On Hybrid Connectionist-Symbolic Models *

Petr Sosík †‡

Abstract

There are many similarities between grammar systems and artificial neural networks: parallelism, independently working elements (grammars/neurons), communication of the elements, absence of centralized control. On the other hand, there are crucial differences between symbolic and quantitative data processing.

We will try to give a brief overview of the methods of "building bridges" between symbolic and connectionist paradigm and to propose some recent results. After that, we will touch some essential problems, concerning incorporation of accepting/generating grammar systems and neural network models.

Keywords: *grammar system, artificial neural network, finite automaton, grammatical inference.*

1 Introduction

First let's recall some characteristics of the constructions we intend to deal with, emphasizing their aspects important for the following discussion. This is not to replace broad and exact descriptions in the basic literature referred to.

Moreover, we restrict our attention to rather theoretical aspects of interactions of neural nets and grammars/grammar systems in hybrid models. For a broader discussion of linking symbolic and subsymbolic systems we refer e.g. to [38].

Speaking about some types of distributed systems, we will use the term "agent" for some simple, but subsystem, interacting with the environment (including other agents) [14]. In terms of grammar systems the agent represents usually a grammar (Chomsky, Lindenmayer or other type [27]), in terms of neural network it represents one neuron or a small group of them.

Let's keep in mind that the artificial neural networks are (not only) biologically motivated, so that each model is looked at as a mathematical machine, possibly physically implementable. That's why this "implementing" point of view will be emphasized in the following paragraphs.

*Presented at workshop *Grammar Systems: Recent Results and Perspectives, July 26-27, 1996, Budapest.*

†Institute of Mathematics and Computer Science, Silesian University at Opava, Bezruv covo nám. 13, 74601 Opava, Czech Republic, email: Petr.Sosik@fpf.slu.cz

‡Research supported by the Grant Agency of Czech Republic, grant No. 201/95/0134.

1.1 Automata

A formal automaton is an accepting device, which is primarily intended to classify strings as *accepted* or *rejected.* Its input string (over a finite alphabet) is accessed *sequentially.* At each step the automaton reads one symbol of the string (and eventually performs some other actions), at the next step it can read the neighboring symbol only. Generally it needs a read/write memory of an unlimited size, but in the most cases of interest in this paper (languages within the context-sensitive class), the amount of memory actually needed can be limited to the length of the processed string. For basic description of formal automata and grammars we refer to [19], [39].

1.2 Grammars

A formal grammar is a form of syntactical description of some formal language. The most important component is a set of rewriting rules (over some finite alphabet). We can interpret it as a generating device, at each step applying some of its rules to a processed sentential form. Due to the fact, that such a device must be eventually able to produce any string of its (often infinite) language, it is mostly nondeterministic (which concern the selection of a rule and a position within the processed string where it is applied). An implementation of a grammar would require an unlimited size memory for reading/writing the string just generated. The memory must allow in principle random access, i.e. at each step any symbol(s) of the string generated can be processed. Some types of grammars can be transformed to a normal form suffering with sequential access. Moreover, the memory must be able to insert new symbols into a processed string.

1.3 Neural networks

An *artificial neural network* (ANN) is a finite set of interconnected autonomous agents – *neurons.* There exist also models of infinite size, but they are rather special [37]. Typically all neurons in the network compute the same function (1), where x_j are the inputs, the y_i is the output of i-th neuron in the network, θ_i is the *threshold function,* see Fig.1. Constants w_{ij} are called *weights* of the inputs. For a more detailed description we refer to [18], [34].

$$y_i = \theta_i(\sum_{j=1}^{N} w_{ij}x_j) \tag{1}$$

The input of ANN is some n-tuple and the output some m-tuple of real-valued signals. There is no relation between n and m. The neurons are often grouped to *layers,* each layer being connected by its inputs to its lower and by its outputs to its upper neighbor. Then the topmost layer is called the *output layer,* connecting its outputs to the outer environment, the most bottom one with its inputs incoming from the environmen is the *input layer* and the others are the *hidden* ones. Within

