Infinite RAAM: A Principled Connectionist Basis for Grammatical Competence

Simon Levy, Ofer Melnik and Jordan Pollack
levy, melnik, pollack@cs.brandeis.edu
Dynamical and Evolutionary Machine Organization
Volen Center for Complex Systems,
Brandeis University, Waltham, MA 02454, USA
February 6, 2000

Abstract

This paper presents Infinite RAAM (IRAAM), a new fusion of recurrent neural networks with fractal geometry, allowing us to understand the behavior of these networks as dynamical systems. Our recent work with IRAAMs has shown that they are capable of generating the context-free (non-regular) language $a^n b^n$ for arbitrary values of n. This paper expands upon that work, showing that IRAAMs are capable of generating syntactically ambiguous languages but seem less capable of generating certain context-free constructions that are absent or disfavored in natural languages. Together, these demonstrations support our belief that IRAAMs can provide an explanatorily adequate connectionist model of grammatical competence in natural language.

Natural Language Issues

In an early and extremely influential paper, Noam Chomsky (1956) showed that natural languages (NL's) cannot be modeled by a finite-state automaton, because of the existence of center-embedded constructions. A second and equally important observation from this work was that a minimally adequate NL grammar must be ambiguous, assigning more than one structure (interpretation) to some sentences, for example, *They are flying planes.*

The first observation led to the development of Chomsky's formal hierarchy of languages, based on the computational resources of the machines needed to recognize them. In this hierarchy, Chomsky's observation about center-embedding is expressed by saying that NL's are non-regular; i.e., they cannot be generated by a grammar having only rules of the form $A \rightarrow bC$, where A and C are non-terminal symbols and b is a terminal symbol.

Whether NL's are merely non-regular, belonging in the next, context-free (CF) level of the Chomsky hierarchy, or are more powerful, belonging further up in the hierarchy, became the subject of heated debate (Higginbotham 1984; Postal and Langendoen 1984; Shieber 1985). Non-CF phenomena such as reduplication/copying (Culy 1985) and crossed serial dependencies (Bresnan, Kaplan, Peters, and Zaenen 1982) suggested that a more powerful approach, using syntactic transformations (Chomsky 1957) was called for, but some researchers criticized transformations as having arbitrary power and thus failing to constrain the types of languages that could be expressed (Gazdar 1982). Further criticism of the entire formal approach came from observing that even CF grammars (CFGs) had the power to generate structures, such as a sequence followed by its mirror image, that did not seem to occur in NL (Manaster-Ramer 1986), or which placed an extraordinary burden on the human parsing mechanism when they did occur (Bach, Brown, and Marslen-Wilson 1986).

Connectionism and Natural Language

While debates about the complexity of NL were raging, connectionism was beginning to awaken from a fifteen-year sleep. In connectionist models many researchers found a way of embodying flexibility, graceful degradation, and other non-rigid properties that seem to characterize real cognitive systems like NL. This research culminated the publication of a highly controversial paper by Rumelhart and McClelland (1986) which provided a connectionist account of part of the grammar of English using a feed-forward neural network. The paper was soon criticized by more traditional cognitive scientists (Fodor and Pylyshyn 1988; Pinker and Prince 1988), who cited the non-generative nature of such connectionist models as a fundamental shortcoming of the entire field.

Partly in response to these criticisms, many connectionists have spent the past decade investigating network models which support generativity through recurrent (feedback) connections (Lawrence, Giles, and Fong 1998; Rodriguez, Wiles, and Elman 1999; Williams and Zipser 1989). The research we present here is an attempt to contribute to this effort while focusing as strongly as possible on the natural language issues described above. Such an attempt faces a number of challenges.

First, despite analysis of how a network's dynamics contribute to its generativity, it is often uncertain whether the dynamics can support generation of well-formed strings beyond a certain length. That is, it is unknown whether the network has a true "competence" for the language of which it has learned a few exemplars, or is merely capable of generating a finite, and hence regular, subset of the language. [1] Second, it is often easier to model weak, rather than strong generative capacity, by building networks that generate or recognize strings having certain properties, without assigning any syntactic structure to the strings. Third, this lack of syntactic structure inhibits the formulation of an account of syntactic ambiguity in such networks, making them less plausible as models of NL.

