

MATHEMATICAL STRUCTURE OF SYNTACTIC MERGE

MATILDE MARCOLLI, NOAM CHOMSKY, ROBERT C. BERWICK

ABSTRACT. The syntactic Merge operation of the Minimalist Program in linguistics can be described mathematically in terms of Hopf algebras, with a formalism similar to the one arising in the physics of renormalization. This mathematical formulation of Merge has good descriptive power, as phenomena empirically observed in linguistics can be justified from simple mathematical arguments. It also provides a possible mathematical model for externalization and for the role of syntactic parameters.

1. INTRODUCTION

Within the context of generative linguistics, the *Minimalist Model* was introduced in the '90s, [5], as a formalism that analyzes the generative process of syntax in terms of a basic fundamental operation, referred to as *Merge*, that generates, combines, and transforms syntactic trees. The formulation of Minimalism underwent some significant changes in recent years, after a simplifying reformulation, [6], [7], [8], [9], see also [3], [4], [29], where the Merge operation is described as a combinatorial binary set formation.

Our main goal here is to present a mathematical formulation of the Merge operator in syntax, based on Hopf algebras.

The reason why Hopf algebras are the suitable mathematical setting comes from the fact that grafting operations on trees such as Merge provide a natural strategy for generating a hierarchy of recursively defined structures. This idea has been widely developed within theoretical physics, where Hopf algebras of rooted trees and of Feynman graphs are used to analyze the combinatorics of perturbative expansions in quantum field theory and the formalism of renormalization, [15], [16], [30]. The Hopf algebra formalism of perturbative quantum field theory and renormalization has also found applications to the theory of computation, see for instance [18], [32], [33], [39].

In particular, an important case in physics, where recursive structures are built out of operators formally resembling the syntactic Merge, is the recursive construction of solutions to the quantum equations of motion (Dyson–Schwinger equations), [20], [51]. The Dyson–Schwinger equations can be seen as a way of implementing recursively at the quantum level the variational least action principle characterizing the classical equations of motion (see §9.6 of [43]).

The fact that the Hopf algebra formalism provides a natural setting for the formulation of recursive operations that build hierarchical structures based on trees strongly suggests that this should also be the natural setting for describing the properties of the Merge operators of syntax. We show in this paper that this is indeed the case, namely that the same mathematical formalism that governs the recursive structures of quantum field theory also governs Merge in the Minimalist Model.

The mathematical formulation of Merge that we obtain here is not just a convenient, mathematically elegant, rephrasing of [7], [8], but it has good descriptive and predictive power. We demonstrate that by showing that some empirically observed linguistic phenomena acquire a simple and direct mathematical explanation in this model. For example, we show that some of the

properties usually required of Merge, such as not decreasing the size of workspaces, or cancellation of deeper copies of accessible terms, follow directly from the mathematical formalism (in particular from the structure of the coproduct of the Hopf algebra). We also show that a hypothetical Merge operator that instead of binary would be n -ary for any $n \geq 3$ would necessarily suffer from both undergeneration and overgeneration with respect to the binary Merge. This property again follows immediately from counting arguments in the Hopf algebra and confirms and explains in clear computational terms some empirically observed linguistic phenomena.

We also show that, within this mathematical setting, one can formulate a possible model for the externalization process, that interfaces the core computational mechanism of Merge with the syntactic constraints of specific languages. We describe the model of externalization in the form of a correspondence, rather than a function, where one side of the correspondence implements the linear order of sentences, which is not present at the level of the deeper structure of Merge, while the second step of the correspondence implements other constraints (in addition to word order) on the syntactic trees that come from syntactic parameters.

We formulate some possible questions for future study, involving possible approaches to models of syntactic-semantic interface, and a characterization of Merge as a solution to an optimization problem.

1.1. Summary of Merge. We summarize briefly the structure of the Merge operation of syntax according to the more recent formulation of the Minimalist Model of syntax, following [7], [8].

One considers a given set of *lexical items* and *syntactic features* like $N, V, A, P, C, T, D, \dots$. One also starts with an independently constructed set of *syntactic objects* (SO) obtained recursively, by adding to the above set all syntactic objects created by application of a basic Merge operation. This is defined in terms of the *binary set formation* that assigns to two arguments α and β the *unordered set*

$$(1.1) \quad \mathfrak{M}(\alpha, \beta) := \{\alpha, \beta\},$$

so as to include, for example, sets like $\{N, V\}$, etc. We denote by \mathcal{SO} the set obtained in this way, starting from lexical items and syntactic features. We write \mathcal{SO}_0 for the initial set of lexical items and syntactic features.

The *accessible terms* of a syntactic object SO are proper nonempty subsets of SO . (A definition in terms of binary rooted trees is given below, in Definition 2.4.) We write $Acc(SO)$ for the set of accessible terms of a syntactic object SO . A *Work Space* WS is a finite (multi)set of syntactic objects in \mathcal{SO} . The size of the workspace WS is the sum of the number of syntactic objects and the number of accessible terms, where the set $Acc(WS)$ of accessible terms of the workspace is

$$Acc(WS) := \bigcup_{SO \in WS} Acc(SO).$$

Merge acting on workspaces consists of a collection of operations $\mathfrak{M} = \{\mathfrak{M}_A\}$, parameterized by sets A consisting of two syntactic objects α, β . These operations have as input a workspace and produce as output a new workspace, by searching for accessible terms in the given workspace matching the selected objects α, β , producing a new object in the workspace obtained by applying binary set formation, and cancelling the remaining deeper copies of the accessible terms used.

The Merge action on workspaces can be given an axiomatic formulation by imposing a list of desired properties. Some of the fundamental required properties of Merge are:

- (1) it is a binary operation (it applies to only two arguments in WS);
- (2) any generated syntactic object remains accessible for further applications of Merge;

- (3) every accessible term only appears once in the workspace;¹
- (4) the result of Merge applied to two arguments α, β does not add any new syntactic properties to α and β nor it removes any of their existing properties (structure preserving principle);²
- (5) workspace size does not decrease and increases at most by one.

The last condition on the size of workspaces can be broken down into two separate conditions on the number of syntactic objects in the workspace and on the number of accessible terms. The number of syntactic objects is expected to be non-increasing, and overall decreasing in the course of the derivation, for derivations to terminate and “converging” thought to be generated, while at the same time the overall number of accessible terms will be non-decreasing, and overall increasing in the course of a derivation. (The term derivation is meant here in the same sense as in logic and theory of computation). The condition stated as above in terms of size of the workspace is consistent with, for example, [22]. Separating conditions on number of syntactic objects and of accessible terms, we can formulate the last condition in the different form

- (5') the number of syntactic objects in the workspace is non-increasing, and the number of accessible terms is non-decreasing.

We will give a more precise definition of syntactic objects, accessible terms, and workspace size in §2.1 and §2.2, and we will discuss more in detail conditions (5) and (5') in §2.5.1.

For the underlying linguistic justification for the desirable properties of the action of Merge on workspaces we refer the reader to [7], [8], [11], and the forthcoming [14]. For our purposes here, we take this list as an assigned guideline, a kind of “axiomatic template” on how to construct a mathematical model for the action of Merge on workspaces. Some caveats and some more specific interpretation of the items listed above will be discussed in the following sections.

A goal of this paper is to show that such a list of fundamental properties leads naturally to a mathematical formulation of Merge in terms of magmas and Hopf algebras, and that this formulation turns out to be in fact the same very basic mathematical structure that arises naturally in the description of fundamental interaction in physics.

2. A MATHEMATICAL MODEL OF SYNTACTIC MERGE

We consider here the formulation of Minimalism as presented in [7], [8], with a fundamental Merge operation of binary set formation.

2.1. Syntactic objects and the Merge magma. As in [7], [8], one considers, as the starting point in the construction of the set of *syntactic objects* \mathcal{SO} , an initial set, which we denote by \mathcal{SO}_0 consisting of lexical items and syntactic features.

Definition 2.1. *The set \mathcal{SO} of syntactic objects is the free, non-associative, commutative magma over the set \mathcal{SO}_0 ,*

$$(2.1) \quad \mathcal{SO} = \text{Magma}_{na,c}(\mathcal{SO}_0, \mathfrak{M}),$$

with the binary Merge operation

$$(2.2) \quad \mathfrak{M}(\alpha, \beta) = \{\alpha, \beta\}.$$

¹It is important to distinguish here between *copies* and *repetitions*: the workspace is a multiset of syntactic objects, hence repetitions are allowed, while this property refers to copies. This will be made more precise in §2.2 and §2.5.2 below.

²One should interpret this principle in the sense that, for example, we cannot add features, or transform α, β into new α', β' . However, syntactic properties do change through Merge, as the following simple example suggested to us by Sandiway Fong shows: in $YP = \{R, XP\}$ we have that YP is a theta-configuration, and XP has now acquired the syntactic property that it is the theta-marked object of R .

This means that, as described in [7], [8], the set \mathcal{SO} is obtained from the initial set \mathcal{SO}_0 through iterations of the Merge operation (2.2). This procedure generates elements of \mathcal{SO} of the form $\{\alpha, \beta\}$, $\{\alpha, \{\beta, \gamma\}\}$, for $\alpha, \beta, \gamma \in \mathcal{SO}_0$, and so on. The Merge operation (2.2) acts on the set \mathcal{SO} , giving it the structure of non-associative, commutative magma³.

Remark 2.2. The description of the set \mathcal{SO} of syntactic objects given in Definition 2.1 above gives an identification

$$(2.3) \quad \mathcal{SO} \simeq \mathfrak{T}_{\mathcal{SO}_0}$$

of the set of syntactic objects with the set of binary, non-planar, rooted trees, with leaves labelled by elements of \mathcal{SO}_0 .

By non-planar we mean that we regard trees $T \in \mathfrak{T}_{\mathcal{SO}_0}$ as abstract trees, without fixing a choice of a planar embedding. This implies that there is no choice of a linear ordering on the leaves of the trees. As in [7], [8], the word order structure, that is, the linearly ordered form of sentences, is considered a part of the externalization process, not of the core computational mechanism of syntax given by Merge.

2.1.1. *Planarity and lists versus sets.* In the formulation above, in Definition 2.1 and Remark 2.2 we identify syntactic objects with *non-planar* binary rooted finite trees with leaves labelled by the set \mathcal{SO}_0 , where non-planar means that no choice of a planar embedding is taken for the tree. These are often also referred to as “abstract trees”. This is the usual mathematical description of the elements of the free, non-associative, commutative magma on a given set.

While planar trees (trees together with a choice of a planar embedding) and abstract trees appear to be similar mathematical objects, their combinatorial properties are very different, and this accounts for several significant differences, in linguistics, between older forms of Minimalism and the newer form we discuss in this paper. This is discussed more explicitly in our companion paper [38].

In the linguistics literature, the passage from planar trees in the older versions of Minimalism to abstract trees, is usually discussed using the terminology *sets* to refer to the abstract trees as elements of the free, non-associative, commutative magma. The reason for the use of this terminology is that in dropping the planar structure one replaces an identification of the set of leaves with parenthesized *lists* (*ordered sets*, often referred to in the linguistics literature as “strings”) with just sets (in fact more precisely *multisets*). In order to avoid the conflict of terminology between sets and multisets, we prefer here to follow the standard mathematical terminology and refer to the syntactic objects as abstract binary rooted trees (with no assigned planar embedding).

It is important to note that, because all the trees are binary, the clash of terminology between sets and multisets is very mild when one considers syntactic objects. Indeed, since trees are binary rather than n -ary with some $n \geq 3$, the only repetitions of labels that give rise to multisets can be on two consecutive ones, so there is an unambiguous way of labeling the same objects by sets. For example, a multiset of the form $\{\{a, a\}, b, \{c, d\}\}$ can be written equivalently as the set $\{\{a\}, b, \{c, d\}\}$ with the convention that a set of the form $\{a\}$ stands for the abstract tree \widehat{a}_a .

However, even with binary trees, the clash of terminology between sets and multisets becomes seriously problematic when it comes to describing *workspaces*, as we will see in §2.2 below. These are genuinely multisets that do not have an equivalent description as sets, hence the mathematically correct notion to use is *binary forests* (disjoint unions of a finite collection of abstract binary

³The presence of magma structures in generative linguistics was also recently observed independently in [17], in a somewhat different context.

rooted trees) rather than sets. Indeed, forests are multisets where the same tree (the same syntactic object) may appear more than once. This is expected as the same syntactic object may be used repeatedly, in different ways, in the course of a derivation. For this reason, we will not be using the “sets” terminology that is more common in the linguistics literature and we prefer to adopt the mathematical notation of trees (with no planar structure) and forests.

2.2. Workspaces: product and coproduct. We next introduce workspaces, as in [7], [8] and the action of Merge on workspaces. We first introduce workspaces with a bialgebra structure related to the combination of workspaces and the extraction of accessible terms with cancellation of copies.

Definition 2.3. *Workspaces are nonempty finite sets of syntactic objects. The identification (2.3) between syntactic objects and binary, non-planar, rooted trees, with leaves labelled by elements of \mathcal{SO}_0 , induces an identification*

$$(2.4) \quad \mathcal{WS} \simeq \mathfrak{F}_{\mathcal{SO}_0}$$

between the set \mathcal{WS} of all workspaces and the set $\mathfrak{F}_{\mathcal{SO}_0}$ of binary non-planar forests (disjoint unions of binary, non-planar, rooted trees) with leaf labels \mathcal{SO}_0 .

Note that, with this definition of workspaces, we allow for the presence of repeated copies of the same syntactic object in a workspace, since a forest can have multiple connected components that are isomorphic to the same tree. This is needed for the operations of combination of workspaces and extraction of accessible terms described below to be well defined, as the result of these operation can produce repeated copies of the same tree, even when starting with a forest that has none.

Definition 2.4. *Given a binary non-planar rooted tree $T \in \mathfrak{T}_{\mathcal{SO}_0}$, let $V_{int}(T)$ denote the set of all internal (non-root) vertices of T . For $v \in V_{int}(T)$, let $T_v \subset T$ denote the subtree consisting of v and all its descendants. Let $L_v = L(T_v)$ be the set of leaves of T_v . The set of accessible terms of T is given by*

$$(2.5) \quad \text{Acc}(T) = \{L_v = L(T_v) \mid v \in V_{int}(T)\}.$$

For a workspace given by a forest $F = \sqcup_a T_a \in \mathfrak{F}_{\mathcal{SO}_0}$, the set of accessible terms is

$$(2.6) \quad \text{Acc}(F) = \bigcup_a \text{Acc}(T_a),$$

so that we have the total number of vertices of the forest given by the sum

$$(2.7) \quad \#V(F) = b_0(F) + \#\text{Acc}(F),$$

where $b_0(F)$ is the number of connected components (trees) of the forest F . We define the size of a workspace F by

$$(2.8) \quad \sigma(F) := \#V(F) = b_0(F) + \#\text{Acc}(F),$$

namely the number of syntactic objects plus the total number of accessible terms. We also define another counting function, which is given by

$$(2.9) \quad \hat{\sigma}(F) := b_0(F) + \#V(F).$$

The size $\sigma(F)$ of the workspace, defined as in (2.8) is consistent with [7], [8] and agrees with the definition of size used in [22]. As pointed out to us by Riny Huijbregts, it may be preferable to consider the effect of Merge on workspaces in terms of the counting of accessible terms $\#\text{Acc}(F) = \#V_{int}(F)$, rather than in terms of the size $\sigma(F)$. We will discuss and compare the effect on various size-counting in §2.5.1 below.

The set of workspaces is endowed with two operations. A product operation that combines workspaces by taking their union is simply given by the disjoint union on the set of forests. It is a commutative and associative product, with unit given by the empty forest. The second operation is a coproduct, which provides all the possible extractions of admissible terms. In order to be able to consider all accessible terms simultaneously, one considers, instead of the set $\mathfrak{F}_{\mathcal{SO}_0}$, as above, a space of formal linear combinations of elements in this set. Namely, we denote by $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0})$ the \mathbb{Z} -module freely generated by elements of $\mathfrak{F}_{\mathcal{SO}_0}$ (formal linear combinations of binary non-planar forests with integer coefficients), so that one can sum over all the possible extractions of accessible terms (see (2.10) below).

In the following, for a given rooted binary tree (with no assigned planar embedding) $T \in \mathfrak{T}_{\mathcal{SO}_0}$, we write T_v for the subtree consisting of v and all its descendants, as in Definition 2.4. In addition to considering these sub-objects $T_v \subset T$, we also consider corresponding quotient objects T/T_v . Linguistically, the T_v are the accessible terms, as described above, while the T/T_v implement the cancellation of the deeper copy of T_v from the resulting workspace, after application of Merge. Mathematically, as we discuss below, the pairs T_v and T/T_v correspond to the two terms of a coproduct applied to T . The quotient object T/T_v is no longer a sub-object of T .

The usual way of defining the quotient T/T_v , in the context of Hopf algebras of rooted trees in mathematics and theoretical physics, is to contract the entire tree T_v to a single vertex, so that the root vertex of T_v becomes a leaf of T/T_v . With this definition of the quotient, in particular, one has $T/T = \bullet$, the tree consisting of a single root vertex. However, this choice of how to define T/T_v is not the best one in our setting. One reason is that, with this definition, the new leaf of T/T_v (the root vertex of T_v) needs to be labeled by an element of \mathcal{SO}_0 , hence some calculus of labels of internal vertices is required. While such a projection mechanism for labeling internal vertices is required in other versions of Minimalism, in the version we are considering we can dispense with that, and labeling of internal vertices should not be required. The other reason, as we will discuss more in detail below, is that for mathematical consistency and for a simple unifying view of Internal and External Merge (see Proposition 2.12 below), it is necessary that the quotient $T/T = 1$ is also the unit 1 of the magma \mathcal{SO} , and such a unit 1 is provided not by the single vertex tree but by formally adding an empty tree. Thus, the consistent way of defining the quotient object T/T_v is provided by the following definition.

Definition 2.5. *Given a rooted binary tree (with no assigned planar embedding) $T \in \mathfrak{T}_{\mathcal{SO}_0}$ and a subtree $T_v \subset T$ consisting of a vertex v of T and all its descendants, the quotient T/T_v is the rooted binary tree obtained by removing the entire tree T_v from T . There is then a unique maximal rooted binary tree that can be obtained from the complement $T \setminus T_v$ via contraction of edges. That resulting rooted binary tree is what we call T/T_v .*

Lemma 2.6. *Given a rooted binary tree (with no assigned planar embedding) $T \in \mathfrak{T}_{\mathcal{SO}_0}$, a subforest $F_{\underline{v}} \subset T$ is a union of subtrees $T_{v_1} \cup \dots \cup T_{v_k}$, for $\underline{v} = (v_1, \dots, v_k)$, with the property that $T_{v_i} \cap T_{v_j} = \emptyset$ for all $i \neq j$. The quotient $T/F_{\underline{v}}$ given by*

$$T/F_{\underline{v}} = (\dots (T/T_{v_1})/T_{v_2} \dots)/T_{v_k},$$

with each quotient of trees as in Definition 2.5, is well defined and independent of the order of v_1, \dots, v_k . This extends to quotients of the form $F/F_{\underline{v}}$ where $F = \sqcup_a T_a$ is a forest with each component T_a a rooted binary tree (with no assigned planar embedding) and $F_{\underline{v}}$ is a subforest of F with $F_{\underline{v}} \cap T_a$ a subforest in the sense above. In particular, these quotients of forests satisfy $F_{\underline{v}, \underline{w}}/F_{\underline{v}} = F_{\underline{w}}$.

