

# An Attempt at a Gentle Introduction to Hopf Algebras for Syntax\*

Avery Andrews  
ANU, Sep 27 2023

Marcolli et al. (2023b), henceforth MCB, and Marcolli et al. (2023a), henceforth MBC, use a kind of mathematical system called a Hopf algebra to present certain aspects of (recent) Minimalist syntactic theory in a new way. Hopf algebras are normally encountered by people at a considerably later stage of mathematical development than the average ‘ordinary working grammarian’<sup>1</sup> is likely to have attained, and that would most definitely include me. However, after a certain amount of poking around, it began to seem evident that they really were not that much different from polynomials in multiple variables as encountered in secondary school. So here I will attempt to present the basics, assuming secondary school algebra plus typical ‘math for linguists’ course content, including the rudiments of abstract algebra (groups, rings and fields, as terminology for things you have already encountered), basic set theory including functions, composition of functions, and cartesian products, isomorphisms and homomorphisms, and the  $\Sigma$  notation for sums.

There are four sections. In the first, we develop algebras as found in Hopf algebras as an elaboration of polynomials in SSA. In the second, we consider the tensors, in third, coalgebras, and begin on their integration with algebras to produce bialgebras. The fourth section finishes this, and waves a hand at what you need to add to a bialgebra to get a Hopf algebra, which the system developed in MCB already has. There is then an appendix detailing a few types in MCB.

A particularly useful source for further study is Federico Ardila’s lectures from 2012, with notes, homeworks and more at <https://fardila.com/Clase/Hopf/lectures>. and lectures starting at <https://www.youtube.com/watch?v=FzVhjCRuXus>.<sup>2</sup>

## 1 Polynomials in SSA vs. Algebras in MCB

Secondary school algebra (SSA) polynomials are made of numbers and variables, with operations of addition and multiplication, and the taking of additive and multiplicative inverses. In the contexts we will be using them

---

\*The major acknowledgement here so far would be to András Kornai for running a course on this topic over Zoom, and Blanka Kóvér for discovering the Ardila lectures, and also LaTeXing and giving a presentation on Tim Gowers’ very useful discussion of tensor products

<sup>1</sup>Term by Chris Potts <http://ordinaryworkinggrammarian.blogspot.com/2023/>.

<sup>2</sup>The webpage contains links to the lectures, but they don’t seem to load; get them from youtube.

in, the numbers would tend to be called ‘scalars’, and can be taken to be either the so-called ‘field’  $\mathbb{Q}$  of rational numbers, or ‘ring’  $\mathbb{Z}$  of integers.<sup>3</sup>

## 1.1 Scalar Rules

The operations on scalars are stipulated to obey a variety of laws, which are all true in terms of elementary school arithmetic:

- (1) Addition Laws (where  $a, b, c$  are arbitrary numbers/scalars):
  - a. Addition is commutative:  $a + b = b + a$
  - b. Addition is associative:  $(a + b) + c = a + (b + c)$
  - c. There is an ‘additive identity’ 0, such that  $a + 0 = a$
  - d. Every  $a$  has an ‘additive inverse’  $-a$ , such that  $a + (-a) = 0$ . Usually,  $a + (-b)$  is written  $a - b$ .

A system in which all of these laws, and possibly others, are obeyed is called a ‘commutative’, or ‘Abelian’ group.<sup>4</sup>

Moving on to multiplication, we find these laws:

- (2) Multiplication Laws (where  $a, b, c$  are arbitrary numbers/scalars):
  - a. Multiplication is commutative:  $ab = ba$
  - b. Multiplication is associative:  $(ab)c = a(bc)$
  - c. There is an ‘multiplicative identity’ 1, such that  $a1 = a$
  - d. Every  $a$  other than 0 has an ‘Multiplicative inverse’  $\frac{1}{a}$ , such that  $a\frac{1}{a} = 1$ . Usually,  $a\frac{1}{b}$  is written  $\frac{a}{b}$ .

So we see that the nonzero scalars also form an Abelian group.