[1] To be fair, not all connectionists, or cognitive scientists, take seriously the notion that human language has infinite generative capacity. Though we obviously do not have the resources to argue the issue here, we are certain that a model with a provably infinite competence would be more persuasive to the cognitive science community as a whole than would a model without one.

In sum, we are concerned with formulating a recurrent network model that rigorously addresses the set of criteria that emerged from the long debate over the complexity of NL. As an candidate,the remainder of this paper presents a new formulation of RAAM (Pollack 1990), a recurrent network model that addresses the NL issues in a principled way.

Traditional RAAM

Recursive Auto-Associative Memory or RAAM (Pollack 1990) is a method for storing tree structures in fixed-width vectors by repeated compression. Its architecture consists of two separate networks – an encoder network, which can construct a fixed-dimensional code by compressively combining the nodes of a symbolic tree from the bottom up, and a decoder network, which decompresses a fixed-width code into its two or more components. The decoder is applied recursively until it terminates in symbols, reconstructing the tree. These two networks are simultaneously trained as an autoassociator with time-varying inputs. If the training is successful, the result of bottom up encoding will coincide with top down decoding.

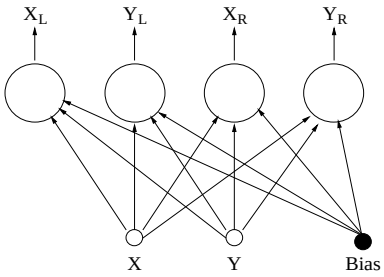
Following the publication of (Pollack 1990), RAAM gained widespread popularity as a model of NL syntax. Some researchers (Blank, Meeden, and Marshall 1991) found it an attractive way of "closing the gap" between the symbolic and sub-symbolic paradigms in cognitive science. Others (Van Gelder 1990) saw in RAAM a direct and simple refutation of the traditional cognitive scientists' backlash against connectionism, and went as far as to show how traditional syntactic operations like transformations could be performed directly on RAAM representations (Chalmers 1990). As the power of the RAAM model became apparent, variants began to emerge. These included the Sequential RAAMs of (Kwasny and Kalman 1995), which showed how a RAAM could behave like a linked list, and the Labeling RAAMs of (Sperduti 1993), which encoded labeled graphs containing cycles.

In short, RAAM seemed to hold a great deal of promise as a general connectionist solution to encoding not just NL syntax, but all sorts of structured representations.

Still, RAAM was plagued by an apparently diverse set of problems, most notably a failure to scale up to realistically large structures. We believe that these problems can be traced to the original formulation of the RAAM decoder, which works in conjunction with a logical "terminal test", answering whether or not a given representation requires further decoding. The default terminal test merely asks if all elements in a given code are boolean, e.g. above 0.8 or below 0.2. This analog-to-binary conversion was a standard interface in back-propagation research of the late 1980's to calculate binary functions from real-valued neurons. However, although it enabled the initial discovery of RAAM training, it led to several basic logical problems which prevented the scaling up of RAAM: 1) The "Infinite Loop" problem is that there are representations which "break" the decoder by never terminating. In other words, some trees appear "infinitely large" simply because their components never pass the terminal test. This behavior breaks computer program implementations or requires depth checking. 2) The "Precision vs. Capacity" problem is that tighter tolerances lead to more decoding errors instead of a greater set of reliable representations. 3) The "Terminating Non-Terminal" problem arises when there is a "fusion" between a non-terminal and a terminal, such that the decoding of an encoded tree terminates abruptly.

In the following section of this paper we present a new formulation of RAAM networks based on an analysis of the iterated dynamics of decoding, that resolves all these problems completely. This formulation leads to a new "natural terminal test", a natural labeling of terminals, and an inherently higher storage capacity.

New RAAM Formulation



$$X_L = \frac{1}{1+e^{-(w_{LXX}x+w_{LXY}y+w_{LX})}}$$
$$Y_L = \frac{1}{1+e^{-(w_{LYX}x+w_{LYY}y+w_{LY})}}$$
$$X_R = \frac{1}{1+e^{-(w_{RXX}x+w_{RXY}y+w_{RX})}}$$
$$Y_R = \frac{1}{1+e^{-(w_{RYX}x+w_{RYY}y+w_{RY})}}$$

Figure 1: An example RAAM decoder that is a 4 neuron network, parameterized by 12 weights. Each application of the decoder converts an (X, Y) coordinate into two new coordinates.

