

Becoming Recursive: Toward a Computational Neuroscience Account of Recursion in Language and Thought

Simon D. Levy

1 Introduction

First you curse... and then you recurse.

– Harry Mairson, “In Praise of the Research University”

After remaining more or less in the background for several decades, the issue of recursion in human languages has returned to the fore. The current, vigorous debate is about not just recursion, but the very idea of linguistic universals as a coherent scientific notion, and the nature of language itself. The prevailing view is that recursion – the ability to produce an unbounded number of well-formed sentences through the combinatorial operations of syntax – is a fundamental property of the “faculty of language in the narrow sense” (FLN), *i.e.*, that recursion is what differentiates human language from other forms of animal communication (Hauser, Chomsky, and Fitch 2002). As an explicit challenge this view, Everett (2005) has documented the syntax of Pirahã, a language that appears to have no embedded clauses, recursive possessives, or other syntactic structures suggesting this capacity. Everett offers Pirahã as a fatal counter-example to the universality of recursion, and questions the utility of searching for other sorts of linguistic universals as well.

Although this debate has involved linguistics, and sometimes politics, we suggest that it might also be fruitfully informed by neuroscience. In particular, we will present a model of event semantics based on neurally plausible role/filler bindings, and describe how recursion, like other syntactic processes, can be viewed as the by-product of “unpacking” and serializing the contents of these representations. A rough outline of our argument: we start by enumerating some basic assumptions or ground rules about what any model of human language and cognition should look like. We then provide a review of simple vector arithmetic for the benefit of readers unfamiliar with this branch of mathematics, and we show how high-dimensional vectors can be used as a plausible model of role/filler semantics and other mental representations without violating any of our stated assumptions. Next we sketch the rudiments of a model of language that treats grammar as a process of transduction between these mental representations and sequences of symbols. We present a brief account of how this language model can be related to traditional syntactic categories like agent and theme, and conclude by setting the model in the context of current research on the sources of linguistic universals.

2 Basic assumptions

We believe that any theory of language or other mental activity must be constrained by a basic set of ground rules. Some of these rules come from general scientific principles, others from concerns specific to cognitive linguistic modeling.

The most basic principle is Occam’s Razor: *entities should not be multiplied without necessity*. In more modern terms, a model with N components that explains a phenomenon adequately should be favored over one with $N+k$ components, for positive integers N and k . We are thinking here of the fact (noted by others in this volume) that although recursive grammars

can generate infinitely long sentences and sentences with unbounded recursive embedding, ordinary spoken language consists of short sentences with little or no recursion. A standard account of this somewhat embarrassing fact posits an infinite “competence” grammar constrained by a “performance” component such as short-term or working memory (Chomsky 1965). Perhaps because of the popularity of the concept of short-term memory in psychology as a whole, this two-component model has become the standard justification for recursive grammars, despite the strange empirical predictions they make. Though we do not wish to dispute the usefulness of concepts like short-term memory as high-level descriptions, we would argue for the advantages of a model in which the limitations on embedding and related operations arise from the same component or substrate that supports these operations.

Second, we take it as a given that all human beings think recursively, *i.e.*, that they can and often do have thoughts about the thoughts or mental states of others. Further, we note that many human languages express these thoughts recursively (*I think that you think that you're smarter than me.*) As the title of our article suggests, we therefore take it as an important challenge to our research program to account for how languages might come to represent recursive thoughts in recursive symbol sequences.

Our third basic assumption is that the task of grammar or any grammar-like model is to provide a mapping between structured meanings and sequences of symbols. By structure we mean the whole apparatus of semantic roles and fillers, variable binding, and unification that has been used by linguists and cognitive scientists to represent “who did what to whom” event semantics, logical propositions, and the like. Though it may not seem controversial, this assumption bears mentioning here. There is a long-standing generative tradition, echoed in recent connectionist modeling, that views languages as sets of strings and the task of a grammar for a language as generating all and only the strings belonging to the set. Such a model need take no account of the (propositional) meanings conveyed by these strings. Following a suggestion by Pullum and Scholz (this volume), we choose instead to model grammars as transducers between form and meaning. The constraints that our model places on recursion and other processes will fall out as the consequence of the limitations that our representations impose on the transduction process.

The issue of representation brings us to our final assumption: the primitives of our model, and the operations supported by these primitives, should be biologically plausible. More strongly: in the words of Kanerva (2008), we ask *what kinds of things suggested by the architecture of the brain, if we modeled them mathematically, could give some properties that we associate with mind?* By *architecture* we take Kanerva to refer not to specific anatomical structures, but instead to general properties of how the brain seems to represent conceptual information. Among such properties we would include content-addressability, robustness / graceful degradation in the presence of noise, and low-precision encoding distributed over a large number computing elements with a high degree of interconnection. These properties are notably absent in the representations employed in traditional symbol-manipulating systems like generative grammars and symbolic programming languages (LISP, Prolog), and in the ubiquitous symbol-manipulating devices called digital computers. First, manipulation of representations in such systems requires “chasing down pointers” to get at content (*e.g.*, tracing from the root of a tree structure to its leaf nodes). Second, because representations in a computer memory are stored locally at specific addresses, small changes to the representation can cause performance to degrade catastrophically. For example, changing the value of a single bit of the representation can produce a “dangling pointer”, causing arbitrarily large amounts of structure to become inaccessible. Third, implementing theoretically unbounded recursion in a finite, discrete device makes recursion work perfectly up to some arbitrary, fixed memory limit, after which it fails completely (stack overflow). In the next section we describe a representational scheme that exhibits the desired architectural properties.