And, finally, these two operations are tied together by the Distributive Law:

- (3) Distributive Law:  
$$a(b + c) = ab + ac$$

---

<sup>3</sup>MCB starts by talking about a vector space over  $\mathbb{Z}$ , which is terminologically wrong, because the equivalent of a vector space over a ring is a ‘module’, but later switches to  $\mathbb{Q}$ , while MBC refers to  $\mathbb{Q}$  from the beginning. I don’t think this makes any substantive difference at all, although it is certainly distracting to people who are not very developed as mathematicians.

<sup>4</sup>Non-commutative groups exist, and are extremely important, but some of the other laws need to be made more complicated to make up for the lack of commutativity.

Versions of this one play a large role in developments to come.

It is perhaps worth noting that I am being a bit sloppy about parentheses: somebody who was being more meticulous would say that all the operations put parentheses around the items operated on or combined, but that we leave off the outermost ones, and any others which make no difference to the result, given the laws. That is, we can write  $a+b+c$  instead of  $(a+(b+c))$  or  $((a+b)+c)$ , which are identical due to the associativity of addition.

Much of the subject matter of ‘abstract algebra’, covered by undergraduate math majors, often also to some extent in math for linguists courses), is about what happens when various of these laws are missing. Some more terminology:

- (4) a. If they are all present, we have a ‘field’. The fields you run into in SSA would be the rational numbers ( $\mathbb{Q}$ ), the real numbers, and perhaps the complex numbers. (These too have their fancy one-letter symbols, but they are irrelevant here.)
- b. If everything is present but perhaps multiplicative inverses, we have a ‘commutative ring with identity’, of which the most prominent example would be the integers ( $\mathbb{Z}$ ). So every field is also a commutative ring with identity.
- c. Losing commutativity of multiplication and the multiplicative identity, we get rings in general, not relevant here.

A useful consequence of the ring axioms is that  $0a = 0$  for any  $a$ , which can be proved by the following chain of equalities, each with its justification:

$$\begin{aligned} (5) \quad 0a &= (1 - 1)a && \text{additive inverse} \\ &= 1a - 1a && \text{distributive law} \\ &= a - a && \text{multiplicative identity} \\ &= 0 && \text{additive inverse} \end{aligned}$$

We’re not going to spend much time on proofs, but this is a useful one to have under your belt, and a basic abstract algebra course will introduce many more like this.

## 1.2 Vector Spaces

The next step in our climb is ‘vector spaces’, where we will start with something very familiar to most children, ‘concrete collections’, such as of Barbie dolls & accessories, Star Wars figures, etc (and, on a more adult level, collections of nuts, bolts and screws). The critical thing here is that you are not just interested in the number of objects in the collection, but the number of objects of each kind. 2 Luke Skywalkers & 1 Princess Leia, or whatever. Then we can plausibly describe the collection as a kind of sum of products of scalars and a kind of item:

$$(6) \ 2LS + 1PL + \dots$$

A practical reason for describing it in this way is that we might want to know how many figures of all kinds are in it, which we can ascertain by adding up the numerical coefficients. In math, such things tend to be called ‘formal sums’ (ignoring the scalar multiplications), and you can learn techniques for managing them abstractly from books on Universal Algebra,<sup>5</sup> but perhaps a term such as ‘typed sums’ would be better, since what they are about is counting the number of things of various kinds that are currently of interest. The idea of ‘sum’ becomes relevant when we group all the types under a supertype, and want to know how many items of that type the collection contains.

In fact, at this point, we almost have something called a ‘Module’, but for that, one more thing is needed: negative coefficients. These would not appear in descriptions of a normal child’s collection, but a more advanced and adventurous collector might do something like promise to provide a Han Solo Frozen in Carbonite at some future date, believing he could source a relatively cheap one. Before he has actually obtained it, we could describe his collection as involving a term  $-1HNFIC$ . We now have a ‘module over the (ring of) integers  $\mathbb{Z}$ ’. Modules over the integers share a lot of behavior with ‘vector spaces’, the difference being that the latter involve a field rather than a ring.

