NETWORKS OF EVOLUTIONARY PROCESSORS AS NATURAL LANGUAGE PARSERS

Gemma Bel-Enguix, M. Dolores Jiménez-López, Robert Mercaş and Alexander Perekrestenko
GRLMC, Rovira i Virgili University, Pl. Imperial Tàrraco 1, Tarragona, Spain

Keywords: Networks of Evolutionary Processors, Natural Language Processing, Parsing.

Abstract: Networks of Evolutionary Processors (NEPs) introduced in Castellanos et al. (2001) are a new computing mechanism directly inspired from the behaviour of cell populations. In the paper, we explore the possibility of using Networks of Evolutionary Processors (NEPs) for modelling natural language an entity generated in parallel by a modular architecture and specially syntax a modular device of specialized processors inside the modular construct of language. An implementation of NEPs for parsing of simple structures is suggested. Moreover, we introduce the concepts of parallel processing and linearity in the formalization of NEPs as accepting devices, and suggest a new line of research by applying these networks to natural language.

1 INTRODUCTION

Networks of Evolutionary Processors (NEPs) are a new computing mechanism directly inspired from the behaviour of cell populations. Every cell is described by a set of words (DNA) evolving by mutations, which are represented by operations on these words. At the end of the process, only the cells with correct strings will survive. In spite of the biological inspiration, the architecture of the system is directly related to the Connection Machine (Hillis, 1985) and the Logic Flow paradigm (Errico and Jessope, 1994). Moreover, the global framework for the development of NEPs has to be completed with the biological background of DNA computing (Păun et al., 1998), membrane computing (Păun, 2000) – that focalizes also in the behaviour of cells –, and specially with the theory of grammar systems (Csuhaj-Varjú et al., 1994), which share with NEPs the idea of several devices working together and exchanging results.

First precedents of NEPs as generating devices can be found in (Csuhaj-Varjú and Salomaa, 1997) and (Csuhaj-Varjú and Mitrana, 2000). The topic was introduced in (Castellanos et al., 2001), and further developed in (Martín-Vide et al., 2003; Castellanos et al., 2003; Castellanos et al., 2005).

NEPs can be defined as a graph whose nodes are processors performing some very simple operations on strings and sending the resulting strings to other nodes. Every node has filters that block some strings from being sent and/or received. This functioning allows the specialization of each processor, which is an interesting feature for natural language processing.

In this paper, we propose to use Networks of Evolutionary Processors (NEPs) as a modular description device in linguistics. The idea of using HNEPs as recognizers is not original. A preliminary approach to accepting NEPs was already introduced in (Margenstern et al., 2004) and developed in a series of contributions (Castellanos et al., ; Manea, 2004; Manea and Mitrana, 2007). Nevertheless, this is the first attempt to deal with natural language processing issues from a NEPs perspective.

As was shown in (Castellanos et al., 2003), HNEPs with regular filters are Turing-equivalent. This fact suggests that they can be used as a formal base for a "programming language" for processing natural language information on all levels. In this context, (H)NEPs have the following advantages:

- the basic structures of NEPs –nodes, filters and the graph-based structure– can be used in a very natural way for modelling different kinds of linguistic issues;
- a HNEP simulator can be easily implemented as a software environment for constructing language-processing modules, and in particular, syntactic parsers;
- we can specify the output format of such a parser according to the needs of the setting for which it is being developed.

In this paper, we introduce a formalization of

Bel-Enguix G., Jiménez-López M., Mercaş R. and Perekrestenko A. (2009).
NETWORKS OF EVOLUTIONARY PROCESSORS AS NATURAL LANGUAGE PARSERS.
In *Proceedings of the International Conference on Agents and Artificial Intelligence*, pages 619-625
DOI: 10.5220/0001809306190625

NEPs suitable for natural language purposes. We illustrate the applicability of this framework for linguistic processing with an example of a NEP that recognizes simple syntactic structures using linear input and output.

The paper is organized as follows. Section 2 presents the general definition of NEPs. In section 3, some of the main features of NEPs are discussed. In section 4, a new variant of NEPs for sentence recognition is introduced. An example of the functioning of this new model is shown in section 5. Finally, section 6 is devoted to conclusions and directions for future work.

2 NEPs: DEFINITION

Following (Castellanos et al., 2003) we introduce the basic definition of NEPs.

