

A Lie-Algebraic Perspective on Tree-Adjoining Grammars

Isabella Senturia
Yale University

Elizabeth Xiao
California Institute of Technology

Matilde Marcolli
California Institute of Technology

Abstract

We provide a novel mathematical implementation of tree-adjoining grammars using two combinatorial definitions of graphs. With this lens, we demonstrate that the adjoining operation defines a pre-Lie operation and subsequently forms a Lie algebra. We demonstrate the utility of this perspective by showing how one of our mathematical formulations of TAG captures properties of the TAG system without needing to posit them as additional components of the system, such as null-adjoining constraints and feature TAG.

1 Introduction

Formal languages have long been one of the most prominent tools used by mathematical linguistics in attempts to develop formal systems generating natural language (Harrison, 1978; Sipser, 1996; Hopcroft et al., 2001). Different grammars enable descriptions of different phenomena in natural language to varying degrees. While the goal is often conceptualized as generating a given language as string of words, the inherent hierarchical nature of language leads to another aim: to generate a language via the appropriate tree structures modeling those hierarchical relationships within natural language. Tree-Adjoining Grammars (TAGs) (Joshi, 1987; Joshi and Schabes, 1997; Kroch and Joshi, 1985) are one approach to such a goal. Structure building via Merge in Minimalism is another approach.

At the same time, mathematicians have worked to reframe various linguistic structures and formulations as mathematical systems. With respect to formal languages, and in particular tree languages, Giraud (2019) demonstrates that the algebraic structures known as colored operads can successfully define various string languages (such as context-free languages) and tree languages (including those generated by tree grammars, regular

tree grammars, and synchronous grammars). Most recently, Marcolli et al. (2025a) have algebraically formalized Merge in the framework of the Strong Minimalist Thesis, modeling syntactic derivations in Hopf-algebraic terms. Also relevant to the Lie algebra formalism considered here, Marcolli and Port (2015) use different combinatorial definitions of graphs to show specific context-free and context-sensitive graph grammars have associated Lie algebras.

With respect to linguistic structure, operations to build larger trees from smaller trees are fundamental in capturing the compositional nature of syntax. In Minimalism, two trees are combined together by Merge by creating a new node and two edges, each of which joins to the two respective roots of the other trees. A complimentary perspective to this external composition, where the combining operation does not affect the internal structure of either of the two independent components, would be internal composition, where one tree is inserted *into* another. This operation is exactly the way that TAG operates, and pre-Lie operations' ability to capture this underpins the relevance of Lie algebras to syntactic structure-building. Moreover, there are two complimentary types of insertion of one tree into another, i.e. at a node (as in TAGs) or at an edge.¹ This paper explores the pre-Lie operation derived from node-insertion.

One mathematical motivation for studying Lie algebras is their close and richly-studied relationship with Hopf algebras. Every Lie algebra has a Hopf algebra associated to it through its universal enveloping algebra; conversely, the Milnor-Moore theorem provides conditions for a Hopf algebra to occur as the universal enveloping algebra of some Lie algebra. Pre-Lie algebras are also closely tied

¹Edge-insertion provides another example of a pre-Lie operation which can be compared and contrasted to the TAG pre-Lie operation. Due to issues of space, we are unable to expand upon this in this work.

to rooted trees in that the free pre-Lie algebra generated by a single element is isomorphic to the vector space of nonplanar rooted trees equipped with an insertion pre-Lie operator, which also has an inherent operadic description (Chapoton and Livernet, 2001). In fact, the Connes-Kreimer Hopf algebra of rooted trees is dual to the universal enveloping algebra of the Lie algebra from a free pre-Lie operator (Connes and Kreimer, 1998).

In this work, we explore the formal properties of tree-adjointing through the mathematical lens of Lie algebras. Although we initially define adjunction on trees in a more general setting, we can constrain our analysis to a particular TAG by restricting the set of generators to the initial and auxiliary trees which occur in that TAG. By recasting TAGs as an algebraic system, we demonstrate that the most appropriate mathematical representation of TAG that yields a Lie algebra requires TAG to be defined using the combinatorial definition of graphs in terms of corollas and half-edges (typically used in theoretical physics but not in computer science or formal language theory, so we refer to it as the “physics definition”). By allowing a single edge to be described as comprising of two half-edges, this provides a more elegant description of the elementary and auxiliary trees and their compositions. With this physics definition of trees, components of the TAG system such as null-adjointing constraints and specification of insertion positions naturally follow and do not need to be postulated ad-hoc (as they do with standard formulations of TAG). Not only this, but elaborations of TAG such as feature-TAG are easily implementable.

From the algebraic side, we show that various alternatives to the physics definition (undirected binary tree graphs and an elaboration on these, which we coin *double vertex* trees) cause problems with the Lie algebra structure (in particular, the pre-Lie operation) and also converge to the physics definition as the most natural algebraic description. We also show that colored operads can model TAG insertion via the composition of two insertion operations at the leaf.

A formulation of TAGs in terms of Lie algebra and operad insertions was suggested in Section 2.5.1 of Marcolli et al. (2025a), for the purpose of comparison with Merge, and is implemented here. While in formal languages generative models are usually compared in terms of their weak and strong generative capacity, a more refined comparison is possible, in the sense of being able to

more explicitly define and compare the underlying algebraic structures.

2 Background

In this section, we introduce the linguistic and mathematical concepts and tools necessary for the paper. We begin with TAGs, before defining graphs and Lie algebras.

2.1 Tree-Adjoining Grammars

We follow the presentation of tree-adjointing grammars as given in Joshi and Schabes (1997). We define the grammar below, before defining the two types of operations that allow building of larger trees out of insertion/composition of smaller trees.

Definition 2.1. *A tree-adjointing grammar (TAG) is a 5-tuple (Σ, N, I, A, S) , where Σ and N are finite sets of terminal and nonterminal symbols, respectively ($\Sigma \cap N = \emptyset$); $S \in N$ is a distinguished nonterminal symbol known as the start symbol; I and A are finite sets of (finite) trees called initial and auxiliary trees, respectively, whose interior nodes are labeled by nonterminal symbols, leaves of initial trees are labeled by either terminal or nonterminal symbols and those labeled by nonterminal symbols are marked for substitution. The auxiliary trees’ leaves are all marked for substitution except one, denoted by an asterisk and required to be labeled with the same label as the root.*

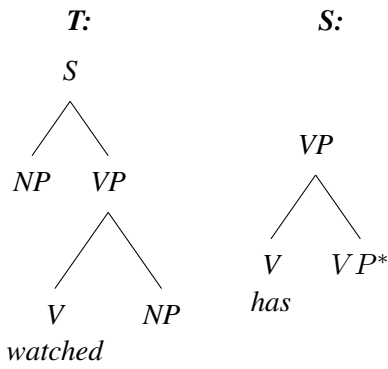
Lexicalized TAG requires at least one terminal symbol to appear at a leaf of every initial or auxiliary tree. *Elementary trees* are those contained in $I \cup A$, and *derived trees* are those built by composition of two or more trees. The two composition operations are defined as follows:

Definition 2.2. *The adjoining operation builds a new tree from an auxiliary tree S and an initial, auxiliary or derived tree T . This is done by inserting S into T at a node α when the root of S and a leaf of T are both labeled with α . In other words, the node labeled α in T is replaced by the tree S . We denote this operation, somewhat nontraditionally, as*

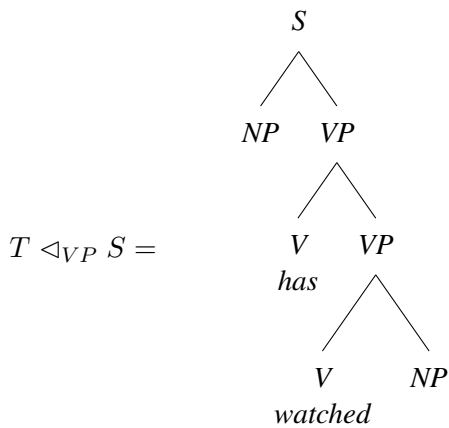
$$T \triangleleft_{\alpha} S,$$

in order to introduce the insertion operation which is relevant for the Lie algebra.