Consider the RAAM decoder shown in figure 1. It consists of four neurons that each receive the same (X, Y) input. The output portion of the network is divided into a right and a left pair of neurons. In the operation of the decoder the output from each pair of neurons is recursively reapplied to the network. Using the RAAM interpretation, each such recursion implies a branching of a node of the binary tree represented by the decoder and initial starting point. However, this same network recurrence can also be evaluated in the context of dynamical systems. This network is a form of *iterated function system* or IFS (Barnsley 1993), consisting of two pseudo-contractive transforms which are iteratively applied to points in a two-dimensional space.

In the past we have examined the applicability of the IFS analogy to other interpretations of neural dynamics (Blair and Pollack 1997; Kolen 1994; Melnik and Pollack 1998; Stucki and Pollack 1992). But in the context of RAAMs the main interesting property of contractive IFSes lies in the trajectories of points in the space. For contractive IFSes the space is divided into two sets of points. The first set consists of points located on the underlying attractor (fractal attractor) of the IFS. The second set is the complement of the first, points

that are not on the attractor. The trajectories of points in this second set are characterized by a gravitation towards the attractor. Finite, multiple iterations of the transforms have the effect of bringing the points in this second set arbitrarily close to the attractor.

As noted before, the Infinite Loop and Terminating Nonterminal problems arise from an insufficient terminal test. Since some trajectories never leave the attractor and all others eventually hit the attractor. The only terminal test that guarantees the termination of all trajectories of the RAAM (IFS) is a test that includes all the points of the attractor itself.

By taking the terminal test of the decoder network to be "on the attractor", not only are problems of infinite loops and early termination corrected, but it is now possible to have extremely large sets of trees represented in small fixed-dimensional neural codes. The attractor, being a fractal, can be generated at arbitrary resolution. In this interpretation, each possible tree, instead of being described by a single point, is now an *equivalence class* of initial points sharing the same tree-shaped trajectories to the fractal attractor. For this formulation, the set of trees generated and represented by a specific RAAM is a function of the weights, but is also governed by how the initial condition space is sampled, and by the resolution of the attractor construction. Note that the lower-resolution attractors contain all the points of their higher-dimensional counterparts (they cover them); therefore, as a coarser terminal set, they terminate trajectories earlier and so act to "prefix" the trees of the higher-dimensional attractors.

Two last pieces complete the new formulation. First, the encoder network, rather than being trained, is constructed directly as the mathematical inverse of the decoder. The terminal set of each leaf of a tree is run through the inverse left or right transforms, and then the resultant sets are intersected and any terminals subtracted. This process is continued from the bottom up until there is an empty set, or we find the set of initial conditions which encode the desired tree.

Second, using the attractor as a terminal test also allows a natural formulation of assigning labels to terminals. Barnsley (1993) noted that each point on the attractor is associated with an address which is simply the sequence of indices of the transforms used to arrive on that point from other points on the attractor. The address is essentially an infinite sequence of digits. Therefore to achieve a labeling for a specific alphabet we need only consider a sufficient number of significant digits from this address.

Example of New RAAM Formulation

In this section, we describe how we obtain the attractor and the trees for a RAAM decoder of the sort shown in figure 1. The decoder weights in the present example were obtained by a hill-climbing search for an aesthetically appealing attractor, but the demonstration is valid for any set of decoder weights.

Recall that we are treating the decoder as an IFS that maps each input point (X, Y) in the range [0,1] to two other points (X_L, Y_L) and (X_R, Y_R) in the same range. To generate the attractor of the IFS, we first apply the two mappings (transforms) to the entire unit square at some fixed resolution. We then re-apply the transforms to the resulting set of points. We repeat this operation until the transforms do not change the set of points any further at that resolution. Hence, we can visualize the behavior of the decoder in the unit square by examining the set of points obtained through iterated applications of the two transforms.

In figure 2, we have applied the transforms once to all points in the unit square, obtaining two large, overlapping regions, corresponding to the left and right transforms of all the original points. Note that some points are part of both the left and right regions.