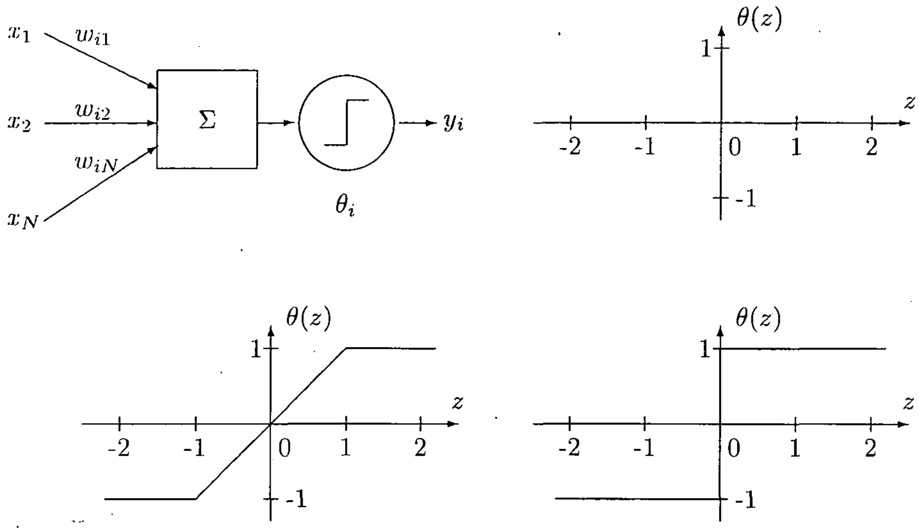

Figure 1: The basic model of neuron and the common types of threshold function.

one-layer networks, the same categorization can be applied to single neurons. There are also intra-layer or bidirectional connections in some models.

The topology of the network (with respect to the paths of passing signals) can be *feedforward* or *feedback*. In the later case, the existence of the feedback loops can lead to the complex dynamical behavior of the network.

There is a *pre-defined communication graph* of the neurons. It can evolve during a training process (some new connections can appear, some old ones can vanish, existing ones can change their strength), but generally these changes are much slower than the normal flow of information (signal levels) through the network.

Thus we distinguish between the *learning mode* of the network when the weights (and sometimes other parameters or even topology) change, and the *recall mode,* when only signal level changes. There is a lot of learning (training) algorithms, various heuristics, which we will not discuss here in general, although it is just the training algorithm, what mostly influences the network behavior.

From another point of view, it is possible to use ANNs with a certain set of threshold functions for approximating any real function with arbitrary small error [23], [30], but this approach is far from symbolic information processing we intend to deal with.

In the most of the known models all the neurons operate *in parallel* (except, for instance, Hopfield network with asynchronous mode) and there is no centralized control of the network activity; the neuron's behavior is influenced by its neighbours only. Neurons can operate in both continuous and discrete time, in the latter case often with a common clock.

1.4 Grammar systems

A grammar system usually consists of some set of (relatively autonomous) components (grammars, agents), performing some operations with a processed string. The notion, originally, was introduced for modelling syntactic properties of multi-agent systems (for details see [6] and [9]), but many variants of grammars (rewriting systems) with regulation and/or modularization can be considered as grammar systems, too (confer to [31], [8]). Here we only emphasize some special capabilities of grammar systems, different from those of single grammars.

- *Parallelism*: while grammars with parallel derivation are rather special, some of basic grammar system models are inherently parallel, e.g. Lindenmayer systems (although they are rather a kind of parallel grammar than a grammar system [27]), parallel communicating grammar systems (PCGS), ...
- In the case of these parallel grammar systems, a *memory* for storing the processed string must allow multiple agents to access different parts of the string at the same time.
- In some models (Lindenmayer systems [31], colonies with terminal mode of rewriting [21], ...) there is a virtually *unlimited potential* of rewriting rules (each rule must be simultaneously at arbitrary number of symbols in generated string).
- There are various forms of *communication* between agents; the simplest form is probably synchronization. A communication graph of agents can be predefined (e.g. colonies, eco-grammar systems [5]) or dynamically defined during the work of the system (e.g. PCGS).

