Modeling morphological affixation with interpretable recurrent networks: sequential rebinding controlled by hierarchical attention

Colin Wilson (colin.wilson@cogsci.jhu.edu)
Cognitive Science Department, Johns Hopkins University
Baltimore, MD 21218 USA

Abstract

This paper proposes a recurrent neural network model that learns to perform morphological affixation, a fundamental operation of linguistic cognition, and has interpretable relations to descriptions of morphology at the computational and algorithmic levels. The model represents morphological sequences (stems and affixes) with distributed representations that support binding of symbols to ordinal positions and position-based unbinding. Construction of an affixed form is controlled at the implementation level by shifting attention between morphemes and across positions within each morpheme. The model successfully learns patterns of prefixation, suffixation, and infixation, unifying these at all levels of description around the theoretical notion of a pivot. Connections of the present proposal to neural coding of ordinal position, and to computational models of serial recall, are noted.

Keywords: morphology; distributed representations; recurrent networks; neural attention; multi-level descriptions

Introduction

Affixation operations play a central role in the inflectional and derivational morphology of natural languages. In elementary cases of prefixation and suffixation, a fixed sequence is concatenated to the beginning (e.g., unkind) or end (e.g., kindness) of a morphological stem. Circumfixation, in which a single morpheme adds material to both edges of the stem (e.g., German gesinge ‘singing’), can be analyzed as a combination of prefixation and suffixation. Infixation adds material at a designated position in the stem that does not consistently align with either the left or right edge. For example, some infixes are placed immediately after the first stem consonant (e.g., Mlabri pfluut ‘carving’; Rischel, 1995), while others appear immediately before the first vowel (e.g., Chamorro trumisti ‘becomes sad’; Topping & Dungca, 1973).

In many empirical domains that involve sequence data (e.g., Bahdanau, Cho, & Bengio, 2015; Chorowski, Bahdanau, Serdyuk, Cho, & Bengio, 2015; Graves & Schmidhuber, 2009; Sutskever, Vinyals, & Le, 2014; Chiu et al., 2017), deep recurrent networks currently outperform other models on large-scale tasks of morphological learning and computation (e.g., Faruqui, Tsvetkov, Neubig, & Dyer, 2016; Cotterell et al., 2016; Cotterell, Vylomova, Khayrallah, Kirov, & Yarowsky, 2017). However, the architecture and mechanics of these networks are currently disconnected from rich, typologically-informed linguistic theories of morphology (e.g., Spencer & Zwicky, 1998; Spencer, 1991) and from a long tradition of symbolic approaches to computational morphology (e.g., Sproat, 1992; Ritchie, Russell, Black, & Pulman, 1992; Beesley & Karttunen, 2003). Despite continuing efforts to visualize and interpret deep neural networks (e.g., Karpathy, Johnson, & Li, 2015; Li, Chen, Hovy, & Jurafsky, 2016; Liu, Mao, Sha, & Yuille, 2017), the way in state-of-the-art models perform even simple concatenation operations remains largely opaque.

This paper develops a recurrent network model of affixation that is fully interpretable, in the sense that each of its representations (patterns of activity) and processes (operations on activity patterns) has a clearly specified role in morphological computation. Several theoretical ideas, drawn from across the cognitive sciences and from applied research on neural networks, are integrated in the model. The most crucial elements are a continuous implementation of symbolic read/write operations and a control system that employs hierarchical neural attention.

The following subsections introduce the model at the computational and algorithmic levels (Marr, 1982). The body of the paper then provides a description at the implementation level — discussing in detail how the representations and processes of the implementation approximate those of the affixation algorithm — and demonstrates that the model is capable of learning affixation patterns, including infixation, from positive examples. While the model does not attempt to match the achievements of previous deep networks, its success in this restricted but important domain shows that interdisciplinary, multi-level research on linguistic cognition need not sacrifice interpretability to produce interesting performance. Directions for further integrating the model with what is known about neural coding of ordinal position, and other types of serial behavior, are noted in the final section.

Computational level

Described at the most abstract level, each instance of the present model computes an input-output function of the form:

\[ \text{stem} \rightarrow \text{stem} \odot [0, \text{pivot}] \odot \text{affix} \odot \text{stem} \odot [\text{pivot} + 1, \ell - 1] \]  

where \text{stem} and \text{affix} are sequences of symbols from a fixed alphabet \( \Sigma \), concatenation is denoted by \( \odot \), and \( \ell = |\text{stem}| \) is the number of symbols in the stem.

