A Feature-Driven Tensor Semantics for Minimalist Grammars

John Paulson
Dept. of Linguistics
University of Utah
john.paulson@utah.edu

Aniello De Santo
Dept. of Linguistics
University of Utah
aniello.desanto@utah.edu

Jon Rawski
Dept. of Linguistics
& Language Development
San Jose State University
jon.rawski@sjsu.edu

Abstract

This paper shows how tensor-based distributional semantics can be incorporated into Minimalist Grammars (MGs). We embed the MG feature calculus with a tensor algebra and give a joint tensor-based representation where compositional semantics is guided by the minimalist syntax. By bridging syntactic and semantic operations in tensor spaces, we aim to contribute to the broader enterprise of neurosymbolic approaches to linguistic cognition.

1 Introduction

Many theories in formal semantics put *compositionality* at the center, in order to map complex linguistic expressions to truth values based on the semantics of their subparts (Lambek, 1958; Montague, 1970; Heim and Kratzer, 1998; Coppock and Champollion, 2025). On the other hand, implementations of the *distributional hypothesis* (Harris, 1954), i.e. approximating the meaning of expressions through distributions estimated via their co-occurrences in corpora, have been successful in computational linguistics (Erk, 2012; Clark, 2015; Lenci, 2018). By encoding distributional information in a high-dimensional vector spaces, distributional approaches to lexical semantics are easily deployable, and have become the de-facto standard as models of lexical meaning in many areas of (computational) linguistics (Mikolov et al., 2013; Boleda, 2020; Erk, 2022; Fodor and Suzuki, 2026, a.o.).

While attempts to extend distributional models to directly capture the meaning of more complex linguistic expressions exist, enhancing distributional lexical-semantics with some degree of compositionality would allow for the development of models that leverage insights from these two slightly disconnected traditions (see Baroni, 2013; Boleda, 2020, and references therein). Beyond potential benefit for Language Modeling applications, bridging the gap between propositional and distributional frameworks thus has the potential for demonstrating the compatibility of supposedly at-odds lines of inquiry into the nature of meaning (Baroni et al., 2014; Boleda and Herbelot, 2016), while contributing to the development of neuro-symbolic perspectives into human linguistic cognition more generally (Smolensky, 1990; Smolensky and Legendre, 2006; Tabor, 2009; Martin, 2020; Smolensky et al., 2022; Garcez and Lamb, 2023; Soulos, 2025; Quigley, 2025).

While a number of suggestions have been made for how to combine lexical vectors (Mitchell and Lapata, 2008; Clarke, 2009, a.o.), we are particularly interested in those proposals arguing that such combinatorial operations should be guided more or less explicitly by formal semantics and/or the steps of the syntactic derivations (Clark and Pulman, 2007; Coecke et al., 2010; Clarke et al., 2011, a.o.). For instance, Maillard et al. (2014) shows that compositional structure can be added to distributional semantics by integrating Combinatory Categorial Grammar (CCG; Steedman, 2001; Steedman and Baldridge, 2011) with distributed lexical representation encoded as tensors (see also Coecke et al., 2010; Grefenstette et al., 2011; Grefenstette, 2013b). The ability to combine vector space encoding with the semantic/syntactic derivations of CCGs is an important step towards bridging the two traditions, since CCG as a grammatical formalism is argued to be expressive enough to capture the full extent of human syntactic abilities, and has recently been proven insightful in studying the cognitive reality of linguistic structure (Stanojević et al., 2023; Isono, 2024; Ozaki et al., 2024).

However, a number of other highly expressive formalisms exist in the formal grammar literature. Among these, Stabler (1996)'s Minimalist Grammars (MGs) incorporate many concepts of modern generative linguistics, and have been gaining traction as a useful formalism for mathematically rigorous studies of the formal properties of syntactic

Proceedings of the Society for Computation in Linguistics (SCiL) 2026, pages 219–229

patterns (Graf, 2022), as well as for the underpinning of models connecting syntactic representations to sentence processing results (Kobele et al., 2012; De Santo, 2020, 2025). An open question, then, is whether formalisms like MGs can accommodate tensor space semantics as straightforwardly as CCG. Crucially, some work already exists proving that the syntactic feature calculus of MGs can be reinterpreted as operations in vector spaces (beim Graben and Gerth, 2012). In this paper, we answer this question by directly leveraging beim Graben and Gerth (2012) to extend Maillard et al. (2014)'s results to MGs. Specifically, we show how to integrate tensor-based semantics into a basic version of MGs, by associating the word meanings to tensors whose semantic features match the syntactic features of the corresponding MG lexical items. Like the CCG case, the core intuition is that since tensors are multilinear maps, they can be directly manipulated using the feature-driven combinatorial calculus of MGs.

2 Mathematical Preliminaries

Here we overview some basic notions of tensors and multilinear algebra (see Kolda and Bader, 2009; Lang, 2002, for more details). We assume familiarity with vector spaces and linear algebra.

For finite-dimensional real vector spaces V, W, the *tensor product* $V \otimes W$ is the vector space spanned by $v \otimes w = vw^T$ for all $v \in V$, $w \in W$, i.e. it is a generalization of the outer product. Iterating this product yields a vector space $V_1 \otimes \cdots \otimes V_k$, whose elements are *order-k tensors*. Order-k tensors can also be considered as a k-dimensional array $\mathcal{T} \in \mathbb{R}^{d_1 \times \cdots \times d_k}$, as they can be fully described using k indices (i.e. rows, columns, etc). Order-0, order-1, and order-2 tensors correspond to scalars, vectors, and matrices, respectively.

Much like the outer product creates tensors of a higher order, the inner product $v \cdot w = v^T w$ decreases, or *contracts* tensor order. *Tensor contraction* is a generalization of the inner product to tensors of any order. Contracting order-2 and order-1 tensors is matrix-vector multiplication and yields a vector, contracting two order-2 tensors is matrix multiplication, yielding a matrix, and so on.

For a matrix $M \in W \otimes V$ and vector $v \in V$, we say the (right) contraction $M \cdot v \in W$ is $M_{ij} v_j$. In general, we say for a tensor $\mathcal{T} \in V_k \otimes \cdots \otimes V_1$ and a vector $v \in V_1$, the contraction $\mathcal{T} \cdot v \in V_k \otimes \cdots \otimes V_2$. Iterating such contractions will always mean right iteration, with the usual convention that repeated indices are implicitly summed. As a notational convenience when contracting, for instance, tensors A_{ijk} and B_k we sometimes write $A_{ijk} \cdot B_k = A_{ij}$ (i.e., $A_{ij} = \sum_k A_{ijk} B_k$).

