Artificial Grammar Learning and Neural Networks

Karl Magnus Petersson (karl.magnus.petersson@fcdonders.ru.nl)
F.C. Donders Centre for Cognitive Neuroimaging,
Radboud University Nijmegen, The Netherlands
CSI, Center for Intelligent Systems, Universidade do Algarve, Portugal

Peter Grenholm (peter_grenholm@yahoo.se)
Cognitive Neurophysiology Research Group, Karolinska Institutet
171 76 Stockholm, Sweden

Christian Forkstam (christian.forkstam@cns.ki.se)
Cognitive Neurophysiology Research Group, Karolinska Institutet
171 76 Stockholm, Sweden

Abstract

Recent FMRI studies indicate that language related brain regions are engaged in artificial grammar (AG) processing. In the present study we investigate the Reber grammar by means of formal analysis and network simulations. We outline a new method for describing the network dynamics and propose an approach to grammar extraction based on the state-space dynamics of the network. We conclude that statistical frequency-based and rule-based acquisition procedures can be viewed as complementary perspectives on grammar learning, and more generally, that classical cognitive models can be viewed as a special case of a dynamical systems perspective on information processing.

Keywords: artificial grammar learning, neural network, dynamical systems.

Introduction

According to Chomsky a core feature of natural language processing is the 'infinite use of finite means'. The family of right-linear phrase structure grammars, implementable in the finite-state architecture (FSA), is a simple formal model of this idea. The work of Reber (1967) suggested that humans can learn AGs implicitly and that the relevant structure is abstracted from the input. Reber (1967) proposed that this process is intrinsic to natural language learning and it has been suggested that AG learning (AGL) is a relevant model for aspects of language acquisition (Gomez & Gerken, 2000). Recent FMRI studies indicate that language related brain regions are engaged in AG processing (Petersson et al., 2004). Here we investigate the Reber grammar (Figure 1) by means of formal analysis and network simulations and outline an approach to grammar extraction based on the network state-space dynamics.

Elementary formal analysis

We begin by showing that the Reber grammar, and in certain respects similar AGs, can be learned by acquiring a finite set of n-grams (for details see Grenholm, 2003). To this end, we modify the Reber machine (RM0) by connecting the final to the initial state with a #-labeled transition (Figure 1). The modified RM1 outputs an infinite symbol string: first an end-of-string symbol '#', then a Reber string, a '#', a new string, and so on indefinitely. It turns out that to recognize all possible output strings of RM1, a necessary and sufficient condition is to know a set, TG, of 48 trigrams. A string is generated by RM1 if and only if the string starts with '#' and only contains trigrams in TG. To show this, we observe that the Reber grammar yields exactly the same strings as RM2 (Figure 2). Assume that we know RM1 and that we know the latest two symbols in an output string from RM2. This determines the internal state of RM1 and RM2. The set of possible bigrams is {MT,MV,…,#V}.

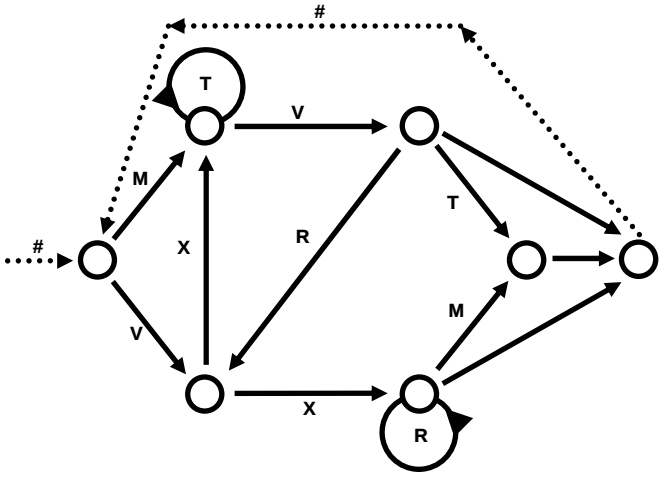

Figure 1: The transition graph representation of RM1 corresponding to the Reber grammar.