Especially the requirements for the dynamic communication graph, the unlimited potential of rules, but also for the parallel memory capable of inserting symbols can be a source of implementational difficulties [24]. Among physically existing systems, most close to these requirements seem to be those with a great number of elements moving freely (immune system [29], splicing systems [12], ...).

Accepting grammar systems have also been defined [10], [11], relaxing some of these requirements (from the implementation point of view) of the generating systems: the degree of nondeterminism (however its definition could be problematic [10]) is often lower, there is no necessity of inserting new symbols into the processed string if we do not admit λ-rules.

2 A motivation for hybrid models

2.1 Goals and expectations

Both grammar and neural systems, representing symbolic and quantitative data manipulation, are inspired (partially) from the areas of AI and AL. Indeed in biological systems we can find connection of both approaches. For example: specific immunity could well be viewed as a competition between grammars, generating polysaccharide strings covering the antigen, and lymphocyte automata attempting to recognize them. And this competition causes adaptation of both of these systems [29].

Among the advantages of ANNs the most important are adaptability, generalizability, massive parallelism, noise tolerance and graceful degradation (robustness). Main disadvantages are difficult understanding and explanation of the results and time-consuming learning. The symbol processing systems have their strength in easy manipulation with symbolic and rule knowledge and wide base of theoretical results, their weaknesses dwell in sensitivity to noise and difficult, often almost impossible adaptation to a novel data. As the disadvantages of one type systems are balanced by the advantages of second type ones, the hybrid architectures seem to be a very promising way [4]. Another advantage of hybrid architectures can be their scalability. Finally, some authors find them crucial to understanding and constructing cognitive models [17].

2.2 Some basic ideas

The first step of a neural symbolic processing is to find a proper coding of symbols. In animal nerves all signals are of the same amplitude and stimuli intensity is expressed by their frequency [25]. Moreover, coding via the signal intensity leads to loss of robustness due to the necessity of distinguishing between close signal levels. Symbols are coded most often by a group of parallel stimuli, each with clearly distinguishable intensity levels (typically binary). Very often a "one-hot" coding is used (each symbol is assigned a separate input of the ANN).

With a piece of abstraction we can think about symbol-processing structures within the brain, with their abilities having been obtained by training. Training is mostly much more effective with a teacher: a neural network obtains some stimuli and after processing them gives output. Then the teacher produces a training signal, expressing the difference between desired output and output given by net. This signal is utilized by the to change its internal structure slightly, so that, after many such cycles, resulting changes converge to correct (desired) outputs. From this point of view it seems that the trainable neural network should be closer to an accepting symbolic system than a generating one, because it is easier to obtain a training signal in the accepting case. This is indeed the approach of [1], [2], [16], [36] and many others.

Now, how complex languages can biological neural nets "recognize"? Strictly speaking, all the languages we can ever recognize should be regular, because at some moment we are able to keep in mind only a limited amount of information. Really most of the languages we recognize (for example languages of syntactical description of visual images, which we know the brain uses [13]) are finite or without the need of recurrent application of the same rule(s) many times, which is typical for formal languages. This, on the one hand, could in an extreme case lead to an impression of the brain as a system of *finite state automata* (FSA). On the other hand, the effectiveness of neural processing of such languages clearly couldn't be by using FSA machinery. To give a rather expressive example, we know how long the transfer of one stimulus through the neuron and the following relaxation takes – something like "neural step" [3]. A simple reflex reaction, of a child who suddenly runs into the road, takes only about 40–60 such time steps [20], done of course massively parallel!

We often perform tasks that are clearly non-regular. Hardly anybody is capable of multiplying two arbitrary ten digit numbers in his mind, even after long training – this seems to be a rather "nonneural" task. The most primitive tool we can use is paper and pencil – external memory for symbol manipulation. Imagine, how we would recognize e.g. strings of language $\{0^n1^n|n \in \mathcal{N}\}$. Moreover, the brain also contains some very specialized structures, created not by training of an individual, but by evolution [25].

To conclude, we use specialized tools for accepting non-regular language, that were not created by a simple training within our brains. This again leads us to the necessity of hybrid architectures, already expressed in the previous section.