3 Vector-based architectures for symbol manipulation

Insofar as linguists know about connectionist/distributed models of language, they typically are aware of the work of Rumelhart and McClelland (1986) on learning the past tense of English verbs, and the work of Elman (1990) on learning structure via sequence prediction. Though we share with these “connectionists” an appreciation of the appeal of distributed representations, we note that (1) neither model addresses the encoding and transduction of propositional meaning that concerns us here; (2) both models rely on the back-propagation algorithm (Rumelhart, Hinton, and Williams 1986), which is generally considered biologically and psychologically implausible (Crick 1989); and (3) as stated above, we consider symbol manipulation essential to cognition. We therefore agree with Holyoak and Hummel (2000), Marcus (1998), and others that the “eliminative connectionism” of Rumelhart and McClelland, Elman, *et al.* cannot account for the phenomena that interest us here. In the remainder of this section we present recent work in connectionism that we feel holds significant promise for providing such an account. Because this work builds on vector arithmetic, we start with a very brief overview of that topic. Readers familiar with vector arithmetic may want to skip to section 3.2.

3.1 Review of vector arithmetic

For our purposes here, a *vector* can be defined as an ordered list of items or *elements*. The number of items in the vector is called the vector’s *dimensionality*. Anyone who has taken high-school trigonometry will be familiar with the two-dimensional vectors of numbers that make up the Cartesian plane. Such vectors can be visualized either as points, or as line segments with an arrow on the end. Visualizing the vectors as points helps us to see the relationships (*e.g.*, clustering) among a large number of vectors. Visualizing the vectors as line segments helps us to compare the directions represented by a small number of vectors, using a familiar trigonometric function like cosine. Figure 1 shows these two different methods of visualization.

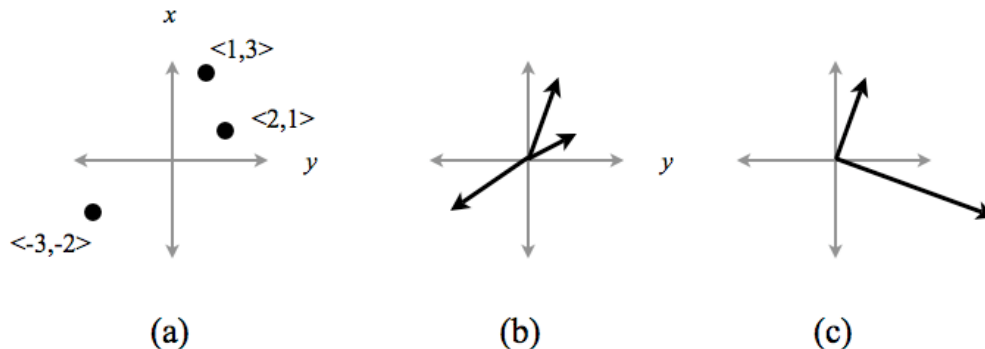


Figure 1. Two-dimensional vectors: (a) represented as points and (b) as line segments. (c) A pair of mutually orthogonal vectors.

If the cosine between two vectors is zero (*i.e.* they are at a right angle to each other), the vectors are said to *be mutually orthogonal*. Mutual orthogonality is very useful property: if we are using vectors to represent symbols, then mutually orthogonal vectors help to keep the symbols for different concepts distinct. Conversely, two similar concepts can be represented by vectors with a higher cosine (degree of similarity), as in the method of Latent Semantic Analysis (Foltz and Laham 1998).

3.2 Computing with high-dimensional, low-precision vectors

Although two- and three-dimensional vectors are easily visualized, they make a poor choice for representing large numbers of symbols: to distinguish among a large number of vectors we need a high degree of numerical precision, and there are, for example, only four mutually orthogonal directions for vectors in two-dimensional space. As discussed earlier, high-precision representations are biologically unrealistic and non-robust in the presence of noise. For this reason, we use high-dimensional vectors, typically 10,000 dimensions or more, with a very low numerical precision for each element. The lowest precision possible (one bit of precision, in information-theoretic terms) distinguishes between only two values. If the choice of value is made randomly for each element (“flipping a coin”), we can obtain very large numbers of mutually orthogonal vectors that are highly robust to noise. A surprisingly large number of the elements in a given vector can be arbitrarily changed before the vector becomes more similar to another vector than to its original form (Kanerva *to appear*).

Single-bit precision is most often associated with the numerical values 0 and 1. For the purposes of symbol manipulation, however, the values -1 and +1 give the vectors a useful property: every vector is its own multiplicative inverse; *i.e.*, multiplying a vector by itself elementwise produces a vector of all 1’s. This allows multiplication to act as a very fast form of associative memory. If we wish to associate two vectors A and B, we simply take their elementwise product, which we represent as $A \otimes B$. The item associated with A can then be retrieved by multiplying this association vector by A; that is, $A \otimes B \otimes A = B$. In a similar manner, multiplying by B retrieves A.

This very general method of association can be used to bind semantic roles to their fillers, to associate a concept with the word representing it, etc. To represent several pairs of associations, we can simply add them together elementwise; *e.g.*, the sum $A \otimes B + C \otimes D + E \otimes F$ represents simultaneously the association of A with B, C with D, and E with F. Summing vectors in this way produces values other than -1 and +1, but as the cosine function is not sensitive to magnitude, such values do not affect the basic operation of association. (To maintain realistically low precision, we can also normalize the sum to a small number of fixed values in the interval [-1,+1].) As with ordinary algebra, multiplication has precedence over and distributes over addition, and both operations are commutative and associative: for three vectors A, B, and C of the same dimensionality, $A \otimes B = B \otimes A$, $A+B = B+A$, $(A \otimes B) \otimes C = A \otimes (B \otimes C)$, $(A+B) \otimes C = A \otimes C + B \otimes C$, and $A \otimes (B+C) = A \otimes B + A \otimes C$. These properties make such representations easy to work with.