This schema unifies diverse affixation patterns around the notion of a pivot (Yu, 2007), defined here as the position in the stem immediately after which the affix is placed. In prefixation, the pivot is consistently 0, the position occupied by a special stem-initial symbol (\( \ast \)). In suffixation, the pivot is always \( \ell - 2 \), so that the affix is inserted immediately before the stem-final symbol (\( \ast \)). Typological surveys of infixation patterns (e.g., Ultan, 1975; Yu, 2007) support a parametric theory of additional pivot positions: attested infixes are located relative to the first or last instance in the stem of one of
a small set of linguistic units (e.g., consonant, vowel, syllable, or stressed foot).

Regardless of the type of affixation operation that is involved, the pivot can be found by a leftward or rightward scan of the stem. For example, the pivot for the Chamorro infix um can be located by iterating rightward from the beginning of the stem and halting when the subsequent symbol is a vowel (i.e., a member of the vowel natural class \( V \subseteq \Sigma \)).

**Algorithmic level**

The model has an algorithmic description in terms of three unbounded arrays, or tapes, that are read from and written to by a hierarchical control system (see Figure 1). The input to the algorithm is a delimited sequence on the stem tape (e.g., \( \times 0k1 z n d s \times s \)). As part of its parameterization for a specific affixation operation, the model has an internal sequence on the affix tape (e.g., \( m 0 e 1 5 2 5 3 \)). It is convenient to treat these two tapes as members of a list or array morphs = [stem, affix] (line 1). A subroutine LOCATEPIVOT identifies the pivot in the input stem (line 2) with a leftward or rightward scan as mentioned above.

Four integer indices are manipulated by the controller. The integer \( a \) indexes the morphemes in morphs (i.e., morphs[0] = stem and morphs[1] = affix). The integers \( b_0, b_1 \), and \( c \) index ordinal positions on the stem, affix, and output tapes, respectively. The morpheme-level index \( a \in \{0,1\} \) can be used to select one of the two position-level indices \( b_0 \) or \( b_1 \), as indicated by the notation \( b_g \).

The controller initializes all indices to the first position in each sequence (line 3), then enters a loop in which symbols are sequentially copied from morphs to output (lines 4-13). In each pass through the loop, a single symbol \( y \) is read from the stem or affix tape — according to the current values of \( a, b_0, b_1, c \) — and written to the current position of the output tape (line 5). If \( a = 0 \), as initially, the symbol read out is morphs[0][b_0] (= stem[b_0]); otherwise, \( a = 1 \) and the symbol morphs[1][b_1] (= affix[b_1]) is read out. In either case, the symbol is written to output[\( c \)].

The controller switches the morpheme-level index from the value for the stem (= 0) to that of the affix (= 1) after the symbol at the pivot position is copied (lines 6-7), and switches it back to the stem value after copying the final symbol of the affix (lines 9-10). At each step the position index for the current morpheme is iterated (lines 8 and 11), as is the position index for the output (line 12). The algorithm terminates when the stem-final symbol \( \times \) is written to the output (line 13). Figure 1 shows the input/output mapping, with explicit index values, for the Chamorro example trumisti.

**Implementation level**

In the network implementation of the model, symbol sequences are realized by continuous embeddings and discrete read/write operations are replaced by gradient counterparts. The network has the status of an implementation because, in clearly-specified limiting cases, its behavior is identical to that of the algorithm (for outputs up to a maximum length).

The representations and update equations of the network also stand in a close correspondence to those at the algorithmic level, with a particularly tight relation holding for the representations of morphological symbol sequences.

**Continuous embedding of symbol sequences**

Previous research on continuous models of cognitive representation provide many potential embeddings of sequences and other structured objects (Kanerva, 1988; Plate, 2003; Botvinick & Watanabe, 2007; Cox, Kachergis, Recchia, & Jones, 2011; Kelly, Mewhort, & West, 2014). Many of these are exact or approximate instances of tensor-product representations (TPRs) (Smolensky, 1990; Tesar & Smolensky, 2006). In a TPR, each symbol \( x_i \) of the alphabet \( \Sigma \) is realized by a vector \( x_i \). The symbol vectors can be localist (‘one-hot’), random, or substantive encodings that express meaningful similarity relations (e.g., shared phonological features). Here we assume only that the symbol vectors are distinct, non-zero, and of dimensionality \( m = |\Sigma| \).