2.3 Theoretical computational power of the neural nets

Now, let's briefly characterize ANN models with enough computational power to accept four basic classes of formal languages. The models are size-independent on the length of the processed string. In the later text a *recurrent neural network* (RNN) is referred to. It is one-layer fully interconnected ANN with N neurons and n inputs, with a dynamics characterized by the equation (2). In addition to (1), u_j are the inputs of the network with weights v_{ij} for i-th neuron, c_i is the threshold value. In the following four cases of equivalence the bottom-left threshold functions of the Fig.1 has The network operates in discrete time and the input string is presented sequentially, i.e. one symbol at each step. The precise definition of language acceptance can be found in [32], [33]. Here we only note, that the used RNN has two binary outputs, the "data" one with accepted/rejected signal and the "validation" one, stating when the data are valid.

$$y_i^{(t+1)} = \theta_i(\sum_{j=1}^{N} w_{ij}y_j^{(t)} + \sum_{j=1}^{n} v_{ij}u_j^{(t)} + c_i), \quad i = 1, 2, \ldots, N. \tag{2}$$

1. Acceptance of regular languages: RNN with discrete outputs of all neurons, i.e. output of each neuron is 0 or 1. A result (output of the network: accepted/rejected) is provided in the next step after the last input symbol has been presented. This well-known result can be found in [22].

2. Context-free languages: RNN with rational-valued outputs and weights of the neurons. As above, a result is provided immediately after the input string has been presented. The precision of computations needed is dependent on (and limited by) the length of the input string. This is rather trivial consequence of the proof of the main result in [32], the principle of which is given in section

3.1. It is clear, that the stack of a pushdown automaton could be implemented by the same way as the tape of a *Turing machine* (TM) in [32].

3. Context-sensitive languages: RNN with rational-valued outputs and weights of the neurons, providing a result after some delay after the input string has been presented. Again the precision of computations needed is by the length of the input string. For deriving this result from [32], it is enough to note the difference between TM and the linear-bounded automaton [39].

4. Recursively enumerable languages: recurrent ANN with rational-valued outputs and weights of the neurons, providing a result after a delay, which can grow over any limit. The necessary precision of computation is also unbounded. This is proven in [32].

3 An overview of hybrid architectures

Architectures coupling symbolic and "neural" methods contain mostly separate neural and symbolic modules. The input of neural net has usually a form of symbolic knowledge (a string or a set of them, properly coded), its output is again interpreted in terms of symbolic knowledge (rules, semantic knowledge...) [38]. Rarely we can meet direct incorporating of symbolic and neural computation principles in one module [22], [35],

3.1 Computational power studies

The aim of these studies is mostly to give a proof of computational power of particular type of ANN. The proofs are constructive, i.e. the result is an ANN modelling the behavior of the the original system (usually some formal automaton), without care of computational complexity and/or model size. Some models are even of infinite size, although the original system is finite [37]. As an example of this approach we present results in [32]. The authors proved that RNN restricted to rational-valued weights and signals has the universal computational power, by constructing a neural model of any TM. The principle of the model (highly simplified) is as follows: First, the tape alphabet is coded by a set of natural numbers less than some b, the empty symbol being coded as 0. Then any string $r_1 r_2 \dots r_k$ over the tape alphabet can be coded as

$$C(r_1 r_2 \dots r_k) = \sum_{i=1}^{k} \frac{r_i}{b^i} \in (0,1). \qquad (3)$$

(In fact the coding is more complicated to avoid the necessity of distinguishing between two very close numbers.) Now let p be the position of the head on the tape of TM. Let output value of neuron α code the portion $r_{p+1} r_{p+2} \dots$ of the tape by the coding (3). Let code portion $r_1 r_2 \dots r_p$ of the tape *in reverse order*. Then we can extract the representation of the symbol read by the head as $\lfloor b o_\beta \rfloor$, where

o_β is the output value of neuron β. Movement head to the right is represented by setting

$$o_\beta := (o_\beta + \lfloor bo_\alpha \rfloor)/b, \qquad o_\alpha := bo_\alpha - \lfloor bo_\alpha \rfloor, \tag{4}$$

and analogously to the left. Here $\lfloor x \rfloor$ is the integer part of x. Rules of TM are then represented by groups of neurons realizing proper logic function. The whole model consists of a fixed number (some hundreds) of neurons. The neurons α, β play the role of wheels rewinding the TM tape, the head being placed between them. Thus the unbounded length of tape is replaced by the unbounded precision of rational number.