3.3 Cleanup memory

Summation does however introduce noise into the representation. Consider, for example, the result of retrieving the value associated with A from the summation $A \otimes B + C \otimes D + E \otimes F$:

$$(1) \quad A \otimes (A \otimes B + C \otimes D + E \otimes F) = B + A \otimes C \otimes D + A \otimes E \otimes F$$

This operation correctly retrieves the value B, but also retrieves the undesired associations $A \otimes C \otimes D$ and $A \otimes E \otimes F$. These bogus associations can be removed from the retrieved vector by passing it through a “cleanup memory” that stores only the meaningful vectors (A, B, C, D, E, F). Cleanup memory can be implemented in a biologically plausible way as a Hopfield network (Hopfield 1982) that maps each meaningful vector to itself, a variant of Hebbian learning (Hebb 1949). Such networks can reconstruct the original form of a vector or image from a highly degraded exemplar, via self-reinforcing feedback dynamics. How the meaningful vectors come to be in the memory is a topic of current research.

A cleanup memory enables other interesting operation as well; for example, the crucial operation of variable substitution used in the unification algorithm. Unlike classical unification, however, this vector-based substitution supports simultaneous substitution of an arbitrary number of new values. Consider for example a vector representing the association of the element X with A and Y with B. Say we wish to substitute P for A and Q for B. We can do this by multiplying our original vector by a vector that encodes the substitutions as pairwise representations:

$$(2) \quad \begin{aligned} & (X \otimes A + Y \otimes B) \otimes (P \otimes A + Q \otimes B) = \\ & X \otimes A \otimes P \otimes A + X \otimes A \otimes Q \otimes B + Y \otimes B \otimes P \otimes A + Y \otimes B \otimes Q \otimes B = \\ & X \otimes P + X \otimes A \otimes Q \otimes B + Y \otimes B \otimes P \otimes A + Y \otimes Q = \\ & X \otimes P + Y \otimes Q + NOISE \end{aligned}$$

where *NOISE* is material that is not in the cleanup memory and hence will be removed by it.

Crucially, none of the operations (association, summing, cleanup) changes the dimensionality of the vectors, unlike earlier forms of vector-based associative memory (*e.g.* Smolensky 1990). Such fixed-size representations also have a higher degree of biological plausibility than classical symbolic (tree, graph) representations, where the storage required increases with the amount of information being represented. It is even possible to implement such high-dimensional vector representations as spiking neurons (Eliasmith 2004). Just as important, however, is the fact that everything (symbols, associations, sets of associations) is represented by objects (vectors) of the same size. This uniformity of representation leads to an extraordinary flexibility in the behavior of systems built this way: combinations can act as entities in their own right, symbols can serve as both variables and constants, fillers can act as roles, etc. This behavior opens up possibilities for exploring issues like analogy, metaphor, and learning, as well as grammaticalization and other forms of language change, in a novel way (well beyond the scope of the present article).

3.4 Embedding

More to the point of the volume in which this article appears, the uniform representation of conceptual entities opens the door to embedding linguistic and conceptual entities inside similar entities, *i.e.*, to recursion. We start with simple embedding. Suppose that instead of associating one symbol with another, we wish to associate a symbol with an entire set of associations – as for example, the predicate BELIEVE embeds an entire proposition. This can be done by introducing a symbol representing the embedded set. Consider, for example a vector containing:

$$(3) \quad A \otimes B + C \otimes D + E \otimes (P \otimes Q + R \otimes S + T \otimes V)$$

where E is a symbol used to embed the association of P with Q, R with I, and T with V. From this vector we can retrieve the item associated with A by multiplying the vector by A, giving us B plus noise. The same goes for retrieving the items associated with B, C, and D. If however we try to retrieve the item associated with P by multiplying the vector by P, we get back nothing but noise, because no part of the resulting product reduces to a single symbol:

$$(4) \quad \begin{aligned} & P \otimes (A \otimes B + C \otimes D + E \otimes (P \otimes Q + R \otimes S + T \otimes V)) = \\ & P \otimes A \otimes B + P \otimes C \otimes D + P \otimes E \otimes P \otimes Q + P \otimes E \otimes R \otimes S + P \otimes E \otimes T \otimes V = \\ & P \otimes A \otimes B + P \otimes C \otimes D + E \otimes Q + P \otimes E \otimes R \otimes S + P \otimes E \otimes T \otimes V = \\ & NOISE \end{aligned}$$

We can think of this situation as analogous to infelicity of an exchange in which someone is told *John wonders whether Bill likes Mary* and responds by asking *WHO does Bill like?*

If we wish to retrieve the item associated with P, we must instead multiply by $E \otimes P$; in other words, by the embedded representation of P:

$$(5) \quad \begin{aligned} E \otimes P \otimes (A \otimes B + C \otimes D + E \otimes (P \otimes Q + R \otimes S + T \otimes V)) = \\ E \otimes P \otimes A \otimes B + E \otimes P \otimes C \otimes D + E \otimes P \otimes E \otimes P \otimes Q + E \otimes P \otimes E \otimes R \otimes S + E \otimes P \otimes E \otimes T \otimes V = \\ E \otimes P \otimes A \otimes B + E \otimes P \otimes C \otimes D + Q + E \otimes P \otimes E \otimes R \otimes S + E \otimes P \otimes E \otimes T \otimes V = \\ Q + NOISE \end{aligned}$$