In addition to the vectors that realize atomic symbols, TPRs employ vectors realizing the structural roles that symbols occupy. A natural role set for symbol sequences is provided by the ordinal positions \( \{0,1,\ldots,n-1\} \), where \( n \) is the maximum sequence length that the network can represent. Each position \( r_j \) is realized by the vector \( x_j \) — here we assume only that these vectors are linearly independent, unit length, and of dimensionality \( n \).

The TPR embedding of a symbol sequence is constructed by binding each symbol vector to its corresponding role vector with the tensor-product operator \( \times \), and summing the results. For example, the embedding of the stem \( \text{kind} \), delimited by start and end symbols, is \( S[\times 0k1 z n d s] = x \times r_0 + k \times r_1 + i \times r_2 + n \times r_3 + d \times r_4 + \times \times r_5 \) where \( x_i \times r_j \) denotes the outer product \( x_i \otimes r_j \). The embedding of an affix such as \( \text{um} \), \( A[\times 0 n] \), is constructed in the same way. Note that the embedding of every morpheme, regardless of its length in symbols, is a matrix of fixed dimensionality (namely, \( m \times n \)).

The assumption that the position vectors are linearly independent is important, because it guarantees that a symbol vector can be faithfully extracted from an embedding or unbound from its role (Smolensky, 1990). This is accomplished by defining an unbinding vector \( u_j \) for each role \( r_j \) such that \( r_j \cdot u_j = 1 \) and \( r_k \cdot u_j = 0 \) for all \( k \neq j \). If the role vectors are arranged into the columns of a matrix \( \mathbf{R} \), the unbinding vectors are exactly the columns of \( \mathbf{U} = (\mathbf{R}^T)^{-1} \). Then the matrix-vector product \( S u_j = SU[,j] \) returns the vector \( x_i \) that is bound to \( r_j \) in \( S \) (or the zero vector \( 0_u \) if there is no symbol \( x_i \) bound to \( r_j \) in the sequence represented by \( S \)).

Binding atomic symbols to ordinal roles gives rise to second-order tensors. Because the binding operation can be applied recursively (Smolensky, 1990), expressions such as:

\[
M = S (\times 0k1 z n d s \times s) \otimes r_0 + A[\times 0 n] \otimes r_1
\]

are well-defined third-order tensors in which the embeddings of a stem and an affix are bound to positions 0 and 1, respec-
tively. In the same way that $\mathbf{S}\mathbf{u}_j$ extracts the vector realizing the $j$th symbol (if any) in a morpheme, a suitably defined unbinding vector $\mathbf{u}_c$ can be used to extract the matrix realizing the $k$th morpheme (i.e., $\mathbf{M}\mathbf{u}_c$).

### Attention-based indexing

The discussion so far has presupposed that integer indices are available to the model (e.g., to select the $j$th column of the role matrix $\mathbf{R}$ for the purpose of binding a symbol vector to it, or to select the $k$th column of the matrix $\mathbf{U}$ for the purpose of unbinding a symbol vector). However, in order to ensure that the input/output function is differentiable, the implementation instead uses real-valued indices. Therefore, exact column selections such as $\mathbf{R}[j, \cdot ]$ and $\mathbf{U}[\cdot, k]$ — and the binding and unbinding operations they support — must be replaced by approximate alternatives. This can be accomplished with neural attention, a type of soft indexation that has been been successfully applied in a wide variety of deep network models (e.g., Graves, Wayne, & Danihelka, 2014; Luong, Pham, & Manning, 2015; Faruqui et al., 2016; Graves et al., 2016).

The essential idea is that exact extraction of the $j$th column of a matrix $\mathbf{X}_{m \times n}$ can be performed with $\mathbf{X}\mathbf{e}_j$, where $\mathbf{e}_j$ is the canonical basis vector with 1 in the $j$th position and 0 elsewhere. Notice that $\mathbf{e}_j$ satisfies the properties $0 \leq \mathbf{e}_{jk} \leq 1$ (for $0 \leq k < n$) and $\sum_{k=1}^n \mathbf{e}_{jk} = 1$; thus $\mathbf{e}_j$ can be interpreted as a maximally-peaked distribution of attention over the $n$ columns of $\mathbf{X}$. In the soft approximation to column extraction, real-valued indices are mapped to generic attention distributions — distributions that spread attention out more broadly, becoming maximally peaked only in the limit — over the ordinal positions $\{0, 1, \ldots, n-1\}$.