There is an isomorphism between tensors and multilinear maps: a map $f : V_1 \to \ldots \to V_j \to V_k$ can be represented as a tensor $\mathcal{T}^f \in V_k \otimes V_j \otimes \ldots \otimes V_1$. This means that tensor contraction acts as function application, i.e. for any $v_1 \in V_1, \ldots, v_j \in V_j$, $f v_1 \ldots v_j = v_k = \mathcal{T}^f \cdot v_1 \cdot \ldots \cdot v_j$.

3 CCG and Tensor Semantics

Our approach to incorporating tensor-based semantics with MGs draws heavily from previous results on CCG. In this section, we give an intuitive sketch of CCG's type-driven calculus, and we walk the reader through an example of how it can be tied to tensor-based semantics. The reader is referred to Steedman and Baldridge (2011) and Maillard et al. (2014) for more detailed introductions to CCGs and to vector space operations over CCG derivations, respectively.

CCG (Steedman, 2001) is a radically lexicalized, mildly-context sensitive grammar formalism in which syntactic types (categories) identify syntactic objects as *primitive categories* or *functions* (Steedman and Baldridge, 2011, p. 184). Primitive categories are those such as noun (N), noun phrase (NP), sentence (S), etc. Functions take arguments of a particular category (either a primitive or function) and outputs something of another category. Thus, function types identify the category of their output, the category of their argument(s), as well as the order in which the arguments must combine. For instance, A/B and $A\backslash B$ are functions that take an element of category B (to the right and to the left, respectively) and return an element of category A. As a more concrete example, transitive verbs can be defined by a syntactic category $(S\backslash NP)/NP$.

Primitive categories and functions can be combined through a set of syntactic rules, which include *functional application* of functions to arguments, as well as combinatory rules like *type-raising* and *functional composition*. Crucially, the application of syntatic rules is fully dependent on the categories of the lexical items involved in the derivation.

For the sake of illustration, let us briefly exemplify functional application. Consider a lexicon with three lexical items: *Alice* and *Bob* both with category NP, and *sees*, with category $(S\backslash NP)/NP$. We can then derive *Alice sees Bob*

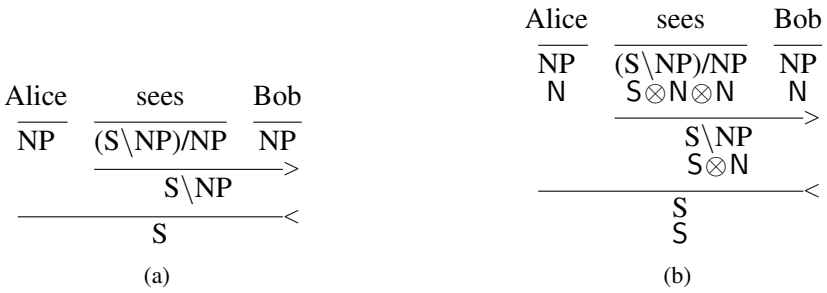

Figure 1: A CCG derivation for the sentence "Alice sees Bob", and the same derivation, with the tensor-semantics shown for each substructure.

by first combining *sees* with *Bob* to produce the expression *sees Bob* (a function of type $S\backslash NP$). Then, this function can combine with *Alice* to yield *Alice sees Bob* with type S (see Figure 1a).

Importantly, one of the core features of CCG is that it commits to a a transparent interface between syntax and semantics, in that the semantic type of a linguistic expression is entirely determined by the syntactic category. Therefore, the combinatorial rules illustrated above perform syntactic and semantic composition in parallel. In turn, this tight link between syntactic and semantic types seemingly makes CCG uniquely suited to the integration of combinatorial operations with vector-based semantics. In this sense, Maillard et al. (2014) exemplifies how to perform tensor-based operations in parallel with the core combinatorial rules of CCG (Coecke et al., 2010; Grefenstette, 2013a). Essentially, each primitive category is associated to a corresponding (distinct) vector space, e.g. S to S and NP to N. Suppose *Alice* and *Bob* have category NP, as before. The corresponding vectors in N would be A_j and B_k.

The vector spaces associated to functions' more complex types are determined through the tensor product of the spaces of the primitive categories each type is composed of. That is, each primitive type is associated with a corresponding vector space, and the slash operators are replaced with the tensor product operator. For instance, the transitive verb category $(S\backslash NP)/NP$ is associated to the vector space $\mathsf{S}\otimes\mathsf{N}\otimes\mathsf{N}$. Thus, the transitive verb *sees* has tensor $E_{ijk}\in\mathsf{S}\otimes\mathsf{N}\otimes\mathsf{N}$, with a distinct index for each component's vector space.

When function application happens in CCG, the corresponding tensors are contracted. In the first step of the derivation for *Alice sees Bob*, we get the tensor $E_{ijk}\cdot B_k = E_{ij}\in\mathsf{S}\otimes\mathsf{N}$, and then $E_{ij}\cdot A_j = E_i\in\mathsf{S}$ (see Figure 1b). Maillard et al. (2014) show how the other combinatory rules map to vector space operations in similarly straighforward ways.

4 Minimalist Grammars and Tensor Representations

The goal of this paper is to incorporate tensor-based semantic representations with Minimalist Grammars. In the previous section, we outlined the first piece of the puzzle by showing how tensor-semantics can be tied to CCG-like categories and type-driven combinatorics. We now turn to our grammar formalism of interest, sketching the intuition behind its structure-building operations and how the syntactic calculus can be mapped to vector space operations.

MGs are a derivational, lexicalized, and feature-driven mildly-context sensitive formalism able to express the complex structural analyses of generative syntax (Stabler, 1996, 2013). Intuitively, an MG grammar consists of a set of lexical items associated with a non-empty, finite string of syntactic features. Lexical items and more complex syntactic objects can be combined via two feature-driven operations — merge and move.[1] CCG-style complex categories for syntactic objects are represented in MG simply by an item's feature string. In this sense, MGs are type-driven: all of the information that is input to the combinatorial rules is contained within the feature strings themselves.