3.5 Recursion and permutation

With a model of embedding in place, we can turn our attention what happens when a symbol embeds itself; *i.e.*, recursive embedding. As a simple example, consider the vector

$$(6) \quad E \otimes (A + E \otimes (B + C))$$

Careful readers may already have anticipated the problem that arises here: the symbol E cancels itself out, losing the inner embedding of B and C:

$$(7) \quad \begin{aligned} E \otimes (A + E \otimes (B + C)) = \\ E \otimes (A + E \otimes B + E \otimes C) = \\ E \otimes A + E \otimes E \otimes B + E \otimes E \otimes C = \\ E \otimes A + B + C \end{aligned}$$

One way of overcoming this problem is to apply a *permutation* (re-ordering) operator to the vector representation of an embedding before adding it to the vector representing the non-embedded material (Gayler 1998). A permutation is just a re-ordering of the elements of the vector; hence, like the other operators, permutation does not change the dimensionality of the vector. Typically, a random re-ordering is used, though systematic orderings like left- or right-shift work equally well. Functionally, permutation as like a quoting or bracketing operation that insulates the embedded material from the effects of self-cancellation. We can use the Greek letter Π to represent permutation; hence, the self-embedding expression above could be represented as $E \otimes (A + \Pi (E \otimes (B + C)))$. For greater depth of embedding, more permutations can be used:

$$(8) \quad E \otimes (A + \Pi_1 (E \otimes (B + C + \Pi_2 (E \otimes (D))))))$$

where the subscript on Π indexes a different permutation (and hence the depth of embedding). The permuted material can be made available for retrieval by applying the inverse permutation appropriate to that stage of retrieval.

A few points about this permutation-based recursive encoding are worth noting. First, despite the functional similarity that permutation and its inverse bear to pushing and popping operations in a traditional stack automaton, in no sense is the representation of the embedded structure “contained” inside or below the representations of the higher-level material. All representations are spread across an entire, single vector.

Second, permutation distributes over addition and multiplication, and (with the degenerate exception of the identity permutation) the composition of a permutation with itself is another permutation different from the original. Hence, a single permutation suffices to encode arbitrarily many levels of recursion; for example:

(9)

$$\begin{aligned}\Pi (A + (\Pi (B + \Pi (C)))) &= \\ \Pi (A) + \Pi (\Pi (B)) + \Pi (\Pi(\Pi (C))) &= \\ \Pi (A) + \Pi \circ \Pi (B) + \Pi \circ \Pi \circ \Pi (C) &= \\ \Pi^1 (A) + \Pi^2 (B) + \Pi^3 (C) &= \end{aligned}$$

where \circ stands for operator composition, and $\Pi^i(X)$ is the i^{th} permutation of vector X .

Finally, although permutation may seem like an ad-hoc solution to a technical problem, it has a good deal of neural plausibility. In a biological neural network, each neuron typically computes the sum (integral) over the input from many others (Dayhoff 1990). Hence, it is reasonable to model the mapping between two layers of neurons (each corresponding roughly to a hyperdimensional vector) as a sum of permutations from the first layer to the second. Recursive structure can then be built from the bottom up, using a feedback circuit like the one shown in Figure 2. This circuit accepts three vectors as inputs, computes the sum of the second two, multiplies this sum by the first (embedding) vector, permutes the resultant product, and passes the permuted vector back as the third input on the next pass. It can be used to encode right-branching recursive structures like the ones above.

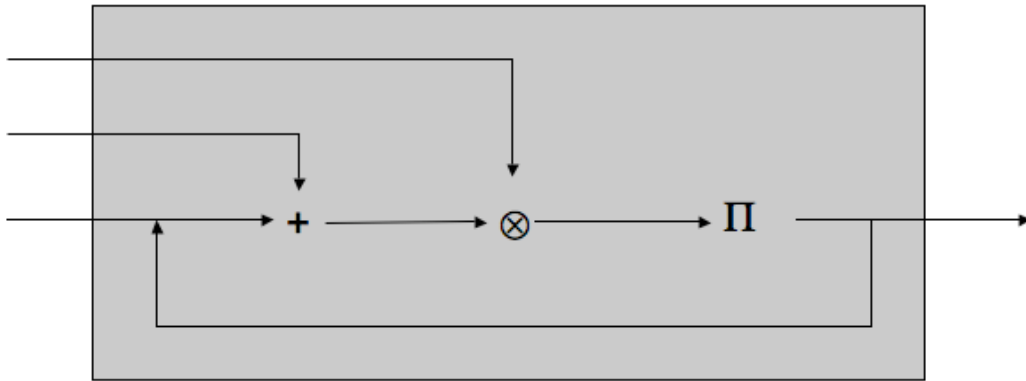


Figure 2. Hypothetical neural circuit for recursive embedding. See text for explanation.

3.6 Representing sequences

Thus far we have shown how sets of symbolic associations, including recursively structured sets, can be encoded using the vector operations multiply, add, and permute. What if we wanted to encode not just sets, but ordered sets; *i.e.*, sequences, such as the sequences of words in a sentence? Vector addition by itself is insufficient owing to its commutativity: $(A + B + C)$ is the same as $(A + C + B)$, $(B + C + A)$, etc. Following Plate (2003), we can use a positional or “trajectory” encoding for sequences in which the position of each element is represented as a position vector multiplied by that element. The sequence A, B, C can be encoded $P_1 \otimes A + P_2 \otimes B + P_3 \otimes C$, and in general, for an arbitrary vector P_i , the product $P_i \otimes X$ can be used to represent the element (symbol, association, set of associations, etc.) X being in the i^{th} position in a sequence. An actual temporal sequence can be generated from such an encoding by multiplying it by the

successive P_i , running the result through the cleanup memory, and outputting the cleaned-up result. If the product of P_i and the cleaned-up output is subtracted from the sequence vector on each such iteration, the values in the vector will eventually reach zero, allowing the process to halt. In this way the vector can be considered an abstract “motor program” for producing the sequence. A hypothetical neural feedback circuit implementing this algorithm is shown in Figure 3.