Example 2.3. Let S and T be the following trees:²



Then inserting S into T at the VP node of T yields



The second operation available in TAGs, *substitution*, was not included in the original definition of TAGs and in fact does not provide any additional expressive power. As such, our Lie-algebraic formalization of TAGs explicitly implementing only the adjoining operation (as the pre-Lie operation) is sufficient to express the entirety of the TAG formalism.

Typically, the arity of TAG trees is unary-binary—that is, nodes can have 0, 1 or 2 children. One can restrict to (full) binary trees (all non-leaves have two children and leaves have zero children) to simplify the set of objects or for comparison to other analyses (Lie-algebras of binary trees and linguistic analyses which consider Merge to be binary, such as Marcolli et al., 2025a).

TAG trees have labeled insertion spots at interior nodes. In our formulation, one can either have uniform insertions at every interior node, or labeled insertion: both cases are realized as a sum over all possible insertions. Similarly, we allow re-attaching of the remainder (part beneath the insertion spot) of the tree being inserted into to be reattached at any leaf, and denote this as a sum

over all leaves. We discuss variations of this labeling later in the paper when we demonstrate the ability of the physics graph formulation of TAG to seamlessly incorporate null-adjoining constraints and feature-TAG.

2.2 Math

We now present the mathematical concepts utilized in this paper: graphs, Lie algebras, and operads.

2.2.1 Graphs

The usual combinatorial definition of graphs is in terms of vertices and edges, namely we define a graph $G = (V(G), E(G))$, where $V(G) = \{v_1, v_2, \dots, v_n\}$ is a set of n vertices, $E(G) = \{e_1 = (v_a, v_b), \dots, e_m = (v_p, v_q)\}$ is a set of m edges. If the edges are undirected, the edge pair (v_i, v_j) is unordered, whereas if the edge is directed, the edge pair is ordered (*start, end*).

The *degree* d_v of a vertex v is the number of edges connected to that node. A *leaf* is a node of degree 1. Two *adjacent* vertices are connected by an edge. A *path* from some vertex v_i to another v_j is the sequence of edges connecting adjacent nodes between v_i and v_j . A graph is *connected* if there is a path from every node to every other node.

The class of *trees* is the class of connected *acyclic graphs* $T = (V, E)$ defined by the existence of exactly one path connecting any two distinct vertices $v_1, v_2 \in V$ —that is, they have no loops. A *directed tree* is a tree with directed edges. A *rooted tree* is a tree for which a specific node has been designated as the root, and is graphed with this root at the top or bottom. We refer to the set of nodes adjacent to, and below, another node v as the children of v . Any rooted tree can be viewed as a directed tree, where all edges are either uniformly directed towards the root, or away from it. A tree is *planar* (or, planarly embedded) if for all v , the children of v have a linear order.

2.2.2 Lie algebras

We follow the definition of Lie and pre-Lie algebras given in Procesi (2007) and Cartier and Patras (2021). All vector spaces are assumed to be over the real numbers \mathbb{R} , although our exposition will work for any field of characteristic 0.

A *Lie algebra* is a vector space V equipped with a bilinear product $[a, b]$ satisfying the Lie axioms of *antisymmetry*,

$$[a, b] = -[b, a],$$

²Adapted from Joshi and Schabes (1997).

and the *Jacobi identity*,

$$[a, [b, c]] + [b, [c, a]] + [c, [a, b]] = 0.$$

This product is called the *Lie bracket*. Due to antisymmetry, the Lie bracket is not commutative unless it is uniformly zero.

A vector space V is *graded* if it decomposes into a direct sum indexed by the non-negative integers: $V = \bigoplus_{n=0}^{\infty} V_n$, where each V_n is a subspace of V and $V_i \cap V_j = \{0\}$ whenever $i \neq j$. We say a graded vector space V is *connected* if V_0 is one-dimensional.³ If $a \in V_n$, then a is called a *homogeneous* (or *pure*) *element of degree n* . If each V_n is finite-dimensional, then V is *locally finite*.

A Lie algebra L is *graded* if the Lie bracket respects the grading: if $a \in L_m$ and $b \in L_n$, then $[a, b] \in L_{n+m}$.

Example 2.4. Any associative algebra A has a Lie bracket defined by $[a, b] := ab - ba$, which is called the commutator of a and b . For instance, let $A = M_2(\mathbb{R})$, the space of 2×2 matrices. Then

$$\left[\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix}\right] = \begin{pmatrix} 2 & 1 \\ 0 & 0 \end{pmatrix} - \begin{pmatrix} 2 & 0 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

We can hence say the following about a commutative algebra:

Example 2.5. If A is a commutative algebra, then $[a, b] = 0$ for all $a, b \in A$ and the commutator Lie bracket is trivial.

More generally, the Lie bracket is the commutator of a binary operation (such as a product or a pre-Lie operation), meaning it can be considered to be a measure of the degree to which the operation is not symmetric.

Let L, M be two Lie algebras. A linear map $\phi : L \rightarrow M$ is called a *Lie algebra homomorphism* if it commutes with the Lie brackets; that is, for $a, b \in L$,

$$\phi([a, b]_L) = [\phi(a), \phi(b)]_M.$$

2.2.3 Pre-Lie algebras

Let V be a vector space. A bilinear product $\triangleleft : V \times V \rightarrow V$ is called a (*right*) *pre-Lie operator* if it satisfies the (*right*) *Vinberg identity*

$$\begin{aligned} (a \triangleleft b) \triangleleft c - a \triangleleft (b \triangleleft c) \\ = (a \triangleleft c) \triangleleft b - a \triangleleft (c \triangleleft b), \end{aligned}$$

³This terminology originates from topology, where the number of generators of the zeroth homology group of a topological space counts its connected components.

making (V, \triangleleft) a (*right*) *pre-Lie algebra*. The expression $(a \triangleleft b) \triangleleft c - a \triangleleft (b \triangleleft c)$ is referred to as the *associator* $A(a, b, c)$ with respect to \triangleleft ; if \triangleleft is associative, then $A(a, b, c) = 0$ uniformly. The Vinberg identity asserts that the associator is unchanged when the second and third arguments are swapped; that is, for all $a, b, c \in V$,

$$A(a, b, c) = A(a, c, b).$$

A pre-Lie operator induces a Lie bracket defined by $[a, b] := a \triangleleft b - b \triangleleft a$, which automatically satisfies antisymmetry and the Jacobi identity, making A a Lie algebra. We do note that not all Lie algebras arise from a pre-Lie operator.