Definition 1. *A Network of Evolutionary Processors of size n is a construct:*

$$\Gamma = (V, N_1, N_2, ..., N_n, G),$$

where:

- *V is an alphabet and for each $1 \leq i \leq n$,*
- *$N_i = (M_i, A_i, PI_i, PO_i)$ is the i-th evolutionary node processor of the network. The parameters of every processor are:*
  - *M_i is a finite set of evolution rules of one of the following forms only:*
    - *i. $a \rightarrow b$, where $a, b \in V$ (substitution rules),*
    - *ii. $a \rightarrow \varepsilon$, where $a \in V$ (deletion rules),*
    - *iii. $\varepsilon \rightarrow a$, where $a \in V$ (insertion rules).*
  - *A_i is a finite set of strings over V. The set A_i is the set of initial strings in the i-th node.*
  - *PI_i and PO_i are subsets of V^* representing the input and the output filter, respectively. These filters are defined by the membership condition, namely a string $w \in V^*$ can pass the input filter (the output filter) if $w \in PI_i(w \in PO_i)$.*
- *$G = (\{N_1, N_2, \ldots, N_n\}, E)$ is an undirected graph called the underlying graph of the network. The edges of G, that is the elements of E, are given in the form of sets of two nodes. The complete graph with n vertices is denoted by K_n.*

A configuration of a NEP is an n-tuple $C = (L_1, L_2, \ldots, L_n)$, with $L_i \subseteq V^*$ for all $1 \leq i \leq n$. It represents the sets of strings which are present in any node at a given moment.

A given configuration of a NEP can change either by an evolutionary step or by a communicating step. When changing by an evolutionary step, each component L_i of the configuration is changed in accordance with the evolutionary rules associated with the node i. The change in the configuration by an evolutionary step is written as $C_1 \Rightarrow C_2$.

When changing by a communication step, each node processor N_i sends all copies of the strings it has, able to pass its output filter, to all the node processors connected to N_i and receives all copies of the strings sent by any node processor connected with N_i, if they can pass its input filter. The change in the configuration by a communication step is written as $C_1 \vdash C_2$.

3 NEPs FOR MODELLING NATURAL LANGUAGE

This formal construct can provide a good framework for attempting a new description and formalization of natural language. Three features of NEPs are crucial for their application to language processing and, especially, to parsing technologies. NEPs are *specialized*, *modular* communicating systems that work in *parallel*.

NEPs are *modular* devices because they can be described as distributed systems of contributing nodes, each one of them carrying out just one type of operation. Every node should be defined depending on the specific domain we aim to tackle. Moreover, the processors of the network are *specialized*, since each one is designed for a specific task, in a way that the final success of the system depends on the correct working of every agent and on the correct interaction between them.

It is a commonplace belief in cognitive science that complex computational systems are at least weakly decomposable into components (Fodor, 1983). In general, modular theories in cognitive science propose a number of independent but interacting cognitive 'modules' that are responsible for each cognitive domain.

The theory of modularity is also present in linguistic approaches. In fact, the modular approach to grammar has been shown to have important consequences for the study of language (cf. (Sadock, 1991)). This has led many grammatical theories to use modular models. The idea of having a system made up of several independent components (syntax, semantics, phonology, morphology, etc.) seems to be a good choice to account for linguistic issues.

Several authors have defended as well internal modularity in the different dimensions of grammar (Everaert et al., 1988; Harnish and Farmer, 1984; Weinberg, 1987). In (Crocker, 1991), for example,

a highly modular organization of *syntax* is suggested where modules are determined by the representations they recover.

Communication is the feature that accounts for the social competences of modules. By means of communication, agents can interact to achieve common goals, work for their own interests, or even isolate. Although different processes and operations are done in parallel, NEPs also need to define some type of *coordination* between nodes, since alternative steps of *communication* have to be synchronized. *Nodes communication* is said to be:

- *graph-supported*, and
- *filter-regulated*.

We say that the communication among nodes is *graph-supported* because the edges of the graph govern its social interactions. Therefore, if there are non-connected nodes in a NEP, these nodes do not communicate with other processors.

The fact that the communication is *filter-regulated* means that it is driven by means of input and output filters.

- The goal of the *input filter* is to control the information or structures entering the node. This helps the specialization, by the selection of the strings/structures the node can process, and protects the module from possible harmful items.
- With its *output filter* the node selects the information it wants to share and also when it wants to share it.