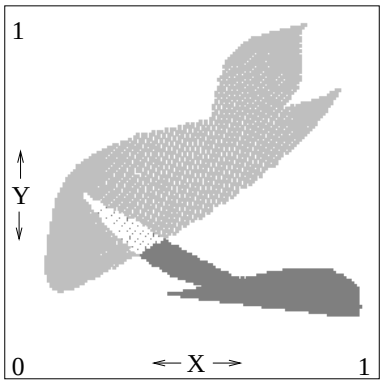


Figure 2: The unit square after one application of the transforms. The attractor is shown in gray: dark gray = points reachable from attractor on left transform, light gray = points reachable on right. The small white wedge where the gray areas overlap contains "ambiguous" attractor points reachable on both transforms.

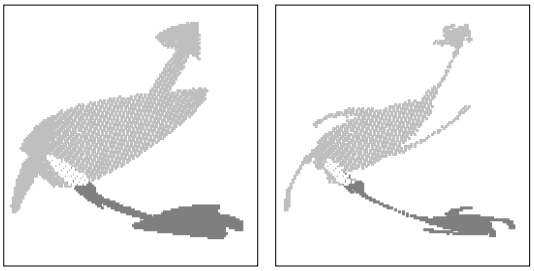


Figure 3: The unit square after two and five applications of the transforms.

Figure 3 shows the unit square after another iteration of the transforms, and after five such iterations. Figure 4 shows the final "Galaxy" attractor obtained when further iterations fail to produce any more contraction. Like any fractal, this attractor exhibits self-similarity, with the two longest arms of the galaxy ending in shapes like that of the whole attractor.

Figure 4 also shows how we derive the tree (1 (1 2)) from a point not on the attractor. Starting at a point not on the attractor (the small circle at the top of the figure), the left transform (dashed line) takes us immediately to the attractor; specifically, to an attractor region labeled 1, indicating that this region is reachable from the other attractor points on the left (first) transform only. Hence our tree so far is (1...). The *right* transform of the point at the top takes us to another point

not on the attractor, indicated by the circle in the lower left part of the figure. Like the first point, this point goes to the attractor region labeled 1 on its left transform; however, it also goes to the attractor on its right transform; specifically, to the region labeled 2, which indicates that this region is reachable from the other attractor points on the right (second) transform only. So this second point decodes the tree (1 2), and its parent tree is (1 (1 2)), completing the derivation.

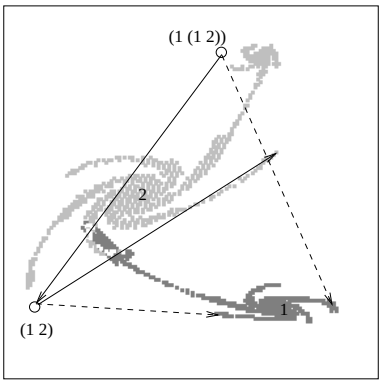


Figure 4: The final attractor, showing derivation of the tree (1 (1 2)) and its daughter tree (1 2). The left transform is shown as a dashed line, and the right transform as a straight line.

By repeating this process for every point not on the attractor, we can map out the set of all trees decoded by the RAAM at a given resolution. As described earlier, each tree in this set corresponds to an equivalence class of points that all decode to that tree. Points in the same class tend to cluster together, giving us an interesting way of laying out the RAAM's language spatially. Figure 5 shows this phenomenon for a RAAM that we hill-climbed to decode the language $a^n b^n$ (described in the next section), with grayscale denoting tree equivalence classes rather than attractor points. The dramatic striping pattern of the equivalence classes in this figure is not inherent in the fractal RAAM model, but derives from the comparatively elegant solution that hill-climbing produced for this language.

Linguistic Advantages of New RAAM

As we described earlier, the new RAAM formulation thoroughly addresses the three shortcomings of the traditional RAAM model. Infinite loops and terminating non-terminals are both eliminated by making the terminal test be a test of whether or not a point is on the fractal attractor of the RAAM decoder.

Furthermore, the new formulation provides a principled account of generativity (grammatical competence). By treating the RAAM as a fractal that can be generated at any arbitrary resolution, we can increase the generative capacity of the RAAM without bound, giving us a model that scales perfectly: hence the name Infinite RAAM (IRAAM). As we have recently shown (**?**), it is a straightforward matter to hill-climb the weights for an IRAAM that generates all and only the strings in the language $a^n b^n \cup a^n b^{n+1}, n \leq 5$.