We obtain the set of possible trigrams from RM2 if we take these 21 bigrams and extend them according to Figure 2, yielding TG={MTT,MTV,…,#VX}. It is clear that a string is an output of RM1, if it starts with '#' and only contains trigrams in TG. Conversely, assume that a string begins with one of the 48 trigrams, the first symbol of which is '#'. The only possibilities are '#MT', '#MV', and '#VX', which

determine unique states in RM2. Recursively, assume that the last three symbols were $C_1C_2C_3$. Then C_2C_3 is a possible bigram, hence determines a unique internal state S in RM2. If the next symbol is C_4, by assumption $C_2C_3C_4$ is a possible trigram and C_4 must be on one of the transitions from S, and thus assigns a unique state to RM2. In this way, we can traverse RM2 in a way that yields the desired output string. It follows that the string is a possible output of RM1. This suggests one reason for the success of fragment-learning methods in acquiring the Reber grammar and similar AGs: a machine that memorizes TG can in principle recognize the Reber language.

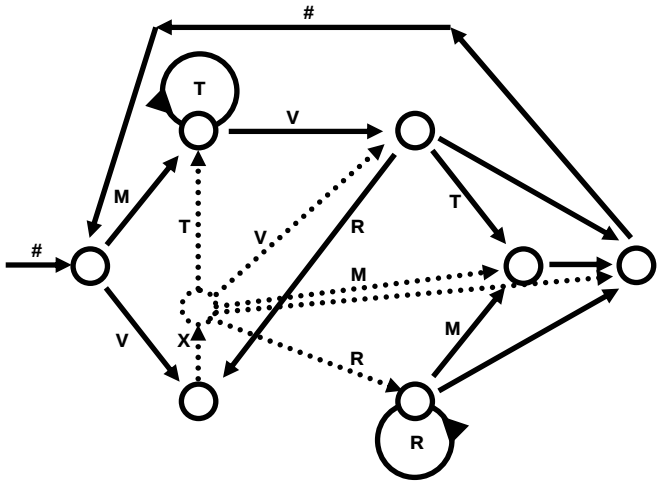

Figure 2: RM2 – The modified RM1.

We can further characterize the properties of RM1 by information theoretic tools. We associate a random process Ψ with RM1 by setting Ψ[0]='#' with probability one. Then beginning at the start state of RM1 and randomly selecting one of the possible transitions with equal probability. Let the value of Ψ[1] be the corresponding symbol. Continuing in this manner, randomly traversing RM1 yields a sequence of random symbols Ψ[n]. The random process obtained will be referred to as the Reber process, a hidden Markov process. We note that while RM1 and RM2 generate the same output strings, the corresponding random processes have different probability distributions.

Information theoretic analysis

In certain cases it is possible to define the entropy of a discrete-time random process. This procedure starts with the entropy of Ψ[n] that remains after conditioning on Ψ[0],…, Ψ[n-1] and this conditional entropy will in our case converge as $n\to\infty$. The elementary analysis implies that, given two symbols of Ψ, the probabilities for the next symbol are determined. Hence, Ψ is second-order Markov and consequently we can determine its entropy (Goldie & Pinch, 1991). In computing entropies, we followed the procedures in section 2.7 of Goldie & Pinch (1991). Higher-order Markov processes were transformed into Markov processes of order one by considering strings as output of the process instead of individual symbols (e.g., the output MTVT is transformed to MT–TV–VT). The limit distribution of the new Markov process was computed by an iterative procedure yielding a numerical limit distribution. The probabilities were rounded and verified to be stationary. The entropy of Ψ was then calculated according to $H(\Psi) = -\Sigma_{ij}\pi_{ij}\log(p_{ij})$, where π_{ij} is the stationary distribution and p_{ij} the transition probabilities of Ψ. We can thus measure how much information that is contained in the last and next-to-last symbol etc. Since the process takes its values among six symbols, the entropy of a single symbol = 2.46 ($\cong \log 6$), implies that these are approximately equally distributed. If we know the latest symbol, the entropy of the next decreases to 1.54 ($\cong \log 3$), while given 2 symbols, which is all there is to know for a second-order Markov process, the entropy = 1.00 bits/symbol. This makes intuitive sense, since most states of RM1 have two exit transitions.