If we extend domain of computation to real numbers, we obtain even more powerful (from the computational complexity point of view) devices than TMs (the power equals to TM consulting sparse oracle) [33].

There are (from the implementational point of view) two major drawbacks of this approach: Firstly, there is the necessity of unbounded precise rational number computation. Secondly, the architecture is "hardwired" and practically involves no adaptation. Moreover, it would be very difficult to extend this approach towards parallel working grammar system due to sequential access to the processed string.

3.2 Topology preserving models

These models are topologically similar to the original system, including both number of elements (neurons, states, rules) and the connections between them. Typically each element of one system is modelled by one or a group of the model. The advantage of this approach is often the possibility of and giving a proof of functional equivalence of the systems in both directions, with respect to the computational complexity. Also direct manipulation with a symbolic knowledge within the ANN is then possible.

As an example the classical paper [22] can be presented, establishing equivalence between finite automata and RNNs with discrete-valued outputs. In [15], neural structures equivalent to the state transitions of FSA are used for inserting symbolic knowledge into a RNN. Also some of the models referred to in [38] have this property, what can lead to using a-priori training (see end of this section). In [35] (see Fig. 2) we deal with computational equivalence of eco-grammar systems and ANNs in both directions. Here parallel data processing is fully preserved, which allows to construct model of computational complexity very close to original system.

Again, the disadvantage of this approach is the fact that neural structures topologically similar to the original system are often heterogeneous, what can cause problems with learning, because the most of learning algorithms require a (computationally) homogeneous network for propagating a learning signal through it and making gradient-directed steps towards the solution.

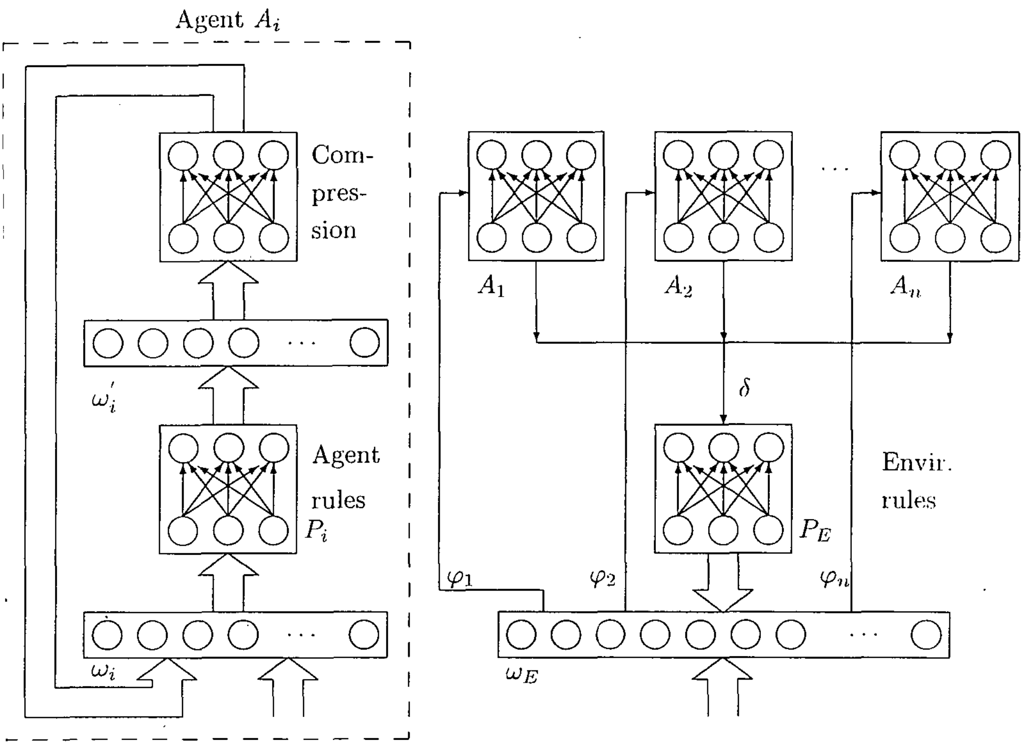

Figure 2: The neural model of an eco-grammar system.