A real life example of a concrete type-based vector space as opposed to a module would be a collection of cooking ingredients, involving fractional amounts of various kinds of flour, etc, with perhaps some negative coefficients arising from the cook borrowing and using some ingredients that they will have to provide replacements, no substitutions allowed. Here the field could be the rational numbers  $\mathbb{Q}$ , since there is probably no concrete use for non-rational real amounts such as  $\sqrt{2}$  tablespoons of butter. I am not (yet) aware that negative coefficients play any role in the syntactic application of Hopf Algebras, but they do play a role in defining the ‘antipode’, the feature that distinguishes Hopf Algebras from the more general Bialgebras, as we will mention later. The antipode also has not been assigned any definite role in the syntactic application (so far, as far as I know).

So much for examples, now for a definition. For a module/vector space you need:

- (7) a. an abelian group (recall that group needs an additive identity and inverses for all). For a short time, we will put little arrow symbols over vectors, eg  $\vec{v}, \vec{w}$ .
- b. a ring/field, called ‘scalars’, usually represented here by  $r, s, t$ .

---

<sup>5</sup>Such as Burris & Sankappanavar (1981), or the more hardcore Grätzer (1978).

- c. There is a ‘scalar product’ such that for scalar  $r$  and vector  $\vec{v}$ ,  $r\vec{v}$  is a vector.
- d.  $0\vec{v} = \vec{0}$  (the identity element of the group)
- e.  $1\vec{v} = \vec{v}$
- f.  $(rs)\vec{v} = r(s\vec{v})$
- g.  $(s+r)\vec{v} = r\vec{v} + s\vec{v}$
- h.  $r(\vec{v} + \vec{w}) = r\vec{v} + r\vec{w}$

When all this holds, we say that we have a vector space/module over the field/ring. Note that (d) allows us to omit terms with 0 as scalar coefficient from the typed sums, and (e) allows us to leave off 1 as scalar coefficient. E.g.  $LS = 1LS$ .

A very important feature of vector spaces is that they have one or more ‘bases’, which are sets of elements such that:

- (8) a. Any element of the space can be expressed in a unique way as a formal sum of members of the basis.
- b. (a) ceases to hold if any member of the basis is dropped.

A basis for a typed sum vector space will be the number of different types that are available in it, since any collection can be expressed as a finite sum of scalar multiples of the types. For the workspace algebras to come, the basis will be infinite, while for Star Wars figures collections, it is presumably finite, at least at any one time (and will probably not grow indefinitely in the future).

And the most important fact about bases is that they all have the same number of elements, called the ‘dimensionality’ of the space. The proof of this is a significant part of courses on linear algebra, although how it is done in general is well beyond the scope of this presentation.

Another point is that any field or ring with identity is a vector space or module *over itself*, since all the laws of (8) hold. This is sometimes useful. (What will be the dimensionality of this space?) Observe also that the elements of a module or vector space do not cease to be such if there is also a way of multiplying them rather than just adding.

### 1.3 Algebras and Variables

And that is exactly what we do to get an ‘algebra’ (over a field or ring), as a term for a specific type of mathematical system, alongside of groups, rings etc., rather than a subject in the math curriculum). And the non-trivial algebra, in this sense, that most people encounter in secondary school is the

one you get by adding ‘variables’ to the system, to produce polynomials. When these are presented in school, it is not really all that clear (or at any rate, wasn’t clear to me), what they ‘really’ were, but it is clear that they are meant to be manipulated in accord with pretty much all of the rules applying to ordinary numbers, with the aim of reducing expressions to the simplest possible form that is useful for finding solutions to equations. Eg  $4 + y = 3x$  and be converted into  $y = 3x - 4$ , which can then be used to easily program a computer to produce a graph of solutions for some values of  $x$ . They are in fact genuine formal sums, with unrestricted rather than only scalar multiplication thrown in (and I cannot (yet) think of any concrete manifestation of them in everyday life and experience; note that multiplication of inventories, collections, etc. by each other does not make sense).

The basic idea of an algebra over a field or ring is that it is a system that is simultaneously a ring (not necessarily commutative) and a vector space or module. ‘Simultaneously’ means that we can multiply the vectors (if only formally), rather than just add them. So note that there will be two rings involved, the base ring/field, participating in scalar multiplication with the vectors, and the multiplication of the vectors. In SSA polynomial algebras, these look the same, but in the ‘workspace algebra’ proposed for syntax, they are completely different.