Specifically, let $\mathbf{rbf}_e$ denote a layer of classical radial basis units (Broomhead & Lowe, 1988; Moody & Darken, 1989), with one unit centered at each ordinal position $\mu$ from 0 to $n-1$. A real-valued input $b$ is passed through the layer and the activities of the units are normalized to yield an attention distribution $\beta = \mathbf{norm-rbf}_e(b)$. For Gaussian units, the mapping from a real-valued index to an attention distribution is defined by:

$$s_{\mu} = -\tau(b - \mu)^2 \quad \text{for } \mu \in \{0, 1, \ldots, n-1\}$$

$$\beta = \text{softmax}(s)$$

where $\tau > 0$ is a precision parameter that determines the selectivity of each unit in the layer and $\text{softmax}(s)_{\mu} = \exp(s_{\mu})/\sum_{\nu} \exp(s_{\nu})$. In the limit $\tau \to \infty$, the vector returned by (2) places all of its attention on the ordinal position that has the smallest squared distance to $b$. Otherwise, attention is distributed across multiple positions in a way that favors those closer to the input. If $b$ is closest to integer index $j$ ($0 \leq j < n$), then $\hat{\mathbf{r}}_j = \mathbf{R}\beta$ will be approximately equal to the $j$th role vector (i.e., $\mathbf{r}_j = \mathbf{R}[j, \cdot ]$). Similarly, the $j$th unbinding vector $\mathbf{u}_j = \mathbf{U}[\cdot, j]$ is approximated by $\hat{\mathbf{u}}_j = \mathbf{U}\beta$.

### Sequential rebinding

The preceding development is sufficient to understand how the model’s implementation can, approximately, unbind the vector realizing a symbol in a particular position of the stem or affix and then bind the vector to a (possibly different) position in the output. Let $\mathbf{M}$ be the third-order tensor defined previously (i.e., $\mathbf{M} = [\mathbf{S}_0, \mathbf{A}_1]$, where matrices $\mathbf{S}$ and $\mathbf{A}$ realize a stem and affix, respectively), and let $\mathbf{Y}$ be a second-order tensor realizing the output of the affixation operation (initially, $\mathbf{Y} = \mathbf{0}_{m \times n}$). Further let the morpheme index $a$ and the position indices $b_0, b_1, c$ be as defined in the algorithmic level description of the model, except that each is now real-valued.

The distribution $\mathbf{a} = \mathbf{norm-rbf}_e(a)$ divides attention between the two morphemes realized within $\mathbf{M}$, while the distributions $\mathbf{b}_0 = \mathbf{norm-rbf}_e(b_0), \mathbf{b}_1 = \mathbf{norm-rbf}_e(b_1)$, and $\mathbf{c} = \mathbf{norm-rbf}_e(c)$ allocate attention to ordinal positions in the stem, affix, and output sequences realized by $\mathbf{S}, \mathbf{A}$, and $\mathbf{Y}$, respectively. The following equations then define the soft re-binding operation that approximately implements copying from a morpheme to the output:
\[\tilde{u}_0 = U\beta_0\]
\[\tilde{u}_1 = U\beta_1\]
\[\tilde{r} = R\omega\]
\[\tilde{x}_0 = M_0\tilde{u}_0 \approx (M_0U\beta_0 = SU\beta_0)\]
\[\tilde{x}_1 = M_1\tilde{u}_1 \approx (M_1U\beta_1 = AU\beta_1)\]
\[\tilde{y} = g_0\tilde{x}_0 + g_1\tilde{x}_1\]
\[Y = Y + \tilde{y} \otimes \tilde{r}\]

The first three equations retrieve soft unbinding and binding vectors with attention distributions over ordinal positions. The two unbinding vectors are then used, again approximately, to unbind a symbol vector from each of the morpheme embeddings. The convex combination of these vectors, with weights determined by the attention distribution over morphemes, determines the vector \(y\) that is ‘read’. Finally, this vector is ‘written’ to the output by binding it to the soft role \(\tilde{r}\) and accumulating the result into the output embedding \(Y\).