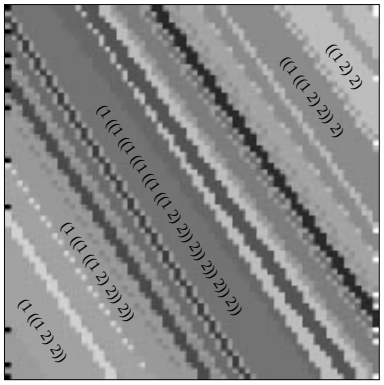


Figure 5: Tree equivalence classes for the $a^n b^n$ system. Attractor points cluster at extreme left (colored black, labeled 1 or a) and right (colored white, labeled 2 or b).

Briefly, the dynamics of the network are such that for any point in the unit square, one of the two transforms of the point is guaranteed to be on the attractor. This behavior corresponds to the terminal component of a recursive grammar in Chomsky Normal Form for the language. In addition, the left transform of any point ends up on the left side of the unit square ($x = 0$) and the right transform ends up on the right side ($x = 1$). Hence, successive application of left/right/left... transforms leads to a zigzag dynamics that balances a's on the left with b's on the right, until a zig or zag lands on the attractor and terminates the oscillation. This behavior corresponds to the recursive component of the grammar. In (**?**), we provide a constructive proof for obtaining these behaviors at any resolution.

The proof gives us an exact IRAAM "competence" model for this non-regular CF language. Specifically, we show that there exists a set of weights for which a RAAM with an attractor generated at a predetermined resolution contains all and only the trees in the $a^n b^n$ language. Performance limitations on the sizes of the trees actually produced derive from the resolution at which the non-attractor unit space is sampled, and not from an arbitrary stipulation or a breakdown of the model.

This infinite competence is not the only thing that IRAAM brings to connectionist NL modeling, however. Because IRAAM is a method of encoding and decoding *trees*, not just strings, its strong generative capacity is known. We can therefore use IRAAM as a direct model of hierarchical linguistic structure. An immediate implication of this result is that an IRAAM can be used as a parser and not just a recognizer. To the extent that real NL processing involves the assignment of meaning to strings based on structure, and not merely grammaticality judgments, this ability represents a significant advance in the application of connectionism to NL.

Finally, and perhaps most interesting, is the way in which IRAAM handles syntactic ambiguity. Consider the fractal addressing scheme that we described earlier. Each terminal point (word) on the attractor is associated with an address which is simply the sequence of indices of the transforms taken to arrive on the attractor point from other points on the attractor. Given K transforms, we would therefore assume

each digit in the sequence would fall in the range $1, 2, ..., K$. For example, a binary-branching IRAAM, with two transforms, would have terminals with address digits 1 and 2. Using a one-digit address, this effectively puts each word into one of K "part of speech" equivalence classes.

This is not the whole story, though. Because there can be more than one path to a given terminal from some other terminal on the attractor, some terminals will have "ambiguous" addresses, containing digits out of the range $1..K$, to express the fact that more than one transform was taken to arrive at that point in the sequence. Continuing the linguistic analogy, this ambiguity corresponds to a given word's belonging to more than one part of speech, as in Chomsky's "flying planes" example, where *flying* can be either a verb or an adjective. For the binary-branching IRAAM example, if a given point had both a left and right inverse on the attractor, a one-digit address for that point would have to be a symbol other than 1 or 2. In general, for a K-ary IRAAM, there are $2^K - 1$ possible one-digit addresses, consisting of K unambiguous values and $2^K - K - 1$ ambiguous values.

This fact has great linguistic importance for IRAAM, for the following reason: typically (but not exclusively), an IRAAM decoder will favor putting the kth non-ambiguous terminal class in the kth position in a string of terminals, because the same set of weights is used to generate the attractor and the transients to the attractor. The likeliest nonterminal structure of a binary-branching IRAAM will therefore be (1 2), with structures (1 1), (2 1) and (2 2) being possible but less likely to occur. If, however, this IRAAM contains ambiguous terminals, it will very likely decode the structures (1 3), (3 2) and (3 3) as well.

Returning to the "flying planes" example, let us assign unambiguous verbs like *are* the category 1, unambiguous nouns like *planes*[2] the category 2, and the ambiguous *flying* the category 3. With this assignment, the natural ability of a binary IRAAM to decode the structures (1 (3 2)) and ((1 3) 2) gives us both parses of the expression *are flying planes*. Hence, we have an existence proof of a RAAM that can deal with syntactic ambiguity and non-deterministic grammars.