There are however a number of different definitions of algebras in circulation, whose relationships are discussed by Ardila (Lecture notes pg 7-9). For SSA polynomials, it is probably best to use his simplest/least flexible definition:

- (9) The vector space (or module)  $A$  is a  $K$ -algebra if
- a.  $K \subseteq Z(A)$ ,  $Z(A)$  being the subset of  $A$  for which multiplication with anything in  $A$  is commutative.
  - b.  $1_K = 1_A$  (the multiplicative identities of the ring and vector space are the same)

This is very natural, because we want to think of SSA polynomials as consisting of ordinary numbers with variables thrown in.

It might be useful however to consider them as vector spaces, to help develop Ardila’s second characterization of algebras, which is more suited to the ‘workspace polynomials’ found in MCB. Suppose we have one variable,  $x$ , and disallow multiplication of any expression containing a variable. Then we have a 2 dimensional space, because anything in it can be expressed as a sum of a scalar and a scalar multiplied by a vector, but losing either kind of element will leave somethings unexpressable. Then, if we allow multiplication of expressions containing variables, the dimensionality of the vector space goes to infinite, since we have expressions in  $x, x^2, x^3, \dots$ . But

we can also regard the expressions without any variable as actually having the variable component  $x^0 = 1$ , which makes at least some sense because we have a more uniform system and characterization of the basis:  $x$  with a non-negative exponent (and we could let in negative exponents if we wanted to).

This makes the vectors more different from the scalars, making a different characterization (Ardila's 2nd) natural, although not truly required:

(10) (The ring)  $A$  is a  $K$ -algebra if there is a ring homomorphism  $u: K \rightarrow A$  such that:

a.  $u(K) \subseteq Z(A)$

b.  $u(1_K) = 1_A$

In case you have forgotten the notation  $u: K \rightarrow A$ , it says that  $u$  is a function from  $K$  to  $A$ , and 'ring homomorphism' means that it preserves the addition and multiplication of the field (or commutative ring with identity)  $K$ . That is:

(11) a.  $u(r + s) = u(r) + u(s)$

b.  $u(rs) = u(r)u(s)$

We can now regard  $u$ , so named because it is the 'unit' of the algebra, as the map taking  $r$  to  $rx^0$ . It should be evident that we can apply this characterization to cases where we want to consider  $A$  as an extension of  $K$  by taking  $u$  to be the identity map, from  $K$ , a commutative subring of  $A$  that contains the identity, into all of  $A$ .

A slight subtlety is that to regard  $A$  as a vector space over  $K$ , we might want to define scalar multiplication as:

(12)  $r\vec{a} = u(r)\vec{a}$

where multiplication on the right is the one in  $A$ .

This lets us think of the scalars and vectors as different kinds of things, for which a multiplication does not exist by any kind of conceptual necessity.

## 1.4 Workspaces in Polynomials

In SSA, when variables are multiplied, they just sit next to each other in a formal combination with no 'substantive meaning', only that provided by the magic of equivalence classes in Universal Algebra, a topic we won't go into here.<sup>6</sup> Whereas, if we multiply two scalars, we get another scalar.

---

<sup>6</sup>Until, of course, we substitute numerical values for the variables, in which case they get added and multiplied in the normal way. Substitution is another complex topic we won't look into.

In MCB polynomials, instead of SSA variables, we have workspaces, and there is again a ‘multiplication’ with a substantive meaning, usually called a ‘product’, but a different in nature than multiplication of scalars. This product is ‘disjoint union’. Unfortunately, to procede, it is probably best to spend some time contemplating what workspaces actually are, a topic presented with less than the highest possible degree of clarity in MCB.