Proof. If no pair T_{v_i}, T_{v_j} has a common vertex v_{ij} of T adjacent to both roots, then it is clear that $(T/T_{v_i})/T_{v_j} = (T/T_{v_j})/T_{v_i}$. If a pair has such common vertex then we can still see that this works

since in this case we have

$$\begin{aligned}
 T &= \begin{array}{c} \diagup \quad \diagdown \\ \text{T}_i \quad \text{T}_j \quad \text{T}_a \quad \text{T}_b \end{array} & T/T_i &= \begin{array}{c} \diagup \quad \diagdown \\ \text{T}_j \quad \text{T}_a \quad \text{T}_b \end{array} & T/T_j &= \begin{array}{c} \diagup \quad \diagdown \\ \text{T}_i \quad \text{T}_a \quad \text{T}_b \end{array} \\
 & & (T/T_i)/T_j &= (T/T_j)/T_i &= \begin{array}{c} \diagup \quad \diagdown \\ \text{T}_a \quad \text{T}_b \end{array}.
 \end{aligned}$$

The extension from trees to forests follows similarly by components. For $F_{\underline{v}, \underline{w}}$ a subforest of T (or of a forest F) on the set of vertices $(\underline{v}, \underline{w})$ with the disjointness property above, we have that $F_{\underline{v}}$ is a subforest given by a subcollection of components, each of which is removed in the quotient operation, so that one is left with $F_{\underline{w}}$. \square

With these definitions of the trees T_v and T/T_v , and forests $F_{\underline{v}}$ and $T/F_{\underline{v}}$, we then have the following structure.

Lemma 2.7. *The operation $\Delta : \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0}) \rightarrow \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0}) \otimes \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})$ defined on trees $T \in \mathfrak{T}_{S\mathcal{O}_0}$*

$$(2.10) \quad \Delta(T) = \sum_{\underline{v}} F_{\underline{v}} \otimes (T/F_{\underline{v}}),$$

with the sum over subforests with quotients as in as in Lemma 2.6, and extended to forests by $\Delta(F) = \sqcup_a \Delta(T_a)$ for $F = \sqcup_a T_a$, which we write as

$$(2.11) \quad \Delta(F) = \sum_{\underline{v}} F_{\underline{v}} \otimes F/F_{\underline{v}},$$

with subforests and quotients as in Lemma 2.6. This defines a coproduct on $\mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})$, which endows $\mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})$ with the structure of an associative, commutative, coassociative, non-cocommutative bialgebra $(\mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0}), \sqcup, \Delta)$.

Proof. The multiplication given by disjoint union is both associative and commutative, in the case of non-planar forests. We check that the coassociativity

$$(2.12) \quad (\text{id} \otimes \Delta) \circ \Delta = (\Delta \otimes \text{id}) \circ \Delta$$

of the coproduct (2.10) is verified. We have

$$(\Delta \otimes \text{id}) \circ \Delta(T) = (\Delta \otimes \text{id}) \sum_{\underline{w}} F_{\underline{w}} \otimes T/F_{\underline{w}} = \sum_{\underline{v}, \underline{w}} F_{\underline{v}} \otimes (F_{\underline{w}}/F_{\underline{v}}) \otimes T/F_{\underline{w}},$$

where the sums are over subforests with the disjointness condition as in Lemma 2.6, where the subforests $F_{\underline{v}} \subset F_{\underline{w}}$ consists of either full components of $F_{\underline{w}}$ or of subforests of the components. The first case gives terms of the form $F_{\underline{v}} \otimes F_{\underline{u}} \otimes T/F_{\underline{v}, \underline{u}}$ for $\underline{w} = (\underline{v}, \underline{u})$. On the other hand, we have

$$(\text{id} \otimes \Delta) \circ \Delta(T) = (\text{id} \otimes \Delta) \sum_{\underline{v}} F_{\underline{v}} \otimes T/F_{\underline{v}} = \sum_{\underline{u}, \underline{v}} F_{\underline{v}} \otimes (T/F_{\underline{v}})_{\underline{u}} \otimes (T/F_{\underline{v}})/(T/F_{\underline{v}})_{\underline{u}}.$$

We distinguish among these terms the case where the subtrees of T with root at u_i are disjoint from the trees of $F_{\underline{v}}$, where we have $(T/F_{\underline{v}})/(T/F_{\underline{v}})_{\underline{u}} = T/F_{\underline{v}, \underline{u}}$, and the remaining cases where some vertices u_i in \underline{u} , as vertices of T , are above some vertices v_j of the components of $F_{\underline{v}}$, in which case the corresponding quotient is $(T/T_{v_j})/T_{u_i} = T/T_{u_i}$ and $(T/T_{v_j})_{u_i} = T_{u_i}/T_{v_j}$. Thus, we see that we obtain the same two types of terms with the same counting.

For a vector space \mathcal{V} let $\tau : \mathcal{V}^{\otimes 4} \rightarrow \mathcal{V}^{\otimes 4}$ denote the permutation of the two central factors,

$$(2.13) \quad \tau(X_1 \otimes X_2 \otimes X_3 \otimes X_4) = X_1 \otimes X_3 \otimes X_2 \otimes X_4.$$

Multiplication and comultiplication satisfy the compatibility,

$$(2.14) \quad \Delta \circ \sqcup = (\sqcup \otimes \sqcup) \circ \tau \circ (\Delta \otimes \Delta),$$

since $\Delta(T \sqcup T') = \Delta(T) \sqcup \Delta(T') = \sum_{\underline{v}, \underline{v}'} F_{\underline{v}} \sqcup F'_{\underline{v}'} \otimes (T/F_{\underline{v}}) \sqcup (T'/F'_{\underline{v}'}) = (\sqcup \otimes \sqcup) \circ \tau \sum_{\underline{v}} F_{\underline{v}} \otimes (T/F_{\underline{v}}) \otimes \sum_{\underline{v}'} F'_{\underline{v}'}) \otimes (T'/F'_{\underline{v}'})$. It is convenient to include in the spanning set of $\mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})$ the unit 1 of the product given by the trivial (empty) forest, which also spans the range of the counit of the coproduct. Moreover, the vector space is graded by the number of leaves (the length of sentences), $\mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0}) = \bigoplus_{n \geq 0} \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})_n$, and the product and coproduct are compatible with the grading, in the sense that $\sqcup : \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})_n \otimes \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})_m \rightarrow \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})_{n+m}$ and $\Delta(T) = T \otimes 1 + 1 \otimes T + \sum T' \otimes T''$ where T', T'' are of strictly lower degree than T , and similarly for forests. Thus, the antipode of the Hopf algebra structure can be defined inductively by $S(T) = -T - \sum S(T')T''$. \square

We can write the coproduct Δ of (2.10) in the form

$$(2.15) \quad \Delta(T) = \sum_{n \geq 2} \Delta_{(n)}(T),$$

where the terms $\Delta_{(n)}$ involve extraction and quotient of subforest of size (number of components) $n - 1$, namely with terms of the form $F_{\underline{v}} \otimes T/F_{\underline{v}}$ with $v = (v_1, \dots, v_{n-1})$. For any given T the expression (2.15) is a finite sum. In particular the first term

$$(2.16) \quad \Delta_{(2)}(T) = \sum_v T_v \otimes T/T_v$$

corresponds to the extraction of subtrees (including the case of the trivial tree 1 and of the full tree T). Note that (2.16) does not suffice for coassociativity, for which the full coproduct (2.10) is needed, which is a form of the usual coproduct by admissible cuts on Hopf algebras of rooted trees. The leading term (2.16) will be the relevant one for the Merge operation.

2.3. Action of Merge on Workspaces. We introduce the action of Merge on workspaces by introducing an operator that performs a search for matching terms. This will be applied to the terms of the coproduct, that is, to the accessible terms that Merge applies to. Indeed, the left-hand-side of the coproduct produces the list of accessible terms, over which the search runs, while the right-hand-side of the coproduct keeps track of the corresponding cancellation of copies. We will introduce the action of Merge with some preliminary steps.

Suppose then given two syntactic objects, that is, two $S, S' \in \mathfrak{F}_{S\mathcal{O}_0}$. We define a linear operator

$$\delta_{S, S'} : \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0}) \otimes \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0}) \rightarrow \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0}) \otimes \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})$$

by defining it on generators in the following way. Let

$$(2.17) \quad \mathfrak{F}_{S\mathcal{O}_0}^\Delta = \{(F_1, F_2) \in \mathfrak{F}_{S\mathcal{O}_0} \times \mathfrak{F}_{S\mathcal{O}_0} \mid \exists F \in \mathfrak{F}_{S\mathcal{O}_0}, F_{\underline{v}} \subset F : F_1 = F_{\underline{v}} \text{ and } F_2 = F/F_{\underline{v}}\}.$$

For $F_1, F_2 \in \mathfrak{F}_{S\mathcal{O}_0}$, we set

$$(2.18) \quad \delta_{S, S'}(F_1 \otimes F_2) = 0 \quad \text{for } (F_1, F_2) \notin \mathfrak{F}_{S\mathcal{O}_0}^\Delta.$$

For $(F_1 = F_{\underline{v}}, F_2 = F/F_{\underline{v}}) \in \mathfrak{F}_{S\mathcal{O}_0}^\Delta$ with $F = \sqcup_{i \in \mathcal{I}} T_i$, we set

$$(2.19) \quad \delta_{S, S'}(F_{\underline{v}} \otimes F/F_{\underline{v}}) = S \sqcup S' \otimes T_a/S \sqcup T_b/S' \sqcup F^{(a,b)}$$

with $F^{(a,b)} = \sqcup_{i \neq a, b} T_i$, if there are indices $a, b \in \mathcal{I}$ such that $T_{a, v_a} \simeq S$, $T_{b, v_b} \simeq S'$. If there is more than one choice of indices a, b for which matching pairs $T_{a, v_a} \simeq S$, $T_{b, v_b} \simeq S'$ exist, then the right-hand-side of (2.19) should be replaced by the sum over all the possibilities. We do not write that out explicitly for simplicity of notation. In all other cases (where no matching terms for S and S' are found) we set

$$(2.20) \quad \delta_{S, S'}(F_{\underline{v}} \otimes F/F_{\underline{v}}) = 1 \otimes F.$$

Next observe that the operation (2.2) on syntactic objects factors through the grafting operator B^+ on forests.

Definition 2.8. Let $\mathfrak{T}_{\mathcal{SO}_0}^{\mathbb{N}}$ denote the set of all n -ary finite rooted trees with arbitrary $n \in \mathbb{N}$, with no assigned planar structure and with labels labelled by the set \mathcal{SO}_0 . Let $\mathfrak{F}_{\mathcal{SO}_0}^{\mathbb{N}}$ be the set of finite forests with connected components in $\mathfrak{T}_{\mathcal{SO}_0}^{\mathbb{N}}$. Let $\mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{\mathbb{N}})$ and $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{\mathbb{N}})$ denote the \mathbb{Q} -vector spaces spanned by these sets. The grafting operator $B^+ : \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{\mathbb{N}}) \rightarrow \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{\mathbb{N}})$ is the linear operator defined on generators by

$$(2.21) \quad B^+(T_1 \sqcup T_2 \sqcup \cdots \sqcup T_N) = \begin{array}{c} \diagup \quad \diagdown \\ T_1 \quad T_2 \quad \cdots \quad T_N \end{array} .$$

The grafting operator B^+ is well known in the mathematical formulation of perturbative quantum field theory, as it is the operator that defines the recursive structure of Dyson–Schwinger equations, see [1], [20].

Lemma 2.9. The Merge operator \mathfrak{M} of (2.2), namely the multiplication operation in the magma $\text{Magma}_{na,c}(\mathcal{SO}_0, \mathfrak{M})$, determines a bilinear operator $\mathfrak{M} : \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}) \otimes \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}) \rightarrow \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0})$ defined on generators as

$$(2.22) \quad \mathfrak{M} : T \otimes T' \mapsto \mathfrak{M}(T, T') = \widehat{T \ T'} ,$$

where we set $\mathfrak{M}(1, 1) = 1$ and $\mathfrak{M}(T, 1) = \mathfrak{M}(1, T) = T$. This operator factors through the grafting operator B^+ restricted to the range of multiplication \sqcup , namely the diagram

$$\begin{array}{ccc} \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}) \otimes \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}) & \xrightarrow{\mathfrak{M}} & \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}) \\ & \searrow \sqcup & \nearrow B^+ \\ & & \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}) \end{array}$$

Proof. Since trees and forests do not have an assigned planar structure, both \mathfrak{M} and the operator B^+ do not depend on the order of the trees. Moreover, since the image of $cV(\mathfrak{T}_{\mathcal{SO}_0}) \otimes \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0})$ under \sqcup consists of forests with two connected components, so that their image under B^+ is still a binary tree, which is of the form (2.22). \square

The operation (2.19) in combination with the operation (2.2) on syntactic objects and the bialgebra structure on workspaces contribute to the definition of the action of Merge on workspaces as described in [7], [8], which we can define in the following way.

Definition 2.10. The action of Merge on workspaces consists of a collection of operators

$$\{\mathfrak{M}_{S,S'}\}_{S,S' \in \mathfrak{T}_{\mathcal{SO}_0}}, \quad \mathfrak{M}_{S,S'} : \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}) \rightarrow \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}),$$

parameterized by pairs S, S' of syntactic objects, which act on $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0})$ by

$$(2.23) \quad \mathfrak{M}_{S,S'} = \sqcup \circ (B^+ \otimes \text{id}) \circ \delta_{S,S'} \circ \Delta ,$$

with B^+ the grafting operator of Definition 2.8.

Note that by the definition of $\delta_{S,S'}$ the operator $B^+ \otimes \text{id}$ applied to elements of the form $\delta_{S,S'}(F_{\underline{v}} \otimes F/F_{\underline{v}})$, for $F \in \mathfrak{F}_{\mathcal{SO}_0}$, produces elements $X \otimes Y$ with X in $\mathfrak{T}_{\mathcal{SO}_0}$ and Y in $\mathfrak{F}_{\mathcal{SO}_0}$, hence $\mathfrak{M}_{S,S'}$ maps $\mathfrak{F}_{\mathcal{SO}_0}$ to itself.

The expression (2.23) agrees with the description of the action of Merge on workspaces in [7], [8], namely the Merge operator $\mathfrak{M}_{S,S'}$ searches for copies of the syntactic terms S and S' in the accessible terms of a given workspace F , extracts those accessible terms to perform the Merge operation on, and cancels copies from the workspace, producing the new resulting workspace.

In (2.23) the first operation, the coproduct Δ , produces the list of all the accessible terms T_v that can be used by Merge and of the corresponding remaining terms T/T_v where the cancellation of copies of accessible terms is performed. Note that these terms correspond to just the part $\Delta_{(2)}$ of the coproduct as in (2.16), since it is in this part that the nontrivial terms selected by the operator $\delta_{S,S'}$ reside, which searches for matching terms among the accessible terms. If no matching terms are found, the action is the identity and the resulting workspace is not changed. If matching terms are found, they are merged using \mathfrak{M} (or equivalently B^+). The final application of the product \sqcup produces the new resulting workspace. The trees in the initial workspace F that do not contain a pair of accessible objects matching the pair (S, S') remain unchanged in the workspace, in agreement with the formulation of [7], [8], while the trees that contain matching accessible terms are replaced by a new syntactic object given by merging the matching terms and by cancellation of the deeper copies, in the form

$$(2.24) \quad \sum_{v,w:T_v=S,T_w=S'} \mathfrak{M}(T_v, T_w) \sqcup (T/T_v) \sqcup (T/T_w).$$

We describe more in detail in §2.4 the various cases.

2.4. Forms of Merge and Minimal Search. One of the drawbacks of the formulation (2.23) of Definition 2.10 is that it allows for additional forms of Merge, besides internal and external Merge, which are not desirable in linguistic terms, such as “sideward Merge” or “countercyclic Merge”. We discuss here how a simple modification of Definition 2.10 that incorporates a formulation of “Minimal Search” suffices to eliminate these cases and retain only the linguistically desirable cases of External and Internal Merge. This is the usual argument given in linguistics, where only External and Internal Merge are retained based on Minimal Search, except that here we reformulate it in a way that fits our algebraic setting. First we review how the different cases of Merge are incorporated in (2.23), then we describe how Minimal Search is implementable in our Hopf algebra setting, and then we show that this has the effect of only retaining the correct forms of external and internal Merge.

2.4.1. Different forms of Merge. In the description of Merge in Definition 2.10 we can distinguish several cases. We recall here the various cases, and we show how they occur in the formulation given above.

Two syntactic objects $\alpha, \beta \in \mathcal{SO} = \mathfrak{T}_{\mathcal{SO}_0}$ can occur in a workspace $F \in \mathfrak{F}_{\mathcal{SO}_0}$ either as elements (that is, as connected components of the forest F), or as accessible terms of some elements. We write $T \in F$ to indicate that a certain syntactic object $T \in \mathcal{SO}$, seen as a tree, is a connected component of the forest F . We write $T \in \text{Acc}(T')$ to indicate that T occurs as an accessible term T'_v of a syntactic object $T' \in F$.

Thus, we have the following three possibility

- (1) $\alpha = T_i$ and $\beta = T_j$ with $T_i, T_j \in F$ and $i \neq j$;
- (2) $\alpha = T_i \in F$ and $\beta \in \text{Acc}(T_j)$ for some $T_j \in F$, with two sub-cases:
 - a) $i = j$
 - b) $i \neq j$
- (3) $\alpha \in \text{Acc}(T_i)$ and $\beta \in \text{Acc}(T_j)$ for some $T_i, T_j \in F$, with two sub-cases:
 - a) $i = j$
 - b) $i \neq j$

Case (1) describes External Merge: for a workspace $F = \sqcup_a T_a$, the Merge operation \mathfrak{M}_{T_i, T_j} replaces the pair T_i, T_j of elements of F with a new syntactic object given by the tree $T_{ij} =$

$\{T_i, T_j\} = \mathfrak{M}(T_i, T_j)$, and produces the new workspace

$$F' = T_{ij} \sqcup \bigsqcup_{a \neq i, j} T_a,$$

where the two components T_i, T_j of F have been removed and replaced by the new tree $T_{ij} = \{T_i, T_j\}$. External Merge decreases by one the number of syntactic objects, and increases by two the number of accessible terms, by adding T_i and T_j to the set $\text{Acc}(T_i) \cup \text{Acc}(T_j)$. In the case of External Merge the following is immediately evident.