Table 1: Measures of the conditional entropy.

Machine	0	1	2	3	4
RM1	2.46	1.54	1.00	1.00	1.00
RM2	2.25	1.58	1.19	1.14	1.13

We see that more of the information is contained in the last compared to the next-to-last symbol, implying that one can expect reasonably good performance by recognizing bigrams alone. Similar results hold for RM2.

Neural network model

We proceed to outline a series of computer experiments using predictive networks based on the simple recurrent network (SRN) architecture (Elman, 1990). The networks were trained on acquisition data generated from RM1 (Grenholm, 2003) with a gradient-descent error back-propagation algorithm with adaptable learning rate and momentum. The discrete-time SRN can be viewed as a simple network analogue of the FSA. More specifically, at time point n+1, the output $\lambda(n+1)=N[\omega(n),\sigma(n)]$ of the SRN is a function of the input σ(n) and the previous internal state ω(n). The internal state $\omega(n)=[h(n),c(n)]$ consist of two components: the state of the hidden layer of computational units h(n) and the state of a 1-step short-term memory layer c(n)=h(n-1). If we introduce the time-shift operator Δ (i.e., $\Delta\beta(n)=\beta(n+1)$), the dynamics of the state transitions are described by: $\Delta h(n):=A[\omega(n),\sigma(n)]$, $\Delta c(n):=C[\omega(n)]:=h(n)$; equivalently,

$$\Delta\omega(n) = U[\omega(n),\sigma(n)] \quad [1]$$
$$\Delta\lambda(n) = N[\omega(n),\sigma(n)] \quad [2]$$

where $U[\omega(n),\sigma(n)]:=(A[\omega(n),\sigma(n)],C[\omega(n)])$. It is clear that the form of [1,2] is identical to that of the dynamical formulation of the FSA (cf., Eqs. [4,5]). This is an example of a well-known result of McCulloch and Pitts (1943), who showed that a particular class of networks is equivalent to the FSA.

Generally, learning and development can be implemented by making the system function, for example U and N of equation [1,2], dependent on adaptive parameters α, thus yielding a model space $M:=\{U_\alpha, N_\alpha|\ \alpha$ realizable by the system$\}$. Learning (development) can then be

conceptualized as a trajectory in the model space M, determined by a learning dynamics L of the system:

$$\Delta\alpha(n) = L(\alpha(n), \sigma(n), n) \quad [3]$$

given an initial state $\alpha=\alpha_0$; here L is explicitly dependent on time in order to capture the idea of innately specified developmental processes and (possible) dependence on the previous developmental history. In general, information processing and learning can be viewed as a coupled dynamical system, where the processing dynamics [1,2] is coupled with the learning dynamics [3]. If we replace the discrete time with a continuous time and specify the differential time changes, we have the general continuous-time case. We note that the SRN differs in processing capacity from the FSA to the extent that it can be viewed as an analog model; in other words, to the extent that the SRN takes advantage of infinite processing precision. If this is not the case, or proper care is not taken with finite-precision simulations, the SRN simulation reduces to a FSA simulation, which might or might not capture relevant model behavior (McCauley, 1993).

