \chi_2 = \emptyset$, we denote by $\Gamma_1 \sqcup \Gamma_2 = (V_1, U_1 \cup U_2, \chi_1 \cup \chi_2, N, \alpha, x_{In}^1, Out_2)$, where $\circ\,|_{\chi_i} = \circ_i$ for all $\circ \in \{N, \alpha\}$ and $i = 1, 2$.

3 Computational Power of Regular ANNIEPs

We start with a relation between the strong and weak acceptance modes.

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

Proof. Let $L \in \mathcal{L}_{wa}(regANNIEP)$. Then $L = L_{wa}(\Gamma)$ for some regular ANNIEP $\Gamma = (V, U, \chi, N, \alpha, x_{In}, Out)$. Let $N(x) = (M_x, \varphi^{R_x}, \psi^{R'_x})$ for a node x of

χ. Without loss of generality we may assume that $M_x = \emptyset$ for all $x \in Out$. We now construct the regular ANNIEP

$$\Gamma' = (V, U \cup \{Z\}, \chi \cup \{x_{Out}\}, N', \alpha', x_{In}, \{x_{Out}\}),$$

where $N'(x) = N(x)$ for $x \in \chi \setminus Out$, and

$$\begin{aligned} y : \{a \to Z \mid a \in U\},\ R_y,\ Z^*, \alpha'(y) = * \text{ for } y \in Out, \\ x_{Out} : \emptyset,\ Z^*,\ \emptyset,\ \alpha'(x_{Out}) = *. \end{aligned}$$

Obviously, if there is a non-empty node y of Out in some configuration of Γ, then y contains some word in some configuration of Γ', too. If this word is ε, then ε is not changed and sent to x_{Out}. If the word in y is non-empty, then all its letters are replaced by Z (note that it cannot leave the node as long as it still contains letters different than Z) and it is send to x_{Out}. Conversely, if a word eventually arrives in x_{Out}, then it contains only Z's which means that it was in a node from Out at some previous step. Thus Γ' accepts the same language as Γ does. Moreover, since the set of output nodes of Γ' is a singleton, we have $L_{wa}(\Gamma) = L_{wa}(\Gamma') = L_{sa}(\Gamma')$. □

Note that we have shown a stronger result than given in Theorem 1 because we have shown that the number of output nodes of an ANNIEP accepting in the weak mode can be decreased to one only.

We now compare the families of languages generated by ANNIEPs with the family of context-sensitive languages denoted here by $\mathcal{L}(CS)$.

Theorem 2. $\mathcal{L}(CS) \subseteq \mathcal{L}_{wa}(regANNIEP)$.

Proof. Let L be a context-sensitive language. Then $L = L(G)$ for some context-sensitive grammar $G = (N, T, P, S)$ in Kuroda normal form, i.e., all its rules are of the form $A \to a$, $A \to BC$ and $AD \to BC$ with $A, B, C, D \in N$ and $a \in T$. Let P' be the set of rules of the form $A \to BC$ and $AD \to BC$. For every $p \in P'$ with its right-hand side BC we set

$$\begin{aligned} R_p &= (N \cup T)^* \{B_p\} (N \cup T)^*, \\ R'_p &= (N \cup T)^* \{B_p C_p\} (N \cup T)^*, \\ R''_p &= (N \cup T)^* \{C_p\} (N \cup T)^* \end{aligned}$$

and $R = \bigcup_{p \in P'} R_p$. We construct the ANNIEP $\Gamma = (T, U, \chi, H, \alpha, x_{In}, \{x_{Out}\})$ with

$$\begin{aligned} U &= N \cup T \cup \{B_p, C_p \mid p = AD \to BC \text{ or } p = A \to BC\}, \\ \chi &= \{x_{In}, x_{Out}\} \cup \{p, p', p'' \mid p \in P'\}, \\ x_{In} &: M_{x_{In}},\ (N \cup T)^*,\ R,\ \alpha = * \\ M_{x_{In}} &= \{a \to A \mid A \to a \in P\} \cup \{B \to B_p \mid p = AD \to BC \text{ or } p = A \to BC\}, \\ p &: \{C \to C_p\},\ R_p,\ R'_p, \alpha = * \text{ for } p = AD \to BC \text{ or } p = A \to BC, \end{aligned}$$