2.2.4 Tree-insertion as a free pre-Lie operator

Let $V = \text{span}(B)$ be a vector space, say of countable dimension, with basis given by $B = \{a, b, c, \dots\}$. We use $\text{span}(B)$ and $\langle B \rangle$ interchangeably to refer to the span. The *free associative algebra over V* has, as its basis, words over the alphabet B , such as $aabc$, $accbba$ or the empty word \emptyset ; the product is defined over basis elements by concatenation of words with no other relations. This construction is often denoted by $T(V)$ and called the *tensor algebra over V* ; if V is d -dimensional, we often call $T(V)$ the *free associative algebra on d generators* (or letters), which is unique up to isomorphism. Any tensor algebra has a natural grading over \mathbb{N} , where the n -degree component of $T(V)$ is generated by words of length n . There is an injective copy of V embedded in $T(V)$, and any extension of V to an associative algebra is a quotient of $T(V)$.

We can similarly define $L(V)$, the *free pre-Lie algebra over V* , with pre-Lie operator \triangleleft satisfying the Vinberg identity on the associators and no other relations. If $V = \text{span}(B)$ is d -dimensional, then we will refer to $L(V)$ as a *free pre-Lie algebra on d generators* (or letters), the elements of B . However, words over B are no longer sufficient to form a basis for $L(V)$, since \triangleleft is not associative; instead, we require strings to be parenthesized such that each pair of parentheses corresponds to one application of the \triangleleft operator. One interpretation would be as planar full binary trees with leaves labeled by elements of B , since such trees with n leaves are in correspondence with complete parenthesizations of a string of length n . This basis works for any non-associative algebra over V with no further relations, and it is reminiscent of the Merge operation since it combines left and right arguments into a single

tree by grafting them together as the left and right child, respectively, of a new root node. However, this basis does not provide us with a convenient geometric or combinatorial interpretation of the associator, since they are defined over a difference of basis elements.

Another interpretation of the free pre-Lie algebra over V is due to [Chapoton and Livernet \(2001\)](#), where basis elements are nonplanar rooted trees (not necessarily binary), with vertices labelled by basis elements of V . The pre-Lie operator can now be interpreted geometrically as insertion: if T and S are two trees, then $T \triangleleft S$ is the sum of all possible trees that result from grafting S to T by joining a node of T to the root of S with a new edge.

Since a new edge is produced in every nontrivial insertion, trees are graded by numbers of vertices and \triangleleft respects the natural grading. The degree zero component of $L(V)$ is generated by the empty tree \emptyset ; for any tree T , we have $T \triangleleft \emptyset = T$ and $\emptyset \triangleleft T = \emptyset$, which extends to the entire vector space by linearity. In the following example, note that the trees are non-planar and so we are currently ignoring linear ordering of the leaves.

Example 2.6. *Let*

$$T = \begin{array}{c} \textcircled{a} \\ / \quad \backslash \\ \textcircled{b} \quad \textcircled{c} \end{array}, \quad S = \textcircled{b}.$$

Then

$$T \triangleleft S = \begin{array}{c} \textcircled{a} \\ / \quad \backslash \\ \textcircled{b} \quad \textcircled{c} \\ | \\ \textcircled{b} \end{array} + \begin{array}{c} \textcircled{a} \\ / \quad \backslash \\ \textcircled{b} \quad \textcircled{c} \\ | \\ \textcircled{b} \end{array} + \begin{array}{c} \textcircled{a} \\ / \quad \backslash \\ \textcircled{b} \quad \textcircled{c} \quad \textcircled{b} \end{array}.$$

This model provides a geometric interpretation for the associator. If T_1, T_2, T_3 are three trees, then $(T_1 \triangleleft T_2) \triangleleft T_3$ is a sum of all trees obtained by first inserting T_2 into T_1 , and then inserting T_3 into the resulting tree. The associator consists exactly of the trees produced by inserting T_2 and T_3 into distinct vertices of T_1 .

Fix a label $\alpha \in B$ from the basis. We can define a restricted insertion operator \triangleleft_α where grafting only occurs at nodes in the first argument labelled α ; this is still a pre-Lie operator. The general pre-Lie operator allowing insertion at every node is the sum of the restricted operator over all basis elements: $\triangleleft = \sum_{\alpha \in B} \triangleleft_\alpha$.

The insertion operators here will serve as a framework for our definition of TAG as a pre-Lie operator. Tree-adjointing will be viewed as a kind of “insertion” where, instead of grafting one tree to a node of the other, the inserted tree becomes embedded inside another one.

2.2.5 Tree-adjointing as a pre-Lie operator

Let X be a collection of trees. For now, we will assume the trees in X are binary, planar and unlabelled. Let $\mathcal{T} = \mathcal{T}_{bin,pl}$ be the free vector space with X as its basis. We define \triangleleft on basis elements $T, S \in X$ by adjoining S at every possible position in T over every leaf of S ; then, $T \triangleleft S$ is the sum of all trees produced by this process.

Note that we permit adjunction of S into T at leaves of T . This procedure superficially resembles the process of substitution, where a leaf is replaced by a larger tree. However, adjunction at leaves is different because it implicitly pairs the root of the inserted tree with one of its leaves, whereas substitution only considers the root of the inserted tree. Hence the coefficient of a tree obtained in such a way in $T \triangleleft S$ will be at least the number of leaves in S .

In general, a tree having a coefficient larger than 1 in $T \triangleleft S$ means it is obtainable from adjunction in multiple different ways.

If T has n vertices and S has ℓ leaves, then the sum of the coefficients of $T \triangleleft S$ will be $n\ell$. The operation \triangleleft is then extended to the whole vector space \mathcal{T} by linearity in both arguments.

Example 2.7. *Let*

$$T = \begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array} \quad \text{and} \quad S = \begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array},$$

which respectively have 5 nodes and 2 leaves. We show the result of adjoining S to T . The copy of S in each term is drawn with white nodes and dashed edges purely for illustrative purposes; the actual trees are unlabelled.

$$\begin{aligned} T \triangleleft S &= \begin{array}{c} \circ \\ / \quad \backslash \\ \bullet \quad \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array} + \begin{array}{c} \circ \\ / \quad \backslash \\ \bullet \quad \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array} + \begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array} + \begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array} \\ &+ 2 \begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array} + 2 \begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array} + 2 \begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array} \\ &= \begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array} + 4 \begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array} + 3 \begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array} + 2 \begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array} \end{aligned}$$

In the first expression for $T \triangleleft S$, the last three terms each have a coefficient of 2 because adjunction occurs at a leaf of T . There is no subtree to be attached to any leaf of S ; however, a leaf of S still needs to be chosen to ensure consistency in the associator, so the two copies correspond to the choices of leaf in S .

We reiterate that each tree has a fixed planar embedding; indeed, if the trees were nonplanar, then the first three terms in the last line would represent the same tree.

The insertion operation above has the following property:

Theorem 2.8. \triangleleft is a pre-Lie operator.

See [proof](#) on page 13.