For convenience, the features relevant for the Merge and Move operations are standardly taken to be different. With respect to Merge, each lexical item will contain one simple categorical feature acting as the selectee feature (denoted f), and 0 or more selector features (denoted $=\mathrm{f}$). With respect to Move, a lexical item might contain 0 or more licensee features (denoted f^-) and licensor features

[1] We set aside extensions to MG that accommodate the Adjoin operation (Frey and Gärtner; Fowlie, 2013), which is also not included in beim Graben and Gerth (2012)'s geometric MG formalization.

(denoted f^{+}). The operation Merge then takes two structures, one whose feature string begins with a basic category (f) feature, and one whose string begins with a selector (=f) feature, combines them, and deletes both features. Move operates similarly with licensor (f^{+}) and licensee (f^{-}) features, except that the structure with the targeted licensee feature is already a substructure of the one with the licensor. Thus, Merge is a binary operation encoding subcategorization, while Move is a unary operation allowing for the re-ordering of already incorporated items.

In this sense, MG derivations can be interpreted as acting on term algebras of trees and feature arrays, using two partial operations:

$$\begin{aligned} \text{merge} &: T_A \times T_A \rightharpoonup T_A, \\ \text{move} &: T_A \rightharpoonup T_A, \end{aligned}$$

where T_A is the set of well-formed trees over a fixed minimalist grammar G, and whose domains are $\mathrm{dom}(\text{merge})$ and $\mathrm{dom}(\text{move})$, respectively (beim Graben and Gerth, 2012).

Importantly, for each lexical item, only one feature is *accessible* to merge or move at each step: the currently accessible feature is the feature at the beginning (left-most) position of the feature string. This implies that some features will only available for checking after others have been checked/deleted, based on their order in their respective feature strings. As an illustrative example, Figure 2 shows an MG derivation for the sentence "*What did Alice eat*", under standard Minimalist assumptions for the structure of the derivation (Koopman and Sportiche, 1991; Adger, 2003). Note how MGs allow for phonologically empty (Hornstein et al., 2005) but syntactically relevant lexical items, as common in many versions of minimalist theories.

Crucially, beim Graben and Gerth (2012) show that the operation merge and move over feature-strings can be mapped to vector space operations acting over tensor-product representations tied to structural positions over MG derivation trees. Specifically, it is possible to define a vector space $\mathscr{F}$, an embedding $\boldsymbol{\Psi} : T_A \to \mathscr{F}$, and partial operations

$$\begin{aligned} \textbf{merge} &: \mathscr{F} \times \mathscr{F} \rightharpoonup \mathscr{F} \\ \textbf{move} &: \mathscr{F} \rightharpoonup \mathscr{F} \end{aligned}$$

whose domains are $\mathrm{dom}(\textbf{merge})$ and $\mathrm{dom}(\textbf{move})$, respectively. Notationally, merge and move are the two structure-building operations on the tree term algebra and **merge** and **move** their corresponding

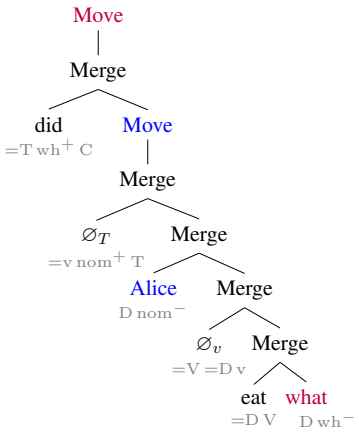

Figure 2: An MG derivation of the sentence "*What did Alice eat*". On each leaf, we show in lighter gray the feature string associated to a lexical item below its corresponding phonological content.

operations on the tensor algebra. beim Graben and Gerth (2012) prove that this mapping forms a homomorphism between the two term algebras:

Theorem 4.1 (beim Graben and Gerth (2012), Thm. 1)**.** $\boldsymbol{\Psi}$ is a homomorphism from $(T_A, \text{merge}, \text{move})$ to $(\mathscr{F}, \textbf{merge}, \textbf{move})$, where for $(t_1, t_2) \in \mathrm{dom}(\text{merge})$ and $(t) \in \mathrm{dom}(\text{move})$, $\boldsymbol{\Psi}(\text{merge}(t_1, t_2)) = \textbf{merge}(\boldsymbol{\Psi}(t_1), \boldsymbol{\Psi}(t_2))$ and $\boldsymbol{\Psi}(\text{move}(t)) = \textbf{move}(\boldsymbol{\Psi}(t))$.

We directly leverage this result in our construction of an MG tensor-based semantics.

5 A Tensor Semantics for MGs

With this background in place, we get to the core of this paper's contribution. We now show how to enhance distributional semantics with compositional operations guided by the structure-building steps of MGs. Our construction proceeds in analogy to Maillard et al. (2014), where syntactic features determine the semantic tensor types (Kobele, 2006) and semantic composition is transparently reflected in the syntactic composition (Hunter, 2007; Kobele, 2012; Tomita, 2016) via tensor contraction (beim Graben and Gerth, 2012; Maillard et al., 2014).

We consider an MG with a finite set $\mathcal{S}$ of selectee features and a corresponding lexicon Lex. Following Maillard et al. (2014), who assigned a vector space for basic CCG categories, we assign a vector space f for each selectee feature $\mathrm{f} \in \mathcal{S}$, and, using the direct sum, let $\mathsf{S} := \bigoplus_{\mathrm{f} \in \mathcal{S}} \mathsf{f}$. We then embed every lexical item in $\mathsf{L} := \bigoplus_{k \geq 1} \mathsf{S}^{\otimes k}$, where $\mathsf{S}^{\otimes k}$ is the tensor product of S with itself k times. Since selector features are checked left-to-right, their

corresponding space is in reverse order in the tensor product, so the currently active selector always corresponds to the rightmost tensor contraction.

Definition 5.1. A *lexical embedding* is a map $\varphi : Lex \to \mathsf{L}$ such that for each lexical item $l \in Lex$, if l's feature string contains selectee feature $\mathrm{f} \in \mathcal{S}$ and selector features $=\mathrm{f}_1,\ldots,=\mathrm{f}_k$, then $\varphi(l) \in \mathsf{f} \otimes \mathsf{f}_k \otimes \cdots \otimes \mathsf{f}_1 \subseteq \mathsf{L}$.

Note that φ maps selector features (e.g. $=\mathrm{N}$) to the same space of the "corresponding" selectee feature (e.g. N). We can extend this to a general compositional semantics over derivation trees.