$$p' : \{B_p \to A\},\ R'_p,\ R''_p,\ \alpha = * \text{ for } p = AD \to BC \text{ or } p = A \to BC,$$
$$p'' : \begin{cases} \{C_p \to D\},\ R''_p,\ (N \cup T)^*,\ \alpha = * \text{ for } p = AD \to BC, \\ \{C_p \to \varepsilon\},\ R''_p,\ (N \cup T)^*,\ \alpha = * \text{ for } p = A \to BC, \end{cases}$$
$$x_{Out} : \emptyset,\ \{S\},\ \{S\},\ \alpha = *.$$

The network simulates a derivation in G backwards. Let w be the input word; we claim that for any word $z \in (N \cup T)^+$ in x_{In} at any computation step we have that $z \Longrightarrow^* w$ in G. Initially, this assertion is true as w lies in x_{In}. Assume that a word $z \in (N \cup T)^+$ is in the node x_{In} at some step. If we apply a rule $a \to A$ to z, the new word remains in x_{In} and the assertion holds for this new word.

Now assume that we apply $B \to B_p$ to z for a rule $p = AD \to BC$. Then the obtained word $z' = z_1 B_p z_2$, where $z = z_1 B z_2$, is sent to the node p, where some C is replaced by C_p. If $B_p C_p$ is not a subword, then the word cannot go out from this node; moreover any word further obtained from this word can never go out from the node p. If $B_p C_p$ is a subword, the word is sent out to the node p', where B_p is replaced by A. This new word is sent out to p''. There C_p is either replaced by D, provided that $p = AD \to BC$, or deleted provided that $p = A \to BC$. Finally, the obtained word, say z', is sent to x_{In}. Altogether, we started with $z = vBCu$ and obtained $z' = vADu$, which implies that $z' \Longrightarrow z \Longrightarrow^* w$.

Moreover, since a word only reaches x_{Out}, if it is S, we infer that a word is weakly accepted by Γ if and only if it is generated by G. Thus $L_{wa}(\Gamma) = L(G)$. □

Theorem 3. $\mathcal{L}_{sa}(regANNIEP) \subseteq \mathcal{L}(CS)$.

Proof. For an ANNIEP $\Gamma = (V, U, \chi, N, \alpha, x_{In}, Out)$, we construct a linearly bounded automaton, which accepts $L_{sa}(\Gamma)$. We do not give a complete formal construction; we only give an informal description of the automaton and leave the details of the construction to the reader.

Let $r = card(Out)$. The automaton has r tapes, and on each tape it nondeterministically follows the itinerary of a copy of the input word. The states are vectors of size $2r$, each ith entry, $1 \leq i \leq r$, being associated with the node containing the word on the tape i, and each ith entry, $r + 1 \leq i \leq 2r$, being 0 or 1 that indicates whether the node associated with the $(i - r)$th entry has finished its task on the word on tape i (in this case the entry is 1) or not. Initially, all tapes contain the input word w, the first r entries of the initial states are associated with the input node x_{In}, and the last r entries are 0.

Let us now consider an arbitrary configuration of the automaton: the first r elements of the current state state are associated with the nodes $x_1, x_2, \ldots x_r$, the last r elements are 0, and on the i-th tape, $1 \leq i \leq r$, the word w_i stands. Now the automaton performs on each tape i the following actions:

- Changes the word w_i according to an application of a rule in M_{x_i}; let v_i be the result.
- Checks whether v_i can pass the output filter of x_i. In the non-affirmative case the automaton blocks the computation. In the affirmative case, the

automaton changes the ith entry of the state into an entry associated with the node y_i, which is a nondeterministically chosen node among the nodes of $\chi \setminus \{x_i\}$.
- Check whether v_i can pass the input filter of y_i. In the non-affirmative case the automaton blocks the computation. In the affirmative case, the $i+r$ entry becomes 1. From now on, no move is observed on the ith tape and no change is made for the entries i and $i+r$, until all the entries $r+1, r+2, \ldots, 2r$ are 1.
- Checks whether the state with the last r entries 1 has its first r entries associated with all output nodes of Γ. In the affirmative case the automaton accepts the input; otherwise it changes the last r entries into 0 and resumes the actions explained above.