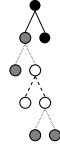
Example 2.9. Let

$$T_1 = \begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array}, \quad T_2 = \begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array}, \quad T_3 = \begin{array}{c} \circ \\ / \quad \backslash \\ \circ \quad \circ \end{array}.$$

Then



appears as a term of $(T_1 \triangleleft T_2) \triangleleft T_3$ only, since adjunctions occur at different vertices of T_1 , while



appears in both $(T_1 \triangleleft T_2) \triangleleft T_3$ and $T_1 \triangleleft (T_2 \triangleleft T_3)$, hence will be cancelled out in the associator.

An immediate question is whether or not tree-adjointing, in any formulation (planar or non-planar, labelled or unlabelled), is isomorphic as a pre-Lie algebra to a free one. This question is relevant to an algebraic comparison between TAGs and the insertion Lie algebra dual to the Hopf algebra of workspaces in Minimalism. Before we can answer this question, we will need to fix some more conditions on the basis elements of \mathcal{T} and how exactly the tree-adjunction operation is defined. Several different possibilities will be considered in §3 on the mathematical formulation of TAG, but we will use the binary planar unlabelled case as our first example.

We note the behaviour of the single-node tree in insertions:

Example 2.10. Let T be a binary planar unlabelled tree, and let \bullet be the tree consisting of a single node. Then

$$\begin{aligned} T \triangleleft \bullet &= |T|T \\ \bullet \triangleleft T &= \ell(T)T \end{aligned}$$

where $|T|$ is the number of nodes in T , and $\ell(T)$ is the number of leaves. Hence the Lie bracket of T and \bullet is

$$[T, \bullet] = (|T| - \ell(T))T.$$

Theorem 2.11. $\mathcal{T}_{bin,pl}$ is not isomorphic as a pre-Lie algebra to L_{free} , the free pre-Lie algebra on one generator.

See [proof](#) on page 14.

To prevent overgeneration, we can introduce labels to $\mathcal{T}_{bin,pl}$ and modify the adjunction operation so that adjunction only occurs when an insertion spot of T has the same label as the root of S , and moreover that S has at least one leaf with the same label, to which the subtree of T will be attached. If S has a unique such leaf, then it represents the distinguished foot node of an auxiliary tree in TAG. See 3.1 for an example.

2.2.6 Operads

As mentioned in the introduction, operads have recently been connected to mathematical linguistics (Giraud, 2019; Marcolli et al., 2025a). Colored operads have most recently been used in formal language theory to obtain a new proof of the well-known Chomsky-Schützenberger representation theorem (Melliès and Zeilberger, 2025). Colored operads are also used in Marcolli and Larson (2025) and Marcolli et al. (2025b) to model theta roles and phases in Minimalism, as outlined in Marcolli et al. (2025a). Indeed, in this work we also utilize colored operads as operations on the vector space containing TAG trees that can mimic the TAG pre-Lie insertion operation as a combination of two colored-operad compositions. We define the notion of a colored operad below, before returning to it in section 3.3.2 to demonstrate how it appropriately captures the Lie-algebraic TAG system.

We provide here a preliminary introduction to the theory of colored operads, and we refer the reader to Giraud (2019) for a more detailed formulation in a formal languages context.

An operad organizes the compositional structure of operations that take multiple inputs and produce a single output. It is a collection \mathcal{D} of sets $\mathcal{D}(n)$,

$\mathfrak{D} = \{\mathfrak{D}(n)\}$ that consist of operations $T \in \mathfrak{D}(n)$ with n inputs and one output. The operad has an algebraic structure defined by composition operations

$$\gamma : \mathfrak{D}(n) \times \mathfrak{D}(k_1) \times \dots \times \mathfrak{D}(k_n) \rightarrow \mathfrak{D}(k_1 + \dots + k_n)$$

These composition operations take the single output out of an operation in $\mathfrak{D}(k_i)$ and plug it into the i -th input of an operation in $\mathfrak{D}(n)$. This results in a new operation in $\mathfrak{D}(k_1 + \dots + k_n)$ whose set of inputs is the union of all the inputs of the operations $\mathfrak{D}(k_i)$, and a single output, which is the output of the operation in $\mathfrak{D}(n)$ they are composed with. The compositions γ are associative. Operads can be *unital*, *symmetric*, etc., but we omit these definitions here as they are not relevant for the current work.

An *algebra* A over an operad \mathfrak{D} is a set that can serve as inputs and outputs for the operations in \mathfrak{D} , namely for which there are maps

$$\gamma_A : \mathfrak{D}(n) \times A^n \rightarrow A$$

compatible with the γ compositions of \mathfrak{D} .

A *colored operad* $\mathcal{O} = \{\mathcal{O}(c, c_1, \dots, c_n)\}$ is an operad where inputs and outputs are labelled (by a set of colors $c, c_i \in \Omega$) and composition can only occur when the output color c of one operation matches the color c_i of the i -th input it feeds to.

In fact, the concept of a pre-Lie operator is inherently operadic. [Chapoton and Livernet \(2001\)](#) explain that pre-Lie algebras naturally arise as algebras over a binary quadratic operad.

In the next section, we mathematically define TAGs in three different ways, and show that only the third mathematical formulation of TAG (using the physics definition of graphs) suffices when we would like to require the TAG to be a Lie algebra.

3 Mathematical Formulation of TAG

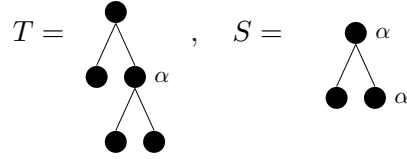
To begin formulating TAG as a mathematical system, we use the combinatorial graph definition of trees as given in section 2.2.1.

3.1 Normal single vertex method

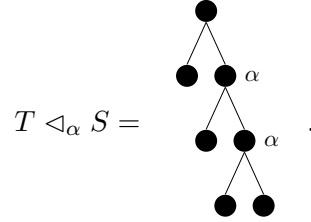
In what follows, we will demonstrate insertion at a specific (labeled) location in the tree, instead of giving the sum over all possible insertions. We label the spot we insert at in T , as well as the two locations (root and specified leaf) that the adjoining occurs at in S , with α . Note also that usually we would be summing over all leaves for the lower

part of T to re-attach to, but again for simplicity we omit this.

Example 3.1. We formulate the trees from the initial TAG example (2.3) as graphs. Let



Then



The problem with this comes when we think about the grading of the vector space. We are motivated to introduce a grading to our vector space because trees are fundamentally combinatorial objects with a notion of size. Furthermore, provided that the set of nodes labels is finite, then the vector space will be locally finite. Ideally we would also want the degree-0 part of the vector space to be generated by a single element, such as the empty tree, so that the vector space is connected, much as in the case of the free pre-Lie algebra. Indeed, if V is connected as a graded vector space, then the universal enveloping algebra of $L(V)$ is a graded connected Hopf algebra, which carry many desirable properties. The pre-Lie operator and Lie bracket with respect to the degree-0 generator should be simple enough that their outputs can be described in closed-form, potentially allowing us to determine whether or not the (pre-)Lie algebra is isomorphic as (pre-)Lie algebras to another one.