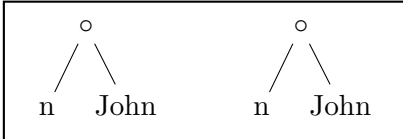
The basic idea is that they are collections of binary trees of some kind, but Chomsky has always been enthusiastic about the use of sets, with the result that his current idea of the output of Merge has been that  $\text{Merge}(X, Y) = \{X, Y\}$ . There are some recent changes to Minimalism involved in this, for example that labelling is no longer accomplished by ‘projecting’ a label into the structure produced by Merge, but simply by searching the items merged (Chomsky et al., 2019). But physicists and mathematicians writing on this topic seem to prefer graphs, which are sets of nodes connected by links (e.g. Ardila and Foissy (undated)). One reason for going with graphs is that people have a reasonable amount of experience working with them, including representing trees with them, whereas there clearly needs to be some work done on using raw sets to build trees as components of workspaces. For example, to implement trees in a workspace, there needs to be a constraint that a set can appear as a subset of at most one other set in that workspace (one, if it’s a daughter of something, none, if it’s the root of a tree). I won’t speculate about what more needs to be done; somebody should do it, but it would be an unnecessary distraction for Hopf algebra beginners.

But MCB’s trees have two further restrictions over those of Foissy:

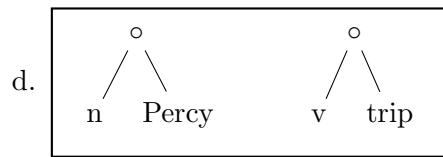
- (13) a. The trees are strictly binary (a node has 2 daughters, or none).  
 b. If a node has no daughters, it has ‘content’ of some kind, presumably provided by the lexicon.

There is a minor terminological issue associated with (b): MCB describe the terminals as being either ‘features’, or ‘lexical items’, without clarifying exactly what they mean. My suggestion is that the lexical items should be roots in the sense of Harley (2014), the features, everything else. This is consistent with the specific examples MCB provides.

Here are a few sample workspaces, aka forests:

- (14) a.   
 b.   
 c. 

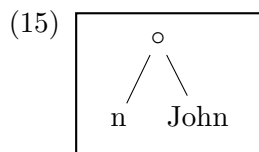




(a) is the empty forest (the 1 in the algebra), (b) is the root ‘John’, (c) two copies of ‘John’ combined with the categorizer for ‘noun’, and (d) a noun and verb in a forest unmerged. Observe, for the reason noted above, that nodes with daughters have no inherent properties, only such as can be found by searching their daughters.

But there is however a foundational issue we should consider before proceeding, which is that a workspace is not just a set of trees, but an isomorphism class of such collections, constituted by all such sets that are isomorphic (according to whatever definition of isomorphism your account of the graphs makes appropriate). This is a problem because due to Russell’s paradox, a class cannot be a member of a set. The most straightforward fix for this problem is to choose our nodes from some infinite set that is generally accepted as existing, such as the natural numbers. Evidently, nobody actually worries about this issue, so neither shall we (“If it is a problem for everybody, it is not a problem for you” – wise advice to PhD students (and probably, these days, ECRs) from Howard Lasnik, once upon a time).

So we finally define the multiplication/product. This is simply ‘disjoint union’, which is set union of sets that have no members in common, with replacement of any sets that do overlap with other, isomorphic ones that don’t. This is actually a bit of an intrusion from category theory, where it is a basic idea that isomorphic objects are essentially the same, and can and should be regarded as equal for (almost) all practical purposes. So we can immediately see that (a) above is the multiplicative identity, and that (c) above would be the results of multiplying (15) below by itself:



It should be evident that this product is intrinsically associative and commutative, and also has an identity, the empty workspace. However it does not have inverses: once something is in a workspace, there is no way to remove it. The definition of an algebra requires the product to be associative and have a unit, but not for it to be commutative, only for there to be a subset of the algebra for which the product is commutative, into which the field or commutative ring with unity can be embedded.

However, we have only defined the product on pairs of workspaces, not the whole algebra. To fix this, we extend it to the entire algebra by decreeing it to be ‘bilinear’, which means ‘linear’ in each of its two arguments/inputs,

but since workspace and SSA polynomial algebras are commutative, we only have to worry about one, say, the first, meaning that the following two equations hold, where  $p, q, r$  are workspace polynomials (including scalars),  $a, b$  scalars only:

$$(16) \quad \begin{array}{l} \text{a. } p(q + r) = pq + pr \\ \text{b. } a(pq) = (ap)q \end{array}$$

In the more general case, where the product does not have to be commutative, we would need two more equations to manage the second input to the product.