It is rather plain that the automaton accepts $L_{sa}(\Gamma)$. Since in any evolutionary step one deletes or substitutes one letter, the length of the words on any tape is bounded by the length of the input word. Thus the workspace of this automaton is linearly bounded. □

By the Theorems 1, 2 and 3, we get immediately the following two statements.

Corollary 1

1. $\mathcal{L}_{wa}(regANNIEP) = \mathcal{L}_{sa}(regANNIEP) = \mathcal{L}(CS)$.
2. Every language in $\mathcal{L}_X(regANNIEP)$, $X \in \{wa, sa\}$, *can be weakly/strongly accepted by a regANNIEP* Γ *such that the action mode of every node of* Γ *is* $*$. □

4 Computational Power of Random Context ANNIEPs

We start with two statements that immediately follows from Theorems 1 and 3.

Theorem 4

1. $\mathcal{L}_{wa}(rcANNIEP) \subseteq \mathcal{L}_{sa}(rcANNIEP)$.
2. $\mathcal{L}_{sa}(rcANNIEP) \subseteq \mathcal{L}(CS)$. □

We do not know whether the second inclusion is proper or equality holds. Thus we give some further relations to other known language families inside $\mathcal{L}(CS)$ and some closure properties which give some more information about the classes $\mathcal{L}_{wa}(rcANNIEP)$ and $\mathcal{L}_{sa}(rcANNIEP)$.

Theorem 5

1. $\mathcal{L}_{wa}(rcANNIEP)$ *includes the class of linear context-free languages.*
2. $\mathcal{L}_{wa}(rcANNIEP)$ *contains non-semilinear languages.*

Proof. 1. Let $G = (N, T, S, P)$ be a linear context-free grammar; without loss of generality we may assume that the following conditions hold:

– Every rule in P is of one of the following three forms: $A \to aB$, $A \to Ba$, $A \to a$, where $A, B \in N$ and $a \in T$,

– If both rules $A \to aC$ and $B \to Db$ belong to P, then $A \neq B$,

– The set of nonterminals N of G is $\{A_1, A_2, \ldots, A_n\}$ for some $n \geq 1$ and $S = A_1$,

– There is no rule $A \to aA$ or $A \to Aa$ for any $A \in N$ and $a \in T$.

We construct the following ANNIEP with the input alphabet T, the working alphabet $U = T \cup \{a_i, a_i' \mid 1 \leq i \leq n\} \cup \{Z\}$, and only one output node x_{Out}.

$$x_{In} : \begin{cases} M = \{a \to a_1 \mid a \in T\}, \\ PI = T, FI = \{a_i \mid a \in T, 1 \leq i \leq n\}, \\ PO = \emptyset, FO = T, \\ \alpha = *,\ \beta = w, \end{cases} \quad x_{Out} : \begin{cases} M = \emptyset, \\ PI{=}\{Z\}, FI{=}U \setminus \{Z\}, \\ PO = U, FO = \emptyset, \\ \alpha = *,\ \beta = s, \end{cases}$$

If there exists $A_i \to aA_j \in P$ for some $a \in T$ and $1 \leq j \neq i \leq n$, then the node x_i is defined by

$$x_i : \begin{cases} M = \{a_i \to a_j' \mid A_i \to aA_j \in P\}, \\ PI = \{a_i \mid a \in T\}, FI = U \setminus \{a_i \mid a \in T\}, \\ PO = \{a_j' \mid a \in T, 1 \leq j \neq i \leq n\}, FO = \emptyset, \\ \alpha = l,\ \beta = w, \end{cases}$$