Definition 5.2. Given a lexical embedding φ, we define $\mathbf{\Phi} : T_A \to \mathsf{L}$ as:

$$\begin{aligned} \mathbf{\Phi}(l) &= \varphi(l) \\ \mathbf{\Phi}(\mathrm{move}(t)) &= \mathbf{\Phi}(t) \\ \mathbf{\Phi}(\mathrm{merge}(t_1,t_2)) &= \mathbf{\Phi}(t_1) \cdot \mathbf{\Phi}(t_2) \end{aligned}$$

where $l \in Lex$, $t \in \mathrm{dom}(\mathrm{move})$, $(t_1, t_2) \in \mathrm{dom}(\mathrm{merge})$.

We note that this function $\mathbf{\Phi}$ is unique. We also note that (right) contraction $(\cdot)$ works on the current selector/selectee pair, so is well-typed for $\mathbf{\Phi}(t_1), \mathbf{\Phi}(t_2)$ when $(t_1,t_2) \in \mathrm{dom}(\mathrm{merge})$.

We now combine the syntactic embedding $\mathbf{\Psi}$ with the semantic embedding $\mathbf{\Phi}$ into a paired representation.

$$\begin{aligned} \mathbf{\Omega} : T_A &\to (\mathscr{F} \times \mathsf{L}) \\ t &\mapsto (\mathbf{\Psi}(t), \mathbf{\Phi}(t)) \end{aligned}$$

Furthermore, we define the partial functions $\mathcal{M}erge$ and $\mathcal{M}ove$:

$$\begin{aligned} \mathcal{M}erge &: (\mathscr{F} \times \mathsf{L}) \times (\mathscr{F} \times \mathsf{L}) \rightharpoonup \mathscr{F} \times \mathsf{L} \\ &\quad ((a,A),(b,B)) \mapsto (\mathbf{merge}(a,b), A \cdot B) \\ \mathcal{M}ove &: \mathscr{F} \times \mathsf{L} \rightharpoonup \mathscr{F} \times \mathsf{L} \\ &\quad (a,A) \mapsto (\mathbf{move}(a), A) \end{aligned}$$

Following beim Graben and Gerth (2012), we show this "joint" embedding is a homomorphism.

Proposition 5.3. $\mathbf{\Omega}$ is a homomorphism from $(T_A, \mathrm{merge}, \mathrm{move})$ to $((\mathscr{F} \times \mathsf{L}), \mathcal{M}erge, \mathcal{M}ove)$, where for $(t_1, t_2) \in \mathrm{dom}(\mathrm{merge})$ and $t \in \mathrm{dom}(\mathrm{move})$, $\mathcal{M}erge(\mathbf{\Omega}(t_1), \mathbf{\Omega}(t_2)) = \mathbf{\Omega}(\mathrm{merge}(t_1,t_2))$ and $\mathcal{M}ove(\mathbf{\Omega}(t)) = \mathbf{\Omega}(\mathrm{move}(t))$.

Proof. Assume $(t_1, t_2) \in \mathrm{dom}(\mathrm{merge})$. By Thm 4.1 and Def 5.2, $\mathbf{merge}(\mathbf{\Psi}(t_1), \mathbf{\Psi}(t_2)) = \mathbf{\Psi}(\mathrm{merge}(t_1, t_2))$ and $\mathbf{\Phi}(\mathrm{merge}(t_1, t_2)) = \mathbf{\Phi}(t_1) \cdot \mathbf{\Phi}(t_2)$. Therefore $\mathcal{M}erge(\mathbf{\Omega}(t_1), \mathbf{\Omega}(t_2)) = \big(\mathbf{\Psi}(\mathrm{merge}(t_1, t_2)), \mathbf{\Phi}(\mathrm{merge}(t_1, t_2))\big) = \mathbf{\Omega}(\mathrm{merge}(t_1,t_2))$. The move case is analogous. □

Our MG tensor semantic formulation has several nice properties. First, every merge step applies a tensor meaning to a vector argument by contraction, and so we only need tensor-vector maps. Moreover, this construction highlights the fact that MG's merge is essentially functional application (Kobele, 2006), and as such merge steps are the only ones that "add" lexical semantic content to a derivation. The relationship between move and propositional semantics is less straightforward, as move is also argued to determine scope-taking. Here, we focused on the case in which the lexical semantic content of a mover is fully integrated in the derivation at the time the mover is first merged. We return to these more complicated effects of move on the semantics of a derivation in Section 6.

5.1 A Merge-Only Example

To better illustrate the above results, we first start by exemplifying the tensor semantics calculus over a minimal MG derivations, simplifying most syntactic assumptions proper of minimalist syntax in order to only focus on our treatment of merge.

Suppose we have three lexical items (with their corresponding MG feature strings): *Alice* :: [D], *Bob* :: [D], and *sees* :: [=D, =D, C], along with their syntactic and semantic embeddings:

$$\begin{aligned} &\mathit{Alice} :: [\mathrm{D}], \\ &\mathbf{\Psi}(\mathit{Alice}) = \mathbf{Alice}, \\ &\mathbf{\Phi}(\mathit{Alice}) = A_j \in \mathsf{D} \end{aligned}$$

$$\begin{aligned} &\mathit{Bob} :: [\mathrm{D}], \\ &\mathbf{\Psi}(\mathit{Bob}) = \mathbf{Bob}, \\ &\mathbf{\Phi}(\mathit{Bob}) = B_k \in \mathsf{D} \end{aligned}$$

$$\begin{aligned} &\mathit{sees} :: [=\mathrm{D}, =\mathrm{D}, \mathrm{C}], \\ &\mathbf{\Psi}(\mathit{sees}) = \mathbf{sees}, \\ &\mathbf{\Phi}(\mathit{sees}) = G_{ijk} \in \mathsf{C} \otimes \mathsf{D} \otimes \mathsf{D} \end{aligned}$$

Thus we have $\mathbf{\Omega}(\mathit{Alice}) = (\mathbf{Alice}, A_j)$, $\mathbf{\Omega}(\mathit{Bob}) = (\mathbf{Bob}, B_k)$, $\mathbf{\Omega}(\mathit{sees}) = (\mathbf{sees}, G_{ijk})$. We can now derive the expression *Alice sees Bob* (see Figure 3).

Step 1. First, we merge *sees* and *Bob*, checking (deleting) the $=\mathrm{D}$ feature from *sees* and the D feature from *Bob*. This gives us a subtree for the expression *sees Bob* with MG feature string $[=\mathrm{D}, \mathrm{C}]$.