Lemma 2.11. *External Merge is achieved by the operators \mathfrak{M}_{T_i, T_j} of Definition 2.10 when the syntactic objects (trees) T_i, T_j match two different connected components of the workspace $F = \sqcup_a T_a$.*

Case (2a) describes Internal Merge: in this case the new workplace F' contains a new component of the form $\mathfrak{M}(\beta, T_i/\beta)$ for $\beta \in \text{Acc}(T_i)$ an accessible term of T_i and T_i a component of the given workspace F . The quotient T_i/β to indicate that the deeper copy of β as an accessible term of T_i is no longer an accessible term of $\mathfrak{M}(\beta, T_i/\beta)$ in F' as β already occurs as accessible term in a higher level in the new syntactic object $\mathfrak{M}(\beta, T_i/\beta)$ formed by Merge. In this case, the realization of Internal Merge by the operators $\mathfrak{M}_{S, S'}$ of Definition 2.10 is more interesting, as it involves a composition of two such operators and the role of the multiplicative unit 1 of the Merge magma, given by the trivial tree.

Proposition 2.12. *Internal Merge is realized by the operators of Definition 2.10 as a composition*

$$\mathfrak{M}_{T/\beta, \beta} \circ \mathfrak{M}_{\beta, 1},$$

where 1 is the unit of the Merge magma, where the tree β is an accessible term of a connected component of F isomorphic to T .

Proof. The operator $\mathfrak{M}_{\beta, 1}$ acting on the workspace F will act on a term of the form $\beta \otimes T/\beta$ in $\Delta(F)$, producing two new components $\beta = \mathfrak{M}(1, \beta)$ and T/β in the resulting new workspace. The operator $\mathfrak{M}_{T/\beta, \beta}$ can then be applied to these two components, as an external Merge, and the resulting workspace will now contain the resulting term $\mathfrak{M}(\beta, T/\beta)$ which is the internal Merge. \square

Remark 2.13. Note that in this formulation internal Merge is just repeated application of two external Merge operations, one of them involving the magma unit, which has the effect of extracting an accessible term and adding it, together with the cancellation of its deeper copy, to the new workspace. However, we will see that by Minimal Search, as well as by counting of size and number of accessible terms, the operation $\mathfrak{M}_{\beta, 1}$ in fact can only occur in the combination $\mathfrak{M}_{T/\beta, \beta} \circ \mathfrak{M}_{\beta, 1}$ that is Internal Merge and not on its own.

Case (2b) corresponds to a case of Sideward Merge. In this case one obtains in the new workspace F' a component of the form $\mathfrak{M}(T_i, \beta)$ and a component of the form T_j/β . Similarly, case (3b) also represents a case of Sideward Merge where in the resulting workspace F' one has new components $\mathfrak{M}(\alpha, \beta)$, as well as T_i/α and T_j/β . These cases of Sideward Merge also occur in the formulation of Merge of Definition 2.10, as the following statement clearly shows.

Lemma 2.14. *The two cases of Sideward Merge (2b) and (3b) are realized by the Merge operators of (2.23) with $\mathfrak{M}_{T_i, \beta}$ with T_i occurring as a component of F and β as an accessible term of a different component T_j of F , and $\mathfrak{M}_{\alpha, \beta}$ with $\alpha \in \text{Acc}(T_i)$ and $\beta \in \text{Acc}(T_j)$, for two components $i \neq j$ of F .*

The last remaining case (3a) corresponds to what is called Countercyclic Merge. In this case the new workspace F' contains new components $\mathfrak{M}(\alpha, \beta)$ and $T_i/(\alpha, \beta)$, where we write $T_i/(\alpha, \beta)$ for

the cancellation from the accessible terms of the copies of α and β inside T_i . This type of Merge can also be obtained through our formulation (2.23). In this case, as for the Internal Merge, one uses a composition of operators $\mathfrak{M}_{S,S'}$.

Lemma 2.15. *Countercyclic Merge is realized by a composition $\mathfrak{M}_{\alpha,\beta} \circ \mathfrak{M}_{\alpha,1} \circ \mathfrak{M}_{\beta,1}$ for α and β accessible terms of the same component T_i of F , with α seen as an accessible term of T_i/β after the application of the first $\mathfrak{M}_{\beta,1}$ operation.*

Proof. This is analogous to the Internal Merge case, with the first operator $\mathfrak{M}_{\beta,1}$ acting on a term $\beta \otimes T_i/\beta$ in the coproduct $\Delta(F)$, giving rise to two new components β and T_i/β in the new workspace. The second operator $\mathfrak{M}_{\alpha,1}$ can then be applied to this new workspace. This acts on the term of the form $\alpha \otimes (T_i/\beta)/\alpha$, producing a new workspace that contains components of the form α , β , and $T_i/(\alpha, \beta) = (T_i/\beta)/\alpha$. The last operation $\mathfrak{M}_{\alpha,\beta}$ then acts as External Merge on the components α , β of this workspace, producing the desired new component $\mathfrak{M}(\alpha, \beta)$, as well as retaining the component $T_i/(\alpha, \beta)$ with the cancellation of the deeper copies. \square

This brief discussion shows that, in addition to Internal and External Merge, our construction allows for extensions of Merge, such as Sideward and Countercyclic Merge that are not desirable from the linguistic perspective. We show in the next subsection how one can introduce a simple modification of the Merge operation described in (2.23) that will retain only Internal and External Merge. This will make use of the grading of the Hopf algebra, to introduce a Minimal Search that eliminates the other extensions of Merge.

To avoid misunderstandings as to the purpose of the construction in the coming §2.4.2 and §2.4.3, it is worth stressing that our goal here is simply to implement the usual mechanism by which, in linguistics, only Internal and External Merge are retained, and not other extensions of Merge, namely the mechanism of *Minimal Search*. Where our presentation differs from the usual formulation in linguistics, is that we provide a somewhat different-looking, but in fact equivalent, description of Minimal Search. The reason why we introduce a reformulation of Minimal Search is the following. We are arguing here that all the key properties of Merge follow directly from underlying Hopf-algebraic properties. In particular, in order to show that this is the case, we need to reformulate all the necessary aspects of the linguistic description of the key Merge operation in such algebraic terms, including Minimal Search. This requires describing the “minimality” property of Minimal Search in terms of a minimization procedure that can be made sense of entirely in terms of the algebraic structure. We argue in §2.4.2 and §2.4.3 below that this can indeed be done, with minimality expressed in the form of extraction of leading order, with respect to a suitable grading (cost function) associated to the terms of the coproduct.

2.4.2. Minimal search. In the formulation of Merge in [7], [8], the search for matching copies of S, S' in the workspace components and accessible terms, for the application of $\mathfrak{M}_{S,S'}$ is performed according to a “Minimal Search” principle, according to which accessible terms in the higher levels of trees are preferentially searched, before those occurring in the deeper levels.

In the formulation given in (2.23) this Minimal Search principle is implicitly built in, through the structure of the coproduct. Our coproduct extracts the entire list of accessible terms (simultaneously implementing the cancellation of copies). However, we can introduce a weight that keeps track of the depth of the accessible terms in each term of the coproduct. This can be done by introducing formal parameters ϵ and η and assigning to the subtrees $T_v \subset T$ a weight ϵ^{d_v} , where d_v is the distance of the vertex v from the root of T , and to the corresponding quotient trees T/T_v a weight η^{d_v} . This can be done by modifying the coproduct to

$$(2.25) \quad \Delta^{(\epsilon,\eta)} : \mathcal{V}(\mathfrak{T}_{S\mathcal{O}_0}) \rightarrow \mathcal{V}(\mathfrak{T}_{S\mathcal{O}_0})[\epsilon] \otimes_{\mathbb{Q}} \mathcal{V}(\mathfrak{T}_{S\mathcal{O}_0})[\eta],$$

$$\Delta^{(\epsilon, \eta)}(T) = \sum_{\underline{v}} \epsilon^{d_{\underline{v}}} F_{\underline{v}} \otimes \eta^{d_{\underline{v}}}(T/F_{\underline{v}}),$$

where for $\underline{v} = (v_1, \dots, v_n)$ a set of vertices $v_i \in V_{int}(T)$, we set $d_{\underline{v}} = d_{v_1} + \dots + d_{v_n}$, with d_v the distance to the root of T as above.

With this simple bookkeeping device, we see that, for instance, for small ϵ the higher levels of the tree T (internal vertices closer to the root), carry the largest weight, while the deeper levels are lower order contributions that can be neglected when one looks at the dominant terms in the coproduct. Note that the introduction of the parameters ϵ, η does not alter the coassociativity property of the coproduct and the compatibility with multiplication of Lemma 2.7. We also assign weight ϵ^d and η^d to the primitive terms of the coproduct $1 \otimes T$ and $T \otimes 1$, with $d = 0$ for both of these terms. Thus, the limit for $\epsilon \rightarrow 0$ and $\eta \rightarrow 0$ of the weighted coproduct $\Delta^{(\epsilon, \eta)}(T)$ retains only the primitive part $\Delta^{(0,0)}(T) = 1 \otimes T + T \otimes 1$.

2.4.3. Internal and External Merge. We now introduce a simple modification of the Merge operation described in (2.23), based on the form of Minimal Search described above, that will retain only External and Internal Merge, eliminating the other forms of Sideward and Countercyclic Merge.

Proposition 2.16. *Consider the modification of (2.23) given by*

$$(2.26) \quad \mathfrak{M}_{S, S'}^{\epsilon} = \sqcup \circ (\mathfrak{M}^{\epsilon} \otimes \text{id}) \circ \delta_{S, S'} \circ \Delta^{(\epsilon, \epsilon^{-1})},$$

with $\Delta^{(\epsilon, \epsilon^{-1})}$ as in (2.25), and with

$$(2.27) \quad \begin{aligned} \mathfrak{M}^{\epsilon} : \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})[\epsilon, \epsilon^{-1}] \otimes_{\mathbb{Q}} \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})[\epsilon, \epsilon^{-1}] &\rightarrow \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})[\epsilon, \epsilon^{-1}] \\ \mathfrak{M}^{\epsilon}(\epsilon^d \alpha, \epsilon^{\ell} \beta) &= \epsilon^{|d+\ell|} \mathfrak{M}(\alpha, \beta). \end{aligned}$$

Then taking compositions of operations of the form (2.26) followed by evaluation at $\epsilon \rightarrow 0$ retains only External and Internal Merge and eliminates all other extended forms of Merge, such as Sideward and Countercyclic.

Proof. For a single application of (2.26), one obtains terms of the form $\epsilon^{d_v+d_w} \mathfrak{M}(T_v, T_w)$, hence the only terms remaining after taking $\epsilon \rightarrow 0$ are of the form $\mathfrak{M}(T, T')$ with T, T' two connected components of the workspace F , which have degree zero in the ϵ variable. These are the External Merge cases. For a composition of two operators of the form (2.26), we regard the result of the first $\mathfrak{M}_{S, S'}^{\epsilon}$ applied to a forest $F \in \mathfrak{F}_{S\mathcal{O}_0}$ as a new workspace, which now carries a dependence on the parameter ϵ . We write such workspaces as $F(\epsilon) = \sqcup_a \epsilon^{d_a} T_a$ in the direct sum (as \mathbb{Q} -vector spaces) $\oplus_a \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})[\epsilon, \epsilon^{-1}]$. The composition with a second operator of the form (2.26), then produces terms of the form $\mathfrak{M}(\epsilon^{d_a} \epsilon^{d_{v_a}} T_{a, v_a}, \epsilon^{d_b} \epsilon^{d_{w_b}} T_{b, w_b})$ for components T_a, T_b and for vertices v_a, w_b in these components. Thus the remaining terms of this composition, after setting $\epsilon \rightarrow 0$ will be those with $d_a + d_b + d_{v_a} + d_{w_b} = 0$. Since $d_{v_a} + d_{w_b} \geq 0$, We need $d_a + d_b \leq 0$ and exactly matching the quantity $-(d_{v_a} + d_{w_b})$. This requires that at least one of the components T_a and T_b (say T_a) of the image of the first operation is a quotient $T_a = T/T_v$ of a component T of the initial workspace, with $d_a = -d_v$. This implies that the first \mathfrak{M}^{ϵ} had the corresponding T_v as one of the two arguments. The other component T_b is either an unchanged component $T_b = T'$ of the original workspace F , or again a quotient $T_b = T'/T'_w$ of a component, if T'_w was the other argument of the first \mathfrak{M}^{ϵ} , or else it can be $T_b = 1$, the trivial tree, so that in all cases either $d_b = 0$ or $d_b = -d_w$. Thus, we obtain either a Merge of the form $\mathfrak{M}(\mathfrak{M}(T', T_v), T/T_v)$ or of the form $\mathfrak{M}(T', \mathfrak{M}(T_v, T/T_v))$. The second case is clearly a composition of internal and external Merge, so it is of the desired form. The first case does not appear to be consisting only of internal/external Merge, but in fact it can be equivalently realized as $\mathfrak{M}(\mathfrak{M}(T', T_v), \tilde{T}/\mathfrak{M}(T', T_v))$, for another tree \tilde{T} , which is obtained by

contracting T_v to its root vertex in T and gluing that root vertex to the root of $\mathfrak{M}(T', T_v)$. Thus, it is also a composition of internal and external Merge. The case of repeated compositions can be analyzed in the same way. \square

Thus, Proposition 2.16 shows that Minimal Search is equivalently described by taking the leading order term for $\epsilon \rightarrow 0$ of the operations formed using $\mathfrak{M}_{S,S'}^\epsilon$.

2.5. Other linguistic properties. We verify here that the action of Merge on workspaces defined as in (2.23) satisfies the desired linguistic properties. First that it accounts for the usual types of Merge: external and internal Merge, and a form of sideways Merge. We then show that several properties that are imposed empirically on Merge are in fact naturally built into the mathematical formulation. In particular, we discuss the requirement that Merge does not decrease the total size of the workspaces and increases it at most by one. We show that this is indeed the case for the dominant (Minimal Search) part $\mathfrak{M}_{S,S'}^\epsilon|_{\epsilon=0}$ obtained as in Proposition 2.16, which recovers internal/external Merge, while violations occur when one also includes Sideward/Countercyclic Merge, confirming what is known from linguistics. Moreover, we show that the cancellation of copies of accessible terms in the resulting workspaces is dictated not only by ‘economy principles’ but also by algebraic constraints, namely by the coassociativity property of the coproduct.

2.5.1. Cases of Merge and size counting. We now analyze the effect of the action of Merge on workspaces in terms of the effect on the size of workspace and on the number of accessible terms.

We show that this property holds for the dominant term (the $\epsilon = 0$ term) of the Merge action of Proposition 2.16, which gives Internal/External Merge, while it generally fails for the other forms of Sideward/Countercyclic Merge. This recovers an observation already known from linguistics.

We first discuss the cases of External/Internal Merge, obtained as $\epsilon \rightarrow 0$ dominant terms as in Proposition 2.16. We then discuss the other forms of Merge that are eliminated by Minimal Search, that is, that do not occur in the $\epsilon \rightarrow 0$ limit of Proposition 2.16.

Proposition 2.17. *Under External and Internal Merge, the effect on the counting functions of Definition 2.4 is given by the following table, where we display the difference between the counting function after application of Merge and before.*

	b_0	$\#Acc$	σ	$\hat{\sigma}$
<i>External</i>	-1	+2	+1	0
<i>Internal</i>	0	0	0	0

Proof. In the case of External Merge, for $F = \sqcup_a T_a$, we have $F' = \mathfrak{M}_{S,S'}(F)$ given by

$$F' = \mathfrak{M}(T_i, T_j) \sqcup \hat{F}^{(i,j)},$$

with $T_i \simeq S$, $T_j \simeq S'$ (where S, S' are assumed to be non-trivial syntactic objects, that is, not equal to 1), and with $\hat{F}^{(i,j)} = F \setminus (T_i \sqcup T_j)$ the remaining components. Thus, the number of connected components (of syntactic objects in the workspace) decreases by one, $b_0(F') = b_0(F) - 1$. The number of accessible terms, on the other hand, satisfies

$$\#Acc(F') = \#V_{int}(F') = \#Acc(F) + 2,$$

as the two root vertices of T_i and T_j become internal vertices of $\mathfrak{M}(T_i, T_j)$, while all the other internal vertices remain unchanged. The size of the workspace satisfies

$$\sigma(F') = b_0(F') + \#Acc(F') = \#V(F') = \sigma(F) + 1,$$

since we have

$$\sigma(\mathfrak{M}(T_i, T_j)) = \sigma(T_i) + \sigma(T_j) + 1.$$

The size function $\hat{\sigma}$ on the other hand gives

$$\hat{\sigma}(F') = b_0(F') + \#V(F') = b_0(F) - 1 + \#V(F) + 1 = \hat{\sigma}(F),$$

hence it is a conserved quantity under External Merge. The special case $\mathfrak{M}_{S,1}$, where $S' = 1$ is the trivial object (empty tree) is discussed in Remark 2.19 below.

In the case of Internal Merge, the number of connected components (number of syntactic objects in the workspace) remains unchanged, as Internal Merge operates on a single tree, so $b_0(F') = b_0(F)$. When counting the number of internal (that is, non-root) vertices, which is the number of accessible terms, we see that the old root of the tree T , which is also the root of T/T_v becomes a new non-root vertex, while in the process of taking the quotient T/T_v according to Definition 2.5, two vertices are identified, hence the overall change in the number of internal vertices is zero, and so is the change in the total number of vertices (the size σ) where we have

$$\sigma(T_v) + \sigma(T/T_v) + 1 = \sigma(T) \text{ since } \#Acc(T_v) + \#Acc(T/T_v) + 2 = \#Acc(T),$$

because T/T_v is obtained by removal of T_v , contraction of the edge above the root of T_v and of the other edge adjacent to it at the vertex above the root of T_v , so that all the vertices of T_v as well as one additional vertex of T are removed to form T/T_v . This gives

$$\sigma(F') = \sigma(\mathfrak{M}(T_v, T/T_v)) + \sigma(\hat{F}) = \sigma(T_v) + \sigma(T/T_v) + 1 + \sigma(\hat{F}).$$

Similarly, $\hat{\sigma}(F') = b_0(F') + \sigma(F') = b_0(F) + \sigma(F)$, so that the size $\hat{\sigma}$ also remains constant. Thus, with our choice of counting measures as in Definition 2.4, all these quantities are preserved unchanged under Internal Merge. \square

Remark 2.18. The fact that under External Merge the number of syntactic objects decreases by one and the number of accessible terms increases by two, while under Internal Merge both the number of syntactic objects and the number of accessible terms remain the same is consistent with the counting in [22]. Note that if one takes the quotient T/T_v in the more sense of contracting T_v to its root vertex, then the number of accessible terms in Internal Merge would increase by one: this is the counting considered by Riny Huijbregts. With this choice of quotient and counting, both Internal and External Merge would increase the number of accessible terms by exactly one. This makes the choice appealing, but we prefer to maintain the counting as in [22] because taking the quotient T/T_v as in Definition 2.5 has advantages over the simple contraction of T_v to the root, in particular not needing projections to assign a syntactic feature label to this root vertex, which would become a leaf in the contraction quotient.