In short, we believe that IRAAM not only solves the problems of the earlier RAAM model, but also addresses the linguistic inadequacies of recurrent neural net models that we discussed earlier.

What IRAAM Can't Do

In the first section of this paper we outlined two linguistic criteria for a plausible NL model: the model should be able to handle "slightly" non-CF phenomena like copying and crossed serial dependencies and should also be incapable of handling CF phenomena absent from or deprecated in NL's, like mirror-image constructions, or should incur a relatively high cost in producing or parsing those structures.

To investigate the latter point, we tested the ability of the IRAAM model shown in figure 1 to "learn" the context-free languages $a^n b^n$ and ww^R, $w \in \{a, b\}$. The training set consisted of the first 14 exemplars of each language (enumerated in increasing order of length)[3], with the fractal address 1 representing a and 2 representing b. Hill-climbing was used to learn the weights. Both the initial weights and the noise added to each weight came from a Gaussian distribution with zero mean and a standard deviation of 5.0, with the added noise's standard deviation being scaled by the fraction of the training set missed. The resulting weights were used to generate trees on an IRAAM with a resolution of 2^{-7}. The attractor was generated at that resolution and the initial starting point space was also sampled at that resolution.

Hill-climbing did not produce good results on either of these languages; the average success was six out of 14 strings covered for both languages. It is, however, instructive to look out *how* those successes were achieved. Comparing the best hill-climbed networks from each language (10 strings covered), we found that most of the strings generated by the $a^n b^n$ network fit the general pattern of the training set: 74% of the strings fit the pattern $a^n b^n$. For the best ww^R network, however, only 14% fit the pattern ww^R. In other words, the $a^n b^n$ network was actually producing mostly "grammatical" strings, whereas the ww^R network was essentially guessing.

We attribute these results to IRAAM's aforementioned tendency to put symbols of one class (a) on the left side of a branch and symbols of another class (b) on the right side. In other words, trees of the form $(a\ b)$, $(a\ (a\ b))$, $((a\ b)\ b)$, $(a\ (a\ (a\ b)))$, $(((a\ b)\ b)\ b)$, are much more "natural" for an IRAAM than are trees of the form $(a\ a)$, $(b\ b)$, $(b\ a)$. But it is precisely the latter types of trees that are used as building blocks for the mirror-image language ww^R. This bias makes the mirror-image language much harder for an IRAAM to learn than the counting language $a^n b^n$, despite the fact that both are expressible by a simple CFG.

Although this result is by no means a proof of any sort, we consider it interesting for two reasons. First, it suggests that the languages generable by an IRAAM share an important formal property with NL, namely, the avoidance of mirror-image constructions. Second, the result illustrates how IRAAM imposes a constraint between the terminal symbol "semantics" and the nonterminal "syntax." This constraint is absent from the definition of CFG's (or of any grammar in the Chomsky hierarchy), where any terminal symbol can appear anywhere. To the extent that individual natural languages favor putting a given part of speech in fixed locations in a sentence or phrase (e.g., English generally has subject-verb-object, Japanese subject-object-verb), IRAAM appears to have an advantage over traditional grammars as a model of NL.

Conclusion and Interpretations

We have demonstrated a new formulation of RAAM, which, by using a fractal attractor as a terminal test, enables the model to show competence and ambiguity, to represent a variety of tree structures, and not to represent deprecated mirror-image structures. We plan to relate this new formula-

[2] Readers troubled by the possibility of *planes* being a singular verb (*The carpenter planes the wood*) can substitute *cars* or some other unambiguous noun here.

[3] The number 14 was chosen because it allowed us to include all the members of ww^R for $|w| \leq 3$. This language has more strings of a given length than the language $a^n b^n$, which meant that the exemplars of the latter had to be longer in order to enumerate the first 14 of them. In effect, this makes the $a^n b^n$ task *harder* than the ww^R task .

tion to linguistic formalisms like Tree-Adjoining Grammars (Joshi and Schabes 1997) and Categorial Grammars (Steedman 1999) having similar properties. We hypothesize that this relation may be achieved through the use of multiplicative connections to gate lexical varieties into naturally recursive dynamics.