## 2 Tensors

Further progress demands that we get more comfortable with tensors, which can be seen as a way of ‘multiplying’ (in yet another sense of multiplication) two vector spaces (or modules) to get a third one, along with a way of multiplying the vectors of the original two spaces to get vectors in the third. This form of ‘multiplication’ is completely different from the multiplication in a ring or algebra, since it is from two systems to a third, rather than within a single system. Tensors are a bit of a stumbling block for many people, given the prevalence of documents and lectures such as Gowers (undated) concerning how to lose your fear of them, but the good news for us is that we can stop with the easier characterization of tensor products in terms of bases rather than the more advanced one that is less reliant on them. We will also present the Universal Property of the tensor, and showing how it allows linear mappings to be tensored, as commonly encountered in presentations of Hopf algebras. This is a significantly more advanced kind of topic than what we have attempted previously.

### 2.1 Bases

We have already observed that vector spaces have a ‘basis’, and, happily for us, workspace algebras have a very natural, although infinite, basis, the workspaces.<sup>7</sup> These and many other vector spaces furthermore have the property that any of their members can be expressed *in a unique way* as a finite sum of basis elements. A further point is that if our algebra is over  $\mathbb{Z}$ , then its basis is also unique, but if it is over  $\mathbb{Q}$ , it is only ‘natural’, since we

---

<sup>7</sup>On the other hand, 3d space, although having a dimensionality of only 3, lacks any truly natural choice of bases, since any three perpendicular vectors of length one, with their origin anywhere, will serve as an adequate one, although in certain situations, some particular three might stand out as the best choice. Lots of entertainment here for a linear algebra course, with applications to computer graphics.

can in a basis substitute for any workspace a scalar multiple of that element, and then get the original element back, if we want it, by division.

In linear algebra, particular importance is attached to the ‘linear’ functions from one vector space to another; these are the functions that obey the following two laws, already stated for the product above, but here put in a more general form:

(17)  $f$  is a linear function from vector space  $V$  to vector space  $W$  over the field  $F$  iff, for all  $v, w \in V, a \in F$ :

- a.  $f(v + w) = f(v) + f(w)$
- b.  $f(av) = af(v)$

Proving that a function is linear tends to be an exercise in applying the distributive and associative laws.

A very important result is that the linear functions have the property that if we define them on the basis elements of an vector space, we have defined them on every element of that vector space. And, since we can restate any workspace polynomial as a sum of scalar products of workspaces, once the value of a linear function is determined for the workspaces, it is thereby determined for the entire workspace algebra (since the algebra product doesn’t create any new workspaces that wouldn’t exist anyway). And every assignment of values to the workspaces defines a linear function on the workspace algebra.

Linear functions have many interesting properties, and it would furthermore be handy, for various reasons, to be able to turn bilinear functions into linear ones. And there turns out to be a way to do this: tensor products. These, with their intimidating symbol  $\otimes$ , have a bit of a reputation, as evidence by the existence of various documents, sections of lectures etc. devoted to helping people ‘lose their fear of them’.<sup>8</sup> Happily for us, we need only the easiest path to tensors, through bases. The basic facts are as follows.

Suppose vector spaces  $V$  and  $W$  have bases  $B$  and  $C$  respectively, such that any member of either space can be expressed as a finite linear combination of elements of its basis. Then it is the case that:

(18) a) There is a vector space notated as  $V \otimes W$  whose basis is the set of ordered pairs whose first member is from  $B$  and second from  $C$ . The basis elements are conventionally written as  $b \otimes c$ , for

---

<sup>8</sup>There are a considerable variety of things called ‘tensors’ floating around in math, for example, in even the simplest forms of linear logic; exactly what unifies them sufficiently to motivate the use of a single term is an interesting question to which I don’t really know the answer, but part of it seems to me to be that it is sort of like multiplication but not necessarily commutative, and produces an object that is in some intuitive way ‘more complex’ than the combined ingredients, which often preserve a degree of partial autonomy, but not full autonomy.

arbitrary basis elements  $b \in B, c \in C$ . But they are really just ordered pairs, and could be written  $\langle b, c \rangle$ .