To illustrate how the syntactic and semantic derivation proceed in parallel to this in the corresponding vector spaces, recall that the vectors $\mathbf{\Omega}(\mathit{Bob})$, $\mathbf{\Omega}(\mathit{sees})$ are in the space $\mathscr{F} \times \mathsf{L}$, where $\mathscr{F}$ represents the *syntactic* component and L represents the (lexical) *semantic* component. Given our construction, when two MG structures are combined with merge, this is reflected by using the **merge** operation in $\mathscr{F}$, and by contracting the tensors in L. To get the result in $\mathscr{F} \times \mathsf{L}$, we use the $\mathcal{M}erge$ operation, which simply applies both operations to the relevant components. Therefore, we can find $\mathbf{\Omega}(\mathit{sees\ Bob})$ as follows:

$$\begin{aligned}&\mathbf{\Omega}(\mathit{sees\ Bob})\\ =&\mathcal{M}erge((\mathbf{sees}, G_{ijk}), (\mathbf{Bob}, B_k))\\ =&(\mathbf{merge}(\mathbf{sees}, \mathbf{Bob}), G_{ij})\end{aligned}$$

As a reminder, G_{ij} is defined to be the contraction $G_{ijk} \cdot B_k$. As expected since *sees Bob* has feature string $[=\mathrm{D},\ \mathrm{C}]$, $\mathbf{\Phi}(\mathit{sees\ Bob}) = G_{ij}$ is an element of $\mathsf{C} \otimes \mathsf{D}$.

Step 2. The next merge checks the remaining $=\mathrm{D}$ feature on *sees Bob* with the D feature from *Alice*, resulting in the expression *Alice sees Bob* On the tensor side, applying again the $\mathcal{M}erge$ operation, we have:

$$\begin{aligned}&\mathbf{\Omega}(\mathit{Alice\ sees\ Bob})\\ =&\mathcal{M}erge((\mathbf{merge}(\mathbf{sees}, \mathbf{Bob}), G_{ij}), (\mathbf{Alice}, A_j))\\ =&(\mathbf{merge}(\mathbf{merge}(\mathbf{sees}, \mathbf{Bob}), \mathbf{Alice}), G_i)\end{aligned}$$

Correctly, G_i, our "final product" of the semantic component, is in C — matching the remaining feature C in the MG feature string of *Alice sees Bob*. By comparing Figure 3 to Figure 1b, it should be evident how our tensor-based semantic calculus mirrors Maillard et al. (2014)'s almost exactly when we limit ourselves to MG's merge operation.

5.2 An Example with Merge and Move

We will now exemplify the tensor calculus over the complete MG derivation introduced earlier (Fig. 2). Suppose we have the following LIs from an MG, along with their images under $\mathbf{\Psi}$ and $\mathbf{\Phi}$:[2]

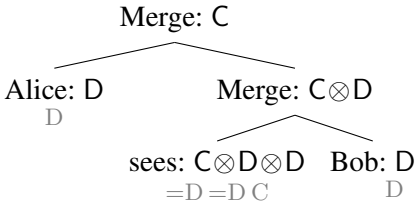

Figure 3: MG derivation for a simplified analysis of "*Alice sees Bob*", showing the semantic vector spaces corresponding to each substructure. Given our focus on the tensor semantics, we leave out the parallel tensor-syntax for ease of visualization.

$$\begin{aligned}\mathit{what} &:: [\mathrm{D}, \mathrm{wh}^-],\\ \mathbf{\Psi}(\mathit{what}) &= \mathbf{what},\\ \mathbf{\Phi}(\mathit{what}) &= W_k \in \mathsf{D}\end{aligned}$$

$$\begin{aligned}\mathit{eat} &:: [=\mathrm{D}, \mathrm{V}],\\ \mathbf{\Psi}(\mathit{eat}) &= \mathbf{eat},\\ \mathbf{\Phi}(\mathit{eat}) &= E_{lk} \in \mathsf{V} \otimes \mathsf{D}\end{aligned}$$

$$\begin{aligned}\varnothing_v &:: [=\mathrm{V}, =\mathrm{D}, \mathrm{v}],\\ \mathbf{\Psi}(\varnothing_v) &= \mathbf{o},\\ \mathbf{\Phi}(\varnothing_v) &= O_{njl} \in \mathsf{U} \otimes \mathsf{D} \otimes \mathsf{V}\end{aligned}$$

$$\begin{aligned}\mathit{Alice} &:: [\mathrm{D}, \mathrm{nom}^-],\\ \mathbf{\Psi}(\mathit{Alice}) &= \mathbf{Alice},\\ \mathbf{\Phi}(\mathit{Alice}) &= A_j \in \mathsf{D}\end{aligned}$$

$$\begin{aligned}\varnothing_T &:: [=\mathrm{v}, \mathrm{nom}^+, \mathrm{T}],\\ \mathbf{\Psi}(\varnothing_T) &= \mathbf{p},\\ \mathbf{\Phi}(\varnothing_T) &= P_{mn} \in \mathsf{T} \otimes \mathsf{U}\end{aligned}$$

$$\begin{aligned}\mathit{did} &:: [=\mathrm{T}, \mathrm{wh}^+, \mathrm{C}],\\ \mathbf{\Psi}(\mathit{did}) &= \mathbf{did},\\ \mathbf{\Phi}(\mathit{did}) &= D_{im} \in \mathsf{C} \otimes \mathsf{T}\end{aligned}$$

We now derive a representation for the expression *What did Alice eat* (see Figure 4).

Step 1. We first merge *eat* and *what*, checking the $=\mathrm{D}$ feature from *eat* and the D feature from *what*. This gives us a subtree for the expression *eat what*, with feature string $[\mathrm{V}]$. In tensor-space,

[2] U is the vector space for the category v (i.e. little v, Koopman and Sportiche, 1991), while $\varnothing_v$ and $\varnothing_T$ are the empty phonological exponents for the syntactic categories v and T, respectively.