3.3 Adaptive models

These models are clearly the most promising ones and are often applied to the grammatical inference problems, natural language understanding or as a part of knowledge and reasoning systems [38], [4].

A lot of recent works deal with a grammatical inference of some accepting device, as FSA [1], [15], or deterministic *pushdown automaton* (PDA) [36]. This approach seems to be most suitable for studying interactions of neural and symbolic parts of the systems, for instance methods for extracting FSA or PDA from trained RNN of various orders. It seems that if one step of the modelled accepting device is governed by n factors (for PDA $n = 3$: a state, a tape symbol, a stack symbol), n-th order neurons suit best [16], [36]. More complex languages (context-sensitive) accepting devices can be also inferred using RNNs, but in these cases extending "nonneural" inferring algorithms are used [2].

As an example we present the results in [1], using the extraction of unbiased FSA from the trained homogeneous first order RNN:

First, RNN (with one output unit giving value in $\langle 0, 1 \rangle$, which corresponds to rejecting/accepting of input string) is trained using positive and negative examples. Then a prefix tree of these strings is built and its nodes are identified with the states of an *unbiased FSA* (UFSA - its definition is given in [1]). Each state is assigned a set of corresponding hidden layer neuron output values, forming a cluster (initially consisting of a single point) in the state space of the hidden layer neuron outputs. Then, a hierarchical clustering in this space is performed. Whenever two clusters are merged, parallel merging of their associated states is performed in the UFSA representation. After each step the consistency of the UFSA with the training set is tested. Whenever an inconsistency occurs, the process is stopped. The advantage of this approach is that there is no estimation of the number of the UFSA states, minimal cluster distance and so on; only the metric for cluster distance must be defined.

A lot of similar results cited above lead us to the opinion that (from the adaptability point of view) there is no need for topological equivalence of the original system and the model. The most natural representation of one state of the UFSA in RNN is a cluster of neuron states, which has nothing to do with the net topology. Generally, even if we haven't any prior knowledge of the complexity of the grammar inferred, the RNN can create a model of this grammar in the space of its internal states. Then we apply some heuristic algorithm for extracting this grammar making it as simple as possible to be consistent with the set of training strings. The drawback of this approach is again the problematic interpretation of the results of ANN and its transformation to a symbolic knowledge.

4 Hybrid models with grammar systems

One possible way of constructing such models could be incorporating some symbol manipulating principles into the neural nets, what's in opposite to usual approaches, integrating neural nets as a part of rather complex symbolic knowledge manipulating architectures. Some problems of this intention are discussed here.

First, a communication between agents in distributed systems (including various kinds of systems from multicomputers to neural nets) could in general be characterized as *by request* (when the agents are passive until any external stimulus) or *by command* (when the agents activelly offers their free cappacity and/or results). The terminology is inspired by [7], [28], where PCGSs are studied. It seems, that the communication by request is closer to the nature of ANNs, when every agent (neuron), through the weights of its connection, can decide which other agents it gets information from.

As it was mentioned in section 1, dynamically defined and parallel communication between agents and parallel access of the agents to different parts of the processed string is typical for some types of grammar systems. This is contrasted to the nature of ANNs with fixed communication graphs. To introduce a "communication dynamics" into ANNs, one have to create highly interconnected structures (including all possible communication branches) equipped with a mechanism of dynamic activation of the connections needed. (This can be directed e.g. by an occurance of some patterns in the string(s) processed by the system) Due to the conclusions in section 2, pre-defined specialized structures allowing this dynamics need to be used as a part of NN. See [36] where a "neural PDA" is presented, equipped with external continuous stack. Unfortunately, the existence of such structures is often in contradiction with the request for homogenity of network dynamics, because many training algorithms require differentiability of the transfer function of the neurons and groups of neurons.