In what refers to the functioning of NEPs, the main feature to be highlighted is the *parallelism*. By means of parallelism, different tasks can be performed at the same time by different processors. Some of linguistic processes, as well as language generation in general, are considered to be parallel. For apparently sequential interactions (i.e. dialogue) parallelism allows working with multi-modal exchanges.

Therefore, taking the four main modules usually considered in linguistics – syntax, semantics, phonology, morphology – and considering the edges of the underlying graph as a way for communicating or not communicating, we can draw the simple scheme of a "language generation NEP" that is shown in Figure 1. *Ph* stands for phonetics, *M* represents morphology, *Sy* is syntax and *Se* refers to semantics. The semantic node only communicates with the phonological and the syntactic ones as there does not seem to be any interaction between semantics and phonetics.

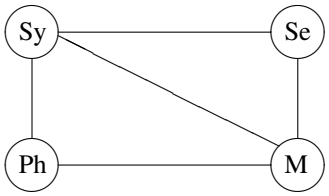

Figure 1: Modular Linguistic NEP.

4 NEPs FOR PARSING

In the sequel we will try to make some modifications in the computational definition of NEPs for them to work as parsers of natural language. For the sake of simplicity, we establish a methodological restriction: we will focus on simple sentences with the shape $[S\ V\ O]$, where $S \rightarrow [NP]$, $O \rightarrow [NP]$, that is, sentences with the form $[[NP]\ V\ [NP]]$. Our purpose is to check whether this mechanism is powerful and simple enough to be used as a model for parsing complex linguistic strings.

As it has been already highlighted, the model hereby presented takes advantage of the main features of NEPs: *modularity*, *specialisation* and *parallelism*. Adopting these characteristics in the modelling of our device one can improve its efficiency and decrease the complexity.

Modularity and specialisation can be useful because they allow designing processors which only accept, recognize and label a single type of syntactic phrases or functions. Such strategy makes easier the first work of classifying the lexical pieces according to their grammatical category. By parallelism, all lexical items will be taken and analyzed at the same time, and afterwards, grammatical units can be packed in different processors.

In general, the system has to perform two main tasks: a) to recognize correct strings, like an automaton could do, and b) to provide an output with labelled elements that give account of the syntactic structure of the input sentence.

To be able to accept and analyze a sentence, the NEP has to perform the following steps: 1) to recognize every word, 2) to make a map of its linguistic features, 3) to gather the non-terminal units in phrases establishing their beginning and end, and finally 4) to give a linear structural version of the whole sentence.

In order to implement such a NEP, we propose several types of specialized nodes:

- nodes specialized in accepting lexical items recognizing their grammatical categories;
- nodes specialized for accepting and operating with different types of phrases, i.e., nominal

phrases (NP), prepositional phrases (PP);

- nodes for accepting sentences.

The structure of the graph is given by the class of sentences to be processed. Many different NEPs could be designed for the parsing of the same type of syntactic structures, but we are looking for the best NEP for every structure in terms of number of nodes and computational efficiency.

Moreover, since the structure we are working with has just two types of sub-structures, namely *NP* and V, at least three specialized nodes are needed, one for sub-components of *NP* (just one element in a minimal *NP*), one for the recognition and labelling of *NP* and the other one for the recognition and analysis of V. The sub-components of *NP* include nodes for lexical units – N for a noun and *ART* for an article. Since a node for packing the final output is also necessary, a graph with at least four nodes has to be designed.

Our device will consist of the following nodes: a) a node for recognizing articles, *ART* b) three processors for labelling nouns Nc, Np, Nv, c) a node for recognizing nominal phrases, *NP*, d) a node for analyzing verbal structures, V, e) a node for analyzing nominal phrases, *NP* and f) a node for labelling sentences, P. Beside these nodes we will also need an output node.

In the input filter of specialized nodes, the only elements accepted will be those that can be part of phrases they can recognize. In the output filter of these nodes, only labelled phrases will be allowed to pass and be sent to the other filters.

To perform the recognition process, two types of alphabets are necessary: V, the alphabet of the input symbols, which are terminal strings, i.e., lexical items, and Σ the alphabet of grammatical types symbols – which correspond to grammatical categories – plus feature symbols. For the simple sentences we are dealing with, we recognize several strings symbols belonging to Σ^* which are needed to process the sentence: $[\,]_N$, $[\,]_V$, $[\,]_{ART}$, $[\,]_{NP}$.