- b) There is a bilinear function  $\Phi$  taking a pair of vectors from  $V$  and  $W$  into a single vector in  $V \otimes W$ , conventionally written  $v \otimes w$ .
- c) Any bilinear function from  $V$  and  $W$  can be expressed in a unique way as  $\Phi$  followed by a linear function from  $V \otimes W$ .

a) follows from the fact that there is a vector space consisting of all formal linear combinations of the pairs, over the field. b) can be constructed as follows: If  $v \in V$ , it can be expressed as the sum  $s_1 b_1 + \dots + s_m b_m$ , where the  $s_i$  are scalars, the  $b_i$  basis elements of  $V$ , and likewise, if  $w \in W$ , it can be expressed as  $t_1 c_1 + \dots + t_n c_n$ . We can then define  $\Phi(v, w)$  to be  $\sum_{i,j} (s_i t_j) b_i \otimes c_j$ .

Here we have used the summation notation, in which the indexes appearing on the  $\Sigma$  and repeated on the terms are taken to range over whatever limits are specified, explicitly or contextually, and the results added up. Here the limits are contextually determined,  $i$  ranging from 1 to  $m$ ,  $j$  from 1 to  $n$ . A very tedious notation without the  $\Sigma$  would be, where  $V$  has  $m$  basis elements and  $W$   $n$ :

$$(19) \quad s_1 t_1 + \dots + s_1 t_n + \dots + s_m t_1 + \dots + s_m t_n$$

We need to show that  $\Phi$  is bilinear, which means that the following four equations hold, for  $v, w, z$  vectors and  $r$  a scalar:

- (20) a.  $\Phi(v + w, z) = \Phi(v, z) + \Phi(w, z)$
- b.  $\Phi(v, w + z) = \Phi(v, w) + \Phi(v, z)$
- c.  $\Phi(rv, w) = r\Phi(v, w) = \Phi(v, rw)$

For (a), we have:

$$\begin{aligned}
 (21) \quad \Phi(v + w, z) &= \Phi((\Sigma_i s_i b_i) + (\Sigma_i r_i b_i), \Sigma_j t_j c_j) && \text{reexpress inputs} \\
 &= \Phi(\Sigma_i (s_i + r_i) b_i, \Sigma_j t_j c_j) && \text{sum shuffle} \\
 &= \Sigma_{ij} ((s_i + r_i) t_j) b_i \otimes c_j && \text{def. of } \Phi \\
 &= \Sigma_{ij} (s_i t_j + r_i t_j) b_i \otimes c_j && \text{distributive law} \\
 &= \Sigma_{ij} (s_i t_j) b_i \otimes a_i + \Sigma_{ij} (r_i t_j) b_i \otimes c_j && \text{sum shuffle} \\
 &= \Phi(v, z) + \Phi(w, z) && \text{reexpress output}
 \end{aligned}$$

The main driver of the proof is the distributive law of the field of the vector space. The proof for (b) is the same, but with the sums in the second argument of  $\Phi$  rather than the first.

And for the first equation in (c) we have:

$$\begin{aligned}
 (22) \quad \Phi(rv, w) &= \Phi(r(\Sigma_i s_i b_i), \Sigma_j t_j c_j) \\
 &= \Phi(\Sigma_i r s_i a_i, \Sigma_j t_j b_j) \\
 &= \Sigma_{ij} (r s_i t_j) b_i \otimes v_j \\
 &= r(\Sigma_{ij} (s_i t_j) b_i \otimes c_j) \\
 &= r\Phi(v, w)
 \end{aligned}$$

The second equation is the same idea. As a consequence, we can write out the scalar coefficients in front of a tensor without indicating the grouping, that is,  $s_i t_j b_i \otimes c_j$  instead of  $(s_i t_j) b_i \otimes c_j$  as above. This approach is used in Ardila's third definition of an algebra.