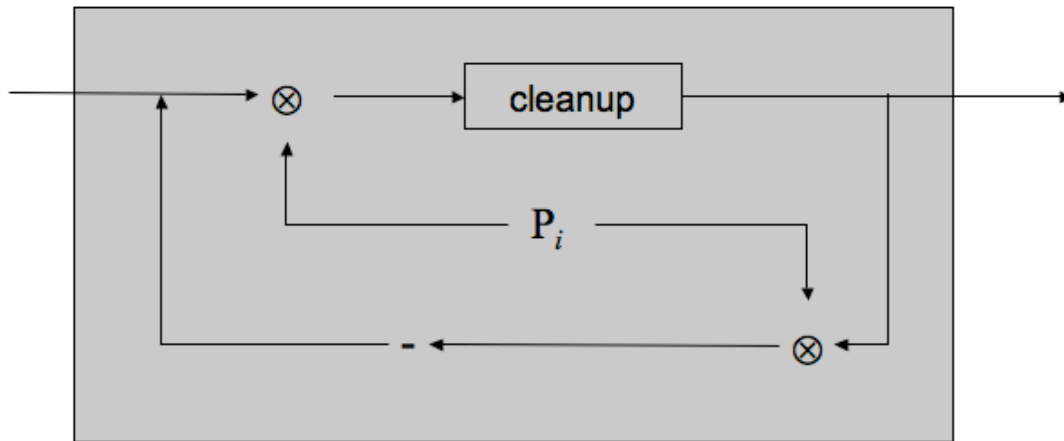


Figure 3. Hypothetical neural circuit for decoding sequences. See text for explanation.

4 Language transduction with the multiply-add-permute architecture

Having outlined our multiply-add-permute framework for symbol association, embedding, and sequencing, we can start applying it toward building a transduction model for language. Our general approach will be to model associations between propositional content and symbol sequences as multiplicative bindings. The simplest associations are between individuals and their names: $\text{JOHN} \otimes \text{John}$, $\text{MARY} \otimes \text{Mary}$, etc., where JOHN and MARY are vectors representing those persons and John and Mary are vectors representing their names. The next step is to build associations between propositions concerning such individuals and the sequences of symbols used to convey those propositions in sentences; for example:

$$(10) \quad (\text{ACTION} \otimes \text{KISS} + \text{AGENT} \otimes \text{MARY} + \text{PATIENT} \otimes \text{JOHN}) \otimes (\text{P}_1 \otimes \text{Mary} + \text{P}_2 \otimes \text{kissed} + \text{P}_3 \otimes \text{John})$$

The first half of this association is a purely conceptual, nonlinguistic vector representation of an event or logical proposition. Other researchers (Eliasmith and Thagard 2001) have used just this sort of representation to model analogical reasoning, independent of language. To produce a sentence representing this event, we multiply this vector by the association vector, yielding the vector $(\text{P}_1 \otimes \text{Mary} + \text{P}_2 \otimes \text{kissed} + \text{P}_3 \otimes \text{John})$ by self-cancellation. Passing this vector through the circuit in Figure 3 produces the sequence *Mary kissed John*. The knowledge that speakers have of their language can be represented as the set of such associations stored as a content-addressable memory containing the sum of the associations, along with a cleanup memory.

Of course, mappings of this form, specific to a single predicate and set of arguments, ignore the sorts of generalizations that we expect any model of grammatical knowledge to represent. Instead of idiosyncratic mappings like this one, we would like the model to contain more general ones in which the predicates and arguments are abstracted as variables. Even better would be a model capable of acquiring the general mappings based on exposure to specific instances.

Our model's ability to meet these challenges derives from a crucial feature of the multiply-add-permute vector framework that we alluded to earlier: because all entities are vectors of a fixed size, there is no underlying distinction between constants and variables. As we saw in the examples above, the association/substitution mechanism behaves as if there were variables, but these are merely a notational convenience that we use to characterize the mechanism's behavior. As noted by Kanerva (*to appear*) this fact leads to a very powerful ability to acquire a general pattern quickly from even a single instance. The example below illustrates this principle, showing how the correct generalization be derived from exposure to a single mapping involving John and Mary (and the relevant individual concept/word associations), via the multiplicative association operator and self-cancellation. (In this and subsequent examples we assume an implicit noise term, which we leave out to avoid clutter. We also make liberal use of ad-hoc semantic roles and categories for the sake of illustration. We have no commitment to any particular view of semantic roles, an issue orthogonal to our interests here).

$$\begin{aligned}
 (11) \quad & (\text{ACTION} \otimes \text{KISS} + \text{AGENT} \otimes \text{MARY} + \text{PATIENT} \otimes \text{JOHN}) \otimes \\
 & (\text{KISS} \otimes \text{kissed} + \text{JOHN} \otimes \text{John} + \text{MARY} \otimes \text{Mary}) \otimes \\
 & (\text{P}_1 \otimes \text{Mary} + \text{P}_2 \otimes \text{kissed} + \text{P}_3 \otimes \text{John}) = \\
 & (\text{ACTION} \otimes \text{kissed} + \text{AGENT} \otimes \text{Mary} + \text{PATIENT} \otimes \text{John}) \otimes \\
 & (\text{P}_1 \otimes \text{Mary} + \text{P}_2 \otimes \text{kissed} + \text{P}_3 \otimes \text{John}) = \\
 & \text{P}_1 \otimes \text{AGENT} + \text{P}_2 \otimes \text{ACTION} + \text{P}_3 \otimes \text{PATIENT}
 \end{aligned}$$