If we grade over vertices, we see that the insertion operation does not preserve gradation: inserting an n -noded tree S into an m -noded tree T yields an $n + m - 1$ -noded tree, because S is replacing a vertex in T and hence one vertex overall is lost.

A possible repair to this, to preserve the gradation of the vector space, would be to grade over number of edges rather than number of vertices. Then inserting an n -edged tree S into an m -edged tree T yields an $n + m$ -edged tree, which fixes that problem, but there is now a new issue: there are two unique trees with zero edges. These are the empty tree, and the tree comprising of a single node.

The situation is further complicated when trees are labeled, since there is now a distinct single-noded tree for every label. As stated in 2.10, the insertion of any of these single-noded trees will return T multiplied by a factor of $|T|_\alpha$, the number of times α occurs in T , meaning it is not exactly the identity insertion operation and hence even more problematic.

Mathematically, we can solve this ambiguity by equating all single-noded trees with the generator of the degree-0 component, which is achieved by quotienting out the ideal generated by elements $1 - \bullet_\alpha$ for all labels α , where \bullet_α is the single-noded tree where the only node is labelled α . However, this trivializes the Lie bracket with respect to \bullet_α , and insertions of the form $T \triangleleft \bullet_\alpha$ are no longer eligible to be used as a way of counting occurrences of α in T . Thus, we are motivated to reformalize the TAG tree graphs in a different way. This is done with the double vertex method in the following section.

3.2 Double vertex trees

Rambow et al. (2001) suggest that locations within a tree where adjunction was possible, which are normally represented as a node in the tree, are actually two nodes, an upper copy and a lower copy, connected by a dashed line (see figure 1).

Inspired by this and in order to fix the grading issue raised by defining trees as regular graphs, we can introduce a modification on the previous trees which we call *double-vertex trees*. These trees are also graphical trees, with the caveat that any node α that is able to be inserted into is actually a double node, i.e. is counted as two nodes. The maximum count of any vertex is two, meaning we cannot increase the count of a vertex to any $n \in \mathbb{N}$. In order to prevent this unbounded increase, and because we are working with unlabeled trees, i.e. where adjoining can occur freely at any location, this means that in any non-auxiliary tree every node will be a double vertex. In every auxiliary tree, there will be exactly two single vertices: the root,

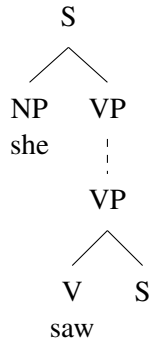
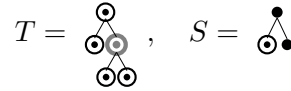


Figure 1: A tree adapted from Rambow et al. (2001) which can be inserted into at the VP node(s).

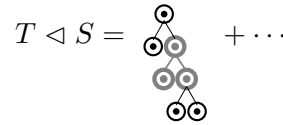
and one leaf. These are the two nodes that will each get one additional count from the double vertex α they are inserting into, meaning that the root and that distinguished leaf will become double-vertex interior vertices in the larger post-adjunction tree.

We can see this in the following example, which recasts example (3.1) in terms of this new counting system.

Example 3.2. Let S, T be as above, i.e.



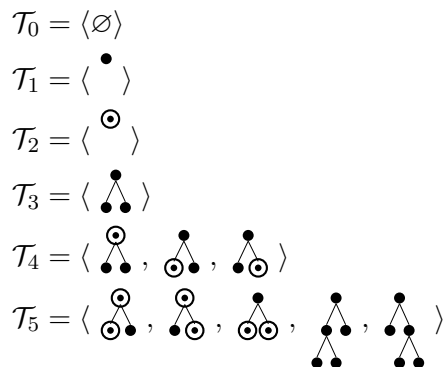
Note that every node in T is doubled and otherwise unlabelled. Inserting S into T at the marked node (which is gray, with a thicker border) yields



However, this model also has its drawbacks. In particular, we are now able to grade by vertices, because we see that the insertion operation does preserve gradation: inserting an n -noded tree S (with $(n - 2)/2$ double-nodes and 2 single nodes) into an m -noded tree T yields an $n + m$ -noded tree (as the double node in T where they are both inserted contributes one node to each of the two single nodes, where adjunction takes place, turning those two single nodes into two double nodes). That being said, there is still a strange effect that this has on the overall gradation of the vector space. All odd-degree components are ignored when restricting to the trees we use in TAG: since all binary trees have an odd number of nodes, odd-degree indicates that there are an odd number of single nodes.

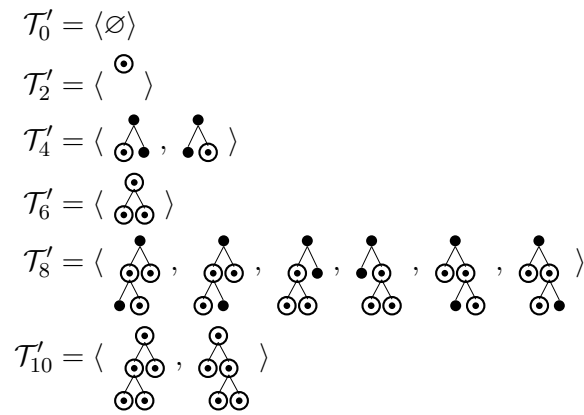
The following is the set of all gradations of the space up to five nodes.

Example 3.3.



We notice that gradation by nodes results in two trees being able to have the same dimension (number of nodes) but they can have different numbers of edges, e.g. in \mathcal{T}_5 there are trees with either two or four edges. This vector space contains many trees that will never be used in tree-adjunction, so we will consider the subspace $\mathcal{T}' \subseteq \mathcal{T}$ restricted to the trees that appear in TAG, which is closed under \triangleleft . Under the grading inherited from \mathcal{T} , the even-degree components of \mathcal{T}' have the following generators; the odd-degree, as mentioned above, are trivial.

Example 3.4.



There is now an alternation in the degrees; the ones which are 0 mod 4 contain the auxiliary trees, whereas the ones which are 2 mod 4 contain the non-auxiliary trees where all nodes are doubled. The difference between the two fundamental types of trees is encoded in the grading. Note that \mathcal{T}' is still not a free pre-Lie algebra since certain insertions of basis elements produce zero: namely, any attempt to adjoin a non-auxiliary tree into another tree is zero since it is an invalid insertion.

This method’s particular utility in being able to encode the different types of trees (auxiliary vs. elementary) in the grading scheme is one reason that these tree graph encodings should continue to be studied in future work. However, the amount of hoops necessary to jump through in order to establish this class of graphs as useful as a mathematical formulation of TAG motivates us to look at a third graphical encoding of TAG trees, which is even simpler than this version and has particularly interesting linguistic implications.

3.3 TAG as physics graphs

We present an alternative combinatorial definition of graphs that is primarily used in theoretical

physics (Marcolli and Port, 2015), which is convenient for defining the colored operads model of tree operations.