Remark 2.19. In the special case of a Merge $\mathfrak{M}_{S,1}$, where $S' = 1$ is the trivial syntactic object (empty tree), if S is matched by a component tree T_i of the workspace forest $F = \sqcup_a T_a$, then $\mathfrak{M}(T_i, 1) = T_i$ so $F' = \mathfrak{M}_{S,1}(F) = F$ and the operation is just the identity. If S is matched by a subtree T_{i,v_i} of a component T_i of F , then

$$\mathfrak{M}_{S,1}(F) = \mathfrak{M}(T_{i,v_i}, 1) \sqcup T_i/T_{i,v_i} \sqcup \hat{F} = T_{i,v_i} \sqcup T_i/T_{i,v_i} \sqcup \hat{F},$$

for $\hat{F} = \sqcup_{a \neq i} T_a$. In this case the number of connected components is growing by one, as the component T_i is separated into two components $T_{i,v_i} \sqcup T_i/T_{i,v_i}$, while the total number of vertices is decreasing by one, since two vertices are identified in taking the quotient $T_i/T_{i,v_i}$ while all other vertices remain unchanged. The number of accessible terms is decreasing by two, as the root vertex of T_v is now the root of a component. Thus, we have the table

	b_0	$\#Acc$	σ	$\hat{\sigma}$
$\mathfrak{M}_{S,1}$	+1	-2	-1	0

If one imposes either that the size of the workspace (total number of vertices) should not decrease or that the number of accessible terms should not decrease, then this implies that a Merge of the form $\mathfrak{M}_{S,1}$ only occurs in compositions such as $\mathfrak{M}_{\beta,T/\beta} \circ \mathfrak{M}_{\beta,1}$ that give an Internal Merge as in Proposition 2.12, but not alone, since otherwise we would violate such conditions. This is consistent with the fact that the weight in ϵ in Proposition 2.16 also excludes the occurrence of a Merge $\mathfrak{M}_{S,1}$ by itself rather than in a composition that forms an Internal Merge.

Remark 2.20. When we consider separately the number of connected components (number of syntactic objects) $b_0(F)$ of the workspace, rather than the size $\sigma(F) = b_0(F) + \#Acc(F)$ we see that, as expected, this decreases by one under External Merge while remaining unchanged under Internal Merge, so that the number of components decreases overall during the course of a derivation, as expected, leading to the desired “convergence”.

The remaining cases of the Merge operation (2.23), which are subdominant in $\epsilon \rightarrow 0$ in (2.26) (hence eliminated by Minimal Search) have a different behavior with respect to the counting functions of Definition 2.4.

Proposition 2.21. *In the cases of Sideward and Countercyclic Merge in (2.23) we have the following change in the counting functions of Definition 2.4.*

	b_0	$\#Acc$	σ	$\hat{\sigma}$
<i>Sideward (3b)</i>	+1	0	+1	+2
<i>Sideward (2b)</i>	0	+1	+1	+1
<i>Countercyclic (3a) (i)</i>	+1	$\#Acc(T_{a,w_a})$	$\sigma(T_{a,w_a})$	$\sigma(T_{a,w_a}) + 1$
<i>Countercyclic (3a) (ii)</i>	+1	$\#Acc(T_{a,v_a})$	$\sigma(T_{a,v_a})$	$\sigma(T_{a,v_a}) + 1$
<i>Countercyclic (3a) (iii)</i>	+1	-2	-1	0

where in case (3a) T_{a,v_a} and T_{a,w_a} are the two subtree of the same component T_a used for Countercyclic Merge.

Proof. In the case of case of Sideward Merge, case (3b) of Section 2.4.1, we have, for $F = \sqcup_i T_i$,

$$F' = \mathfrak{M}(T_{a,v_a}, T_{b,w_b}) \sqcup T_a/T_{a,v_a} \sqcup T_b/T_{b,w_b} \sqcup \hat{F}^{(a,b)},$$

with $\hat{F}^{(a,b)} = \sqcup_{i \neq a,b} T_i$. Thus, the number of connected components increases by one, because of the new component $\mathfrak{M}(T_{a,v_a}, T_{b,w_b})$, while the number of accessible terms is given by

$$\begin{aligned} \#Acc(F') &= \#Acc(\mathfrak{M}(T_{a,v_a}, T_{b,w_b})) + \#Acc(T_a/T_{a,v_a}) + \#Acc(T_b/T_{b,w_b}) + \#Acc(\hat{F}^{(a,b)}) \\ &= \#Acc(T_{a,v_a}) + \#Acc(T_{b,w_b}) + 2 + \#Acc(T_a) - \#Acc(T_{a,v_a}) - 1 \\ &\quad + \#Acc(T_b) - \#Acc(T_{b,w_b}) - 1 + \#Acc(\hat{F}^{(a,b)}) = \#Acc(F). \end{aligned}$$

Thus, $\sigma(F') = b_0(F') + \#Acc(F') = \sigma(F) + 1$ and $\hat{\sigma}(F') = \hat{\sigma}(F) + 2$.

For Sideward Merge, case (2b) of Section 2.4.1 we similarly have

$$F' = \mathfrak{M}(T_a, T_{b,w_b}) \sqcup T_b/T_{b,w_b} \sqcup \hat{F}^{(a,b)},$$

so that $b_0(F') = b_0(F)$, since one new component $\mathfrak{M}(T_a, T_{b,w_b})$ is created and one component T_b is removed. The counting of accessible terms give

$$\begin{aligned} \#Acc(F') &= \#Acc(\mathfrak{M}(T_a, T_{b,w_b})) + \#Acc(T_b/T_{b,w_b}) + \#Acc(\hat{F}^{(a,b)}) \\ &= \#Acc(T_a) + \#Acc(T_{b,w_b}) + 2 + \#Acc(T_b) - \#Acc(T_{b,w_b}) - 1 + \#Acc(\hat{F}^{(a,b)}) = \#Acc(F) + 1. \end{aligned}$$

Thus, the size satisfies $\sigma(F') = \sigma(F) + 1$ and $\hat{\sigma}(F') = \hat{\sigma}(F) + 1$.

In the case of Countercyclic Merge we have, for $F = \sqcup_i T_i$

$$F' = \mathfrak{M}(T_{a,v_a}, T_{a,w_a}) \sqcup T_a/T_{a,v_a,w_a} \sqcup \hat{F}^{(a)},$$

where $T_{a,v_a,w_a} \subset T_a$ is given by

$$(2.28) \quad T_{a,v_a,w_a} := \begin{cases} T_{a,v_a} & \text{case (i): if } T_{a,w_a} \subset T_{a,v_a} \\ T_{a,w_a} & \text{case(ii): if } T_{a,v_a} \subset T_{a,w_a} \\ T_{a,v_a} \sqcup T_{a,w_a} & \text{case (iii): if } T_{a,v_a} \cap T_{a,w_a} = \emptyset, \end{cases}$$

and $\hat{F}^{(a)} = \sqcup_{i \neq a} T_i$. Thus, we obtain one additional connected component $\mathfrak{M}(T_{a,v_a}, T_{a,w_a})$, so that

$$b_0(F') = b_0(F) + 1.$$

The counting of accessible terms is given by

$$\begin{aligned} \#Acc(F') &= \#Acc(\mathfrak{M}(T_{a,v_a}, T_{a,w_a})) + \#Acc(T_a/T_{a,v_a,w_a}) + \#Acc(\hat{F}^{(a)}) \\ &= \#Acc(T_{a,v_a}) + \#Acc(T_{a,w_a}) + 2 + \#Acc(T_a/T_{a,v_a,w_a}) + \#Acc(\hat{F}^{(a)}), \end{aligned}$$

because the root vertices of T_{a,v_a} and T_{a,w_a} also appear as accessible terms in $\mathfrak{M}(T_{a,v_a}, T_{a,w_a})$, while we have

$$\begin{aligned} \#Acc(F) &= \#Acc(T_a) + \#Acc(\hat{F}^{(a)}) = \#Acc(T_{a,v_a}) + 2 + \#Acc(T_a/T_{a,v_a}) + \#Acc(\hat{F}^{(a)}) \\ &= \#Acc(T_{a,w_a}) + 2 + \#Acc(T_a/T_{a,w_a}) + \#Acc(\hat{F}^{(a)}). \end{aligned}$$

Thus, in the first two cases (i) and (ii) of (2.28), we respectively have

$$\#Acc(F') = \begin{cases} \#Acc(F) + \#Acc(T_{a,w_a}) & \text{if } T_{a,v_a,w_a} = T_{a,v_a} \\ \#Acc(F) + \#Acc(T_{a,v_a}) & \text{if } T_{a,v_a,w_a} = T_{a,w_a} \end{cases}$$

This then gives, for these two cases

$$\sigma(F') = \begin{cases} \sigma(F) + \sigma(T_{a,w_a}) & \text{if } T_{a,v_a,w_a} = T_{a,v_a} \\ \sigma(F) + \sigma(T_{a,v_a}) & \text{if } T_{a,v_a,w_a} = T_{a,w_a}, \end{cases}$$

since $\sigma(F') = b_0(F') + \#Acc(F')$, while $\hat{\sigma}(F') = b_0(F') + \sigma(F')$ has an additional increase by +1.

The third case (iii) of (2.28) has two possibilities. If there is a vertex u_a in T_a that is adjacent to both the roots of T_{a,v_a} and T_{a,w_a} then the tree T_a contains a subtree with root u_a that is isomorphic to $\mathfrak{M}(T_{a,v_a}, T_{a,w_a})$ so that we have, in this case,

$$F' = \mathfrak{M}(T_{a,v_a}, T_{a,w_a}) \sqcup T_a/\mathfrak{M}(T_{a,v_a}, T_{a,w_a}) \sqcup \hat{F}^{(a)},$$

so that the counting of accessible terms satisfies

$$\begin{aligned} \#Acc(F') &= \#Acc(\mathfrak{M}(T_{a,v_a}, T_{a,w_a})) + \#Acc(T_a/\mathfrak{M}(T_{a,v_a}, T_{a,w_a})) + \#Acc(\hat{F}^{(a)}) \\ &= \#Acc(T_a) - 2 + \#Acc(\hat{F}^{(a)}) = \#Acc(F) - 2. \end{aligned}$$

The workspace sizes correspondingly change by

$$\sigma(F') = b_0(F') + \#Acc(F') = \sigma(F) - 1 \quad \text{and} \quad \tilde{\sigma}(F') = 2b_0(F') + \#Acc(F') = \tilde{\sigma}(F).$$

The other possibility for case (iii) is that the vertices above the roots of T_{a,v_a} and T_{a,w_a} are different. In this case, $T_a/T_{a,v_a,w_a} = (T_a/T_{a,v_a})/T_{a,w_a} = (T_a/T_{a,w_a})/T_{a,v_a}$. Thus, we have

$$\begin{aligned} \#Acc(T_a/T_{a,v_a,w_a}) + \#Acc(T_{a,w_a}) + 2 &= \#Acc(T_a/T_{a,v_a}) \\ &= \#Acc(T_a) - \#Acc(T_{a,v_a}) - 2, \end{aligned}$$

so that

$$\#Acc(T_a/T_{a,v_a,w_a}) = \#Acc(T_a) - \#Acc(T_{a,v_a}) - \#Acc(T_{a,w_a}) - 4.$$

Thus we have

$$\begin{aligned} \#Acc(F') &= \#Acc(\mathfrak{M}(T_{a,v_a}, T_{a,w_a})) + \#Acc(T_a/T_{a,v_a,w_a}) + \#Acc(\hat{F}^{(a)}) \\ &= \#Acc(T_{a,v_a}) + \#Acc(T_{a,w_a}) + 2 + \#Acc(T_a/T_{a,v_a,w_a}) + \#Acc(\hat{F}^{(a)}) \\ &= \#Acc(T_a) - 2 + \#Acc(\hat{F}^{(a)}) = \#Acc(F) - 2, \end{aligned}$$

so that we obtain the same counting as in the first case of (iii). \square

We see from Proposition 2.21 that various requirements on the counting functions of Definition 2.4 can be used to rule out these forms of Sideward and Countercyclic Merge. In particular we look at the effect of requirements that the number of accessible terms in the workspace should be non-decreasing ($\Delta(\#Acc) \geq 0$) and the number of syntactic objects should be non-increasing ($\Delta b_0 \leq 0$); that the overall size of the workspace does not decrease and does not increase more than one ($0 \leq \Delta\sigma \leq 1$), and the requirement that $\hat{\sigma}$ is a conserved quantity ($\Delta\hat{\sigma} = 0$). Note that all of these conditions are satisfied by Internal and External Merge.

Corollary 2.22. *Constraints on counting functions in the cases of Sideward and Countercyclic Merge are (Y) or are not (N) satisfied according to the following table.*

	$\Delta b_0 \leq 0$	$\Delta(\#Acc) \geq 0$	$0 \leq \Delta\sigma \leq 1$	$\Delta\hat{\sigma} = 0$
<i>Sideward (3b)</i>	N	Y	Y	N
<i>Sideward (2b)</i>	Y	Y	Y	N
<i>Countercyclic (3a) (i)</i>	N	Y	N	N
<i>Countercyclic (3a) (ii)</i>	N	Y	N	N
<i>Countercyclic (3a) (iii)</i>	N	N	N	Y

Thus, all the cases of Sideward and Countercyclic Merge of Proposition 2.21 are ruled out by at least one of these conditions, but not all of them by the same one, and $\Delta b_0 \leq 0$ and $\Delta(\#Acc) \geq 0$ together do not suffice to rule out all of these cases, but conservation $\Delta\hat{\sigma} = 0$ together with any one of the other conditions suffices.

This shows that constraints based on counting are less efficient than the constraint based on Minimal Search (the $\epsilon \rightarrow 0$ limit in Proposition 2.16), which rules out all of these remaining forms of Sideward and Countercyclic Merge.

Remark 2.23. Note however that, if one take quotients by contraction to the root vertex, instead of using the quotient as in Definition 2.5, then the condition that *the number of accessible terms should increase exactly by one* (as suggested by Huijbregts, see Remark 2.18) would suffice to rule out all the cases of Sideward and Countercyclic Merge. Indeed, with the quotient by contraction, we would obtain that the number of accessible terms for both Sideward and Countercyclic Merge would always increase by at least two.

2.5.2. Cancellation of copies. In the form (2.23) of the action of Merge on workspaces, the cancellation of copies of the accessible terms used by Merge is implemented by the coproduct Δ of (2.10) through the quotient terms T/T_v .

Cancellation of copies is usually postulated as an “economy principle” in linguistics, and it is usually assumed that cancellation always happens in the deeper copies.

A first observation is that, in the formalism we are using, the fact that cancellation is implemented in the deeper copy is directly built into the structure of the coproduct and it does not have to be included as an additional requirement, since in the terms $T_v \otimes T/T_v$ the copy of T_v on

the left-hand-side (the one that contributed to Merge) has lower depth than the copy inside T , which is cancelled on the right-hand-side.

A second observation is that cancellation of copies is necessary in order to have a good coassociative coproduct. Indeed, one needs to quotient out the copy of T_v inside T in the right-hand-side of the coproduct for coassociativity to work as shown in Lemma 2.7. One can see that a coproduct of the form $T \mapsto \sum_v T_v \otimes T$ without the cancellation would no longer have this property.

Note moreover that, although we refer to the term T/T_v as ‘‘cancellation’’ of a copy of T_v , nothing is really cancelled, since a copy of T_v remains on the other side of the term $T_v \otimes T/T_v$ of the coproduct. The basic structure of the coproduct separates out trees (in all possible ways) into a subtree and a quotient. One can simply then read the subtree as the ‘‘creation of a copy’’ and the quotient tree as corresponding ‘‘cancellation of the original (deeper) copy’’ when Internal Merge is applied.

Also observe that there are distinct roles in the model we are discussing here for copies and repetitions. Repetitions are accounted for in this setting by the fact that we are defining the workspace as a forest (a disjoint union of trees, that is, of syntactic objects). This allows for the presence of repetitions since a forest is not a set but a multiset of trees. Copies, on the other hand are only created during the application of the coproduct, and ultimately play a role only in the operation of Internal Merge.

3. THE CORE COMPUTATIONAL STRUCTURE OF MERGE

The description of Merge and its action on workspaces that we described above follows closely the formulation presented in [7]. We discuss here a further simplification of the structure of Merge, which extract its core computational structure, as presented in [10].

Let \mathfrak{T} be the set of binary rooted trees without planar structure (and without labeling of the leaves), and $\mathcal{V}(\mathfrak{T})$ the free \mathbb{Z} -module (or the \mathbb{Q} -vector space) spanned by the set \mathfrak{T} . The following description is the analog of Definition 2.1 and Remark 2.2.

Lemma 3.1. *The set \mathfrak{T} is the free non-associative, commutative magma whose elements are the balanced bracketed expressions in a single variable x , with the binary Merge operation $(\alpha, \beta) \mapsto \mathfrak{M}(\alpha, \beta) = \{\alpha, \beta\}$. Correspondingly, $\mathcal{V}(\mathfrak{T})$ is the free commutative non-associative algebra generated by a single variable x .*

Proof. We can identify the binary rooted trees without planar structure with the balanced bracketed expressions in a single variable x . For example

$$\{\{x\{xx\}\}x\} \longleftrightarrow \begin{array}{c} \diagup \quad \diagdown \\ x \quad \quad x \\ \diagdown \quad \diagup \\ x \quad x \end{array} .$$

The Merge operation $\mathfrak{M}(\alpha, \beta) = \{\alpha, \beta\}$ takes two such bracketed expressions α and β and forms a new one of the form $\{\alpha, \beta\}$, which correspond to attaching the roots of the two binary trees to a common root, $\mathfrak{M}(T, T') = T \wedge T'$. \square

Equivalently, $\mathcal{V}(\mathfrak{T})$ is the free algebra over the quadratic operad freely generated by the single commutative binary operation \mathfrak{M} (see [27]).

The generative process for the set \mathfrak{T} via the Merge operation can be equivalently described as a recursive procedure encoded in the form of a fixed point equation.

Proposition 3.2. *Let $\mathcal{V}(\mathfrak{T}) = \bigoplus_{\ell} \mathcal{V}(\mathfrak{T})_{\ell}$ with the grading by length (number of leaves) as before, with $\mathfrak{M} : \mathcal{V}(\mathfrak{T})_{\ell} \times \mathcal{V}(\mathfrak{T})_{\ell'} \rightarrow \mathcal{V}(\mathfrak{T})_{\ell+\ell'}$, where \mathfrak{M} is extended by linearity in each variable. Consider*

formal infinite sums $X = \sum_{\ell \geq 1} X_\ell$ with $X_\ell \in \mathcal{V}(\mathfrak{T})_\ell$ and the recursive equation

$$(3.1) \quad X = \mathfrak{M}(X, X).$$

Then the generative process for \mathfrak{T} via the Merge operation is equivalent to the recursive construction of a solution of (3.1) with initial condition $X_1 = x$.