If there exists $A_i \to A_j a \in P$ for some $a \in T$ and $1 \leq j \neq i \leq n$, then the node x_i is defined by

$$x_i : \begin{cases} M = \{a_i \to a_j' \mid A_i \to aA_j \in P\}, \\ PI = \{a_i \mid a \in T\}, FI = U \setminus \{a_i \mid a \in T\}, \\ PO = \{a_j' \mid a \in T, 1 \leq j \neq i \leq n\}, FO = \emptyset, \\ \alpha = r,\ \beta = w, \end{cases}$$

Moreover, we set

$$x_i' : \begin{cases} M = \{a_j \to a_i \mid a \in T, 1 \leq j \neq i \leq n\}, \\ PI = \{a_i' \mid a \in T\}, FI = \emptyset, \\ PO = \{a_i \mid a \in T\}, FO = \{a_j \mid a \in T, 1 \leq j \neq i \leq n\}, \\ \alpha = *,\ \beta = w, \end{cases} \quad \text{for} \quad 1 \leq i \leq n,$$

$$\bar{x}_i : \begin{cases} M = \{a_i' \to \varepsilon \mid a \in T\}, \\ PI = \{a_i' \mid a \in T\}, FI = \{a_j \mid a \in T, 1 \leq j \neq i \leq n\}, \\ PO = \{a_i \mid a \in T\}, FO = \{a_i' \mid a \in T\}, \\ \alpha = l,\ \beta = w, \end{cases} \quad \text{for} \quad 1 \leq i \leq n,$$

$$\tilde{x}_i : \begin{cases} M = \{a_i' \to \varepsilon \mid a \in T\}, \\ PI = \{a_i' \mid a \in T\}, FI = \{a_j \mid a \in T, 1 \leq j \neq i \leq n\}, \\ PO = \{a_i \mid a \in T\}, FO = \{a_i' \mid a \in T\}, \\ \alpha = r,\ \beta = w, \end{cases} \quad \text{for} \quad 1 \leq i \leq n,$$

$$y : \begin{cases} M = \{a_i \to Z \mid A_i \to a \in P, a \in T, 1 \leq i \leq n\}, \\ PI = \bigcup_{i=1}^{n} \{a_i \mid a \in T\}, FI = U \setminus (\bigcup_{i=1}^{n} \{a_i \mid a \in T\}), \\ PO = \{Z\}, FO = \emptyset, \\ \alpha = r,\ \beta = w, \end{cases}$$

The general idea of this construction is that for every $1 \leq i \leq n$, the following statement holds:

Fact: *If* $S \Longrightarrow^t uA_iv \Longrightarrow^+ uwv = z$ *for some* $t \geq 0$, *with* $|z| = m$, *then* $h_i(w) \in (C^{(z)}_{2m(t+1)+2t}(x_i) \cap C^{(z)}_{2m(t+1)+2t}(y))$, *where* h_i *is a literal morphism from* T *to* $\{a_i \mid a \in T\}$ *defined by* $h(a) = a_i$ *for any* $a \in T$.

This fact can be proved by a standard induction argument on t. Now, if $t = m - 1$, then w is reduced to a letter from T, say a, therefore after the word a_i is transformed into Z in the node y, it arrives in x_{Out} and the computation halts successfully. This means that z is accepted by the network.

On the other hand, if $C^{(z)}_0, C^{(z)}_1, C^{(z)}_2, \ldots, C^{(z)}_p$ is an accepting computation on z and $h_i(w) \in (C^{(z)}_t(x_i) \cap C^{(z)}_t(y))$ for some $t < p$, then the derivation $S \Longrightarrow^* uA_iv \Longrightarrow^+ uwv = z$ holds in G, which concludes the proof of the first statement of the theorem.