3.3.1 Physics definition

A graph $G = (C(G), \mathcal{F}(G), \mathcal{I})$ consists of a set $C(G) = \{v_1, \dots, v_n\}$ of n corollas (nodes distinguished by having half-edges attached to them), a set $\mathcal{F}(G) = \{f_1, \dots, f_m\}$ of m flags (or half-edges), and an involution $\mathcal{I} : \mathcal{F}(G) \rightarrow \mathcal{F}(G)$ on the flags. The external edges $E_{ext}(G)$ of G are the flags $f \in \mathcal{F}(G)$ fixed by the involution, i.e. $\mathcal{I}(f) = f$, while the internal edges $E_{int}(G)$ correspond to the two-element subsets $\{f, f'\} \in \binom{\mathcal{F}(G)}{2}$ such that $\mathcal{I}(f) = f'$ (and correspondingly $\mathcal{I}(f') = f$). The valence of a corolla $v \in C(G)$ is the number of half-edges attached to it.

Operadic insertion is now represented by joining external edges; adjunction involves splitting an internal edge by dividing it into its component flags and joining two external edges of a new graph, one to each flag.

This model also provides a concrete distinction between non-auxiliary trees and auxiliary trees. Auxiliary trees have exactly two half edges; one corresponding to the root, and another corresponding to the leaf that has the same label as the root in Joshi’s original formulation of TAG.⁴

In fact, there is some nuance here. Because we are representing the pre-Lie insertion operation as a summation over all possible insertions, this will be generalized as *all* leaves of auxiliary trees being half edges—the insertion operation of $T \triangleleft S$ then varies across the half edge beneath the insertion point in T being re-attached in turn to each half edge leaf of S. This naturally motivates the colored-operad lens on the TAG Lie algebra—every single leaf is a location where the colored operad will, in turn, insert a terminal node with a half edge attached to it. We return to this idea later on in this section, but note that we will not be writing the trees with terminal symbols and half-edges for all leaves, for sake of convenience.

The concept of half-edges is reminiscent of the usage of half-arcs to represent features in Minimalist grammars, where matching features are joined together. We explore this interpretation further in Section 4.1.

⁴See, for instance, tree S in example (3.6).

3.3.2 Operadic definition of TAG Lie algebra

The following describes how TAG insertion can be implemented via the composition of two unary operad insertions.

Definition 3.5. Consider the insertion $T \triangleleft S$, where insertion location is labeled with α . T was originally derived from composing two trees at α via an operad composition:

$$T = \mathcal{O}(T_1, T_2) = T_1 \circ_{\alpha} T_2. \quad (1)$$

This α composition location is exactly where T will be broken apart in order to insert S between the two:

$$T \triangleleft_{\alpha} S = \mathcal{O}(\mathcal{O}(T_1, S), T_2) = (T_1 \circ_{\alpha_u} S) \circ_{\alpha_l} T_2$$

u and l stand for “upper” and “lower” copies of α , respectively.⁵ Note that α_u actually comes from S , and α_l comes from T , because the edge above α in T is what is split into two half edges in order to insert S .

Because one of the α nodes is coming from the auxiliary tree S , and the other is coming from the tree being adjoined into, T , this results in the grading by vertices is preserved with this formalization of graphs, maintaining the connectedness of the space and hence solving those issues as they pertained to the previous two graph formalisms.

In the following example, we demonstrate the physics graph definition as it is applied to the TAG trees, using our running example, and show how the adjoining operation can be performed via the composition of two operad operations.

Example 3.6. An example of the insertion operation (outside of the operad). Supposed the grading is by nodes.

Let S, T as in the previous examples:

$$T = \begin{array}{c} \diagup \quad \diagdown \\ \alpha \\ \diagdown \quad \diagup \end{array}, \quad S = \begin{array}{c} \uparrow \\ \alpha \\ \diagdown \quad \diagup \end{array}$$

To insert S into T at α , forming $T \triangleleft_{\alpha} S$, we break T into T_1 and T_2 such that $T = T_1 \circ_{\alpha} T_2$, and

⁵One may view conceptualizing a tree containing an insertion spot at an interior node as being comprised of two trees who each contain that node on their periphery (the upper copy being a leaf in T_1 and the lower copy being the root of T_2) as linguistically-alternative, but any tree can be viewed as the composition of a subgraph and the quotient graph over the subgraph.

specifically the edge above α in T is broken into two half edges. Hence, T_2 retains α :

$$T_1 = \begin{array}{c} \diagup \quad \diagdown \\ \end{array}, \quad T_2 = \begin{array}{c} \uparrow \\ \alpha \\ \diagdown \quad \diagup \end{array}$$

Then inserting S into T at α is simply the composition

$$\begin{aligned} T \triangleleft_{\alpha} S &= T_1 \circ_{\alpha} S \circ_{\alpha} T_2 \\ &= \begin{array}{c} \diagup \quad \diagdown \\ \end{array} \circ_{\alpha} \begin{array}{c} \uparrow \\ \alpha \\ \diagdown \quad \diagup \end{array} \circ_{\alpha} \begin{array}{c} \uparrow \\ \alpha \\ \diagdown \quad \diagup \end{array} \\ &= \begin{array}{c} \diagup \quad \diagdown \\ \\ \\ \alpha \\ \diagdown \quad \diagup \\ \\ \\ \alpha \\ \diagdown \quad \diagup \end{array} \end{aligned}$$

There is a distinction that must be made regarding the arity of the colored operads we will use in the context of the TAG Lie algebra. In the Lie algebra formalism we sum over insertions at all possible locations, and readjoining at any possible leaf. Then, in order to complete the TAG tree, after readjoining at one of the leaves, the remaining $l - 1$ leaves must be “capped off” with operadic insertion of $l - 1$ half-edges whose single nodes are labeled with terminals. When we use operadic compositions to describe the insertion operations, we can decompose the composition γ (that fills all inputs at once) into repeated compositions $\circ_i : \mathfrak{D}(n) \times \mathfrak{D}(m) \rightarrow \mathfrak{D}(n + m - 1)$ that perform a single match of output to input. When the color-matching is also taken into account, these give (1). Thus, we can assume that only one leaf is a half-edge and all the other leaves are filled with terminals. In operadic terms, filling with terminals is part of the algebra over an operad structure, see [Giraudo \(2019\)](#).

4 Implications of Mathematical Formalizations of TAG

The physics definition of a graph makes it easier to recast TAG in terms of the graph grammars in

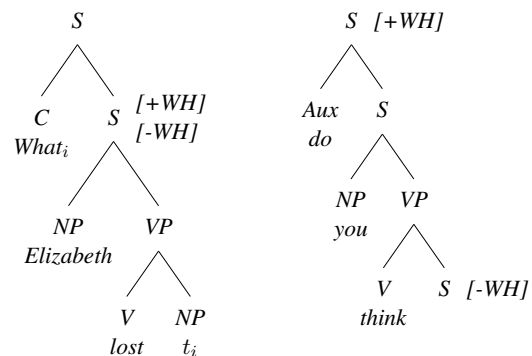
Marcolli and Port (2015). While it was possible to describe the production rules of TAG in terms of the combinatorial definition of graphs, the fact that tree-adjoining must be ultimately viewed as an insertion-elimination operation introduces an unwelcome degree of context-dependence to the formulation. For every label at which adjunction may occur, there needs to be a distinct production rule for every possible degree and every possible combination of labels that the neighbours of this node may have, in order to account for all possible insertions. When using the physics definition, the production rules may be defined in terms of the corollas; in this case, we still need a distinct production rule for each possible valence but the labels of the neighbouring vertices no longer need to be considered.⁶