Proof. We have $\mathfrak{M}(\sum_\ell X_\ell, \sum_{\ell'} X_{\ell'}) := \sum_{\ell, \ell'} \mathfrak{M}(X_\ell, X_{\ell'})$. In particular, the term of degree n in $\mathfrak{M}(X, X)$ is given by

$$\mathfrak{M}(X, X)_n = \sum_{j=1}^{n-1} \mathfrak{M}(X_j, X_{n-j}),$$

so that the fixed point equation (3.1) reduces to the recursive relation

$$X_n = \sum_{j=1}^{n-1} \mathfrak{M}(X_j, X_{n-j}).$$

starting with $X_1 = x$, the recursion produces $X_2 = \{xx\}$, $X_3 = \{x\{xx\}\} + \{\{xx\}x\} = 2\{x\{xx\}\}$, $X_4 = 2\{x\{x\{xx\}\}\} + \{\{xx\}\{xx\}\}$, and so on. These first terms X_n list all the possible non-planar binary rooted trees with n leaves, with multiplicities that account for the different planar structures. Given a non-planar binary rooted tree T with n leaves, we can always write it as $T = \mathfrak{M}(T', T'')$ where T', T'' are the two binary rooted trees with roots at the two internal vertices of T connected to the root of T , with $j = \#L(T')$ and $n-j = \#L(T'')$, for some $j \in \{1, \dots, n-1\}$. Since the Merge product is commutative it does not matter in which order we list T' and T'' . Thus, each $T \in \mathfrak{T}_n$ can be mapped uniquely to an *unordered* pair $\{T', T''\}$ and conversely, any pair of trees T', T'' with numbers of leaves ℓ' and ℓ'' , respectively, determines uniquely a tree $\mathfrak{M}(T', T'')$ with $\ell' + \ell''$ leaves. Thus, inductively, if each X_j for $1 \leq j < n$ consists of a list (formal sum) of all the possible non-planar binary rooted trees with j leaves, then X_n also consists of a sum of all the possible non-planar binary rooted trees with n leaves. One sees similarly that the integer coefficients in the sum count different planar structures. \square

As we discuss further in §6, this shows that the generative process for the core computational structure of Merge in the Minimalist Model of syntax is in fact the most fundamental basic case of the Dyson–Schwinger equations in physics. We give a quick summary here of what we will discuss in more detail in §6.

In general, the Dyson–Schwinger equation implements in perturbative quantum field theory the construction of solutions of the equations of motion. It is a way of encoding the variational principle of least action for equations of motion in classical physics in a form suitable for quantum fields, via a recursive method of solution that can be performed order by order in the perturbative expansion. There are two main conceptual aspects to single out here. One is the fact that the construction of solutions of Dyson–Schwinger equations becomes a combinatorial problem, in terms of Feynman graphs and associated trees, expressible as a solution to a fixed point equation, of which (3.1) is the most fundamental example. The general such combinatorial Dyson–Schwinger equation always involves a form of (possible n -ary) Merge operation, given by the grafting operator B^+ of Definition 2.8, and a polynomial fixed point equation in a Hopf algebra, which takes the general form $X = B^+(P(X))$, for a polynomial P and a variable $X = \sum_\ell X_\ell$ in (a completion of) a Hopf algebra of rooted trees. The equation is solved recursively, as in the fundamental case of (3.1). The other aspect is the usual requirement that for classical solutions of the equations of motion the action functional is stationary under infinitesimal variations. This is transformed in the case of quantum fields into corresponding equations for the quantum correlation functions. In the formulation of perturbative quantum field theory in terms of Hopf algebra, these in turn arise from

the combinatorial solution, which is entirely determined in terms of the underlying Hopf-algebraic structure, together with the evaluation of a (renormalized) Feynman rule, to obtain the actual physical solution from the combinatorial one. As we discuss further in §6, the first observation identifies the generative process of syntactic objects through Merge with the basic case of the structure of generative processes of fundamental physics. The second observation suggests that the optimality that the core computational structure of Merge ought to satisfy is of the same conceptual nature as the least action principle of physics, when the latter manifests itself in a combinatorial form.

4. CONSTRAINTS ON MERGE: THE n -ARITY QUESTION

An important question regarding the Merge operation of syntax is whether the same generative power would be achievable with a similar operation that is n -ary, for some $n \geq 3$, rather than binary.

Riny Huijbregts presented in [28] strong empirical linguistic evidence for why, for example, a ternary Merge would be inadequate, in the sense that such a ternary operation would produce both *undergeneration* and *overgeneration* with respect to the binary Merge. Undergeneration refers to syntactic constructions that can be derived through the binary Merge but would not be generated by a ternary Merge, while overgeneration consists of ungrammatical sentences that would be generated by a ternary Merge, but not by binary Merge. While the undergeneration problem could in principle be bypassed by hypothesizing the simultaneous presence of a binary and a ternary Merge, the overgeneration problem cannot be similarly dealt with.

We discuss here briefly why any n -ary Merge operation, for any $n \geq 3$, would necessarily lead to both undergeneration and overgeneration, as a simple consequence of the algebraic structure described in the previous section. In particular, within this formulation one can see that undergeneration and overgeneration have two somewhat different origins. Undergeneration is a direct consequence of the structure of the magma on the Merge operation, which gives rise to the set of syntactic objects, while overgeneration involves directly the action of Merge on workspaces.

4.1. The n -ary Merge magma. Here we assume the existence of a hypothetical n -ary Merge, for some $n \geq 3$, and we discuss how the structure of the magma of syntactic objects changes with respect to the binary case. We assume the same initial set \mathcal{SO}_0 of lexical terms and syntactic features.

Definition 4.1. *An n -magma consists of a set X together with an n -ary operation*

$$\mathfrak{M}_n : X \times \underbrace{\cdots \times X}_{n\text{-times}} \rightarrow X, \quad (x_1, \dots, x_n) \mapsto \mathfrak{M}_n(x_1, \dots, x_n).$$

We say that (X, \mathfrak{M}_n) is an n -magma over a set Y , if all elements of X are obtained by iterated application of \mathfrak{M}_n starting with n -tuples of elements in Y .

We write $\{x_1, \dots, x_n\} := \mathfrak{M}_n(x_1, \dots, x_n)$ for the element of X that is obtained by applying \mathfrak{M}_n to the n -tuple (x_1, \dots, x_n) . In particular, the set X consists of a subset X_1 consisting of all elements of the form $\{y_1, \dots, y_n\} := \mathfrak{M}_n(y_1, \dots, y_n)$ with all the $y_i \in Y$, a set X_{2n-1} consisting of all elements of the form

$$\mathfrak{M}_n(y_1, \dots, y_{i-1}, \mathfrak{M}_n(a_{i,1}, \dots, a_{i,n}), y_{i+1}, \dots, y_n) = \{y_1, \dots, y_{i-1}, \{a_{i,1}, \dots, a_{i,n}\}, y_{i+1}, \dots, y_n\}$$

for $i = 1, \dots, n$ and with all the $y_i, a_{i,j} \in Y$, a set X_{3n-2} consisting of all elements of the form

$$\{y_1, \dots, y_{i-1}, \{a_{i,1}, \dots, a_{i,n}\}, y_{i+1}, \dots, y_{j-1}, \{b_{j,1}, \dots, b_{j,n}\}, y_{j+1}, \dots, y_n\} \quad \text{and}$$

$$\{y_1, \dots, y_{i-1}, \{a_{i,1}, \dots, a_{i,j-1}, \{b_{j,1}, \dots, b_{j,n}\}, a_{i,j+1}, a_{i,n}\}, y_{i+1}, \dots, y_n\},$$

with $i \neq j$, $i, j = 1, \dots, n$, and all the $y_i, a_{i,k}, b_{j,k} \in Y$, and so on, so that we have

$$(4.1) \quad X = \bigsqcup_{k \geq 1} X_{k(n-1)+1}.$$

We refer to the subset $X_{k(n-1)+1}$ as the set of elements of length $k(n-1)+1$ in the n -magma.

The n -magma is associative if all the elements of length $k(n-1)+1$ are identified, that is, if bracketing is irrelevant. It is commutative if elements $\{x_1, \dots, x_n\}$ with entries that differ by a permutation in the symmetric group S_n are identified, that is, if every set within brackets is unordered.

We then have the following description of the set of syntactic objects produced by a hypothetical n -ary Merge \mathfrak{M}_n .

Definition 4.2. *The set $\mathcal{SO}^{(n)}$ of n -ary syntactic objects is the free, non-associative, commutative n -magma on the set \mathcal{SO}_0 ,*

$$(4.2) \quad \mathcal{SO}^{(n)} = \text{Magma}_{na,c}^{(n)}(\mathcal{SO}_0, \mathfrak{M}_n),$$

with

$$(4.3) \quad \mathcal{SO}^{(n)} = \bigsqcup_{k \geq 1} \mathcal{SO}_{k(n-1)+1}^{(n)}.$$

Remark 4.3. We can identify the elements of $\mathcal{SO}^{(n)}$ with rooted n -ary trees,

$$(4.4) \quad \mathcal{SO}^{(n)} \simeq \mathfrak{T}_{\mathcal{SO}_0}^{(n)},$$

namely trees where all the non-leaf vertices have n descendants, without a planar structure, and with leaves labelled by elements of the set \mathcal{SO}_0 . (Note that what we call here n -ary trees are *full* n -ary trees.)

The set $\mathcal{SO}_{k(n-1)+1}^{(n)}$ is the set of rooted n -ary trees (with no assigned planarity) with $k(n-1)+1$ leaves, and therefore with k non-leaf vertices. The number k of non-leaf vertices is the number of applications of \mathfrak{M}_n in the process of generating elements of $\mathcal{SO}^{(n)}$, where each non-leaf vertex is the graphical representation of a Merge operation.

4.2. Undergeneration. Given the structure (4.3) of the set of n -ary syntactic objects, we can show that there are two different forms of undergeneration (with respect to the binary Merge), and that both of them inevitably occur for any n -ary Merge with $n \geq 3$. The two different forms of undergeneration correspond, respectively, to certain lengths not being achievable through an n -ary Merge construction, and to certain syntactic parsing ambiguities not being accountable for by an n -ary Merge construction.

The first form of undergeneration can be seen as follows.

Lemma 4.4. *Only strings of elements of \mathcal{SO}_0 of length $k(n-1)+1$, for some $k \geq 1$, can be achieved through an n -ary Merge. In particular, only the binary Merge can achieve all lengths.*

Proof. The number of leaves of an n -ary tree with k non-leaf vertices is $k(n-1)+1$. Thus, the only possible strings of elements of \mathcal{SO}_0 that can be obtained through k successive applications of an n -ary Merge \mathfrak{M}_n are of length $k(n-1)+1$, as in the decomposition (4.3) of the set of n -ary syntactic objects. Only in the case $n = 2$ the set $\{k(n-1)+1\}_{k \geq 1}$ contains all positive integers greater than or equal to 2. \square

Known empirical linguistic examples of this kind of undergeneration include, for instance, the fact that sentences like “*it rains*” are in \mathcal{SO}_2 while $\mathcal{SO}_2^{(3)} = \emptyset$.

The second form of undergeneration can be seen through counting and comparing the sizes of the sets $\mathcal{SO}_{k(n-1)+1}$ and $\mathcal{SO}_{k(n-1)+1}^{(n)}$, for $n \geq 3$. The counting formulae for rooted trees are simpler in the case of trees with an assigned planar structure, rather than for abstract trees with no assigned planarity. Thus, we count the resulting trees after the externalization step that introduces planar structures.

Let $\mathfrak{T}_{\mathcal{SO}_0}^{pl} = \sqcup_{\ell} \mathfrak{T}_{\mathcal{SO}_0, \ell}^{pl}$ and $\mathfrak{T}_{\mathcal{SO}_0}^{(n), pl} = \sqcup_k \mathfrak{T}_{\mathcal{SO}_0, k(n-1)+1}^{pl}$ denote, respectively, the sets of binary and of n -ary rooted trees with a choice of planar embedding.

Lemma 4.5. *For any given $n \geq 3$, and for $\ell = k(n-1) + 1$, for any $k \geq 2$, we have*

$$\#\mathfrak{T}_{\mathcal{SO}_0, \ell}^{pl} > \#\mathfrak{T}_{\mathcal{SO}_0, \ell}^{(n), pl}.$$

Proof. The number of planar rooted binary trees with $\ell = r + 1$ leaves (hence r non-leaf vertices) is given by the Catalan number

$$C_r = \frac{1}{r+1} \binom{2r}{r}.$$

Thus, for $\ell = k(n-1) + 1$, we have the counting

$$C_{k(n-1)} = \frac{1}{(n-1)k+1} \binom{2k(n-1)}{k}.$$

The number of planar rooted n -ary trees with $(n-1)k + 1$ leaves (hence k non-leaf vertices) is correspondingly given by the Fuss–Catalan numbers

$$C_k^{(n)} = \frac{1}{(n-1)k+1} \binom{nk}{k}.$$

The different assignments of labels at the leaves contribute in both cases a factor $S^{(n-1)k+1}$, where $S := \#\mathcal{SO}_0$. When we compare the counting we see that

$$(4.5) \quad S^{(n-1)k+1} (C_{k(n-1)} - C_k^{(n)}) = \frac{S^{(n-1)k+1}}{(n-1)k+1} \left(\binom{2k(n-1)}{k} - \binom{nk}{k} \right) > 0$$

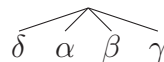
since $2k(n-1) > nk$ for $n \geq 3$. □

Thus, at the level of planar trees, counting detects an undergeneration phenomenon which is present at all levels $k \geq 1$ of the construction of the sets of syntactic objects. This phenomenon shows that there are always strings of elements of \mathcal{SO}_0 of length $k(n-1) + 1$ that have ambiguous parsing when realized in terms of binary Merge, while the ambiguity cannot be accounted for with an n -ary Merge.

As a simple example of this type of undergeneration, the two different parsings of the ambiguous sentence “I saw someone with a telescope” depend on the difference between the two binary trees



which would disappear entirely if the terms $\alpha, \beta, \gamma, \delta$ are assembled through a 4-ary Merge to form the tree



where the ambiguity would no longer be detectable.

4.3. The structure of a hypothetical n -ary Merge. Given the set $\mathcal{SO}^{(n)}$ of syntactic objects associated to a hypothetical n -ary Merge, obtained as in (4.3), we can consider the same type of action of Merge on workspaces that we have introduced above for a binary Merge. We will see in §4.4 below that, when the same structure is implemented through an n -ary Merge with $n \geq 3$, it inevitably leads to an overgeneration phenomenon.

As in the binary case, we introduce the set of workspaces as finite collections of syntactic objects, which in the n -ary case are elements of the set $\mathcal{SO}^{(n)} \simeq \mathfrak{T}_{\mathcal{SO}_0}^{(n)}$ of n -ary non-planar rooted trees. We again consider the vector space $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{(n)})$, where $\mathfrak{F}_{\mathcal{SO}_0}^{(n)}$ is the set of finite forests with connected components in $\mathfrak{T}_{\mathcal{SO}_0}^{(n)}$. In order to write the extraction of accessible terms and the cancellation of copies in the form of a coproduct, and the Merge pairing on accessible terms, we consider the relevant algebraic structure on $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{(n)})$, namely the product given by disjoint union \sqcup and the coproduct as in (2.10) and (2.11).

Remark 4.6. In defining a coproduct of the form (2.10) for an n -ary tree, one no longer has the option of taking the quotient T/T_v in the sense of Definition 2.5, as after removal of the subtree T_v , contractions of edges in the resulting tree $T \setminus T_v$ will produce vertices with either less or more than n descendants. In order to have a quotient T/T_v that is itself an n -ary tree, one can define T/T_v as obtained by contracting T_v to its root vertex. This requires that the root vertex of T_v , which becomes a leaf in T/T_v , need to be labelled by an element in \mathcal{SO}_0 . This requires including in \mathcal{SO}_0 syntactic features of the form XP with $X \in \{N, V, A, P, C, T, D, \dots\}$, and the label of the new leaf in T/T_v , obtained by projection, needs to be computed by inspecting the structure of T_v . This computation of labels of root vertices can be avoided in the case of binary Merge, by performing quotients as in Definition 2.5, but can no longer be avoided in the case of n -ary Merge with $n \geq 3$, where the quotient needs to be taken by contraction to the root vertex.

We can assume that the form of the action of Merge on workspaces will be of the same form as in the binary case of (2.23). Thus, we can write the desired form for the n -ary Merge action on workspaces as follows.

Given a collection $S = (S_i)_{i=1}^n$ of n -ary syntactic objects $S_i \in \mathfrak{T}_{\mathcal{SO}_0}^{(n)}$, we also define an operator

$$\delta_{S_1, \dots, S_n} : \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{(n)}) \otimes \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{(n)}) \rightarrow \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{(n)}) \otimes \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{(n)})$$

in the same way as the $\delta_{S, S'}$ defined in the binary case in (2.18), (2.19), (2.20). As in (2.17) we set

$$(4.6) \quad \mathfrak{F}_{\mathcal{SO}_0}^{\Delta, (n)} = \{(F_1, F_2) \in \mathfrak{F}_{\mathcal{SO}_0}^{(n)} \times \mathfrak{F}_{\mathcal{SO}_0}^{(n)} \mid \exists F \in \mathfrak{F}_{\mathcal{SO}_0}^{(n)}, F_{\underline{v}} \subset F : F_1 = F_{\underline{v}} \text{ and } F_2 = F/F_{\underline{v}}\}.$$

We then set

$$(4.7) \quad \delta_{S_1, \dots, S_n}(F_1 \otimes F_2) = 0 \quad \text{for } (F_1, F_2) \notin \mathfrak{F}_{\mathcal{SO}_0}^{\Delta, (n)},$$

$$(4.8) \quad \delta_{S_1, \dots, S_n}(F_{\underline{v}}, F/F_{\underline{v}}) = S_1 \sqcup \dots \sqcup S_n \otimes T_{a_1}/S_1 \sqcup \dots \sqcup T_{a_n}/S_n \sqcup F^{(a_1, \dots, a_n)},$$

for $F = \sqcup_{i \in \mathcal{I}} T_i$, if there are indices $a_1, \dots, a_n \in \mathcal{I}$ such that $S_i \simeq T_{a_i, v_i}$, and with

$$F^{(a_1, \dots, a_n)} = \sqcup_{i \neq a_1, \dots, a_n} T_i.$$

As in the binary case, if there is more than one choice of indices a_1, \dots, a_n for which matching terms $S_i \simeq T_{a_i, v_i}$ exist, then the right-hand-side of (4.8) should be replaced by the sum over all the possibilities, which we do not write out explicitly. In the remaining case where such matching of terms does not exist one sets

$$(4.9) \quad \delta_{S_1, \dots, S_n}(F_{\underline{v}}, F/F_{\underline{v}}) = 1 \otimes F.$$

Definition 4.7. *The action of Merge on workspaces consists of a collection of operators*

$$\{\mathfrak{M}_{S_1, \dots, S_n}\}_{S'_i \in \mathfrak{T}_{\mathcal{SO}_0}^{(n)}}, \quad \mathfrak{M}_{S_1, \dots, S_n} : \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{(n)}) \rightarrow \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{(n)}),$$

parameterized by n -tuples $(S_i)_{i=1}^n$ of n -ary syntactic objects, which act on $\mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{(n)})$ by

$$(4.10) \quad \mathfrak{M}_{S_1, \dots, S_n} = \sqcup \circ (B^+ \otimes \text{id}) \circ \delta_{S_1, \dots, S_n} \circ \Delta,$$

with the same operation B^+ as in Definition 2.8.