Performance evaluation

Four different methods for performance evaluation were used: (A) The output of the network was compared with the next symbol in the test sequence. This difference is precisely what the learning process aims to minimize. Since the next symbol of the test sequence is a random variable, perfect prediction performance is impossible, and one approach is to compute the theoretically best performance in terms of generalization performance and compare the actual performance with this. Here the output was interpreted using a winner-take-all competitive architecture in the output layer. (B) It is possible to interpret the normalized output of the network as a probability distribution for the next symbol. This is then compared with the theoretical probability distribution of RM1. (C) The third method mimics human experiments on AGL, in which the network is set to discriminate between correct and incorrect strings in a grammaticality classification task. In method C1, we introduced a grammatical string between strings to be classified (cf., below), in order to force the network to settle in a start-of-word state. We denote method C by C2 when we do not use this reset procedure. (D) Here we attempted to measure how far the internal state of the network deviates from what can be considered the typical behavior in terms of its state-space dynamics. This is described in greater detail below and attempts to analyze the internal behavior of trained network. The basic question is whether the nodes in the hidden layer capture features that are interpretable with respect to underlying generative mechanism. To this end, we constructed a competitive network, which was trained to classify the outputs of the hidden nodes.

Simulation results

From the Reber process, a sequence of 1000 symbols was generated (127 strings). In the first series of experiments, five SRNs were defined with the following architecture: 6 nodes in the input/output layers, and 2^k nodes (k = 1-5) in the hidden layer (transfer function in the hidden layer: inverse tangent; output layer: linear). With randomly initialized weights, the mean squared error was used as the objective cost function for the learning process (gradient-descent back-propagation with adaptable learning rate and momentum). The symbols were translated into a vector representation with six elements (0.8 in one position and else -0.2). The networks were trained for 200 epochs using one symbol of the acquisition sequence as input and the next symbol as target. The results of the first series of experiments (mean learning error on the acquisition data for 10 networks of each size) indicate that the smaller networks (2, 4 hidden nodes) have smaller error in the beginning, while the larger networks perform better in the end (8-32). There was no performance gain when the number of hidden nodes increased above 8. In epochs 40-200, the performance curves continue almost horizontally, essentially replicating previous results (data not shown). In a second series of experiments, the networks were evaluated using methods A-D on an independent string set. Evaluation by method A and B is straightforward and in order to evaluate the networks by method C and D we generated a list of Reber and non-Reber strings. The latter were generated from a first-order Markov process with the same bigram statistics as the Reber process. The list contained 1000 symbols (50% strings from the Reber process). In method C, we interpreted the network's output as probabilities and multiplied these for each symbol in the test string, yielding a maximum-likelihood score for each string. These were normalized by taking the logarithm and dividing by string-length. We measured performance according to how many non-Reber strings that scored better than Reber strings. In order to improve performance, resetting was used in C1 by feeding MV#MV#MV#MV as input between each test string. In this way, the network settles in its start-of-word position. For method D, we first trained a competitive network with 24 nodes for 10000 epochs on the activation pattern of the SRN hidden nodes. The adaptive algorithm used was developed from the algorithms described in section 2.5 of Haykin (1998).

Table 2: Performance evaluation according to methods A and B. Mean from 8 networks of each size (± SD).

#Hidden Nodes	A	B
2	.114±.002	.037±.002
4	.094±.004	.017±.004
8	.088±.002	.010±.002
16	.086±.003	.009±.003
32	.085±.002	.008±.002
Best possible	.076	0

We then measured, for each symbol, the state-space distance from the activation of the hidden nodes to the closest node in the competitive network. We computed the mean value of these distances for all symbols in a test string. Heuristically, if the mean distance is large, the word is likely to be non-grammatical. The rest of the evaluation was performed as in method C. We evaluated the generalization capacity on an

independent string set, using the methods A-D described above, see Table 2 and 3.

Table 3: Performance evaluation according to methods C and D. Mean from 8 networks of each size (± SD).