4.1 Linguistic implications

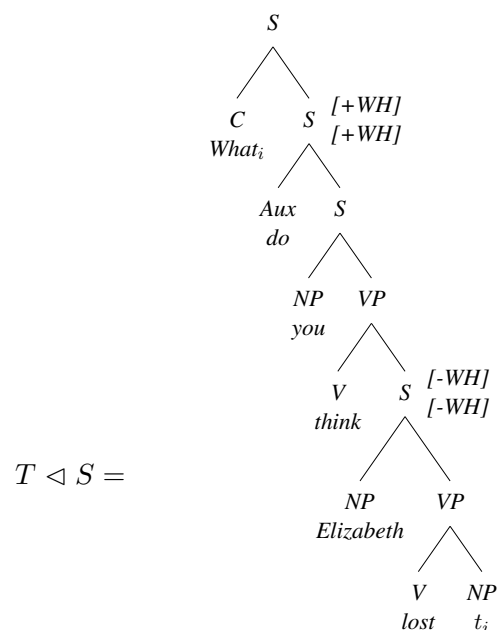
A striking benefit of this model is that null-adjoining constraints are *automatically* granted based on whether or not an edge is allowed to be split into two half edges. If it is restricted from being two half-edges, then adjoining is not possible. If it is required to split, then adjoining is required. This leads into another immediate linguistic payoff: feature-TAG.

In feature-TAG (Vijay-Shanker and Joshi, 1988), the argument is made that tree adjoining is required when there is a featural mismatch in an elementary tree at a given node, e.g. it has [+WH] from above but [-WH] from below. Then in order to resolve this, the insertion of an auxiliary tree which has [+WH] in the upper node and [-WH] in the lower node would fix this, because the two [+WH] would match as well as the two [-WH], resolving the fact that originally the elementary tree had [+WH] and [-WH] in the same node.

Example 4.1. To form the question “What do you think Elizabeth lost?”, we can use the following single elementary tree *T* (left) and auxiliary tree *S* (right):



In the elementary tree, there is a featural mismatch in the lower *S* node, because above this *S* node “What” dictates it is [+WH], but below this *S* node there is no such lexical time/syntactic feature, meaning the lower feature is [-WH]. This featural mismatch prompts the insertion of *S* into *T*, yielding the following well-formed tree with no featural mismatches:



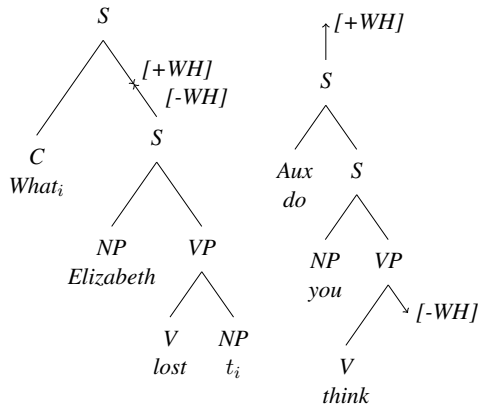
Implementation of feature TAG follows easily from the half-edge model of TAG.⁷ We can think about the half edges as carrying the feature labels, so an original α node’s parent edge is split in half exactly when the upper half edge has a [+F] and the lower half edge is labeled by [-F] (or vice-versa). Then the auxiliary tree’s upper half edge coming out of the root must carry a [+F] and the lower half edge, representing a leaf location, must carry a [-F]. After adjoining the two sets of two half edges will

⁷Note that this implementation is based upon labeling half-edges—this is an inherent component of the physics definition, so no additional functionality is being added to the system.

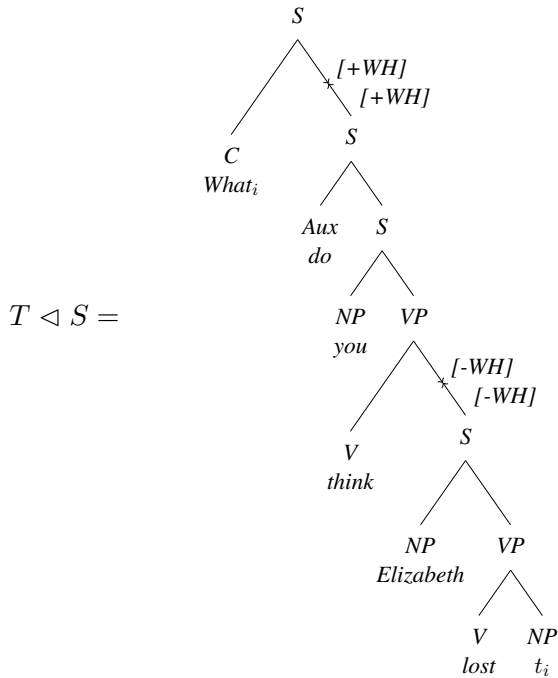
⁶For an example of this, see §3 of Marcolli and Port (2015).

each have featural match. Below, we implement example (4.1) with the physics-based TAG graphs.

Example 4.2. *The trees T and S are, respectively,*



The insertion of S into T follows from the feature mismatch of the two half-edges—and this is resolved with the insertion:



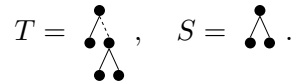
4.2 Mathematical implications

The Lie-algebraic structure provides many further avenues of exploration. The fact that we have managed to define graded and connected Lie algebra structures using TAG is convenient for future investigation of the universal enveloping algebra, since the universal enveloping algebra is graded, connected and furthermore commutative as a Hopf algebra. A graded connected commutative Hopf algebra is known to be free as an algebra (in other words, it is isomorphic to a polynomial algebra), thanks to a theorem of Hopf-Leray, which means that it can easily be implemented in computational

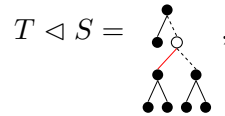
algebra programs. If the Hopf algebra arises from a pre-Lie structure, then the coproduct can be defined in terms of the pre-Lie operator, which in our case has an intuitive combinatorial description in terms of trees. Indeed, any monomial over tree generators can be interpreted as a forest, so the Hopf algebra has a basis indexed by forests. All of these properties facilitate explicit calculations in the Hopf algebra.

As mentioned in the introduction, TAG insertion via nodes is only one type of graphical insertion. The other insertion type involves inserting into existing edges via splitting an edge into two with a node in the middle, and then connecting that node to the tree being inserted via another edge, as depicted in the following example.

Example 4.3. *Let S, T be as previously, i.e.*



Inserting S into T at the dashed edge yields



where the single dashed edge has been split into two dashed edges via the creation of the white node and a new (red) edge has been created connecting the root of S to the new node.

This creation of two new edges allows the insertion operation to preserve the binary nature of the trees. It is worth recasting TAG insertion as an edge-insertion operation (e.g. above the labeled vertex, as the half-edges constructed in the physics formulation were) and exploring the algebraic differences and implications of these two different types of TAG insertion.

Another interesting perspective is due to a universal property of connected commutative Hopf algebras, which states that any such Hopf algebra is characterized by a unique nontrivial 1-cocycle. For the Connes-Kreimer Hopf algebra of rooted nonplanar trees, which arises from the free pre-Lie algebra on one generator, this cocycle acts on forests by grafting the component trees to a common root. When the forest consists of exactly two trees, this provides a description of the fundamental Merge operation in Marcolli et al. (2025a).