In order to model the subject-verb agreement, some of these symbols will be provided with morphological markers. First of all, two different marks will be established for the category $[V]$ in order to distinguish between the two different forms of the English verb in present: s stands for the general form, and p for the third person. In this way, when the node receives a lexical item x, it analyzes it, and inserts the grammatical types symbols giving us $[x]_{vs}$ or $[x]_{vp}$.

In order to fulfill the agreement with the verb, $[N]$ has to be recognized with the same parameters as the verb and moreover with some that give us the agreement with the article, $\{c, p, v\}$. On the other hand, in order to model agreement inside a phrase, we introduce separate nodes for the same phrase with different morphological characteristics.

For distinguishing the article "a" from the articles "an" and "the", the feature $[ART]$ will be $[\,]_{a1}$ for "a", $[\,]_{a2}$ for "an" and $[\,]_{a3}$ for "the", where the absence of any symbol means it works for both singular and plural. If the agreement is not accomplished inside *NP* or between *NP* at the left of the verb and the verb itself, then the sentence will not be recognized.

For delimiting the phrases as a group of several elements belonging to Σ, and sentences as a group of phrases, we introduce in our NEP a rule that will isolate these symbols from the ones belonging to the alphabet V.

With the elements we have just explained, a NEP for sentence analysis can be defined as follows:

Definition 2. *A NEP for the analysis of simple sentences $[[NP]V[NP]]$ is a general structure:*

$$\Gamma = (V, \Sigma, \{ART, N_v, N_c, N_p, V, NP, P, Out\}, G)$$

where:

- *V is the input vocabulary,*
- *Σ is the grammatical type vocabulary,*
- *$\{ART, N_v, N_c, N_p, V, NP, P, Out\}$ are the node processors $N_1, N_2, \ldots, N_7$, Out of the network with the following definition. For every node $N_i = (M_i, A_i, PI_i, PO_i)$:*
  - *M_i is the finite set of evolution rules of the form:*
    - *i. $a \to [a]_n$, where $a \in \{V \cup \Sigma\}^*$ (*insertion rule *with n indicating the indexes of the node),*
    - *ii. $a \to \varepsilon$, where $a \in V$ (*deletion rule *in which all elements of V are erased), or*
    - *iii. $a, b \to [ab]_n$, where $a, b \in V \cup \Sigma$ (adjunction rule in which two elements coming from different nodes are wrapped together);*
  - *A_i is the set of strings over V in the initial configuration,*
  - *PI_i are the input filters over $\{V \cup \Sigma\}^*$, and*
  - *PO_i are the output filters over $\{V \cup \Sigma\}^*$.*
- *Out is the output node that has a special input filter that compares the initial phrase shuffled with Σ^*, with the words that are trying to enter the node ($Inp \sqcup\!\sqcup \Sigma^* = inputword$)*
- *$G = (V, Ev)$ is the network graph where:*
  - *$V = \{N_1, N_2, \ldots, N_7, Out\}$ are its nodes, and*
  - *$Ev = (N_1N_6, N_2N_6, N_3N_6, N_4N_6, N_5N_6, N_6N_7, N_7Out)$ are the arcs.*

The computation works almost like in a regular *NEP*, combining evolutionary steps and communication steps. Moreover, the system is totally parallel, even in the input mechanism, and every node applies, during evolutionary steps, as many rules as it can.

The system stops when no operation can be performed by the *NEP* or the *Out* node receives an input. In the latter case, the recognition process ends, the initial phrase is accepted as correct and its structure is displayed.

5 AN EXAMPLE

In this section, a *NEP* will be implemented for the recognition of sentences $[[NP]V[NP]]$. The example we use is the sentence *The boy eats the apple*.

Our *NEP* contains eight nodes. The alphabet Σ consists of the symbols $\{[,],1,2,3,a,c,f,n,p,s,v\}$ that indicate the type of the structure.