Note that the n -ary analog of Lemma 2.9 also holds, so that (4.10) is obtained analogously.

This action of Merge on workspaces has the same structure as in the binary case, namely, for each of the n input of the n -ary Merge \mathfrak{M}_n a search is made over the workspace by extracting accessible terms and comparing them with the corresponding n -ary syntactic object S_i . Non-matching terms are left unchanged in the new workspace, while the n -ary Merge operation is applied to n -tuples of matching terms among the extracted accessible terms for each Merge input. The new workspace then have these Merge outputs along with the terms coming from the quotient part of the coproducts, where cancellation of the deeper copies of the accessible terms used by Merge is performed.

One can envision other possible generalizations of the binary Merge action on workspaces to the n -ary case, using a coproduct with higher arity instead of Δ . We will not discuss them here, since (4.10) is the simplest direct generalization of (2.23), and it suffices to show the inevitability of overgeneration (which would occur for the same reasons in other generalizations as well).

4.4. Overgeneration. We can now see the overgeneration phenomenon as a different type of comparison between the sets \mathcal{SO} and $\mathcal{SO}^{(n)}$, with respect to the undergeneration discussed above. Unlike undergeneration, overgeneration depends not only on the structure of the set $\mathcal{SO}^{(n)}$ of syntactic objects, but also on the action of on workspaces as described above.

Indeed, consider the following empirical linguistic example of overgeneration by a hypothetical ternary Merge. We take a workspace given by an n -ary forest of the form

$$F = \{\alpha, \beta, \gamma\} \sqcup \delta \sqcup \eta,$$

with $\alpha, \beta, \gamma, \delta, \eta \in \mathcal{SO}^{(3)} \simeq \mathfrak{T}_{\mathcal{SO}_0}^{(3)}$. Consider the action of ternary Merge on workspaces described by (4.10) with $n = 3$ and with $S = (S_1, S_2, S_3)$ given by $S_1 = \alpha$, $S_2 = \beta$, and $S_3 = \{\alpha, \beta, \gamma\}$ gives the internal Merge

$$\mathfrak{M}_{S_1, S_2, S_3}(F) = \{\alpha, \beta, \{\alpha, \beta, \gamma\}\} \sqcup \delta \sqcup \eta.$$

Similarly, the same action with $S_1 = \delta$, $S_2 = \eta$, and $S_3 = \{\alpha, \beta, \gamma\}$ gives the external Merge

$$\mathfrak{M}_{S_1, S_2, S_3}(F) = \{\delta, \eta, \{\alpha, \beta, \gamma\}\}.$$

These ternary Merge operations are responsible for generating ungrammatical sentences such as⁴ *peanuts monkeys children will throw* (as opposed to *children will throw monkeys peanuts*), resulting from

$$(4.11) \quad \{\text{peanuts, monkeys, \{children, will, \{throw, monkeys, peanuts\}\}}\}.$$

In the example of (4.11) one sees that α and β are accessible terms of $\{\alpha, \beta, \gamma\}$, hence with a ternary Merge one can form $\{\alpha, \beta, \{\alpha, \beta, \gamma\}\}$. On the other hand, $\{\alpha, \beta\}$ is not an accessible term of $\{\{\alpha, \gamma\}, \beta\}$.

This example indicates that the overgeneration phenomenon it illustrates is caused by a difference in the size of the sets of accessible terms on which the action of Merge on workspaces

⁴This example was communicated to us by Riny Huijbregts. For a more detailed discussion, see [28].

is based. Indeed, in the general case of an arbitrary hypothetical n -ary Merge with $n \geq 3$, the overgeneration phenomenon is caused by the simple fact that, given a binary tree and an n -ary tree with the same set of leaves, there are fewer pairs of accessible terms (input for binary Merge) in the binary tree than there are n -tuples of accessible terms (input for the n -ary Merge) in the n -ary tree. We can see this more explicitly as follows.

Lemma 4.8. *Let $V_{int}^o(T)$ denote the set of vertices that are neither leaves nor root. Suppose given a set of leaves L with $\#L = \ell = k(n-1) + 1$, for some $k \geq 1$. Let T and T' be, respectively, a binary and an n -ary tree with $L(T) = L(T') = L$. Then for $k \geq n$ we have*

$$(4.12) \quad \#(V_{int}^o(T')^{n-1} \setminus \text{Diags}) > \#V_{int}^o(T),$$

where $\text{Diags} \subset V_{int}^o(T')^{n-1}$ is the union of all the diagonals, where two or more of the entries in $(v_1, \dots, v_{n-1}) \in V_{int}^o(T')$ coincide.

Proof. A binary tree T with ℓ leaves has $\ell - 1$ non-leaf vertices, a total of $2\ell - 1$ vertices, and $2(\ell - 1)$ non-root vertices. An n -ary tree on the same set of leaves with $\ell = k(n-1) + 1$ has k non-leaf vertices, a total of $kn + 1$ vertices, and kn non-root vertices. Note that, in order to have $n - 1$ choices without repetitions in $V_{int}^o(T')$ we need to assume that $k \geq n$. In (4.12) we are then comparing $\#V_{int}^o(T) = k(n-1)$ with

$$\#(V_{int}^o(T')^{n-1} \setminus \text{Diags}) = k(k-1) \cdots (k-n+1) = n! \binom{k}{n} = \frac{k!}{(k-n)!},$$

which is larger than $k(n-1)$. □

We include the leaf-vertices in the counting of accessible terms. The result of Lemma 4.8 is similar.

Corollary 4.9. *If $V_{int}(T)$ is the set of all non-root vertices, then*

$$(4.13) \quad \#(V_{int}(T')^{n-1} \setminus \text{Diags}) > \#V_{int}(T),$$

Proof. Note that, unlike in Lemma 4.8, now $k \geq 1$ is arbitrary. By the same counting as above of non-root vertices, in this case we have $\#V_{int}(T) = 2k(n-1)$ and $\#V_{int}(T') = kn$. In particular, $\#V_{int}(T) > \#V_{int}(T')$, but when counting inputs for internal Merge we obtain

$$\#(V_{int}(T')^{n-1} \setminus \text{Diags}) = kn(kn-1) \cdots (kn-n+1) = n! \binom{kn}{n} = \frac{(nk)!}{(n(k-1))!}$$

which is now larger than $\#V_{int}(T)$. □

The left-hand-side of (4.12) (respectively, (4.13)) is the size of the set of possible inputs for an n -ary internal Merge that can be extracted from the n -ary tree T' , while the right-hand-side of (4.12) (respectively, (4.13)) is the size of the set of all possible inputs for a binary internal Merge that can be extracted from the binary tree T , with the same set of leaves. The discrepancy between these two sizes shows the inevitable presence of overgeneration with an n -ary Merge and quantifies precisely the amount of overgeneration that can occur.

5. A MODEL OF EXTERNALIZATION

The action of Merge on workspaces described in (2.23) and Definition 2.10 can be also interpreted as a representation of a non-associative algebra in the following way. (All vector spaces and algebras are taken over \mathbb{Q} .)

First observe that the magma structure on $\mathcal{SO} = \mathfrak{T}_{\mathcal{SO}_0}$ of (2.1) gives to the vector space $\mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0})$ the structure of a non-associative commutative algebra, see [25], [26], where the binary Merge operation \mathfrak{M} gives the product operation.

Note that the coproduct (2.10) does not induce a bialgebra structure on $\mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})$ because it does not satisfy the compatibility:

$$\Delta \circ \mathfrak{M} \neq (\mathfrak{M} \otimes \mathfrak{M}) \circ \tau \circ (\Delta \otimes \Delta),$$

unlike the compatibility of \sqcup and Δ on $\mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})$ in Lemma 2.7. The reason is because $\Delta(\mathfrak{M}(T, T'))$ has only terms of the form $\mathfrak{M}(T_v, T') \otimes T/T_v$, $\mathfrak{M}(T, T'_w) \otimes T'/T'_w$, $T_v \otimes \mathfrak{M}(T/T_v, T')$, and $T'_w \otimes \mathfrak{M}(T, T'/T'_w)$, while the right-hand-side applied to $T \otimes T'$ also has all terms of the form $\mathfrak{M}(T_v, T'_w) \otimes \mathfrak{M}(T/T_v, T'/T'_w)$. However, a modified form of the coproduct (2.10) does give $\mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})$ the structure of a non-associative, commutative, co-commutative, co-associative Hopf algebra (see [25], [26]), with

$$(5.1) \quad \Delta(T) = \sum_{L \subset L(T)} T|_L \otimes T|_{L^c},$$

where, for a subset $L \subset L(T)$ (with $L^c = L(T) \setminus L$) we write $T|_L$ to denote the binary rooted tree obtained by removing all the leaves in L and then performing the edge contractions needed to obtain a binary tree. The difference between this coproduct and (2.10) lies in the fact that the coproduct of (5.1) would correspond to a notion of accessible terms that include all possible subsets of the set of leaves, not just those of the form $L = L(T_v)$.

Here we only need to consider the non-associative commutative algebra structure $\mathcal{A}_{na,c} = (\mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0}), \mathfrak{M})$, without the comultiplication, but the above remark is included for completeness.

5.1. Merge representation. The notion of representation and module over a non-associative algebra is much weaker than its associative counterpart. If \mathcal{A} is a non-associative algebra and \mathcal{V} is a vector space, an \mathcal{A} -module structure on \mathcal{V} is simply given by a *linear* map

$$\rho : \mathcal{A} \rightarrow \text{End}(\mathcal{V}).$$

This map is not an algebra homomorphism when \mathcal{A} is non-associative. We can equivalently view ρ as a linear map $\rho : \mathcal{A} \times \mathcal{V} \rightarrow \mathcal{V}$. We say that a vector space \mathcal{V} is a module over a non-associative algebra \mathcal{A} if it is endowed with a representation of \mathcal{A} on \mathcal{V} in the sense described here above.

Lemma 5.1. *The vector space $\mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})$ is a module over the algebra $\mathcal{A}_{na,c}$ through the representation given by the maps*

$$(5.2) \quad \rho(T)(F) = \sqcup \circ (\mathfrak{M}^T \otimes 1) \circ \Delta(F) = \sqcup_a(\mathfrak{M}(T, T_{a,v}) \sqcup T_a/T_{a,v}),$$

where $F = \sqcup_a T_a$ and $\mathfrak{M}^T(T_a) := \mathfrak{M}(T, T_a)$.

It then suffices to show that the representation (5.2) is enough to determine the Merge operators $\mathfrak{M}_{S,S'}$ as described in (2.23) in Definition 2.10. The form (2.23) of the action of Merge on workspaces is designed so as to exactly describe the procedure of searching among the accessible terms and syntactic objects of the workspace for copies of the chosen objects S and S' , for each of the two inputs of Merge, and applying Merge with corresponding cancellation of copies of accessible terms. The following observation shows that the same internal and external Merges can be also obtained through the somewhat simplified expression (5.2) of the representation of Lemma 5.1.

Lemma 5.2. *The representation (5.2) suffices to determined the Merge operations (2.23) on workspaces in $\mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})$.*

Proof. Consider the operator $\rho(T)$ of the representation (5.2) restricted to the subspace $\mathcal{V}_T \subset \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0})$ spanned by all the workspaces $F \in \mathfrak{F}_{S\mathcal{O}_0}$ that contain T either as a connected component of F or as an accessible term of one of the connected components. Then the result $\rho(T)(F)$ of applying the operator $\rho(T)$ to elements $F \in \mathcal{V}_T$ gives rise to all the possible Merge operations $\mathfrak{M}_{S,S'}$ with $S = T$ and with S' another component or accessible term of F . \square

This rephrasing of the action of Merge on workspaces in terms of algebras and modules has the advantage that it suggests a possible model for thinking about the process of externalization. This models how the core computational structure of Merge, when implemented in the human brain, needs to be followed by what one calls an “externalization procedure”, that allows for interaction with the sensorimotor system (see [3]). It is in this externalization process that additional constraints are imposed, such as the presence of a linear ordering on sentences (in the form of planar embeddings of binary rooted trees), as well as constraints coming from UG principles. One also needs to account for the syntactic diversity across different human languages (syntactic parameters), see for instance [19].

5.2. Externalization and linear ordering. We can look first at the step of externalization that introduces planar structures, hence linear ordering on the leaves of the trees, that is, an ordering on the resulting sentence. At first it may seem, intuitively, that introducing a linear ordering is a way of imposing a constraint and should therefore give rise to some kind of quotient map, in fact the quotient map goes in the opposite direction, as the map that identifies the abstract (non-planar) tree behind all its different planar embeddings. It can also be described as the quotient that maps non-commuting variables (where order matters) to corresponding commuting variables (where it does not). This means that one has a non-associative and non-commutative algebra $\mathcal{A}_{na,nc}$ together with a projection homomorphism $\mathcal{A}_{na,nc} \rightarrow \mathcal{A}_{na,c}$ that quotients out the commutators and identifies all different planar embeddings to the same abstract (non-planar) tree. The part of the externalization process that fixes a planar structures consists of the choice of a section of this projection morphism. Such a section is not an algebra homomorphism (as that would not map a commutative to a non-commutative algebra). Indeed, this is not surprising, as it is simply expressing the fact that the choice of planar embeddings cannot be universal and is in fact language-dependent, as it involves specific word order structures. Thus, the construction of this section of the projection $\mathcal{A}_{na,nc} \rightarrow \mathcal{A}_{na,c}$ is the first instance where one sees the role of syntactic parameters, in this case specifically in the form of word order parameters.

We denote, as before, by $\mathfrak{T}_{\mathcal{SO}_0}$ and $\mathfrak{F}_{\mathcal{SO}_0}$ the sets of binary rooted trees (respectively, forests) with leaves labels in \mathcal{SO}_0 , and we denote by $\mathfrak{T}_{\mathcal{SO}_0}^{pl}$ and $\mathfrak{F}_{\mathcal{SO}_0}^{pl}$ the corresponding sets of *planar* binary rooted trees (respectively, forests) with leaves labels in \mathcal{SO}_0 .

Proposition 5.3. *At the level of the underlying vector spaces, the quotient map $\mathcal{A}_{na,nc} \rightarrow \mathcal{A}_{na,c}$ is the map $\Pi : \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl}) \twoheadrightarrow \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0})$ that assigns to a planarly embedded tree the underlying abstract tree, forgetting the planar embedding, that is, identifying together all the different planar embeddings of the same abstract tree. There is a corresponding quotient map on workspaces*

$$\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl}) \twoheadrightarrow \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}).$$

The representation (5.2) extends to a representation $\rho^{pl} : \mathcal{A}_{na,nc} \rightarrow \text{End}(\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl}))$ so that the following diagram commutes

$$(5.3) \quad \begin{array}{ccc} \mathcal{A}_{na,nc} \otimes \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl}) & \xrightarrow{\rho^{pl}} & \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl}) \\ \downarrow \Pi \otimes \Pi & & \downarrow \Pi \\ \mathcal{A}_{na,c} \otimes \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}) & \xrightarrow{\rho} & \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}). \end{array}$$

Proof. The algebra $\mathcal{A}_{na,nc}$ is the free non-associative non-commutative algebra generated by the set \mathcal{SO}_0 with a non-associative non-commutative product, which we denote by \mathfrak{M}^{nc} . Unlike the non-associative commutative Merge product \mathfrak{M} of $\mathcal{A}_{na,c}$, we have in general $\mathfrak{M}^{nc}(\alpha, \beta) \neq$

$\mathfrak{M}^{nc}(\beta, \alpha)$. Thus, we can identify $\mathcal{A}_{na,nc}$ with the algebra associated to the non-associative non-commutative magma $\text{Magma}(\mathcal{SO}_0, \mathfrak{M}^{nc})$. We can identify the elements of this magma with *ordered* words in the alphabet \mathcal{SO}_0 with matched parentheses. Equivalently, we can describe the magma $\text{Magma}(\mathcal{SO}_0, \mathfrak{M}^{nc})$ through its Malcev representation, where a new variable c is introduced to mark the opening parenthesis. The position of the closing parenthesis is determined, so for example, instead of $(\alpha, ((\beta, \gamma), \delta))$ one can write $c\alpha c^2\beta\gamma\delta$, see [26]. The magma operation \mathfrak{M}^{nc} in the Malcev representation takes the form

$$\mathfrak{M}^{nc}(\alpha, \beta) = c\alpha\beta.$$

The set of ordered words in \mathcal{SO}_0 with matched parentheses can be identified with the set of binary rooted trees with a choice of planar embedding. Thus, we can identify

$$\mathfrak{T}_{\mathcal{SO}_0}^{pl} = \text{Magma}(\mathcal{SO}_0, \mathfrak{M}^{nc}).$$

In terms of the Malcev representation, the variable c marks the opening parenthesis that corresponds to an internal vertex of the planar tree in $\mathfrak{T}_{\mathcal{SO}_0}^{pl}$. The quotient map $\mathcal{A}_{na,nc} \rightarrow \mathcal{A}_{na,c}$ that kills the commutators has kernel the ideal generated by the elements $\mathfrak{M}^{nc}(T, T') - \mathfrak{M}^{nc}(T', T)$. Elements in this ideal are by construction differences between pairs of trees that differ in planar embeddings at one (or more) of the internal vertices, since every application of \mathfrak{M}^{nc} corresponds to an internal vertex of the resulting planar tree. Thus, the quotient map is exactly Π that identifies different planar embeddings of the same tree. Similarly, in $\mathfrak{F}_{\mathcal{SO}_0}^{pl}$ forests are now planarly embedded, hence the components T_a form an ordered set, which we describe by writing $F = \sqcup_a^{nc} T_a$, where \sqcup^{nc} means that the order of the T_a matters, namely \sqcup^{nc} is the union as planarly embedded trees, in a sequential order compatible with an ordering of the union of their leaves. By defining ρ^{nc} as

$$\rho(T)(F) = \sqcup^{nc} \circ (\mathfrak{M}^{T,nc} \otimes 1) \circ \Delta(F) = \sqcup_a^{nc} (\mathfrak{M}^{nc}(T, T_{a,v}) \sqcup T_a / T_{a,v})$$

with $F = \sqcup_a^{nc} T_a$ and $\mathfrak{M}^{T,nc}(T_a) := \mathfrak{M}^{nc}(T, T_a)$, one obtains compatibility as expressed by the commutativity of the diagram in the statement. \square

An assignment of a planar structure can then be seen as a section σ_L of the projection Π ,

$$(5.4) \quad \mathfrak{T}_{\mathcal{SO}_0}^{pl} \begin{array}{c} \xrightarrow{\sigma_L} \\ \xrightarrow{\Pi} \end{array} \mathfrak{T}_{\mathcal{SO}_0},$$

with $\Pi \circ \sigma_L = \text{id}$, where the section is dependent on a particular language L and exists as a map of vector spaces, but not as a morphism of algebras. These properties express the property that assignment of linear ordering of sentences is not directly generated by Merge itself, but requires an additional mechanism, and cannot be implemented in a universal language-independent way, see the discussion in §5.5.