#Hidden Nodes	C1	C2	D
2	60±4%	60±5%	61±2%
4	79±6%	81±5%	68±6%
8	91±3%	87±5%	75±5%
16	93±2%	74±6%	83±5%
32	94±2%	73±4%	81±5%
Best possible	100%	100%	100%

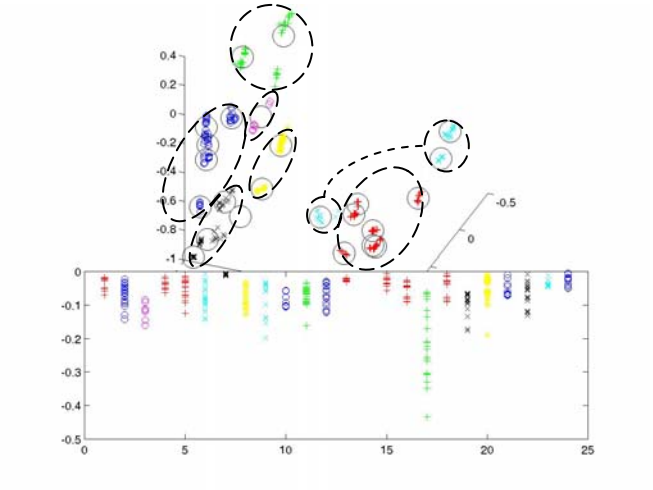

Figure 3: The upper part shows the activation of the hidden nodes plotted along the first three principal axes obtained by PCA. The dots are coded to correspond to the seven internal states of RM1. The diagram shows that dots corresponding to the same internal state are geometrically close (dashed ellipses). The filled circles in the diagram are centered at the 24 nodes of a trained competitive network. The lower part of the diagram plots all activation points according to their distance from the closest node in the competitive network.

The theoretically best performance was calculated from the full knowledge of the state transitions between internal states of RM1. The behavior of the competitive network, characterized by the first three principal components, is illustrated in Figure 3. We see that the competitive network is able to discriminate between activation states of the hidden nodes corresponding to different states of RM1: to every RM1-state corresponds one region of state-space. Thus the state-space dynamics of the network reproduces the dynamics of the Reber machine, and the combined approach of analyzing the behavior of the hidden nodes with a competitive network and PCA provides one approach to grammar extraction from the state-space dynamics of the recurrent network. We note that grammar extraction as outlined here is related to the symbolic dynamics approach used in the analysis of non-linear dynamical systems (cf., Badii & Politi, 1997).

Discussion

There was no significant performance increase with more than 8 hidden nodes on the training data. These nodes have two main tasks: process memory and computation. The information theoretical analysis suggests that the memory required is on the order of 2-3 bits (~2-3 nodes; cf., Table 1). The reason performance improved with 8 hidden nodes is likely related to the requirements of the output representation. In general, there is a risk of over-learning using large networks on small acquisition samples, leading to sub-optimal generalization performance and less efficient acquisition of the underlying structure in the input (Haykin, 1998). In the present study, this was not a major problem (cf., Table 2 and 3), and is likely related to the close match between the SRN architecture and Markov processes. As noted above, the SRN architecture can be viewed as a time-discrete analog version of the FSA. To make this explicit, we take a dynamical systems perspective on the classical computational models (Davis, Sigal, & Weyuker, 1994); let Σ be the input space ($\sigma \in \Sigma$), Ω the space of internal states ($\omega \in \Omega$), and Λ the output space ($\lambda \in \Lambda$). The possible transitions between internal states are determined by a transition function $T:\Omega x\Sigma \rightarrow \Omega$ and the outputs by a function $R:\Omega x\Sigma \rightarrow \Lambda$. In other words, at processing step n, the system receives input $\sigma(n)$ in state $\omega(n)$, and changes its state into $\omega(n+1)$ while generating the output $\lambda(n+1)$ according to:

$$\Delta\omega(n) = T[\omega(n), \sigma(n)] \quad [4]$$

$$\Delta\lambda(n) = R[\omega(n), \sigma(n)] \quad [5]$$

In this way, the system traces a trajectory in state-space, forced by the input, and thus generating an output sequence. Within the framework of Church-Turing computability Σ, Ω, and Λ are all finite and thus T and R are finitely specified. It is clear then that these models represent a special class of dynamical systems (Petersson, 2004b). Furthermore, we note that the form of [4,5] is identical with that of equations [1,2]. Here we have not explicitly described the system's memory organization. In all classical architectures, the transition function $T:\Omega x\Sigma \rightarrow \Omega$ can be realized in a FSA. Thus, with respect to the mechanism subserving transitions between internal states there is no fundamental distinction in terms of machine complexity between the different computational architectures. However, as indicated by the Chomsky hierarchy (Davis, Sigal, & Weyuker, 1994), there are differences in structural expressivity. These differences are fundamentally related to the interaction between the generative mechanism and the available memory organization. The most important determinant of structural expressivity is the availability (or not) of infinite storage capacity. Thus, it is the characteristics of the memory organization, which in a fundamental sense, allow the architecture to recursively employ its processing capacities inherent in T, to realize functions of high complexity or complex levels of expressivity.

In addition, we evaluated network performance on test data with the same bigram statistics as the Reber process. This is a harder task than commonly employed in AGL experiments on humans or with computers. We were able to obtain better than 90% network performance as

characterized by method C1, which is not surprising on theoretical grounds but suggests that the lower SRN performance reported in the literature is related to the evaluation method C2. In particular, when a large SRN is tested using C2, a single incorrect symbol can induce activation patterns that are quite distant from the regions of state-space on which it was trained. The state-space trajectories then remain distant for several subsequent input symbols from 'typical' trajectories to which the SRN has been exposed. A grammatical string that is processed subsequently to an incorrect string can thus obtain a low grammaticality score due to transient network behavior.

Grammar learning

Grammar learning, whether natural or artificial, is commonly conceptualized either in terms of a structure-based ('rule') acquisition mechanisms or statistical learning mechanisms. Our analysis of activation patterns suggests that the SRN architecture can learn to represent the rules of the grammar in its state-space dynamics. Conversely, the elementary analysis shows that a finite set of string fragments is sufficient for Reber language recognition. Thus, it appears that these perspectives on grammar learning are complementary rather than mutually exclusive, at least in the case of finite-state grammars. Some aspects of natural language (e.g., syntax) are amenable to an analysis within the classical cognitive science framework, which suggests that isomorphic models of cognition can be found within the framework of Church-Turing computability (Davis, Sigal, & Weyuker, 1994). These language models typically allow for a greater structural expressivity than can be (strictly) implemented in the FSA. For example, the different architectures of the Chomsky hierarchy allow for different types of recursion, a feature thought to be at the core of the language faculty (Hauser et al., 2002). Unlimited embedding recursion is supported by the push-down architecture, while unlimited cross-dependency recursion requires a linearly-bound architecture (Davis et al., 1994); none of these are characteristic of human performance. In contrast, the FSA supports unlimited concatenation recursion and can support finite recursion of general type. This is also characteristic of human performance. It is well accepted that neurobiological and functional brain constraints have important implications for the characteristics of the language faculty (Hauser et al., 2002). It seems natural to assume that the brain is finite with respect to its memory organization. Now, if one assumes that the brain implements a classical model of language, then it follows immediately from the assumption of a finite memory organization that this model can be implemented in a FSA, although a context-sensitive or any other suitable formalism might be used as long as the finite memory organization is appropriately handled. In short, a finite-state machine will behave as a Turing machine as long as the memory limitations are not encountered. However, one can take the view that classical cognitive models of language are approximate abstract descriptions of brain properties. A complementary perspective is offered by network models of language processing. The network perspective represents a special case of the recently revived dynamical systems perspective on analog information processing (Siegelmann & Fishman, 1998) as a model for cognition (e.g., Petersson, 2004). From a neurophysiological perspective, the natural generalization of the classical finiteness in this context is the property of state-space compactness (Petersson, 2005a). If one assumes that the brain sustains some level of noise or does not utilize infinite precision processing (either of these assumptions seems necessary for physically realizability), it can be argued on qualitative grounds that 'natural language', viewed as a neurobiological system, might be well-approximated by finite-state behavior (Petersson, 2005a).