When all three attention distributions are sharply peaked, soft rebinding closes approximates the algorithmic read/write operations of line 5 in Figure 1. Otherwise, the vector that is unbound will be a blend of the realizations of possibly multiple symbols from both morphemes, and binding to the soft role vector will spread it over multiple ordinal positions. As noted above, the degree of discretization is controlled by the precision parameters of the radial basis pools \(rbf_2\) and \(rbf_n\).

**Hierarchical neural control**

In the algorithm of Figure 1, lines 6–12 specify the control structure that updates integer indices over the course of processing. The implementation replaces discrete increments and decrements with continuous changes of the real-valued indices, as specified by the following series of equations:

\[s_0 = p\cdot\beta_0\]
\[s_1 = \text{CHECKEND}(A, b_1)\]
\[a = a + \alpha_0 s_0 - \alpha_1 s_1\]
\[b_0 = b_0 + \alpha_0\]
\[b_1 = b_1 + \alpha_1\]
\[c = c + 1\]

where in the first equation \(p = \text{LOCATEPIVOT}(S)\) assigns a pivot probability to each position in the stem, and in the second equation \(\text{CHECKEND}\) determines the probability that \(b_1\) is near the end of the affix.\(^1\)

\(1\)The matrix \(R\) of role vectors is the identity matrix \(I_{n\times n}\), then so is the unbinding matrix \(U\). Under this localist encoding of position, \(SU\beta_0\) and \(AU\beta_1\) simplify to \(S\beta_0\) and \(A\beta_1\). The equations in the text generalize to linearly-independent distributed role vectors.

\(2\)\(\text{LOCATEPIVOT}\) and \(\text{CHECKEND}\) were implemented with simple LSTM subnetworks (e.g., Hochreiter & Schmidhuber, 1997), the former scanning the stem bidirectionally and the latter scanning leftward only. For example, \(\text{CHECKEND}(A, b_1)\) maps the soft position after \(b_1\), namely \((b_1 + 1)\), to an attention distribution over unbinding

The morpheme index \(a\) is incremented by about one unit if attention is currently focused on the stem \((a_0 \approx 1)\) and \(b_0\) is close to the pivot point \((s_0 \approx 1)\) — thereby softly switching attention onto the affix. It is decremented by about one unit when attention is currently directed to the affix \((a_1 \approx 1)\) and \(b_1\) is close to the end of the affix \((b_1 \approx 1)\) — softly switching back to the stem. Intermediate changes are possible and result in attention being more evenly divided between the two morphemes. The stem position index \(b_0\) is incremented to the extent that attention is currently focused on the stem, and similarly for the affix position index \(b_1\). In the one residual discrete component of the model, the output index \(c\) always advances in unit increments.

The equations in (3) and (4) together describe a recurrent network that, at each time step, first modifies the output \(Y\) according to the current values of the indices (and the matrices \(S\) and \(A\)) and then updates the indices. The indices are initialized as in the algorithm, with \(a = b_0 = b_1 = c = 0\). The condition for halting the recurrence can be implemented in two ways: by checking for similarity between the current \(y\) and the vector realizing the stem-end symbol \(\text{stem}\) after each update, as in the algorithmic description; or by allowing the recurrence to always proceed for \(n\) time steps, continuing to ‘read’ and ‘write’ vectors close to \(0_n\), after the symbols in the stem and affix have been exhausted.

**Simulation results**

Each instantiation of the model implements a single affixation operation, learned from positive examples of the form \((\text{stem}, \text{output})\). The model was not explicitly told what type of operation the examples illustrate (i.e., prefixation, suffixation, or infixation), and the input contained no marking of the boundaries between stems and affixes. Therefore, the model had to learn both the form of the affix and the function that determines the pivot point in any given stem. The learned parameters of the model were the \(m \times n\) tensor-product representation of the affix, the precisions of the two radial-basis pools, and the parameters of the LSTMs implementing \(\text{LOCATEPIVOT}\) and \(\text{CHECKEND}\). (The parameterization of \(\text{CHECKEND}\) could in principle be hard-coded but was learned in the present simulations.)