The general definition of the system is as follows:

$$\Gamma = (V, \Sigma, \{ART, N_v, N_c, N_p, V, NP, P, Out\}, G)$$

where:

- $V = \{apple, apples, boy, boys, eat, eats, a, an, the\}$
- $\Sigma = \{[,],1,2,3,a,c,f,n,p,s,v\}$
- $Out = \{\{(Inp \sqcup\!\sqcup x) \rightarrow x$, where $x \in \Sigma^*\}, \emptyset, \{The\ boy\ eats\ an\ apple\} \sqcup\!\sqcup \Sigma^*, \Sigma^*\}$
- $ART = \{(a \rightarrow [a]_{a1}, an \rightarrow [an]_{a2}, the \rightarrow [the]_{a3}), \{a, an, the\}, \emptyset, \{[a]_{a1}, [an]_{a2}, [the]_{a3})\}\}$
- $N_v = \{(apple \rightarrow [apple]_{nv}), \{apple\}, \{[(V \cup \Sigma^{)}*]_{nv}\}\}, \{[(V \cup \Sigma^{)}*]_{nv}\}\}$, that is, all third person singular nouns starting with a vowel.
- $N_c = \{(boy \rightarrow [boy]_{nc}), \{boy\}, \{[(V \cup \Sigma)^*]_{nc}, \{[(V \cup \Sigma)^*]_{nc}\}\}$, that is, all third person singular nouns starting with a consonant.
- $N_p = \{(apples \rightarrow [apples]_{np}, boys \rightarrow [boys]_{np}), I \rightarrow [I]_{np}, you \rightarrow [you]_{np}, \{apples, boys\}, \{[(V \cup \Sigma)^*]_{np}, \{[(V \cup \Sigma)^*]_{np}, [I]_{np}, [you]_{np}\}\}$, that is all plural nouns and singular ones that are not third person.
- $V = \{(eats \rightarrow [eats]_{vs}, eat \rightarrow [eat]_{vp}), \{eat, eats\}, \{[(V \cup \Sigma)^*]_{vs}, [(V \cup \Sigma)^*]_{vp}\}, \{[(V \cup \Sigma)^*]_{vs}, [(V \cup \Sigma)^*]_{vp}\}\}$
- $NP = \{([x]_{ax_1}[y]_{ny_1} \rightarrow [[x]_{ax_1}[y]_{ny_1}]_{fy_1}, ([y]_{np} \rightarrow [[y]_{np}]_{fp})), \emptyset, (V \cup \Sigma)^*, \{(V \cup \Sigma)^* \setminus \{[[V \cup \Sigma]_{a2}[V \cup \Sigma]_{n\{c,p\}}]_{\Sigma^*}, [[V \cup \Sigma]_{a1}[V \cup \Sigma]_{n\{v,p\}}]_{\Sigma^*}\}\}\}$
- $P = \{([x]_{nx_1}[y]_{vx_1}[z]_{nz_1} \rightarrow [[x]_{nx_1}[y]_{vx_1}[z]_{nz_1}]_{x_1}), \emptyset, ([V \cup \Sigma)^*]_{fx}[V \cup \Sigma)^*]_{vx}[V \cup \Sigma)^*]_{fy}, (V \cup \Sigma)^*\}$
- $G = (V, Ev)$
  - $V = \{ART, N_v, N_c, N_p, V, NP, P, Out\}$
  - $Ev = (ART\ NP,\ N_vNP,\ N_cNP,\ N_pNP,\ V\ P,\ NP\ P,\ P\ Out)$

Here $x, y, z, x_1, y_1, z_1 \in \Sigma \cup V$.

In order to understand how our example program works we will "run" it on two different sentences. The first one will be the phrase: "The boy eats an apple". Before the first computation step, *ART* will contain the words *the* and *an*, *Nc* will contain the words *boy* and *apple*, and *V* the verb *eats*. The second step is a communication one. The third step, which is a computation step, will produce in *NP* the combinations $\{[the]_{a3}[boy]_{nc}, [an]_{a2}[boy]_{nc}, [the]_{a3}[apple]_{nv}, [an]_{a2}[apple]_{nv}\}$. Using the output filter, *NP* is able to exclude from the next computation the second and the third component. Hence, after the fifth computation step we will have in *P* the string set $\{[[the]_{a3}[boy]_{nc}]_{fs}\ [eats]_{vs}\ [[an]_{a2}[apple]_{nv}]_{fs}, [[an]_{a2}[apple]_{nv}]_{fs}\ [eats]_{vs}\ [[the]_{a3}[boy]_{nc}]_{fs}\}$. The last computation step, the seventh one, is going to output $[[\ ]_{a3}[\ ]_{nc}]_{fs}\ [\ ]_{vs}\ [[\ ]_{a2}[\ ]_{nv}]_{fs}$ which is the structure of the input phrase. This implies that the phrase was correct.