Returning to the issue of grammar learning, it is possible to take a view that is placed somewhere between the two more common conceptualizations. The generative mechanisms of the Reber machine is easily translated into a Minimalist-type or unification-based framework (Chomsky, 2005; Joshi & Schabes, 1997) as suggested in Petersson et al. (2004). Specifically, given a transition from state s_j to s_k when the terminal symbol T is recognized ($s_j \rightarrow^T s_k$ in the transition graph), this is translated into a lexical item or feature vector $[s_j,T,s_k]$, where s_j and s_k should be interpreted as 'syntactic' or 'control' features ('specifier' feature: s_j, and 'complement' feature s_k) and T as a 'surface' or 'phonological' feature. A finite transition graph thus generates a finite number of lexical items. The syntactic features of these representations could very well be generated or estimated based on a statistical learning mechanism. Moreover, there is no need for a specific 'rule' acquisition mechanism, because the parsing process might use general structure integration mechanisms already in place for merging or unifying structured representations. We note that the syntactic features of lexical items have acquired a particular functional role in this picture. This can be described in terms of governing or monitoring of the integration process based on selecting pieces of information that can be merged. In other words, the finite-state control has been distributed over a mental lexicon (long-term memory) among the lexical items in terms of control features. In the unification picture, the lexical item $[s_j,T,s_k]$ corresponds to stored trees ('treelets') with foot- (i.e., s_k) and root-nodes (i.e., s_j), which are merged or unified in a unification space. Thus, again, important aspects of the finite-state control is allocated to the mental lexicon and the foot- and root-nodes of the lexical items now come to govern the unification process based on selecting the representations that can be recursively integrated. This view on grammar acquisition is more akin to lexical learning in that it suggests that simple structured representations $[s_j,T,s_k]$ are created during acquisition. In essence, this re-traces a major trend in theoretical linguistics in which more of the grammar is shifted to the mental lexicon and the distinction between lexical items and grammatical rules is beginning to evaporate.

Information processing in dynamical systems

It is well-known that the class of symbolic processing models can be captured within the Church-Turing framework, which is computationally equivalent to the class of partially recursive functions (Davis, Sigal, & Weyuker, 1994; Rogers, 2002). Hence it is possible to simulate all finitely specified symbolic models as processes on numbers. Furthermore, it is known that these can be emulated in dynamical systems, including low-dimensional smooth dynamical systems (Moore, 1991), analog recurrent networks (Siegelmann & Fishman, 1998), and spiking networks (Maass, 1996). Generally, network approaches offer interesting possibilities to model cognition within a non-classical dynamical systems framework. For example, a rich class of dynamical systems can be implemented in recurrent network architectures (Siegelmann & Fishman, 1998; Legenstein & Maass, 2005). The recurrent network architecture can be viewed as a finite set of analog registers (e.g., membrane potentials) that processes information concurrently and interactively. It is natural to model cognitive brain functions in terms of spiking recurrent networks and since the brain only represents 'numbers' in terms of membrane potentials, inter-spike-intervals, or any appropriate set of dynamical variables, it is clear that the human brain does not represent cognitive structures in a simple transparent manner. However, general dynamical system theory is obviously too rich as a framework for formulating explicit models of cognitive brain functions. For example, it turns out that for any given state-space one can find a universal dynamical system whose traces will generate any dynamics on the state-space (Lasota & Mackey, 1994). Thus, what is needed is a specification of cognitively relevant constraints as well as processing principles relevant for neurobiological networks subserving information processing in the brain (Petersson, 2005b). Any real progress on this front would represent a significant generalization of Chomsky's concept of knowledge and competence. On a final note, the existence of universal dynamical systems, which in general are infinite dimensional, suggests an additional possibility. In general, mapping data non-linearly into a high-dimensional space typically makes the data linearly separable and thus easier to learn. The dynamical version of this in conjunction with the possibility that the neural infrastructure supports dynamics of very high-dimensionality might suggest that brains can learn 'traces' and thus acquire a rich spectrum of cognitive skills using what appears to be surprisingly stereotypic architecture at a microscopic level (Petersson, 2005b).