For each input stem, the model generated a soft TPR as output. This matrix was interpreted probabilistically by iterating through the discrete position roles and transforming the gradient symbol vector that is bound to role \(r_i\) into a probability distribution over symbols in the \(i\)th position.\(^3\)

\(3\)This was done by computing the Euclidean distance between the unbound vector and each symbol vector, then normalizing. Symbol and role vectors were random subject to the requirements specified earlier in the text. Precision parameters were initialized to 1 and other parameters were randomly initialized to small values. Simulations were performed in PyTorch (Paszke et al., 2017) with the Adagrad gradient-based optimizer and an initial learning rate of 0.05.
attempted to minimize the negative log-likelihood of the correct output symbol at each position, and learning halted when this objective (summed across positions and averaged over members of a minibatch) fell below 0.01 or 5000 epochs were completed, whichever came first. At test, a single predicted symbol sequence was generated from each output by unbinding the most probable filler for each role until a symbol was emitted.

The first simulation was performed on 1270 adjective stems and their nominal forms derived with the highly productive suffix -ness (e.g., ⟨kind, kindness⟩), as extracted from CELEX (Baayen, Piepenbrock, & Gulikers, 1995). The dataset was randomly divided into two halves (train vs. test), and in each epoch a minibatch of 40 examples was drawn at random from the training half. The entire training/testing procedure was repeated ten times. In nine of the replications, the model performed perfectly (100% accuracy) on all of the data; in the tenth, the model achieved 96% correct on the training forms and 95% on the held-out testing forms.

The second simulation used 338 adjective stems and their un- prefixed forms (e.g., ⟨kind, unkind⟩), again taken from CELEX. The same split-half procedure with multiple replications was performed. The model performed perfectly on both halves of the data in all replications. In these and the previous simulations, the objective typically fell below the threshold in fewer than 500 epochs and accuracy on the training examples asymptoted even faster. The model thus learns simple prefixation and suffixation patterns from relatively little evidence, and generalizes systematically beyond its training data.

For the third simulation, 124 examples of Chamorro -um- infixation were gathered from a dictionary (Topping & Dungca, 1973) and other sources. Following the same training and testing procedure, the model performed perfectly on both halves of the data in seven out of ten replications. For the remaining three replications it had perfect performance on the training data but 97% accuracy on the held-out data. When the model produced erroneous output sequences, these were one symbol away from the correct outputs. While the present results are encouraging, future work should investigate whether a larger training corpus or changes to the learning regimen can raise the model’s performance for infixation patterns to the same level as for prefixation and suffixation.

**Conclusion and future directions**

Previous studies of the structure and learning of morphology have focused almost exclusively on a single level of description: the computational level adopted in generative linguistics, the algorithmic level of finite-state models, or the implementation level of classic and modern neural networks. The recurrent model of morphological affixation developed in this paper has a consistent interpretation at all three levels, and combines insights from many different perspectives. The elementary step of morphological affixation, as understood here, involves copying a symbol from one sequence to another (i.e., rebinding; cf. Gu, Lu, Li, & Li, 2016). This operation, trivial to characterize at the computational and algorithmic levels, requires neural attention or a similar mechanism when implemented with continuous indices. The notion of pivot, drawn from restrictive theories of linguistic cognition, indicates when copying should switch from one morpheme to another. The switching process can itself be modeled as a reallocation of attention over morphemes.

At the implementation level, the model uses unit types that are familiar from artificial and biological neural networks, and processing interactions among units that may be canonical in brain-based computation (Carandini & Heeger, 2012). Real-valued indices are mapped to attention distributions with units that are tuned to ordinal positions and interact by divisive normalization. Neurons with ordinal preferences have been identified in single-cell recording studies (e.g., Nieder & Miller, 2004; Jacob & Nieder, 2008); however, these biological neurons do not appear to have the symmetric and equal-precision receptive fields assumed here, suggesting a direction for further research at the implementation level. These neurons have been invoked to account for serial recall performance (e.g., Botvinick & Watanabe, 2007) and sequential linguistic behavior (e.g., Dehaene, Meyniel, Wacongne, Wang, & Pallier, 2015). The connection between serial recall and morphology is relatively unexplored but clear from the present analysis: construction of an affixed form can be understand as grammatically-regulated recall of two simultaneously active symbol sequences.

**Acknowledgments**

Discussions related to this research with Marina Bedny, Paul Smolensky, and members of the gradient symbolic computation research group are gratefully acknowledged, as are the comments of four anonymous reviewers and support from NSF INSPIRE grant BCS-1344269.

**References**