This construction can then be used to generate the appropriate symbol sequences for other propositions:

$$\begin{aligned}
 (12) \quad & (\text{P}_1 \otimes \text{AGENT} + \text{P}_2 \otimes \text{ACTION} + \text{P}_3 \otimes \text{PATIENT}) \otimes \\
 & (\text{INSULT} \otimes \text{insults} + \text{FRED} \otimes \text{Fred} + \text{BILL} \otimes \text{Bill}) \otimes \\
 & (\text{ACTION} \otimes \text{INSULT} + \text{AGENT} \otimes \text{FRED} + \text{PATIENT} \otimes \text{BILL}) = \\
 & \text{P}_1 \otimes \text{Fred} + \text{P}_2 \otimes \text{insults} + \text{P}_3 \otimes \text{Bill}
 \end{aligned}$$

Before moving on to syntactic recursion, we would like to highlight a few points about this model. First, what appears as multi-step algebraic derivations in these examples is just another notational convenience. In an actual implementation the entire set of substitutions takes place at once, using elementwise vector multiplication and addition followed by cleanup. Second, the position vectors could equally well represent other sorts of linguistic encoding than sequencing, such as inflectional endings in a language with freer word order than English. Third, the model is by no means restricted to stereotypical transitive constructions. The content-addressable memory could just as easily contain intransitives, ditransitives, noun phrases, etc.:

$$\begin{aligned}
 (13) \quad & (\text{P}_1 \otimes \text{EXPERIENCER} + \text{P}_2 \otimes \text{STATE-CHANGE}) \\
 & \quad \quad \quad [\text{John died}]
 \end{aligned}$$

$$(14) \quad (P_1 \otimes \text{AGENT} + P_2 \otimes \text{ACTION} + P_3 \otimes \text{BENEFICIARY} + P_4 \otimes \text{THEME})$$

[*Mary gave Bill presents*]

$$(15) \quad (P_1 \otimes \text{IDENTITY} + P_2 \otimes \text{ATTRIBUTE} + P_3 \otimes \text{ENTITY})$$

[*a tall man*]

Fourth, we have used the English simple past tense in these examples to illustrate the overall behavior of the model, but it would be straightforward to extend the model to generalize over tense, aspect etc, via lexical associations like $\text{KISS} \otimes \text{PAST} \otimes \textit{kissed}$. Finally, we are not the first to take a data-driven approach to generalizations of this sort. There is a growing literature in this area, mainly using traditional representations like parse trees and grammatical rules (e.g., Bod and Scha 1997; Batali 2002). Our contribution here is to show how this approach can be grounded in a neurally plausible framework.

5 Phrase structure, embedding, and recursion

In an earlier section we presented an account of embedding in which the vector representations of embedded material were permuted to prevent the self-cancellation of recursive structure. In this section we discuss how this mechanism can be exploited in our transduction model.

Consider a representation of the proposition expressing one person’s belief about the beliefs of another:

$$(16) \quad \text{EXPERIENCE} \otimes \text{BELIEF} + \text{EXPERIENCER} \otimes X +$$

$$\text{THEME} \otimes \Pi (\text{EXPERIENCE} \otimes \text{BELIEF} + \text{EXPERIENCER} \otimes Y + \text{THEME} \otimes \Pi (P))$$

where x and y are individuals, p is a proposition, and Π a permutation operator. Even with the permutation operator, a problem arises when we try to use these mappings in the context of the corresponding transduction mappings like $(P_1 \otimes \text{EXPERIENCER} + P_2 \otimes \text{EXPERIENCE} + P_3 \otimes \text{THEME})$ and $(P_1 \otimes \text{AGENT} + P_2 \otimes \text{ACTION} + P_3 \otimes \text{PATIENT})$: multiplying these mappings together results in the self-cancellation of each position vector, obliterating the sequencing information.

5.1 Syntax as planning

The origin of this problem lies in the oversimplified way in which we are associating semantic content with syntactic form. In section 4 above, we alluded to a content-addressable memory that supports this association, but gave no details. To flesh out the details of such a device, we begin by noting that it is unrealistic to expect it to supply the entire set of associations for a given propositional meaning at once, especially when the proposition contains embedding. Instead, following recent work by Steedman (2002), we might view the associative memory as a kind of planning mechanism. This mechanism would retrieve the appropriate mapping for a given semantic content before retrieving the mapping for any part of that content. The execution of intermediate plans – here, the serialization of embedded semantic content – would be accomplished by means of a mechanism that gives precedence to intermediate plans as they are retrieved. Because of the way that we are encoding sequences, “embedded” here refers to any material requiring its own internal sequencing, including noun phrases.

As shown in (Levy 2007), such a mechanism can be modeled in a neurally plausible way using the fixed-size vector apparatus described here. Its operation can be understood by analogy with the foregrounding and backgrounding of visual images. As new images (plans) are overlain onto the fixed-size image buffer, they are foregrounded by amplifying them by some constant. For visual images, this operation manifests itself as greater prominence of the foregrounded

image. For sequential plans like the ones we are working with here, an additional mechanism is needed, to store the current position vector and reset it after the current plan is completed. If we use the sort of low-precision neural encoding that we mentioned in our introduction, we obtain a realistic “soft limit” on the number of such plans that can be superimposed in this way. We also obtain a prediction about the relative ease of processing tail-recursive constructions (*the stick that beat the cat that ate the goat*) as compared with the difficulty of center-embedded constructions (*the goat that the cat that the stick beat ate*). As more position vectors are added together and the sums normalized to a fixed precision, it becomes increasingly difficult to recover earlier plans. In terms of our initial discussion, this means that competence and performance derive from the same source, namely, the mathematics of high-dimensional, low-precision vectors.