Our work is by no means complete; nor do we mean to imply that NL grammar can be represented in four neurons with 12 weights! On the other hand, the principle of contractive maps and the emergence of fractal attractors in the limit behavior of nonlinear systems are mathematical facts, and have been used successfully in image-compression systems. Recent work by Tabor (1998) provides further evidence for the relevance of such principles to connectionist modeling of natural language. We now have reason to believe that these principles, under the right interpretation and scale, can support a neurally plausible universal grammar.

References

Bach, E., C. Brown, and W. Marslen-Wilson (1986). Crossed and nested dependencies in german and dutch: A psycholinguistic study. *Language and Cognitive Processes 1*(4), 249–262.

Barnsley, M. (1993). *Fractals everywhere*. New York: Academic Press.

Blair, A. and J. Pollack (1997). Analysis of dynamical recognizers. *Neural Computation 9*(5), 1127–1142.

Blank, D., L. Meeden, and J. Marshall (1991). Exploring the symbolic/subsymbolic continuum: A case study of raam. Technical Report TR332, Computer Science Department, University of Indiana.

Bresnan, J., R. Kaplan, S. Peters, and A. Zaenen (1982). Cross-serial dependencies in dutch. *Linguistic Inquiry 13*(4), 613–634.

Chalmers, D. (1990). Syntactic transformations on distributed represenations. *Connection Science 2*, 53–62.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on information theory 2*, 113–124.

Chomsky, N. (1957). *Syntactic Structures*. Mouton.

Culy, C. (1985). The complexity of the vocabulary of bambara. *Linguistics and Philosophy 8*, 345–351.

Fodor, J. and Z. Pylyshyn (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition 28*, 3–71.

Gazdar, G. (1982). Phrase structure grammar. In P. Jacobson and G. Pullum (Eds.), *The Nature of Syntactic Representation*. Reidel.

Higginbotham, J. (1984). English is not a context-free language. *Linguistic Inquiry 15*(2), 225–234.

Joshi, A. and Y. Schabes (1997). Tree-adjoining grammars. In G. Rozenberg and A. Salomaa (Eds.), *Handbook of Formal Languages and Automata*, Chapter 3. Berlin: Springer Verlag.

Kolen, J. (1994). *Exploring the Computational Capabilities of Recurrent Neural Networks*. Ph. D. thesis, Ohio State.

Kwasny, S. and B. Kalman (1995). Tail-recursive distributed representations and simple recurrent neural networks. *Connection Science 7*(1), 61–80.

Lawrence, S., C. Giles, and S. Fong (1998). Natural language grammatical inference with recurrent neural networks. *IEEE Transactions on Knowledge and Data Engineering*, to appear.

Manaster-Ramer, A. (1986). Copying in natural languages, context-freeness, and queue grammars. In *Proceedings of the 24th meeting of the Association for Computational Linguistics*, pp. 85–89.

Melnik, O. and J. Pollack (1998). A gradient descent method for a neural fractal memory. In *WCCI 98*. International Joint Conference on Neural Networks: IEEE.

Pinker, S. and A. Prince (1988). On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition 28*, 73–193.

Pollack, J. (1990). Recursive distributed representations. *Artifical Intelligence 36*, 77–105.

Postal, P. and D. Langendoen (1984). English and the class of context-free languages. *Computational Linguistics 10*(3–4), 177–181.

Rodriguez, P., J. Wiles, and J. Elman (1999). A recurrent neural network that learns to count. *Connection Science 11*, 5–40.

Rumelhart, D. and J. McClelland (1986). On learning the past tenses of english verbs. In D. Rumelhart and J. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Volume 2. MIT.

Shieber, S. (1985). Evidence against the context-freeness of natural language. *Linguistics and Philosophy 8*, 333–343.

Sperduti, A. (1993). Labeling raam. Technical Report TR-93-029, International Computer Science Institute.

Steedman, M. (1999). Categorial grammar. In R. Wilson and F. Keil (Eds.), *The MIT Encyclopedia of Cognitive Sciences*. MIT.

Stucki, D. and J. Pollack (1992). Fractal (reconstructive analogue) memory. In *14th Annual Cognitive Science Conference*, pp. 118–123.

Tabor, W. (1998). Dynamical automata. Technical Report TR98-1694, Computer Science Department, Cornell University.

Van Gelder, T. (1990). Compositionality: a connectionist variation on a classical theme. *Cognitive Science 14*, 355–384.

Williams, R. and D. Zipser (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation 1*, 270–280.