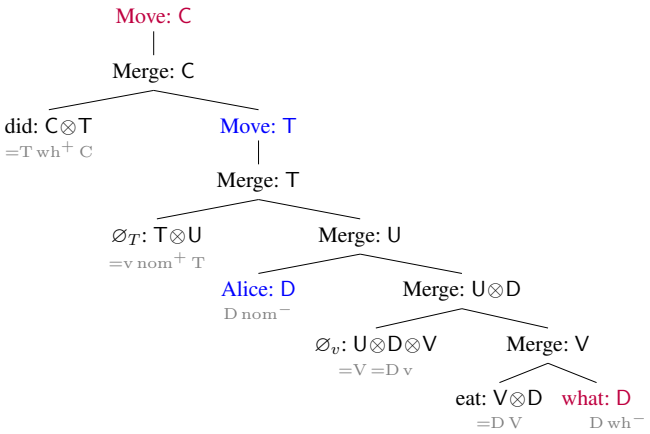

Figure 4: An MG derivation for "*What did Alice eat*", showing the semantic space for each lexical item and for each substructure as the result of merge and move operations.

syntactically we are **merge**-ing **eat** and **what**, and semantically we are contracting E_{lk} and W_k:

$$\begin{aligned}&\mathcal{M}erge((\textbf{eat},E_{lk}),(\textbf{what},W_k))\\=&(\textbf{merge}(\textbf{eat},\textbf{what}),E_l)\end{aligned}$$

with $\boldsymbol{\Phi}(\textit{eat what}) = E_{lk} \cdot W_k = E_l \in \mathsf{V}$, paralleling the remaining [V] feature of the substructure *eat what*. While *what* still has an active wh^- feature to allow for its displacement later in the derivation, recall that movement features are not reflected in the semantic space. For convenience, we write $t_0 := \textbf{merge}(\textbf{eat},\textbf{what})$.

Step 2. We now merge $\varnothing_v$ with *eat what*, checking the =V feature from $\varnothing_v$ and the V feature from *eat what*. The resulting expression $\varnothing_v$ *eat what* has feature string [=D, v]. In tensor-space:

$$\begin{aligned}&\mathcal{M}erge((\textbf{o},O_{njl}),(t_0,E_l))\\=&(\textbf{merge}(\textbf{o},t_0),O_{nj})\end{aligned}$$

Semantically, $\boldsymbol{\Phi}(\varnothing_v \textit{eat what}) = O_{njl} \cdot E_l = O_{nj} \in \mathsf{U}\otimes\mathsf{D}$, as expected. We let $t_1 := \textbf{merge}(\textbf{o},t_0)$.

Step 3. We merge $\varnothing_v$ *eat what* with *Alice*, checking the remaining = D feature from $\varnothing_v$ *eat what* and the D feature from *Alice*. This gives us *Alice* $\varnothing_v$ *eat what*, with feature string [v]. In vector space:

$$\begin{aligned}&\mathcal{M}erge((t_1,O_{nj}),(\textbf{Alice},A_j))\\=&(\textbf{merge}(t_1,\textbf{Alice}),O_n)\end{aligned}$$

with $O_{nj} \cdot A_j = O_n \in \mathsf{U}$, and $t_2 := \textbf{merge}(t_1,\textbf{Alice})$.

Step 4. The next merge gives us a structure for $\varnothing_T$ *Alice* $\varnothing_v$ *eat what*. For the vectors:

$$\begin{aligned}&\mathcal{M}erge((\textbf{p},P_{mn}),(t_2,O_n))\\=&(\textbf{merge}(\textbf{p},t_2),P_m)\end{aligned}$$

with $P_{mn} \cdot O_n = P_m \in \mathsf{T}$, and we let $t_3 := \textbf{merge}(\textbf{p},t_2)$.

Step 5. We encounter our first instance of move, triggered by the nom^+ feature introduced by $\varnothing_T$ checking the remaining nom^- feature on *Alice*. This results in the expression *Alice* $\varnothing_T$ $\varnothing_v$ *eat what*. For the vectors, syntactically we are applying the **move** operation to t_3. Recall that $\textbf{move} : \mathscr{F} \rightharpoonup \mathscr{F}$ is defined such that for any MG tree t, $\textbf{move}(\boldsymbol{\Psi}(t)) = \boldsymbol{\Psi}(\mathrm{move}(t))$. Semantically, there is no change in the vector space. That is:

$$\begin{aligned}&\mathcal{M}ove((t_3,P_m))\\=&(\textbf{move}(t_3),P_m)\end{aligned}$$

Again we let $t_4 := \textbf{move}(t_3)$.

Step 6. We merge *did* and *Alice* $\varnothing_T$ $\varnothing_v$ *eat what*, checking the =T feature from *did* and the T feature from *Alice* $\varnothing_T$ $\varnothing_v$ *eat what*. For the tensors:

$$\begin{aligned}&\mathcal{M}erge((\textbf{did},D_{im}),(t_4,P_m))\\=&(\textbf{merge}(\textbf{did},t_4),D_i)\end{aligned}$$

so that $D_{im} \cdot P_m = D_i \in \mathsf{C}$. We let $t_5 := \textbf{merge}(\textbf{did},t_4)$.

Step 7. Finally, one last move operation is triggered by the remaining wh^- feature on *what*, which is now checked against the active wh^+ feature introduced by *did*. This results in the expression *what did* $\varnothing_T$ *Alice* $\varnothing_v$ *eat*, and the only remaining feature associated with the tree structure is $[\mathrm{C}]$. In the vector space, we have $\mathbf{move}(t_5)$ for the syntactic representation: Semantically, there is again no change in the vector space.

$$\begin{aligned}&\mathcal{M}ove((t_5,D_i))\\ =&(\mathbf{move}(t_5),D_i)\end{aligned}$$

Figure 4 illustrates this full derivation, focusing on how the tensor-semantics happens in parallel to the symbolic feature-checking.

6 Discussion and Conclusion

In this paper, we constructed a tensor-semantics for MGs, illustrating how distributional approaches to lexical semantics can be enhanced by the compositional operations of MGs. Specifically, we built on beim Graben and Gerth (2012)'s formulation of merge and move as tensor-product operations over structural positions, and we added maps to tensor-based semantics representations from MG syntactic features (Maillard et al., 2014). Notably, we restricted our construction to cases in which the lexical semantics of a mover is fully integrated into the derivation when said mover is first merged. This is consistent with an interpretation of move as "internal merge" not contributing any novel lexical semantics to the derivational process.