2. The network with the nodes defined by:

$$x_{In} : \begin{cases} M = \{a \to \bar{a}\}, \\ PI = \{a\}, FI = \{\bar{a}, \tilde{a}\}, \\ PO = \{\bar{a}\}, FO = \emptyset, \\ \alpha = *,\ \beta = s, \end{cases} \quad x_1 : \begin{cases} M = \{a \to \tilde{a}\}, \\ PI = \{\bar{a}\}, FI = \{\tilde{a}\}, \\ PO = \{\tilde{a}\}, FO = \emptyset, \\ \alpha = *,\ \beta = s, \end{cases}$$

$$x_2 : \begin{cases} M = \{\bar{a} \to \varepsilon\}, \\ PI = \{\bar{a}, \tilde{a}\}, FI = \emptyset, \\ PO = \{\tilde{a}\}, FO = \emptyset, \\ \alpha = *,\ \beta = s, \end{cases} \quad x_3 : \begin{cases} M = \{\tilde{a} \to a'\}, \\ PI = \{\tilde{a}\}, FI = \{\bar{a}\}, \\ PO = \{a'\}, FO = \{\tilde{a}\}, \\ \alpha = *,\ \beta = s, \end{cases}$$

$$x_4 : \begin{cases} M = \{a' \to a\}, \\ PI = \{a'\}, FI = \{a, \bar{a}, \tilde{a}\}, \\ PO = \{a\}, FO = \{a'\}, \\ \alpha = *,\ \beta = s, \end{cases} \quad x_{Out} : \begin{cases} M = \emptyset, \\ PI = \{\bar{a}\}, FI = \{a, \bar{a}, \tilde{a}\}, \\ PO = \{\bar{a}\}, FO = \emptyset, \\ \alpha = *,\ \beta = s, \end{cases}$$

weakly accepts the non-semilinear language $\{a^{2^n} \mid n \geq 0\}$. Indeed, the computation of this netwok on every input is divided in two phases. In the first phase, the input word looses one occurrence of a and changes another one to a' by visiting the nodes x_{In}, x_1, x_2, x_3. This process resumes until no occurrence of a is observed in the current word. There are three possiblities: (1) it contains only a's, (2) it contains only a's excepting an occurrence of $\bar{a}$, (3) it equals $\bar{a}$. Now the second phase of the computation starts. In the first case, the word enters x_4 where all a's are transformed into original a's and the first phase resumes from x_{In} with a word that is exactly twice shorter than the word present in the input node in the beginning of the previous first phase. In this case, we have checked whether the length of that word was an even number. In the second case listed above, the computation cannot continue anymore, hence the network will eventually halt without accepting. In the third case, the computation halts accepting the input word. This means that the length of the input word could be divided iteratively by 2 until the result was one, hence the length of the input word was a power of 2. □

Theorem 6
1. The class $\mathcal{L}_{wa}(rcANNIEP)$ is closed under boolean union, literal morphism, inverse weak literal morphism, mirror image.
2. The class $\mathcal{L}_{sa}(rcANNIEP)$ is closed under boolean intersection, literal morphism, inverse weak literal morphism, concatenation, mirror image.

Proof. 1. We give an informal proof for union that can be easily formalized by the reader. Let Γ_1 and Γ_2 be to ANNIEPs; we construct a new ANNIEP Γ that contains three subnetworks. In the input node of the first subnetwork, an arbitrary symbol of the input word is substituted by either its primed copy or its barred copy. All words 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 words 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 word. The word containing primed symbols only is given as an input word to the subnetwork formed from Γ_1 modified accordingly. The other word is processed analogously by the subnetwork formed from Γ_2 modified accordingly. The set of output nodes of Γ is the union of the sets of output nodes of Γ_1 and Γ_2 modified accordingly. Clearly, $L_{wa}(\Gamma) = L_{wa}(\Gamma_1) \cup L_{wa}(\Gamma_2)$.

If $h : V \longrightarrow U$ is a literal morphism and Γ is an ANNIEP with the input alphabet V, then let Γ' be the ANNIEP with the input alphabet U formed by two subnetworks as follows. In the input node of the first subnetwork, each symbol b of the input word 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 word were substituted, all the words obtained are sent to the input node of the subnetwork formed from Γ modified accordingly. It is plain that $h(L_{wa}(\Gamma)) = L_{wa}(\Gamma')$. The construction for the closure under inverse weak literal morphism is pretty similar and left to the reader.

The closure under mirror image follows pretty simple; it suffices to interchange all the action modes l and r of the nodes.