The second phrase we will look at is "The apples eats a boy". It is easy to observe that this phrase is a wrong one. Before the first computation step *ART* will contain the words *the* and *a*, *Nc* the words *boy* and *apples* and *V* the verb *eats*. The third computation step will produce in *NP* the string set $\{[[the]_{a3}[apples]_{np}]_{fp}, [[the]_{a3}[boy]_{nc}]_{fs}, [[a]_{a1}[boy]_{nc}]fs, [[apples]_{np}]_{fp}\}$. By looking at the verb we observe that in the fifth computation step we will get the string set $\{[[the]_{a3}[boy]_{nc}]_{fs}\ [eats]_{vs}\ [[the]_{a3}[apples]_{np}]_{fp}, [[a]_{a1}[boy]_{nc}]_{fs}\ [eats]_{vs}\ [[the]_{a3}[apples]_{np}]_{fp}, [[the]_{a3}[boy]_{nc}]_{fs}\ [eats]_{vs}\ [[apples]_{np}]_{fp}, [[a]_{a1}\ [boy]_{nc}]_{fs}\ [eats]_{vs}\ [[the]_{a3}[apples]_{np}\}$. As no more computation steps are possible and no output is produced the automaton stops and concludes that the input phrase was wrong.

6 DISCUSSION AND FUTURE WORK

In this paper we presented an application of NEPs for the analysis and recognition of sentences of natural language. In the NEPs we have modelled, each processor is specialized in the processing of different syntactic patterns: NP, VP, S. An important feature of the system is that both input and output are linear, the lexical items are the input and syntactic structures are the output.

It is clear that this is not even a preliminary approach, but rather a suggestion of a new research line. We have already highlighted the advantages of such constructs because of their characteristics and functionality. For the future, a more precise analysis of the components of the NEPs for parsing of natural lan-

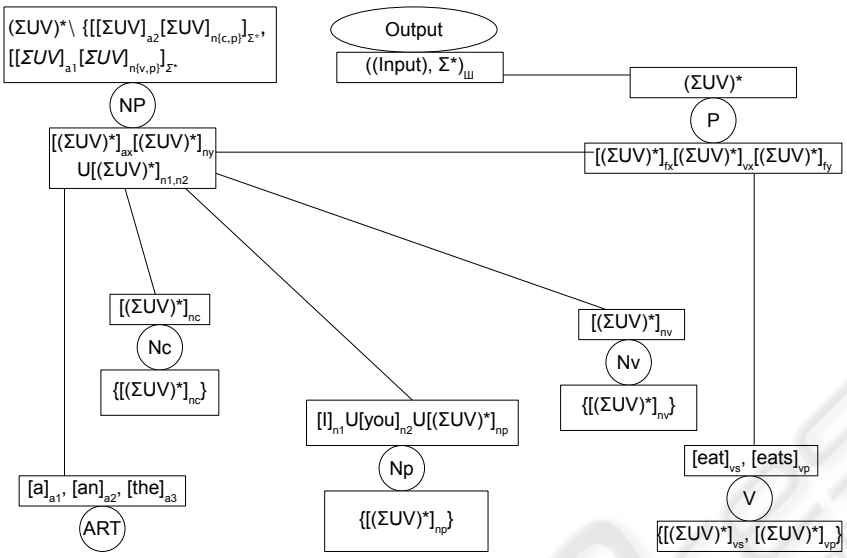

Figure 2: Example.

guage would be necessary as well as a detailed study of their connection with real neuronal capabilities.

There exist also some problems regarding the proposed NEP-based approach to the language processing. As has been mentioned before, the accepting power of NEPs is that of a Turing machine which, on the one hand, allows us to specify a program without any restriction regarding the linguistic framework we want to simulate and the linguistic facts we want to deal with. But on the other hand, it excludes any possibility of the automatic validation of the properties of a program and makes the checking of a large program's correctness very difficult and time-consuming. This concerns in particular the polynomial-time (Turing-)complexity of the running time of a program with respect to the length of the input which is generally considered as crucially important for automatic language processing.

At the same time, the specific structure of the elementary objects used in NEPs, makes out of this formalism a very convenient tool for modelling natural language. From the practical perspective, it means that NEPs can be used as a formal base for specialized task-oriented programming environments for natural language processing.