Another problem can be caused by the fact that the recurrent application of the same rule (group of rules) to the generated or accepted string is typical for grammars (and thus also agents of grammar systems). ANNs, in contrast, obviously reach a stable state in a few steps, due to their nature as an asymptotically stable dynamical system. In the models described in section 3.3 this problem has been overcome by the sequential reading of the input string, providing stimuli not allowing the system dynamics to stabilize. This solution leads nevertheless to the lack of parallelism.

Also a memory for storing the processed string, capable of replacing some substrings by ones of different length, is a very "nonneural" device. Both these problems seem to be best solved by adding another special structure integrating the one-step actions of agents into the processed string and re-loading the result into net inputs. During the phase of learning, such a network could be "unfolded in time", just as in the *Backpropagation algorithm* [18], [36] and similar ones, giving promising results.

5 Conclusion

Although there have been many successfully working hybrid neural-symbolic architectures referred to in the last two sections, it still seems to be possible to find a closer connection of grammar systems and ANNs, particularly exploiting massive parallelism and no need of centralized control in both system types. A highly interconnected RNN looks as a promising way of constructing trainable "neural" accepting symbol-processing system, eventually with a lot of autonomous modules, equipped with some special structures for symbol storing, transfer and some other tasks which are inherently "nonneural". It would be capable of parallel string processing and inter-agent communication.

There are many open problems, including the definition of training hybrid systems and proofs of their stability and convergence. Some of them have been partially solved already, as the training of ANNs is only now beginning to be deeper understood task [4]. Anyway, parallelism and decentralized processing are the main features of present computer architecture trends.

References

[1] R. Alquézar, A. Sanfeliu: A hybrid connectionist-symbolic approach to regular grammatical inference based on neural learning and hierarchical clustering. In: *Grammatical Inference and Applications,* proc. of ICGI-94, R. Carrasco, J. Oncina (eds.), Springer-Verlag 1994, pp.203–211.

[2] R. Alquézar, A. Sanfeliu, J. Cueva: Learning of context-sensitive language acceptors through regular inference and constraint induction. In: *Proc. ICGI'96,* L.Miclet and C.de la Higuera (eds.), Springer-Verlag, Lecture Notes in Artificial Intelligence 1147, pp.134-145, 1996.

[3] P. Århem: On artificial intelligence and neurophysiology: two necessary questions. In: J.L. Casti, A. Karlqvist (eds.): *Real Brains, Artificial Minds.* North Holland, Elsevier Science Publishing Co., Inc., N.Y. 1987.

[4] L.A. Bookman, R. Sun: Editorial: Integrating neural and symbolic processes. *Connection Science,* vol. 5, No. 3-4, p.203, 1993.

[5] E. Csuhaj-Varjú, A. Kelemenová, J. Kelemen, Gh. Păun: Eco-grammar systems: A gramatical framework for life-like interactions. Silesian University CSR-TR-95/2, Opava, 1995.

[6] E. Csuhaj-Varjú, J. Dassow, J. Kelemen, Gh. Păun: *Grammar Systems: a Grammatical Approach to Distribution and Cooperation.* Gordon and Breach Science Publishers, London, 1994.

[7] E. Csuhaj-Varjú, J. Kelemen, Gh. Păun: Grammar systems with WAVE-like communication. *Computers and Artificial Intelligence,* 15 (1996), 419-436.

[8] J. Dassow, Gh. Păun: *Regulated Rewriting in Formal Language Theory.* Springer-Verlag, Berlin, Heidelberg, 1989.

[9] J. Dassow, Gh. Păun, G. Rozenberg: Grammar Systems. In: *Handbook of Formal Languages, Vol. 2,* G. Rozenberg and A. Salomaa (eds.), Springer, Heidelberg, 1997, 155-213.

[10] H. Fernau, H. Bordihn: Remarks on accepting parallel systems. *Intern. J. Computer Math.*, vol. 56, 51–67.

[11] H. Fernau, M. Holzer, H. Bordihn: Accepting multi-agent systems: the case of cooperating distributed grammar systems. *Computers and Artificial Intelligence*, vol. 15 (1996), No. 2–3, 123–139.