However, move is often conceived as interacting with propositional meaning in more complex ways, for instance by affecting quantifier scope taking and variable biding. In this sense, our construction is a tensor-based implementation of Kobele (2012)'s moveEmpty operation, which does not affect the denotational content of a lexical item. Kobele (2012) also allows move to postpone the incorporation of the semantics of a mover to one of its landing sites via a type of storage. This approach is compatible with our framework, and it would require to delay the tensor-contraction associated with the selection of a lexical item containing a movement feature. We conjecture that this extension is straightforward, since the mappings of these extra move operations are still homomorphisms between partial algebras. Future work should define the technical details of this approach, while also evaluating alternative formalizations to the effects of movement on propositional semantics across clause types (see also Kobele, 2006; Hunter, 2007; Tomita, 2016).

In line with Maillard et al. (2014)'s work on CCGs, here we made no assumptions about how the semantic tensors should be interpreted. While we tie our framework to distributional semantics broadly construed, future work should investigate how semantically interpretable vectors for the different types of features can be acquired (Bos et al., 2004; Socher et al., 2007, 2012; Hermann and Blunsom, 2013). Encouragingly, recent work on broad-coverage neural parsers for MGs has shown unexpected success in extracting from corpora super-tags for MG-style lexical items, even for the controversial "phonologically null" syntactic categories (Torr, 2017, 2018; Torr et al., 2019). These results leverage heavily work on broad-coverage CCG parsing via super-tagging (Bangalore and Joshi, 1999), and suggest that several enterprises (learning semantic tensors, super-tag inference, and broad-coverage syntactic and semantic parsing) can benefit from insights across grammar formalisms (Kasai et al., 2017; Prange et al., 2021; Yamaki et al., 2023).

Finally, the addition of tensor-based semantics to the geometric MG formalization of beim Graben and Gerth (2012) opens the way to future work incorporating semantic information into dynamical models of parsing, with potential relevance to cognitively plausible models of sentence processing (Gerth and beim Graben, 2009; Smith and Vasishth, 2020; Villata and Tabor, 2022; De Santo, 2025). Soulos (2025) showcases a general framework for integrating tensor representations of compositional structure with various connectionist accounts of learning and processing, in line with Smolensky et al. (2022) proposal for neurosymbolic computing. Overall then, our results add another piece to the puzzle of how to successfully connect generative syntax to broader work on neuro-symbolic processing in cognitive science (Smolensky et al., 1993; Smolensky, 2012; Martin, 2020; Garcez and Lamb, 2023)

References

David Adger. 2003. *Core syntax: A minimalist approach*. Oxford University Press.

Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational linguistics*, 25(2):237–265.

Marco Baroni. 2013. Composition in distributional

semantics. *Language and Linguistics Compass*, 7(10):511–522.

Marco Baroni, Raffaella Bernardi, and Roberto Zamparelli. 2014. Frege in space: A program for composition distributional semantics. *Linguistic Issues in Language Technology*, 9.

Peter beim Graben and Sabrina Gerth. 2012. Geometric representations for minimalist grammars. *Journal of logic, language, and information*, 21(4):393–432.

Gemma Boleda. 2020. Distributional semantics and linguistic theory. *Annual Review of Linguistics*, 6:213–234.

Gemma Boleda and Aurélie Herbelot. 2016. Formal distributional semantics: Introduction to the special issue. *Computational Linguistics*, 42(4):619–635.

Johan Bos, Stephen Clark, Mark Steedman, James R Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a ccg parser. In *COLING 2004: Proceedings of the 20th international conference on computational linguistics*, pages 1240–1246.

Stephen Clark. 2015. Vector space models of lexical meaning. *The Handbook of Contemporary semantic theory*, pages 493–522.

Stephen Clark and Stephen Pulman. 2007. Combining symbolic and distributional models of meaning. In *Proceedings of AAAI Spring Symposum on Quantum Interaction*.

Daoud Clarke. 2009. Context-theoretic semantics for natural language: an overview. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 112–119, Athens, Greece. Association for Computational Linguistics.

Daoud Clarke, David Weir, and Rudi Lutz. 2011. Algebraic approaches to compositional distributional semantics. In *Proceedings of the Ninth International Conference on Computational Semantics (IWCS 2011)*.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36:345–384.

Elizabeth Coppock and Lucas Champollion. 2025. Invitation to formal semantics. *Manuscript, Boston University and New York University. Last Updated October 2025*.

Aniello De Santo. 2020. *Structure and memory: A computational model of storage, gradience, and priming*. Ph.D. thesis, State University of New York at Stony Brook.

Aniello De Santo. 2025. Capturing online SRC/ORC effort with memory measures from a minimalist parser. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 24–35, Albuquerque, New Mexico, USA. Association for Computational Linguistics.

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

Katrin Erk. 2022. The probabilistic turn in semantics and pragmatics. *Annual Review of Linguistics*, 8:101–121.

James Fodor and Shinsuke Suzuki. 2026. Using computational semantics to study meaning in the brain. *Neuroscience & Biobehavioral Reviews*, 181:106514.

Meaghan Fowlie. 2013. Order and optionality: Minimalist grammars with adjunction. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pages 12–20.

Werner Frey and Hans-Martin Gärtner. On the treatment of scrambling and adjunction in minimalist grammars. In *Proceedings of formal grammar*, volume 2002, pages 41–52.

Artur d'Avila Garcez and Luis C Lamb. 2023. Neurosymbolic AI: The 3rd wave. *Artificial Intelligence Review*, 56(11):12387–12406.

Sabrina Gerth and Peter beim Graben. 2009. Unifying syntactic theory and sentence processing difficulty through a connectionist minimalist parser. *Cognitive neurodynamics*, 3(4):297–316.

Thomas Graf. 2022. Subregular linguistics: bridging theoretical linguistics and formal grammar. *Theoretical Linguistics*, 48(3-4):145–184.

Edward Grefenstette. 2013a. *Category-theoretic quantitative compositional distributional models of natural language semantics*. Ph.D. thesis, Oxford University, UK.

Edward Grefenstette. 2013b. Towards a formal distributional semantics: Simulating logical calculi with tensors. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 1–10, Atlanta, Georgia, USA. Association for Computational Linguistics.

Edward Grefenstette, Mehrnoosh Sadrzadeh, Stephen Clark, Bob Coecke, and Stephen Pulman. 2011. Concrete sentence spaces for compositional distributional models of meaning. In *Proceedings of the Ninth International Conference on Computational Semantics (IWCS 2011)*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Irene Heim and Angelika Kratzer. 1998. *Semantics in Generative Grammar*. Blackwell, Malden, MA.

Karl Moritz Hermann and Phil Blunsom. 2013. The role of syntax in vector space models of compositional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 894–904, Sofia, Bulgaria. Association for Computational Linguistics.