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

It is known that every recursively enumerable language can be written as the image of the intersection of two linear languages through a weak literal morphism. Therefore, the following statement is a consequence of the second statement of Theorem 4 and Theorem 6:

Corollary 2
1. Every recursively enumerable language is the weak literal morphic image of a language in $\mathcal{L}_{sa}(ANNIEP)$.
2. $\mathcal{L}_{sa}(ANNIEP)$ is not closed under weak literal morphism. □

5 Final Remarks

As we showed in this note, the computational power of ANNIEPs is very different than that of generating networks of non-inserting processors. The role of

evolutionary operations in generating networks of evolutionary processors, that is generating networks with nodes specialized in all three evolutionary operations, in two operations out of these three and in only one operation, was considered in [1]. A similar investigation on ANEPs has already started.

References

1. Alhazov, A., Dassow, J., Rogozhin, Y., Truthe, B.: Personal communication
2. Castellanos, J., Martín-Vide, C., Mitrana, V., Sempere, J.: Networks of evolutionary processors. Acta Informatica 38, 517–529 (2003)
3. Csuhaj-Varj, E., Salomaa, A.: Networks of parallel language processors. In: Păun, G., Salomaa, A. (eds.) New Trends in Formal Languages. LNCS, vol. 1218, pp. 299–318. Springer, Heidelberg (1997)
4. Csuhaj-Varj, E., Mitrana, V.: Evolutionary systems: a language generating device inspired by evolving communities of cells. Acta Informatica 36, 913–926 (2000)
5. Csuhaj-Varjú, E., Martín-Vide, C., Mitrana, V.: Hybrid NEPs are computationally complete. Acta Informatica 41, 257–272 (2005)
6. Dassow, J., Truthe, B.: On the power of networks of evolutionary processors. In: Durand-Lose, J., Margenstern, M. (eds.) MCU 2007. LNCS, vol. 4664, pp. 158–169. Springer, Heidelberg (2007)
7. Errico, L., Jesshope, C.: Towards a new architecture for symbolic processing. In: Artificial Intelligence and Information-Control Systems of Robots, vol. 94, pp. 31–40. World Scientific, Singapore (1994)
8. Fahlman, S., Hinton, G., Seijnowski, T.: Massively parallel architectures for AI: NETL, THISTLE and Boltzmann Machines. In: Proc. AAAI National Conf. on AI, pp. 109–113. William Kaufman, Los Altos (1983)
9. Hillis, W.: The Connection Machine. MIT Press, Cambridge (1985)
10. Manea, F., Martín-Vide, C., Mitrana, V.: On the size complexity of universal accepting hybrid networks of evolutionary processors. Mathematical Structures in Computer Science 17, 753–771 (2007)
11. Manea, F., Mitrana, V.: All NP-problems can be solved in polynomial time by accepting hybrid networks of evolutionary processors of constant size. Information Processing Letters 103, 112–118 (2007)
12. Manea, F., Margenstern, M., Mitrana, V., Perez-Jimenez, M.: A new characterization of **NP**, **P**, and **PSPACE** with accepting hybrid networks of evolutionary processors (submitted)
13. Margenstern, M., Mitrana, V., Perez-Jimenez, M.: Accepting hybrid networks of evolutionary systems. In: Ferretti, C., Mauri, G., Zandron, C. (eds.) DNA 2004. LNCS, vol. 3384, pp. 235–246. Springer, Heidelberg (2005)
14. Martín-Vide, C., Mitrana, V.: Networks of evolutionary processors: results and perspectives. In: Molecular Computational Models: Unconventional Approaches, pp. 78–114. Idea Group Publishing, Hershey (2005)
15. Păun, G., Sntean, L.: Parallel communicating grammar systems: the regular case. Annals of University of Bucharest, Ser. Matematica-Informatica 38, 55–63 (1989)
16. Păun, G.: Computing with membranes. Journal of Computer and System Sciences 61, 108–143 (2000)
17. Sankoff, D., et al.: Gene order comparisons for phylogenetic inference: evolution of the mitochondrial genome. In: Proceedings of the National Academy of Sciences of the United States of America, vol. 89, pp. 6575–6579 (1992)