[12] R. Freund, L. Kari, Gh. Păun: DNA computing based on splicing: the existence of universal computers. Technische Universität Wien TR 185-2/FR-2/95, Wien 1995.

[13] K.S. Fu: *Syntactic Pattern Recognition and Applications.* Englewood Cliffs, NJ, Prentice-Hall, 1982.

[14] M.R. Genesereth, N. J. Nilsson: *Logical Foundations of Artificial Intelligence.* Morgan Kaufman, Los Altos, Cal., 1987.

[15] C. Giles, C. Omlin: Extraction, insertion and refinement of symbolic rules in dynamically-driven recurrent neural networks. *Connection Science*, vol. 5, No. 3-4, p.307, 1993.

[16] M. Goudreau, C. Giles, S. Chakradhar, D. Chen: First-order vs. second-order single layer recurrent neural networks. In: *IEEE Trans. on Neural Networks*, vol. 5, No. 3, p. 511, 1994.

[17] J. Hendler: On the need for hybrid systems. *Connection Science*, vol. 1, No. 1, p.0, 1989.

[18] J. Hertz et.al.:*Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.

[19] J.E. Hopcroft, J.D. Ullman: *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading, 1979.

[20] J. Hořejš: A view on neural network paradigms development. *Neural Network World*, IDG Prague, 1991,1992.

[21] A. Kelemenová, E. Csuhaj-Varjú: Languages of colonies. *Theoretical Computer Science* 134, Elsevier, 1994, 119–130.

[22] S.C. Kleene: Representation of events in nerve nets and finite automata. In: C.E. Shannon, J. McCarthy (eds.): *Automata Studies*, 3–41, Princeton Univ. Press 1956.

[23] V. Kůrková: Universal approximation using feedforward neural Gaussian bar units. *Proceedings of ECAI'92*, Vienna, 1992, 193-197.

[24] M. Malita, Gh. Stefan: The eco-chip: A physical support for artificial life systems. In: [26], 260–275.

[25] J.G. Nicholls, A.R. Martin, B.G. Wallace: *From Neuron to Brain.* Third edition, Sinauer Ass., Sunderland, Ma, 1992.

[26] Gh. Păun (ed.): *Artificial Life: Grammatical Models.* The Black Sea University Press, Bucharest, 1995.

[27] Gh. Păun: Foreword. In: [26], v-ix.

[28] Gh. Păun, L. Sântean: Parallel communicating grammar systems: the regular case, *Ann. Univ. Buc., Math.-Informatics Series 38* (1989) 55–63.

[29] A.S. Perelson: Immune network theory. *Immunological Reviews,* Santa Fe Institute, No. 110 (1989), 5–36.

[30] T. Poggio, F. Girosi: Networks and the best approximation property. *Biological Cybernetics*, 63, 1990, 169-176.

[31] G. Rozenberg, A. Salomaa: *The Mathematical Theory of L Systems.* Academic Press, New York 1980.

[32] H. Siegelmann, E. D. Sontag: On the computational power of neural nets. In: *Proc. Fifth ACM Workshop on Computational Learning Theory*, 440–449, Pittsburgh 1992.

[33] H.T. Siegelmann, E. D. Sontag: Analog computation via neural networks. *Theoretical Computer Science*, 1994.

[34] K.P. Simpson: *Artificial neural systems.* Pergamon Press, N.Y., 1990.

[35] P. Sosík: Eco-grammar systems and artificial neural networks. *Computers and Artificial Intelligence*, No. 2–3 (1996), 247–264.

[36] G. Sun, C. Giles, H. Chen, Y. Lee: The neural network pushdown automaton: model, stack and learning simulations. University of Maryland TR Nos. UMIACS-TR-93-77 & CS-TR-3118, 1995.

[37] D. Wolpert: A computationally universal field computer which is purely linear. *Los Alamos National Laboratory report LA-UR-91-2937.*

[38] A. Wilson, J. Hendler: Linking symbolic and subsymbolic computing. *Connection Science*, vol. 5, No. 3-4, p.395, 1993.

[39] D. Wood: *Theory of Computation.* John Wiley & Sons, 1987.