It would be interesting to examine the corresponding cocycles for the Hopf algebras obtained

from the TAG Lie algebras. We expect them to be different from the grafting procedure since the pre-Lie operation based on TAG is not free.

5 Conclusion

In this paper, we formalized tree-adjoining grammars mathematically as Lie algebras where the pre-Lie operation was defined via the adjoining operation of TAG. In order to work with TAGs as mathematical objects, we demonstrated that representing the trees as graphs is only feasible in the Lie-algebra setting when we use the physics definition of graphs, allowing us to express single edges as two half-edges. This mathematical motivation also had striking linguistic payoffs, allowing various components of TAGs usually posited as additional constraints to be inherent via the nature of the graphs.

Acknowledgements

We thank Robert Frank for his useful comments and suggestions. The third author is supported by NSF grant DMS-2104330.

References

- Pierre Cartier and Frédéric Patras. 2021. *Classical Hopf algebras and their applications*, volume 29 of *Algebra and Applications*. Springer, Cham.
- Frédéric Chapoton and Muriel Livernet. 2001. *Pre-Lie algebras and the rooted trees operad*. *Internat. Math. Res. Notices*, (8):395–408.
- Alain Connes and Dirk Kreimer. 1998. *Hopf algebras, renormalization and noncommutative geometry*. *Comm. Math. Phys.*, 199(1):203–242.
- Samuele Giraud. 2019. *Colored operads, series on colored operads, and combinatorial generating systems*. *Discrete Math.*, 342(6):1624–1657.
- Michael A. Harrison. 1978. *Introduction to Formal Language Theory*. Addison-Wesley Publishing Co., Reading, MA.
- John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2001. *Introduction to automata theory, languages, and computation*, 2nd edition. *SIGACT News*, 32(1):60–65.
- Aravind K. Joshi. 1987. *An introduction to tree adjoining grammars*. In *Mathematics of language*, pages 87–114. Benjamins, Amsterdam.
- Aravind K. Joshi and Yves Schabes. 1997. *Tree-adjoining grammars*. In *Handbook of Formal Languages: Volume 3 Beyond Words*, pages 69–123. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Anthony S Kroch and Aravind K Joshi. 1985. *The linguistic relevance of tree adjoining grammar*.
- Matilde Marcolli, Noam Chomsky, and Robert C Berwick. 2025a. *Mathematical Structure of Syntactic Merge: An Algebraic Model for Generative Linguistics*. MIT Press.
- Matilde Marcolli, Riny Huijbregts, and Richard K. Larson. 2025b. *Hypermagnas and colored operads: Heads, phases, and theta roles*. *Preprint*, arXiv:2507.06393.
- Matilde Marcolli and Richard K. Larson. 2025. *Theta theory: operads and coloring*. *Preprint*, arXiv:2503.06091.
- Matilde Marcolli and Alexander Port. 2015. *Graph grammars, insertion Lie algebras, and quantum field theory*. *Math. Comput. Sci.*, 9(4):391–408.
- Paul-André Melliès and Noam Zeilberger. 2025. *The categorical contours of the Chomsky-Schützenberger representation theorem*. *Log. Methods Comput. Sci.*, 21(2):Paper No. 12.
- Claudio Procesi. 2007. *Lie groups*. Universitext. Springer, New York. An approach through invariants and representations.
- Owen Rambow, K. Vijay-Shanker, and David Weir. 2001. *D-tree substitution grammars*. *Computational Linguistics*, 27(1):87–121.
- Michael Sipser. 1996. *Introduction to the Theory of Computation*, 1st edition. International Thomson Publishing.
- K. Vijay-Shanker and A. K. Joshi. 1988. *Feature structures based tree adjoining grammars*. In *Proceedings of the 12th Conference on Computational Linguistics - Volume 2, COLING '88*, page 714–719, USA. Association for Computational Linguistics.

A Appendix: Proofs of Theorems

Theorem 2.8. \triangleleft is a pre-Lie operator.

Proof of Theorem 2.8. It is enough to show that the right Vinberg identity holds for basis elements. Let T_1, T_2, T_3 be trees in X . In the associator $A(T_1, T_2, T_3) = (T_1 \triangleleft T_2) \triangleleft T_3 - T_1 \triangleleft (T_2 \triangleleft T_3)$, the only basis elements with nonzero coefficient are the ones where T_2 and T_3 are adjoined at distinct vertices of T_1 , meaning that T_2 and T_3 are disjoint subgraphs of the resulting tree. These terms will appear in $(T_1 \triangleleft T_2) \triangleleft T_3$, but not $T_1 \triangleleft (T_2 \triangleleft T_3)$. Any term where T_3 is adjoined to T_2 , or a copy of T_2 in a term of $T_1 \triangleleft T_2$, necessarily appears in both terms of the associator, hence are cancelled out. This cancellation ensures that $A(T_1, T_2, T_3) = A(T_1, T_3, T_2)$. \square

Theorem 2.11. $\mathcal{T}_{bin,pl}$ is not isomorphic as a pre-Lie algebra to L_{free} , the free pre-Lie algebra on one generator.

Proof of Theorem 2.11. Suppose, for the sake of contradiction, that $\mathcal{T}_{bin,pl}$ is isomorphic to L_{free} ; then there exists an isomorphism $\phi : L_{free} \rightarrow \mathcal{T}_{bin,pl}$ such that $\phi(x \triangleleft y) = \phi(x) \triangleleft \phi(y)$ for all $x, y \in L_{free}$. (We use the same symbol \triangleleft to denote the pre-Lie operation on both algebras, since context prevents any ambiguity.)

The contradiction is defined by considering the inverse element $\phi^{-1}(\bullet) \in L_{free}$, and showing that it cannot be well-defined.

We will use the basis on L_{free} given by unlabelled nonplanar trees of arbitrary arity, where trees are graded by its number of nodes.

Since L_{free} is graded, $\mathcal{T}_{bin,pl}$ inherits a grading from the isomorphism, although it may not be an obvious one based on any combinatorial properties of trees. In fact, a basis element of $\mathcal{T}_{bin,pl}$ may not even be a homogeneous element under this grading.

Let T be a tree in $\mathcal{T}_{bin,pl}$; then $T \triangleleft \bullet = |T|T$, so

$$\phi^{-1}(T) \triangleleft \phi^{-1}(\bullet) = |T|\phi^{-1}(T).$$

Let $c \in \mathbb{R}$ be the coefficient of \emptyset in $\phi^{-1}(\bullet)$, so that

$$\phi^{-1}(\bullet) = c\emptyset + (\text{higher degree terms}).$$

Let y be a term in $\phi^{-1}(T)$ of minimal degree. Since \emptyset is the unique degree-0 generator in L_{free} , it is the only element that does not raise the degree of y when inserted into it. Since $|T|y$ occurs in the expression $\phi^{-1}(T) \triangleleft \phi^{-1}(\bullet)$, it can only be the result of performing $\phi^{-1}(T) \triangleleft c\emptyset$, hence $c = |T|$. But $\phi^{-1}(\bullet)$ cannot depend on any specific T . \square