Norbert Hornstein, Jairo Nunes, and Kleanthes K Grohmann. 2005. *Understanding minimalism*. Cambridge University Press.

Tim Hunter. 2007. Deriving syntactic properties of arguments and adjuncts from neo-davidsonian semantics. In *Conference on Mathematics of Language*, pages 103–116. Springer.

Shinnosuke Isono. 2024. Category locality theory: A unified account of locality effects in sentence comprehension. *Cognition*, 247:105766.

Jungo Kasai, Robert Frank, R. Thomas McCoy, Owen Rambow, and Alexis Nasr. 2017. TAG parsing with neural networks and vector representations of supertags. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1712–1722, Copenhagen, Denmark. Association for Computational Linguistics.

Gregory M. Kobele. 2006. *Generating copies: An investigation into structural identity in language and grammar*. Ph.D. thesis, University of California, Los Angeles.

Gregory M. Kobele. 2012. Importing montagovian dynamics into minimalism. In *Logical Aspects of Computational Linguistics - 7th International Conference, LACL 2012, Nantes, France, July 2-4, 2012. Proceedings*, Lecture Notes in Computer Science, pages 103–118. Springer.

Gregory M. Kobele, Sabrina Gerth, and John Hale. 2012. Memory resource allocation in top-down minimalist parsing. In *Formal Grammar - 17th and 18th International Conferences, FG 2012, Opole, Poland, August 2012, Revised Selected Papers, FG 2013, Düsseldorf, Germany, August 2013. Proceedings*, Lecture Notes in Computer Science, pages 32–51. Springer.

Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review*, 51(3):455–500.

Hilda Koopman and Dominique Sportiche. 1991. The position of subjects. *Lingua*, 85(2-3):211–258.

Joachim Lambek. 1958. The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3):189–224.

Serge Lang. 2002. The tensor product. In *Algebra*, chapter XVI, pages 601–640. Springer New York, New York, NY.

Alessandro Lenci. 2018. Distributional models of word meaning. *Annual review of Linguistics*, 4:151–171.

Jean Maillard, Stephen Clark, and Edward Grefenstette. 2014. A type-driven tensor-based semantics for CCG. In *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, pages 46–54, Gothenburg, Sweden. Association for Computational Linguistics.

Andrea E Martin. 2020. A compositional neural architecture for language. *Journal of Cognitive Neuroscience*, 32(8):1407–1427.

Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio. Association for Computational Linguistics.

Richard Montague. 1970. English as a formal language. In B. Visentini et al, editor, *Linguaggi nella Società e nella Tecnica*, pages 188–221. Milan: Edizioni di Communita. Reprinted in Thomason (ed.) 1974, pp. 188–221.

Satoru Ozaki, Aniello De Santo, Tal Linzen, and Brian Dillon. 2024. CCG parsing effort and surprisal jointly predict RT but underpredict gardenpath effects. In *Society for Computation in Linguistics*, volume 7, pages 362–364.

Jakob Prange, Nathan Schneider, and Vivek Srikumar. 2021. Supertagging the long tail with tree-structured decoding of complex categories. *Transactions of the Association for Computational Linguistics*, 9:243–260.

Daniel Quigley. 2025. A vector logic for extensional formal semantics. *Journal of Logic, Language and Information*, pages 557–599.

Garrett Smith and Shravan Vasishth. 2020. A principled approach to feature selection in models of sentence processing. *Cognitive science*, 44(12):e12918.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216.

Paul Smolensky. 2012. Symbolic functions from neural computation. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1971):3543–3569.

Paul Smolensky and Géraldine Legendre. 2006. *The harmonic mind: From neural computation to optimality-theoretic grammar (Cognitive architecture), Vol. 1*. MIT Press.

Paul Smolensky, Géraldine Legendre, and Yoshiro Miyata. 1993. Integrating connectionist and symbolic computation for the theory of language. *Current Science*, pages 381–391.

Paul Smolensky, Richard McCoy, Roland Fernandez, Matthew Goldrick, and Jianfeng Gao. 2022. Neurocompositional computing: From the central paradox of cognition to a new generation of ai systems. *AI Magazine*, 43(3):308–322.

Richard Socher, Chris Biemann, and Rainer Osswald. 2007. Combining contexts in lexicon learning for semantic parsing. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA 2007)*, pages 175–182, Tartu, Estonia. University of Tartu, Estonia.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Jeju Island, Korea. Association for Computational Linguistics.

Paul Soulos. 2025. *Unifying Neural and Symbolic Computation for Compositional Generalization: Representation and Processing*. Ph.D. thesis, The Johns Hopkins University.

Edward Stabler. 1996. Derivational minimalism. In *International conference on logical aspects of computational linguistics*, pages 68–95. Springer.

Edward P Stabler. 2013. Two models of minimalist, incremental syntactic analysis. *Topics in cognitive science*, 5(3):611–633.

Miloš Stanojević, Jonathan R Brennan, Donald Dunagan, Mark Steedman, and John T Hale. 2023. Modeling structure-building in the brain with ccg parsing and large language models. *Cognitive science*, 47(7):e13312.

Mark Steedman. 2001. *The syntactic process*. MIT press.

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. In Robert D. Borsley and Kersti Börjars, editors, *Non-Transformational Syntax*, chapter 5, pages 181–224. John Wiley & Sons, Ltd.

Whitney Tabor. 2009. A dynamical systems perspective on the relationship between symbolic and non-symbolic computation. *Cognitive neurodynamics*, 3(4):415–427.

Yu Tomita. 2016. Solving event quantification and free variable problems in semantics for minimalist grammars. In *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation: Oral Papers*, pages 219–227.

John Torr. 2017. Autobank: a semi-automatic annotation tool for developing deep minimalist grammar treebanks. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–86.

John Torr. 2018. Constraining mgbank: Agreement, l-selection and supertagging in minimalist grammars. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 590–600.

John Torr, Miloš Stanojević, Mark Steedman, and Shay B Cohen. 2019. Wide-coverage neural a* parsing for minimalist grammars. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 2486–2505.

Sandra Villata and Whitney Tabor. 2022. A self-organized sentence processing theory of gradience: The case of islands. *Cognition*, 222:104943.

Ryosuke Yamaki, Tadahiro Taniguchi, and Daichi Mochihashi. 2023. Holographic ccg parsing. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 262–276.