5.3 Grammatical categories as attractor basins

Given our use of ad-hoc semantic roles and grammatical constructions, the reader might wonder about the status of grammatical categories (lexical, phrasal) in our model. Rather than positing such categories as primitives, we see them as emergent properties of the coordination between meaning and form that is accomplished by our associative mappings. Specifically, they are the *attractor basins* (Abraham and Shaw 1992), or stable patterns, that emerge in the Hopfield network of the associative cleanup memory as it is exposed to exemplars of such mappings during language acquisition. Repeated exposure to common patterns like ($P_1 \otimes \text{AGENT} + P_2 \otimes \text{ACTION} + P_3 \otimes \text{PATIENT}$) induces (metaphorically) deep, wide attractor basins for such patterns that make them easier to reconstruct from degraded or incomplete data. In practical terms this means that when presented with a novel sequence, such as ($P_1 \otimes a + P_2 \otimes \textit{large} + P_3 \otimes \textit{glarph}$), the associative memory will derive a representation for *glarph* that is in the vector “neighborhood” for concepts that in English are expressed by nouns.

This graded interpretation of grammatical categories is reminiscent of the syntax-lexicon continuum of cognitive/construction grammar (Croft and Cruse 2004). Our contribution here is to propose a biologically plausible mechanism supporting such a continuum.

5.4 Becoming recursive

Our model puts not just grammatical categories, but also recursion itself, in a new light. Because the model makes it necessary to encode all sorts of structure using permutations, there is no overt representation or special status given to “true” recursion, *e.g.*, occurrence of a predicate in the scope of an identical predicate (*John knows that Bill knows that Mary kissed Fred*). Because the model is based on transduction between concrete patterns, as opposed to abstract categories, sets, and rules, the contentious question about whether all (or any) human languages are “infinite” does not arise. Eschewing such traditional mechanisms and their resulting distinctions encourages us to explore the issues surrounding recursion in a novel way.

For example, the role of recursion in language evolution can be recast as the question of how a mechanism used to represent events, states, roles, fillers, and other relational semantic information came to be coupled with a mechanism for planning sequential actions. Perhaps the sequence-processing ability was exapted from other modalities, as one researcher has suggested for the sense of smell (Lorig 1999). With respect to language acquisition, our model gives no reason to expect constructions traditionally viewed as recursive (like relative clauses) to be more difficult to acquire than non-recursive structure of comparable syntactic complexity. This view is consistent with recent experiments in the acquisition of English relative clauses, which found adult-like competence for relative clauses by age three or four (Fragman, Goodluck, and Heggie 2007).

If recursion *per se* does not constitute a fundamental leap in language evolution or acquisition, the absence of recursion in a language would likely be due to non-linguistic

constraints; for example, a cultural prohibition on certain forms of utterance. This is precisely the situation that Everett (2005) describes for Pirahã.

6 Conclusions and future work

In this article we have presented a view in which recursion is not a single faculty or property of language, but rather the result of the way that “quasi-recursive” mental representations and processes are used (quoting Pinker and Bloom 1990) for *the transmission of propositional structures through a serial interface*.

Though other researchers have taken approaches similar to ours, we feel that our model differs in significant respects. Tabor, Juliano, and Tanenhaus (1996) present a back-propagation network model in which lexical and phrasal categories emerges as basins of attraction in the network dynamics, but their task consists of predicting word sequences without modeling the event semantics described by the sequences. Chang, Dell, and Bock (2006) describe a connectionist network that “becomes syntactic” by learning to predict roles and their fillers; however, they too use the biologically implausible back-propagation algorithm, and don’t deal with recursive constructions. Dominey, Hoen, and Inui (2006) model the processing of grammatical constructions using a network that relates roles and fillers to sequences in a way similar to ours. Their network uses a localist indexing scheme and works with relative clauses; it is not clear how well the model would extend to other sorts of embedded/recursive structures.

The view we have sketched is naturally quite tentative, and much work remains to be done. Relative clauses and conjuncts present a challenge for us because they can associate the same role with two different fillers (*The cat chased the rat that chased the mouse; Mary kissed John and John kissed Fred*). It is not clear how to represent the semantic structure conveyed by such utterances, let alone the mapping to a syntactic construction, in the permutation-based approach we have taken here.

Finally, we note the immense gap in explanatory capacity between our model and the rich theoretical apparatus provided by more traditional frameworks like generative grammar. The latter has had five decades to develop accounts of subadjacency, word-order universals, and other important phenomena that we have not even begun to address. Rather than being a single, unitary alternative to this framework, we see our model as part of an overall approach that seeks to explain language patterns as arising from the interaction of multiple biological, psychological, social, and cultural constraints on communication and adaptation. (See, for example, Berwick and Weinberg’s (1984) parsing account of subadjacency, Hawkins’ (1994) performance theory of word-order universals, and Kirby’s (2002) communication-bottleneck model of the emergence of recursive syntax.) We hope to have made some small contribution here to this ongoing and worthwhile effort.