5.3. Correspondences. There is another role for syntactic parameters in the model of externalization process we propose here, where they define a quotient map that significantly cuts down on the combinatorial explosion of Merge. In order to describe this process more precisely, it is useful to recall the mathematical notion of correspondence and how it generalizes the concept of function and mapping.

The notion of correspondence is a natural generalization of the concept of function or map, and has already played a crucial role in contemporary mathematics. It is generally understood that correspondences provide a better notion of morphisms than functions. In the case of a category

of geometric spaces (or the underlying category of sets) one typically replaces the usual notion of a function $f : X \rightarrow Y$ with correspondences that are of the form

$$(5.5) \quad \begin{array}{ccc} & Z & \\ & \swarrow & \searrow \\ X & & Y. \end{array}$$

The case of a function is recovered as the special case where $Z = G(f) \subset X \times Y$ is the graph of the function $G(f) = \{(x, y) \mid y = f(x)\}$, with the two projection maps to X and Y . Correspondences, however, are more general than functions. Given a correspondence Z , one can transfer structures (e.g. vector bundles, spaces of functions, etc.) from X to Y , by pulling them back to Z and then pushing them forward to Y via the two maps of the correspondence.

Thus, in this setting, given a category \mathcal{C} that has pullbacks, one can view correspondences as 1-morphisms in a 2-category of *spans* in \mathcal{C} , namely the 2-category $\text{Spans}(\mathcal{C})$ that has:

- objects given by the objects of \mathcal{C} ;
- 1-morphisms given by \mathcal{C} -diagrams of the form (5.5), with the composition given by the pullback

$$\begin{array}{ccccc} & & Z \times_Y Z' & & \\ & \swarrow & & \searrow & \\ & Z & & Z' & \\ \swarrow & & & & \searrow \\ X & & Y & & X' \end{array};$$

- 2-morphisms between spans $X \leftarrow Z_1 \rightarrow Y$ and $X \leftarrow Z_2 \rightarrow Y$ are morphisms $Z_1 \rightarrow Z_2$ in \mathcal{C} that give a commutative diagram

$$\begin{array}{ccc} & Z_1 & \\ & \swarrow & \searrow \\ X & & Y \\ & \swarrow & \searrow \\ & Z_2 & \end{array}$$

Since correspondences are usually described in this way as *spans* in the case of geometric spaces, they are usually described dually as *cospans* in the case of algebras, namely as diagrams of the form

$$\begin{array}{ccc} & \mathcal{E} & \\ & \swarrow & \nwarrow \\ \mathcal{A} & & \mathcal{B}. \end{array}$$

This construction further extends to the typical case where correspondences of algebras are defined as bimodules. However, one can also consider the case of correspondences (or co-correspondences) given by spans of algebras

$$\begin{array}{ccc} & \mathcal{E} & \\ & \swarrow & \searrow \\ \mathcal{A} & & \mathcal{B}. \end{array}$$

(and co-spans of spaces), with the composition $(\mathcal{B} \xleftarrow{g} \mathcal{E}' \rightarrow \mathcal{A}') \circ (\mathcal{A} \leftarrow \mathcal{E} \xrightarrow{f} \mathcal{B})$ given by the pullback, that is, the restricted direct sum $\mathcal{E} \oplus_{\mathcal{B}} \mathcal{E}' = \{(e, e') \mid f(e) = g(e')\}$. This is the kind of correspondences that we see in the description of the externalization of Merge.

5.4. Externalization as correspondence. The computational mechanism described by the action of Merge on workspaces encodes the fundamental computational structure of syntax, which is independent of the variation of syntactic structures across different languages. Where this variation actually occurs is only in the externalization process. At the level of the syntactic objects, given by the trees in $\mathfrak{T}_{\mathcal{SO}_0}^{pl}$, and of the workspaces, given by the forests in $\mathfrak{F}_{\mathcal{SO}_0}^{pl}$, the externalization that corresponds to a particular language L introduce quotient maps

$$(5.6) \quad \begin{aligned} \Pi_L : \mathfrak{T}_{\mathcal{SO}_0}^{pl} &\twoheadrightarrow \mathfrak{T}_{\mathcal{SO}_0}^{pl,L} \\ \Pi_L : \mathfrak{F}_{\mathcal{SO}_0}^{pl} &\twoheadrightarrow \mathfrak{F}_{\mathcal{SO}_0}^{pl,L}, \end{aligned}$$

where $\mathfrak{T}_{\mathcal{SO}_0}^{pl,L}$ and $\mathfrak{F}_{\mathcal{SO}_0}^{pl,L}$ are the set of planar binary rooted trees (respectively, forests) with leaves labels in \mathcal{SO}_0 , that are possible syntactic trees for the given language L . This quotient map very significantly reduces the combinatorial explosion of Merge, as only a small fraction of all the possible binary rooted trees generated by the Merge magma are realizable as syntactic trees of a specific given language. (We discuss in §5.5 below the role of syntactic parameters in determining the quotient map (5.6).)

In order to formulate this quotient at the level of the algebra $\mathcal{A}_{na,nc}$ and its action ρ^{pl} on the space $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl})$ of workspaces with planar structure, we need to use the notion of *partial algebra*, which is a vector space induced with a partially defined bilinear multiplication. Examples of partial algebras include the span of paths in a directed graph with the composition product.

Lemma 5.4. *The projection map of vector spaces $\Pi_L : \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl}) \twoheadrightarrow \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl,L})$ induced by the quotient map of (5.6) determines a non-associative non-commutative partial algebra $\mathcal{A}_{na,nc,L}$, with an induced action $\rho^{pl,L}$ of $\mathcal{A}_{na,nc,L}$ on $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl,L})$, with a commutative diagram*

$$(5.7) \quad \begin{array}{ccc} \mathcal{A}_{na,nc,L} \otimes \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl,L}) & \xrightarrow{\rho^{pl,L}} & \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl,L}) \\ \Pi_L \uparrow & & \uparrow \Pi_L \\ \mathcal{A}_{na,nc} \otimes \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl}) & \xrightarrow{\rho^{pl}} & \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl}). \end{array}$$

Proof. As a vector space, $\mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl,L})$ is spanned by those trees in $\mathfrak{T}_{\mathcal{SO}_0}^{pl}$ that are realizable as syntactic trees of the given language L , as such it can be viewed either as a quotient space, under the projection Π determined by (5.6), or as a subspace of $\mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl})$. This subspace $\mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl,L})$ is not a priori a subalgebra with respect to the Merge product \mathfrak{M}^{nc} . However, it is a partial algebra, where the induced Merge $\mathfrak{M}^{nc,L}$ acts as \mathfrak{M}^{nc} on the domain given by the set of pairs $T, T' \in \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl,L})$ with the property that $\mathfrak{M}^{nc}(T, T') \in \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl,L})$. This gives a non-associative, non-commutative partial algebra $\mathcal{A}_{na,nc,L} = (\mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl,L}), \mathfrak{M}^{nc,L})$. The vector space $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl,L})$ can similarly be regarded both as a quotient of $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl})$ under the quotient map Π_L or as a subspace. We can consider on $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl,L})$ a coproduct induced by the coproduct Δ of $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl})$, determined by setting

$$\Delta_L(T) = \sum_{v \in V_{int}(T): T_v, T/T_v \in \mathfrak{T}_{\mathcal{SO}_0}^{pl,L}} T_v \otimes (T/T_v).$$

The induced action of $\mathcal{A}_{na,nc,L}$ on $\mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0}^{pl,L})$ is given by

$$\rho^{pl,L}(T)(F) = \Pi_L \circ (\mathfrak{M}^{T,nc} \otimes 1) \circ \Delta_L(F),$$

and satisfies by construction the stated compatibility. \square

We then see that the combination of the procedure described in §5.2 for introducing linear ordering of sentences, with the quotient procedure that eliminates trees that are not realizable as syntactic trees of a specific language, describes the externalization process in the form of a correspondence, in the sense outlined in §5.3 above.

Definition 5.5. *Externalization is a correspondence given by the span of algebras (or partial algebras) and associated modules*

$$(5.8) \quad \begin{array}{ccc} & & \mathcal{A}_{na,nc,L} \otimes \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0}^{pl,L}) \xrightarrow{\rho^{pl,L}} \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0}^{pl,L}) \\ & \nearrow \Pi_L \otimes \Pi_L & \\ \mathcal{A}_{na,nc} \otimes \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0}^{pl}) & \xrightarrow{\rho^{pl}} & \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0}^{pl}) \xrightarrow{\Pi_L} \\ \downarrow \Pi \otimes \Pi & & \downarrow \Pi \\ \mathcal{A}_{na,c} \otimes \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0}) & \xrightarrow{\rho} & \mathcal{V}(\mathfrak{F}_{S\mathcal{O}_0}) \end{array} .$$

5.5. The role of syntactic parameters. In the Minimalist Model, where the core structure of syntax is described by the Merge operation of binary set formation, *syntactic parameters*, which account for syntactic variation across languages, become part of the externalization structure. The notion of syntactic parameters was originally introduced in the context of the Principles and Parameters model, [12], [13]. A recent extensive study of syntactic parameters can be found in [46].

For simplicity, we can assume that syntactic parameters are binary variables. This may not account for phenomena such as some kind of entailment relations between parameters, observed for example in [31], but it is still, to a large extent, accurate. We can describe the set of syntactic parameters as a subset $\mathcal{P} \subset \mathbb{F}_2^N$, where N is a (large) number of binary variables that record various syntactic features of languages, and the locus $\mathcal{P} \subset \mathbb{F}_2^N$ accounts for the set of “possible languages” (possible values of parameters that can be realized by actual human languages, see [41]). The set \mathcal{P} incorporates all the possible relations between parameters. One knows a significant number of relations is expected, for example through the geometric and topological data analysis techniques applied to databases of syntactic features, see for instance [23], [42], [44], [47]. The exact nature of these relations is not known, but one can hypothesize that \mathcal{P} may be realizable as an algebraic set (or algebraic variety) over \mathbb{F}_2 , embedded in the affine space \mathbb{F}_2^N . Regardless of any specific assumption on the geometry of the set \mathcal{P} , we have that a language L determines a corresponding point $\pi_L \in \mathcal{P}$, which is a vector $\pi_L \in \mathbb{F}_2^N$ that lists as entries the binary values of the N syntactic parameters for that particular language.

In the description of externalization proposed in §5.4, one expects that syntactic parameters will be involved in determining both the section σ_L of (5.4) and the projection Π_L of (5.6).

Since the first part of externalization, which corresponds to the section σ_L of (5.4), only depends on syntactic parameters that govern word order, while the projection Π_L of (5.6) depends on all other parameters, we can single out a subset of $M < N$ parameters that affect word-order. We denote by $q : \mathbb{F}_2^N \rightarrow \mathbb{F}_2^M$ the corresponding projection map that only keeps the word-order parameters, and we denote by $\bar{\mathcal{P}} = q(\mathcal{P})$ the image under this projection of the locus of parameters,

with $\bar{\pi} = q(\pi)$, for $\pi \in \mathcal{P}$. The parameters $q(\pi_L)_i$, $i = 1, \dots, M$ of a point $q(\pi_L)$ in this space $\bar{\mathcal{P}} \subset \mathbb{F}_2^M$ cut out a subset of $\mathfrak{T}_{\mathcal{SO}_0}^{pl}$ that consists of those planar structures for trees in $\mathfrak{T}_{\mathcal{SO}_0}$ that are compatible with the word-order properties of the given language L . These define the range of the section σ_L , and similarly for workspaces $\mathfrak{F}_{\mathcal{SO}_0}^{pl}$.

On the other hand, given the set of all planar binary trees and forests in $\mathfrak{T}_{\mathcal{SO}_0}^{pl}$ and $\mathfrak{F}_{\mathcal{SO}_0}^{pl}$, respectively, the syntactic parameters specified by the point $\pi_L \in \mathcal{P}$ have the effect of selecting which syntactic trees are realizable in the given language L , thus determining the sets $\mathfrak{T}_{\mathcal{SO}_0}^{pl,L}$ and $\mathfrak{F}_{\mathcal{SO}_0}^{pl,L}$. We can give the following geometric description of this procedure, which has the advantage that it allows for the possible use of tools from algebraic geometry to model more closely the externalization process. We give below some example of questions that can be naturally formulated in this mathematical framework.

As discussed in the vector spaces $\mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl})$ and $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl})$ are graded by number of leaves (sentence length),

$$(5.9) \quad \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl}) = \bigoplus_{\ell} \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl})_{\ell} \quad \text{and} \quad \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl}) = \bigoplus_{\ell} \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl})_{\ell},$$

with finite dimensional graded pieces.

Proposition 5.6. *Let \mathcal{L} denote the set of languages $L \in \mathcal{L}$. Let $\text{Gr}(d, n)$ denote the Grassmannian of d -dimensional linear subspaces in an n -dimensional space. The identification of the spaces $\mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl,L})$ and $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl,L})$ by specifying the syntactic parameters $\pi_L \in \mathbb{F}_2^N$ for a language L is described by a collection of maps*

$$(5.10) \quad E_{i,\ell} : \mathcal{P} \rightarrow \text{Gr}(d_{\pi_i,\ell}, d_{\ell}),$$

where for $\pi = (\pi_i)_{i=1}^N \in \mathcal{P}$, the image $E_{i,\ell}(\pi) \subset \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl})_{\ell}$ is the subspace spanned by the trees that are compatible with the constraints imposed by the value of the i th syntactic parameter π_i , with $d_{\ell} = \dim \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl})_{\ell}$. Thus, a point $\pi \in \mathcal{P}$ determines a subspace $E_{\ell}(\pi) = \bigcap_i E_{i,\ell}(\pi)$, and similarly for $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl})$. The assignment $\pi : \mathcal{L} \rightarrow \mathcal{P}$ of syntactic parameters to languages $L \mapsto \pi_L$ in turn determines $\mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl,L})$ as

$$(5.11) \quad \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl,L}) = \bigoplus_{\ell} E_{i,\ell}(\pi_L),$$

and similarly for $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl,L})$.

Proof. As in Lemma 5.4, we can view $\mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl,L})$ and $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl,L})$ as subspaces (rather than quotient spaces) of $\mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl})$ and $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl})$, respectively. Each syntactic parameter π_i of a point $\pi = (\pi_i)_{i=1}^N \in \mathcal{P} \subset \mathbb{F}_2^N$ determines a subspace $\mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl})_{\pi_i} \subset \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl})$ (respectively, $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl})_{\pi_i} \subset \mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl})$), such that, for $\pi = \pi_L$ for some language $L \in \mathcal{L}$

$$(5.12) \quad \bigcap_{i=1}^N \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl})_{\pi_{L,i}} = \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl,L}),$$

and similarly for the $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl})_{\pi_{L,i}}$. Given the graded structure (5.9), we can consider the procedure (5.12) of cutting out the subspaces $\mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl,L})$ and $\mathcal{V}(\mathfrak{F}_{\mathcal{SO}_0}^{pl,L})$ step by step by degrees. For a given $\ell \in \mathbb{N}$, there are integers $c_{\pi_i,\ell}$, $i = 1, \dots, N$ that specify the codimensions of the subspaces

$$\mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl})_{\pi_i,\ell} \subset \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl})_{\ell}.$$

If $d_{\ell} = \dim \mathcal{V}(\mathfrak{T}_{\mathcal{SO}_0}^{pl})_{\ell}$ with $d_{\pi_i,\ell} = d_{\ell} - c_{\pi_i,\ell}$ the dimensions, we then have maps

$$E_{i,\ell} : \mathcal{P} \rightarrow \text{Gr}(d_{\pi_i,\ell}, d_{\ell})$$

to the Grassmannian of $d_{\pi_i, \ell}$ -dimensional subspaces inside the d_ℓ -dimensional space $\mathcal{V}(\mathfrak{T}_{S\mathcal{O}_0}^{pl})_\ell$, so that

$$E_{i, \ell}(\pi_L) = \mathcal{V}(\mathfrak{T}_{S\mathcal{O}_0}^{pl})_{\pi_{L,i}} \in \text{Gr}(d_{\pi_{L,i}, \ell}, d_\ell),$$

and similarly for $\mathcal{V}(\mathfrak{T}_{S\mathcal{O}_0}^{pl})_\ell$. Similarly, if $d_{L, \ell} = \dim \mathcal{V}(\mathfrak{T}_{S\mathcal{O}_0}^{pl, L})_\ell$ is the dimension of the resulting intersection (which need not be transversal due to relations between syntactic parameters), the vector $\pi_L \in \mathbb{F}_2^N$ of parameters for the language L determines a map

$$E_\ell \circ \pi : \mathcal{L} \rightarrow \bigcup_d \text{Gr}(d, d_\ell) \quad E_\ell(\pi_L) = \mathcal{V}(\mathfrak{T}_{S\mathcal{O}_0}^{pl, L})_\ell \in \text{Gr}(d_{L, \ell}, d_\ell),$$

with $\mathcal{V}(\mathfrak{T}_{S\mathcal{O}_0}^{pl, L}) = \oplus_\ell E_\ell(\pi_L)$. □

There are natural geometric questions that this viewpoint suggests. For instance, when comparing the syntax of different languages $L, L' \in \mathcal{L}$, one can consider the resulting comparison between the systems of subspaces $\mathcal{V}(\mathfrak{T}_{S\mathcal{O}_0}^{pl, L}) = \oplus_\ell E_\ell(\pi_L)$ and $\mathcal{V}(\mathfrak{T}_{S\mathcal{O}_0}^{pl, L'}) = \oplus_\ell E_\ell(\pi_{L'})$. Syntactic proximity can be viewed in terms of the geometric position of these subspaces. For example, mathematically a special case of pairs E, E' of infinite dimensional subspaces inside an infinite dimensional space \mathcal{V} is given by the Fredholm pairs, where the intersection $E \cap E'$ is finite dimensional and the span of the union $E \cup E'$ has finite codimension. These would represent the situation of maximal differentiation. Moreover, there are models of semantics based on the geometry of Grassmannians, [35], and one can consider in this context the possibility of algebro-geometric models of a syntactic-semantic interface.

6. MERGE AND FUNDAMENTAL COMBINATORIAL RECURSIONS IN PHYSICS

In classical physics, a “least action principle” governs the solutions of equations of motion of physical systems, in the form of minimization (or stationarity) of the action functional, namely a minimization with respect to energy. The equations of motion are then expressed as the Euler–Lagrange equations that describe the stationarity of the action functional under infinitesimal variation. In quantum physics, and more precisely in quantum field theory, the classical equations of motion become equations in the quantum correlation functions of the fields (see [43], §9.6). More precisely, the Euler–Lagrange equations are satisfied by the Green functions of the quantum field theory, up to terms that reflect the noncommutativity of field operators. The resulting quantum equations of motion are known in physics as Dyson–Schwinger equations. They represent the optimization process of the least action principle, implemented at the quantum level.