References

Badii, R., & Politi, A. (1997). Complexity: Hierarchical Structures and Scaling in Physics. Cambridge, UK: Cambridge University Press.

Chomsky, N. (2005). Three factors in language design. Linguistic Inquiry, 36, 1 - 22.

Davis, M. D., Sigal, R., & Weyuker, E. J. (1994). Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science (2 ed.). San Diego, CA: Academic Press.

Elman, J. L. (1990). Finding structure in time. Cognitive Science, 14, 179-211.

Goldie, C. M., & Pinch, R. G. E. (1991). Communication Theory. Cambridge, UK: Cambridge University Press.

Gomez, R. L., & Gerken, L. (2000). Infant artificial language learning and language acquisition. Trends in Cognitive Sciences, 4, 178-186.

Grenholm, P. (2003). Artificial Grammar Learning - Rules and Statistics. Stockholm, Sweden: Karolinska Institutet.

Hauser, M. D., Chomsky, N., Fitch, W. T. (2002). The faculty of language. Science 198, 1569-1579.

Haykin, S. (1998). Neural Networks: A Comprehensive Foundation, 2nd ed. Upper Saddle River, NJ: Prentice Hall.

Joshi, A. K., & Schabes, Y. (1997). Tree-adjoining grammars. In A. Salomaa (Ed.), Handbook of Formal Languages (Vol. 3: Beyond words). Berlin: Springer Verlag.

Lasota, A., & Mackey, M. C. (1994). Chaos, Fractals, and Noise: Stochastic Aspects of Dynamics. New York: Springer-Verlag.

Legenstein, R., & Maass, W. (2005, preprint). What makes a dynamical system computationally powerful? In S. Haykin, J. C. Principe, T. J. Sejnowski, and J. G. McWhirter (eds.), New Directions in Statistical Signal Processing: From Systems to Brain. Cambridge, MA: MIT Press.

Maass, W. (1996). Lower bounds for the computational power of networks of spiking neurons. Neural Computation 8, 1-40.

McCauley, J. L. (1993). Chaos, Dynamics, and Fractals: An Algorithmic Approach to Deterministic Chaos. Cambridge, UK: Cambridge University Press.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophysics 5, 115-133.

Moore, C. (1991). Generalized shifts: Unpredictability and undecidability in dynamical systems. Nonlinearity, 4, 199-230.

Petersson, K. M., Forkstam, C., & Ingvar, M. (2004). Artificial syntactic violations activate Broca's region. Cognitive Science, 28, 383-407.

Petersson, K. M. (2004). The human brain, language, and implicit learning. Impuls - Journal of Psychology (Norwegian), 58(3), 62-72.

Petersson, K. M. (2005a). On the relevance of the neurobiological analogue of the finite-state architecture. Neurocomputing, 65-66, 825-832.

Petersson, K. M. (2005b). Learning and Memory in the Human Brain. Stockholm, Sweden: Karolinska University Press.

Reber, A. S. (1967). Implicit learning of artificial grammar. J. Verb. Learn. & Verb. Behav., 6, 855-863.

Rogers, H. (2002). Theory of Recursive Functions and Effective Computability. Cambridge, MA: MIT Press.

Siegelmann, H. T., & Fishman, S. (1998). Analog computation with dynamical systems. Physica D, 120, 214-235.