In this light, a possible future line of research could be restrictions to the NEPs that, on the one hand, will guarantee that any program executes in polynomial time, based on the length of the input and, on the other hand, will not hinder too much the framework's expressivity for linguistic purposes.

We claim that NEPs are a very convenient system, not only for explaining natural language processing, but also for simulating knowledge representation and cognitive mechanisms. Moreover, NEPs provide a consistent theoretical framework for the formalization of human-computer interfaces. In this model, the human capacity of transforming the world by means of the word and knowledge can be approached by a computational device that is rather simple in terms of its size, structure and implementation.

REFERENCES

Castellanos, J., Leupold, P., and Mitrana, V. (2005). On the size complexity of hybrid networks of evolutionary processors. *Theoretical Computer Science*, 330(2):205–220.

Castellanos, J., Manea, F., de Mingo Lpez, L. F., and Mitrana, V. Accepting networks of splicing processors with filtered connections. In *MCU 2007*, pages 218–229.

Castellanos, J., Martín-Vide, C., Mitrana, V., and Sempere, J. M. (2001). Solving np-complete problems with networks of evolutionary processors. *Lecture Notes in Computer Science*, IWANN 2001(2048):621–628.

Castellanos, J., Martín-Vide, C., Mitrana, V., and Sempere, J. M. (2003). Networks of evolutionary processors. *Acta Informatica*, (39):517–529.

Crocker, M. (1991). Multiple meta-interpreteres in a logical model of sentence processing. In Brown, C. and Koch, G., editors, *Natural Language Understanding*

and Logic Programming, III, pages 127–145. Elsevier Science Publishers B.V., North-Holland.

Csuhaj-Varjú, E., Dassow, J., Kelemen, J., and Păun, G. (1994). *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*. Gordon and Breach, London.

Csuhaj-Varjú, E. and Mitrana, V. (2000). Evolutionary systems: a language generating device inspired by evolving communities of cells. *Acta Informatica 36*.

Csuhaj-Varjú, E. and Salomaa, A. (1997). Networks of parallel language processors. *Lecture Notes on Computer Science 1218*.

Errico, L. and Jessope, C. (1994). Towards a new architecture of symbolic processing. In Plander, I., editor, *ARtificial Intelligence and Information-Control Systems of Robots'94*, pages 31–40, Singapore. World Scientific.

Everaert, M., Evers, A., Hybreqts, R., and Trommelent, M., editors (1988). *Morphology and Modularity: In HOnour of Henk Schultink*. Publications in Language Sciences 29, Foris.

Fodor, J. (1983). *The Modularity of Mind*. The MIT Press, Cambridge, MA.

Harnish, R. and Farmer, A. (1984). Pragmatics and the modularity of the linguistic system. *Lingua*, 63:255–277.

Hillis, W. (1985). *The Connection Machine*. MIT Press, Cambridge, MA.

Manea, F. (2004). Using ahneps in the recognition of context-free languages. In *Proceedings of the Workshop on Symbolic Networks, ECAI 2004*.

Manea, F. and Mitrana, V. (2007). All np-problems can be solved in polynomial time by accepting hybrid networks of evolutionary processors of constant size. *Inf. Process. Lett.*, 103(3):112–118.

Margenstern, M., Mitrana, V., and Jiménez, M. P. (2004). Accepting hybrid networks of evolutionary processors. In *Pre-proceedings of DNA 10*, pages 107–117.

Martín-Vide, C., Mitrana, V., Pérez-Jiménez, M., and Sancho-Caparrini, F. (2003). Hybrid networks of evolutionary processors. In *Proceedings of GECCO 2003*, Lecture Notes in Computer Science 2723, pages 401–412, Berlin. Springer.

Păun, G. (2000). Computing with membranes. *Journal of Computer and Systems Sciences*, 61(1):108–143.

Păun, G., Rozenberg, G., and Salomaa, A. (1998). *DNA Computing. New Computing Paradigms*. Springer, Berlin.

Sadock, J. (1991). *Autolexical Synax. A Theory of Parallel Grammatical Representations*. University of Chicago Press, Chicago.

Weinberg, A. (1987). Modularity in the syntactic parser. In Garfield, J., editor, *Modularity in Knowledge Representation and Natural Language Understanding*, pages 259–276. The MIT Press, Cambridge, MA.