Quantum physics, in the form of perturbative quantum field theory, is governed by a combinatorial generative process that determines the terms of the perturbative expansion. The combinatorial objects involved are the Feynman graphs of the theory, and the generative process can be described either by formal languages (in the form of graph grammars, see [39]) or in a more efficient way in terms of Hopf algebras (the Connes–Kreimer Hopf algebras of Feynman graphs and of rooted trees, see [15], [16]).

These two different descriptions of the generative process that produces the Feynman graphs of quantum field theory can be compared to what happens with older formulations of the Minimalist Model in generative linguistics, where one can give both a formal languages description (see [49]) and a description in terms of (internal/external) Merge operators, where the latter is computationally significantly more efficient (see [2]).

We discuss in a separate companion paper [38] how to compare older versions of the Minimalist Model to the new version of [7], [8] that we analyzed in this paper, at the level of the Hopf algebra structure, and how one sees in those terms the advantage of the more recent formulation.

Here the main point we want to stress is that, in the setting of quantum physics, the best description of the generative process of the hierarchy of Feynman graphs organized by increasing loop number in the asymptotic expansion is also determined by a Hopf algebra. There are two main advantages of this algebraic formalism in physics:

- (1) The algebraic structure governs the construction of the quantum solutions of the equations of motion, through the Dyson–Schwinger equations recalled above, so that solutions can be constructed through a combinatorial recursive procedure.
- (2) The Hopf algebra formalism also transparently explains the renormalization process in physics (namely the elimination of infinities, that is, the consistent extraction of finite (meaningful) values from divergent Feynman integrals).

We will discuss more in detail here the role of the algebraic formalism in quantum field theory in the recursive construction of solutions to the Dyson–Schwinger equations, as this is the aspect that is more closely related to the properties of Merge that we discussed in the previous sections. We will only sketch briefly in §6.2 the possible relevance of the algebraic formulation of renormalization to linguistics models, as we plan to return to discuss that in a separate paper.

6.1. The recursive construction of Dyson–Schwinger equations. In the physics of renormalization in quantum field theory, the generative process for the hierarchical structure of Feynman graphs is described equivalently by the Connes–Kreimer Hopf algebra of Feynman graphs mentioned above [15], or by a Hopf algebra of planar rooted trees (not necessarily binary), where the tree structure describes the way in which subgraphs are nested inside Feynman graphs (see [21], [30]). When formulated in terms of the Hopf algebra of trees, one can obtain a recursive construction of the solutions of the equations of motion of the quantum system, the Dyson–Schwinger equations, in terms of the combinatorics of trees, see [1], [20], [51].

This happens in the following way, as we outlined briefly in §2.3 and §3. The Hopf algebra \mathcal{H} of planar rooted trees and forests has product given by disjoint union and coproduct given by

$$\Delta(T) = \sum_C \pi_C(T) \otimes \rho_C(T),$$

where the left-hand-side $\pi_C(T)$ of the coproduct is a forest obtained by cutting subtrees of T using an “admissible cut” (not two cut legs on the same path from root to leaves) and the right-hand-side $\rho_C(T)$ is the tree that remains attached to the root when the cut is performed. Note that this is a form of the coproduct that we also used in (2.10).

One defines an operators $B^+ : \mathcal{H} \rightarrow \mathcal{H}$, as in Definition 2.8. Namely, B^+ acts on a forest $T_1 \sqcup \dots \sqcup T_m$ by creating a new rooted tree T where all the roots v_{r_1}, \dots, v_{r_m} of the trees T_1, \dots, T_m are attached to a single new root vertex,

$$B(T_1 \dots \dots T_m) = T = \begin{array}{c} \\ \diagup \\ T_1 \quad T_2 \quad \dots \quad T_n \end{array} .$$

As we observed in §2.3 and §3, this has exactly the structure of a Merge operator (though not necessarily binary, as it can take an arbitrary number of input trees). The operator B^+ satisfies the identity

$$(6.1) \quad \Delta(B^+(X)) = B^+(X) \otimes 1 + (Id \otimes B^+) \circ \Delta(X),$$

for all $X \in \mathcal{H}$. This identity is the Hochschild 1-cocycle condition (see [15], [1], [21]).

The combinatorial Dyson–Schwinger equation then takes the form of a fixed point equation

$$(6.2) \quad X = B^+(P(X)),$$

where $X = \sum_{k \geq 1} x_k$ is a formal series of elements $x_k \in \mathcal{H}_k$ in the graded pieces of the Hopf algebra, and $P(t) = \sum_{k \geq 0} a_k t^k$ with $a_0 = 1$ is a formal power series (or polynomial). The simplest and most fundamental such equation is the case where P is quadratic, $P(X) = X^2$, which is the form that we have encountered in (3.1), which governs the generative process of the core structure of Merge, discussed in §3. The equation (6.2) has a unique solution $X = \sum_{k \geq 1} x_k$ ([1], [21]) that can be written in the recursive form

$$(6.3) \quad x_{n+1} = \sum_{k=1}^n \sum_{j_1 + \dots + j_k = n} a_k B^+(x_{j_1} \cdots x_{j_k}),$$

with initial step $x_1 = B^+(1)$. It is shown in [1], [21] that the cocycle property (6.1) of the operator B is required to ensure that the coordinates x_n of the solution of a Dyson–Schwinger equation determine a Hopf subalgebra, though the construction of the solution (6.3) itself does not require the cocycle condition (6.1). In the case of the linguistic Merge the basic combinatorial structure is the same, with the recursion (6.3) corresponding to the core generative process of Merge, as we described in §3.

For a short overview of how this kind of combinatorial Dyson–Schwinger equations recover the physical equations of motion in quantum field theory, see [50]. For a more detailed treatment of combinatorial Dyson–Schwinger equations see [51]. A discussion of the use of Dyson–Schwinger equation in the context of the theory of computation is given in [18], following the approach to Renormalization and Computation developed in [32], [33].

As we mentioned at the beginning of this section, the classical Euler–Lagrange equations of motion express an optimality process given by a least action principle, and the quantum equations of motion given by the recursively solved Dyson–Schwinger equations, reflect this form of optimization in the quantum setting. In this sense the core computational structure of syntax defined by the Merge operator can be seen as being optimal and most fundamental, as it reflects the structure of the physical Dyson–Schwinger equation (for the appropriate Hopf algebra) and for the most basic (quadratic) form of the recursion.

One can ask whether there are any other characterizations of the syntactic Merge by optimality. The optimization is usually done with respect to some real-valued cost functional (energy/action in the case of physical systems). There are also other ways of thinking of optimization that do not require an evaluation through a function with values in real numbers. For example, it is possible to formulate optimization processes in a purely categorical framework (see for instance [37]), and an optimality property for the syntactic Merge may similarly take some more abstract categorical form. On the other hand it is also possible to consider minimization conditions with respect to other types of “action functional” that replace energy in the case of computational systems. For example, it is argued in [34] that complexity provides a suitable replacement for energy in the context of the theory of computation. We leave these questions to further investigation.

6.2. Algebraic renormalization and its relevance to linguistics models. The second aspect we mentioned above of the Hopf algebra formalism in physics is related to the mathematical treatment of renormalization in physics. The renormalization process in quantum field theory is a fundamental process that allows for the evaluation of Feynman integrals through a subtraction of infinities that is compatible with the hierarchical generative process of graphs (that is, with the subdivergences nested inside larger graphs). More precisely, one wants to extract a “meaningful” (finite) part out of the calculation of a priori divergent Feynman integrals, in such a way that this extraction of the “meaningful part” can be performed compatibly with what already assigned to subgraphs inside of larger graphs (that is, renormalization of sub-divergences, in physics terms).

An analogy with the linguistics setting immediately comes to mind, where one literally talks about assignment of meaning (semantics) to syntactic parsing trees of sentences, discarding possibilities that are ruled out by semantics. One similarly encounters the requirement that such assignment be done consistently across subtrees. For the moment this is purely an analogy, but one can see that this can be enriched with actual precise mathematical content, by considering an intermediate step between physics and linguistics, which is the extension of the formalism of renormalization and Hopf algebra from physics to the theory of computation, developed by Manin in [32], [33]. In this setting, one deals, as in physics, with a problem of subtraction of infinities, which arise not from divergence of Feynman integrals but from non-computability (in the form of divergence of computational time in the halting problem). A procedure of extraction of computable “subfunctions” from non-decidable problems is organized in terms of a Hopf algebra of flow charts (of algorithms) replacing the Hopf algebra of Feynman graphs. The role of Dyson–Schwinger relations in this context was analyzed in [18].

More precisely, in the mathematical formulation of renormalization (see [15], [16]), the (regularized) Feynman rules are described as a morphism of commutative algebras $\Phi : \mathcal{H} \rightarrow \mathcal{R}$, where \mathcal{H} is the Hopf algebra of Feynman graphs and \mathcal{R} is an algebra of functions where regularization of Feynman integrals takes place, such as Laurent series, with the structure of Rota–Baxter (RB) algebra of weight -1 . The RB structure captures the properties of the subtraction of the polar part of the Laurent series and more general it models a good process of subtraction of a divergent part. The coalgebra and antipode on \mathcal{H} , together with the Rota–Baxter operator on \mathcal{R} , determine a Birkhoff factorization of Φ into a part Φ_- that carries all the divergences and a part Φ_+ that gives the finite renormalized Feynman amplitudes (the actual physical quantities). This operation carries within itself the consistent assignment of the meaningful convergent part across subgraphs, with the coproduct of the Hopf algebra organizing the subgraphs.

We will not discuss any further in the present paper the linguistic analog for this Birkhoff factorization, although we can mention a possible interesting direction of investigation. A version of Birkhoff factorization taking place in semirings, with applications to the theory of computation was developed in [36], [40]. There are models of syntactic parsing and of semantics that are based on the same mathematical structure of semirings (see for instance [24], [45]). One can expect that it should be possible to construct a mathematical model of a syntactic-semantic interface that resembles the physical model of renormalization, with extraction of semantic meaning replacing the extraction of renormalized physical values. We will consider this problem elsewhere, as a separate paper.

Acknowledgments. We thank Riny Huijbregts and Sandiway Fong for providing extensive comments and very helpful feedback on a draft of this paper. The first author acknowledges support from NSF grant DMS-2104330 and FQXi grants FQXi-RFP-1 804 and FQXi-RFP-CPW-2014, SVCF grant 2020-224047, and support from the Center for Evolutionary Science at Caltech.

REFERENCES

- [1] C. Bergbauer, D. Kreimer, *Hopf algebras in renormalization theory: locality and Dyson-Schwinger equations from Hochschild cohomology*, in “Physics and Number Theory”, pp. 133–164, IRMA Lect. Math. Theor. Phys. 10, Eur. Math. Soc., 2006.
- [2] R.C. Berwick, *Mind the gap*, in “50 Years Later: Reflections on Chomsky’s Aspects”, pp. 1–13, MITWPL, 2015.
- [3] R.C. Berwick, N. Chomsky, *Why only us?* MIT Press, 2017.
- [4] R.C. Berwick, S. Epstein, *On the Convergence of ‘Minimalist’ Syntax and Categorical Grammar*, in “Algebraic Methods in Language Processing 1995”, pp. 143–148, Universiteit Twente, 1995.

- [5] N. Chomsky, *The Minimalist Program*, MIT Press, 1995.
- [6] N. Chomsky, *Problems of projection*, *Lingua*, Vol.130 (2013) 33-49.
- [7] N. Chomsky, *Some Puzzling Foundational Issues: The Reading Program*, *Catalan Journal of Linguistics Special Issue* (2019) 263–285.
- [8] N. Chomsky, *The UCLA lectures*, 2019. <https://ling.auf.net/lingbuzz/005485>
- [9] N. Chomsky, *Simplicity and the form of grammars*, *Journal of Language Modelling*, Vol.9 (2021) N.1, 5–15.
- [10] N. Chomsky, *Genuine explanation*, preprint, 2021.
- [11] N. Chomsky, *Working Toward the Strong Interpretation of SMT*, lecture series Theoretical Linguistics at Keio-EMU, 2023.
- [12] N. Chomsky, *Lectures on Government and Binding*, Dordrecht: Foris Publications, 1982.
- [13] N. Chomsky, H. Lasnik, *The theory of Principles and Parameters*, in “Syntax: An international handbook of contemporary research”, pp.506–569, de Gruyter, 1993.
- [14] N. Chomsky, T.D. Seely, R.C. Berwick, S. Fong, M.A.C. Huybregts, H. Kitahara, A. McInnerney, Y. Sugimoto, *Merge and the Strong Minimalist Thesis*, preprint, 2023.
- [15] A. Connes, D. Kreimer, *Hopf algebras, Renormalization and Noncommutative geometry*, *Comm. Math. Phys* 199 (1998) 203–242
- [16] A. Connes, M. Marcolli, *Noncommutative Geometry, Quantum Fields, and Motives*, Colloquium Publications, Vol.55, American Mathematical Society, 2008.
- [17] C.W. Coopmans, K. Kaushik, A.E. Martin, *Hierarchical structure in language and action: A formal comparison*, preprint, 2022.
- [18] C. Delaney, M. Marcolli, *Dyson-Schwinger equations in the theory of computation*, in “Feynman amplitudes, periods and motives”, pp. 79–107, *Contemp. Math.* 648, Amer. Math. Soc., 2015.
- [19] M.B.H. Everaert, M.A.C. Huybregts, N. Chomsky, R.C. Berwick, J.J. Bolhuis, *Structures, Not Strings: Linguistics as Part of the Cognitive Sciences*, *Trends in Cognitive Sciences*, Vol.19 (2015) N.12, 729–743.
- [20] L. Foissy, *Classification of systems of Dyson–Schwinger equations in the Hopf algebra of decorated rooted trees*, *Advances in Math.* 224 (2010) 2094–2150.
- [21] L. Foissy, *Les algèbres de Hopf des arbres enracinés, I*, *Bull. Sci. Math.* 126 (2002) 193–239.
- [22] S. Fong, R. Berwick, J. Ginsburg, *The combinatorics of merge and workspace right-sizing*, *Evolinguistics Workshop*, 2019.
- [23] S. Gakkhar, M. Marcolli, *Syntactic structures and the general Markov models*, arXiv:2104.08462.
- [24] J. Goodman, *Semiring parsing*, *Computational Linguistics*, Vol.25 (1999) N.4, 573–605.
- [25] R. Holtkamp, *Comparison of Hopf algebras on trees*, *Arch. Math. (Basel)* 80 (4) (2003) 368–383.
- [26] R. Holtkamp, *Rooted trees appearing in products and co-products*, in “Combinatorics and physics”, 153–169, *Contemp. Math.*, 539, Amer. Math. Soc., 2011.
- [27] R. Holtkamp, *A pseudo-analyzer approach to formal group laws not of operad type*, *J. Algebra*, Vol. 237 (2001) 382–405.
- [28] M.A.C. Huybregts, *Empirical cases that rule out Ternary Merge*, notes, 04/19/2021.
- [29] M. Komachi, H. Kitahara, A. Uchibori, K. Takita, *Generative procedure revisited*. *Reports of the Keio Institute of Cultural and Linguistic Studies*, 50 (2019) 269–283.
- [30] D. Kreimer, *On the Hopf algebra structure of perturbative quantum field theories*, *Adv. Theor. Math. Phys.* 2 (1998) N.2, 303–334.
- [31] G. Longobardi, C. Guardiano, *Evidence for syntax as a signal of historical relatedness*, *Lingua*, 119 (2009) 1679–1706.
- [32] Yu.I. Manin, *Renormalization and computation I: motivation and background*, in “OPERADS 2009”, 181–222, *Sémin. Congr.*, 26, Soc. Math. France, Paris, 2013.
- [33] Yu.I. Manin, *Renormalisation and computation II: time cut-off and the Halting problem*, *Math. Structures Comput. Sci.* 22 (2012), no. 5, 729–751.
- [34] Yu.I. Manin, *Complexity vs Energy: Theory of Computation and Theoretical Physics*, 3Quantum: Algebra Geometry Information (QQQ Conference 2012), *J. Phys.: Conf. Ser.* 532 (2014) 012018 [
- [35] Yu.I. Manin, M. Marcolli, *Semantic Spaces*, *Math. Comput. Sci.* 10 (2016), no. 4, 459–477.
- [36] M. Marcolli, *Information algebras and their applications*, in “Geometric Science of Information”, pp. 271–276, *Lecture Notes in Comput. Sci.*, 9389, Springer, 2015.
- [37] M. Marcolli, *Pareto optimization in categories*, arXiv:2204.11931.
- [38] M. Marcolli, R.C. Berwick, N. Chomsky, *Old and New Minimalism: a Hopf algebra comparison*, preprint 2023.
- [39] M. Marcolli, A. Port, *Graph grammars, insertion Lie algebras, and quantum field theory*, *Math. Comput. Sci.* 9 (2015) no. 4, 391–408.

- [40] M. Marcolli, N. Tedeschi, *Entropy algebras and Birkhoff factorization*, J. Geom. Phys. 97 (2015), 243–265.
- [41] A. Moro, *The Boundaries of Babel*, Second Edition, MIT Press, 2015.
- [42] A. Ortegaray, R.C. Berwick, M. Marcolli, *Heat kernel analysis of syntactic structures*, Math. Comput. Sci. 15 (2021), no. 4, 643–660.
- [43] M.E. Peskin, D.V. Schroeder, *An Introduction to Quantum Field Theory*, ABP 1995.
- [44] A. Port, T. Karidi, M. Marcolli, *Topological analysis of syntactic structures*, Math. Comput. Sci. 16 (2022), no. 1, Paper No. 2, 68 pp.
- [45] B. Roark, R. Sproat, *Computational Approaches to Morphology and Syntax*, Oxford University Press, 2007.
- [46] I. Roberts, *Parameter Hierarchies and Universal Grammar*, Oxford University Press, 2019.
- [47] K. Shu, M. Marcolli, *Syntactic structures and code parameters*, Math. Comput. Sci. 11 (2017), no. 1, 79–90.
- [48] E.P. Stabler, *Computational perspectives on minimalism*, in “Oxford Handbook of Linguistic Minimalism” (C. Boeckx, ed.), Oxford University Press, 2010, 616–641.
- [49] K. Vijay-Shanker, D. Weir, *The equivalence of four extensions of context free grammar formalisms*, Mathematical Systems Theory, 27 (1994) 511–545.
- [50] S. Weinzierl, *Hopf algebras and Dyson-Schwinger equations*, arXiv:1506.09119.
- [51] K. Yeats, *Rearranging Dyson-Schwinger Equations*, Memoirs of the American Mathematical Society, 211, American Mathematical Society, 2011.

DEPARTMENTS OF MATHEMATICS AND OF COMPUTING AND MATHEMATICAL SCIENCES, CALIFORNIA INSTITUTE OF TECHNOLOGY, PASADENA, CA 91125, USA

Email address: `matilde@caltech.edu`

DEPARTMENT OF LINGUISTICS, UNIVERSITY OF ARIZONA, TUCSON, AZ 85721, USA

Email address: `noamchomsky@email.arizona.edu`

Email address: `chomsky@mit.edu`

INSTITUTE FOR DATA, SYSTEMS, AND SOCIETY, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAMBRIDGE MA 02141, USA

Email address: `berwick@csail.mit.edu`