Acknowledgments

The author thanks Ross Gayler and Harry Howard for extensive discussions and advice on the material presented in this article, and two anonymous reviewers for helpful comments

References

- Abraham, Ralph and Christopher D. Shaw
1992 *Dynamics: The Geometry of Behavior*. Reading, Massachusetts: Addison Wesley.
- Batali, John
2002 The negotiation and acquisition of recursive grammars as a result of competition among exemplars. In Ted Briscoe (ed.), *Linguistic Evolution through Language Acquisition: Formal and Computational Models*. Cambridge: Cambridge University Press.
- Berwick, Robert C., and Amy S. Weinberg
1984 *The Grammatical Basis of Linguistic Performance: Language Use and Acquisition*. Cambridge, Massachusetts: MIT Press.
- Bod, Rens and Remko Scha
1997 Data-oriented language processing: An overview (Technical Report 38). Amsterdam: NWO Priority Programme in Language and Speech Technology.
- Chang, Franklin, Gary S. Dell, and Kathryn Bock
2006 Becoming syntactic. *Psychological Review* 113(2): 234-272.
- Chomsky, Noam
1965 *Aspects of the Theory of Syntax*. Cambridge, Massachusetts: MIT Press.
- Crick, Francis
1989 The recent excitement about neural networks. *Nature*, 337: 129–132.
- Croft, William A. and D. Alan Cruse
2004 *Cognitive Linguistics (Cambridge Textbooks in Linguistics)*. Oxford: Oxford University Press.
- Dayhoff, Judith
1990 *Neural Network Architectures*. New York: Van Nostrand Reinhold.
- Dominey, Peter F., Michel Hoen, and Toshio Inui
2006 A neurolinguistic model of grammatical construction processing. *Journal of Cognitive Neuroscience* 18: 2088-2107.
- Eliasmith, Chris
2004 Learning context sensitive logical inference in a neurobiological simulation. In Simon D. Levy and Ross Gayler (eds.), *Compositional Connectionism in Cognitive Science: Papers from the AAAI Fall Symposium*, 17-20. Menlo Park, California: AAAI Press.
- Eliasmith, Chris and Paul Thagard
2001 Integrating structure and meaning: a distributed model of analogical mapping. *Cognitive Science* 25(2): 245-286.
- Elman, Jeffrey L.
1990 Finding structure in time. *Cognitive Science* 14: 179–211.
- Everett, Daniel L.
2005 Cultural constraints on grammar and cognition in Pirahã. *Current Anthropology* 46 (4): 621-646.
- Foltz, Thomas K. and Darrell Laham
1998 Introduction to latent semantic analysis. *Discourse Processes* 25: 259-284.
- Fragman, Cathy, Helen Goodluck, And Lindsay Heggie
2007 Child and adult construal of restrictive relative clauses: Knowledge of grammar and differential effects of syntactic context. *Journal of Child Language* 34 (2): 345-380.

Gayler, Ross. W.

1998 Multiplicative binding, representation operators, and analogy [Abstract of poster]. In Keith J. Holyoak, Dedre Gentner & Boicho N. Kokinov (eds.), *Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences*. Sofia: New Bulgarian University.

Hauser, Marc D., Noam Chomsky, and W. Tecumseh Fitch

2002 The faculty of language: What is it, who has it, and how did it evolve? *Science* 298 (5598): 1569-1579.

Hawkins, John

1994 *A Performance Theory of Order and Constituency*. Cambridge: Cambridge University Press.

Hebb, Donald O.

1949 *The Organization of Behavior*. New York: John Wiley and Sons.

Holyoak, Keith J. and John E. Hummel

2000 The proper treatment of symbols in a connectionist architecture. In E. Dietrich and A.B. Markman (eds.), *Cognitive Dynamics: Conceptual and Representational Change in Humans and Machines*. Mahwah, New Jersey: Lawrence Erlbaum Associates.

Hopfield, John J.

1982 Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences* 79: 2554–2558.

Kanerva, Pentti

to appear Hyperdimensional computing: A tutorial on computing in distributed representation based on high-dimensional random vectors. *Cognitive Computation*.

Kanerva, Pentti

2008 Encoding meaning with high-dimensional random vectors. Guest lecture delivered 9 December 2008, Redwood Center for Theoretical Neuroscience, Berkeley, California. Available at http://www.archive.org/details/ucbvs298_neural_comp_2008_12_09.

Kirby, Simon

2002 Learning, bottlenecks and the evolution of recursive syntax. In Ted Briscoe (ed.), *Linguistic Evolution through Language Acquisition: Formal and Computational Models*, 173-204. Cambridge: Cambridge University Press.

Levy, Simon D.

2007 Continuous states and distributed symbols: Toward a biological theory of computation (Poster). *Proceedings of Unconventional Computation: Quo Vadis?* Santa Fe.

Lorig, Tyler S.

1999 On the similarity of odor and language perception. *Neuroscience & Biobehavioral Reviews* 23 (3): 00041-4.

Marcus, Gary. F.

1998 Rethinking eliminative connectionism. *Cognitive Psychology* 37: 243-282.

Pinker, Steven and Paul Bloom

1990 Natural language and natural selection. *Behavioral and Brain Sciences* 13 (4): 707-784.

Plate, Tony

2003 *Holographic Reduced Representation: Distributed Representation for Cognitive Science*. Stanford, California: CSLI Publications.

Rumelhart, David E. and James L. McClelland, James L.

1986 On learning the past tenses of English verbs. In James L. McClelland, David Rumelhart, and the PDP research group (eds.). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models*. Cambridge, Massachusetts: MIT Press.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams

1986 Learning internal representations by error propagation. In Rumelhart and McClelland (1986).

Smolensky, Paul

1990 Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence* 46, 159–216.

Steedman, Mark

2002 Planning, affordances, and combinatory grammar. *Linguistics and Philosophy* 25: 723–753.

Tabor, Whitney, Cornell Juliano, and Michael K. Tanenhaus

1997 Parsing in a dynamical system: An attractor-based account of the interaction of lexical and structural constraints in sentence processing. *Language and Cognitive Processes* 12 (2/3) 211-271.