Note that the well-definedness of  $\Phi$ , given bases for  $W$  and  $V$ , depends on the fact that the vectors in  $W$  and  $V$  have unique representations as linear combinations of basis vectors. If they didn't, the construction would not determine unique values for pairs of vectors  $w$  and  $v$ .

So what does  $\Phi$  do? What it does is allow us to convert bilinear functions from the set of ordered pairs with first member in  $W$  and second in  $V$ , called the cartesian product of  $W$  and  $V$ , symbolized  $W \times V$ , into linear functions from  $V \otimes W$ , and, furthermore do this in a unique way. For suppose we have bilinear  $f: V \times W \rightarrow Z$ . Then:

$$(23) \quad \begin{aligned} f(u, v) &= f(\sum_i r_i b_i, \sum_j s_j c_j) \\ &= \sum_{ij} r_i s_j f(b_i, c_j) \end{aligned}$$

But now we see that the value of  $f$  is determined by its values on the basis pairs, so we get the same values from a linear function  $\tilde{f}: V \otimes W \rightarrow Z$  defined by setting  $\tilde{f}(b_i \otimes c_j) = f(b_i, c_j)$  (recalling that a tensor of basis elements can be regarded as an ordered pair). Therefore, for any  $v \in V$ ,  $f(v) = \tilde{f}(\Phi(v))$ , which would often be written with the 'function composition' symbol ' $\circ$ ', where  $g \circ f$  means 'the function defined by applying first  $f$ , and then  $g$ '. So we get the following statement of the extremely important 'universal property' of the tensor product:

$$(24) \quad \text{For any bilinear function } f: U \times V \rightarrow W, \text{ there is a unique } \tilde{f}: U \otimes V \rightarrow W \text{ such that } \tilde{f} \circ \Phi = f$$

The difference between  $f$  and  $\tilde{f}$  is slight enough so that people tend to forget to notate it, according to Ardila.

For this reason, in presentations of algebras and Hopf algebras, the product is sometimes presented as a bilinear function from the cartesian product  $A \times A$  of the algebra  $A$ , other times as a linear function from the tensor product  $A \otimes A$ .

I will close up this section by observing that tensoring obeys all the rules for multiplication in SSA except for commutativity, and so therefore can be regarded as a somewhat exotic form of multiplication, but one that exists for all pairs of vector spaces, and takes things from pairs of vector spaces into a third, rather than operating within a single system. It is also important to note that  $b \otimes c$  can be regarded as essentially an ordered pair only if  $b$  and  $c$  are elements of the bases, in which case the tensor is called a 'pure tensor'. Otherwise, they can be multiplied out as a sum of scalar products of pure tensors, whereas mere ordered pairs have no multiplication.

'Universal Property' is a more sophisticated concept than what we have previously encountered; what gives it its utility is that there is more than

one way to construct something that has the universal property of the tensor product, but all such constructions have the property that they are isomorphic, so can be regarded as ‘essentially’ the same, even if they are not exactly the same as delivered by the Axiom of Extensionality in set theory. Note, in passing, that if a syntax teacher draws a syntax tree and starts talking about ‘that tree’, they are probably really referring to the class of trees that are isomorphic to (an abstract representation of) what they drew, rather than the literal marks on the board at that place and time. The relations between isomorphism and identity are investigated in a rather advanced subject called ‘Homotopy Type Theory’ (HOTT), which I have not managed to get a grip on.

One use for the universal property of the tensor is the possibility of defining a useful way of combining linear maps, which play a substantial role in formulating Hopf algebras. Suppose we have two linear maps,  $f:U \rightarrow V$  and  $g:W \rightarrow Z$ . Then we have a function designated  $f \times g$  from  $U \times W$  to  $V \times Z$ , defined by applying  $f$  to the first member of a pair in  $U \times W$  and  $g$  to the second, and then sticking the results into an ordered pair, which will be in  $V \times Z$ . So then we have  $\Phi_{V,Z} \circ f \times g:U \times W \rightarrow V \times Z$ . But now, by the universal property of the tensor product, we can define  $f \otimes g$  as:

$$(25) \quad f \otimes g = \widetilde{\Phi_{V,Z}} \circ (f \times g)$$

It would be rather tedious to get this defined by cranking through linear identities, but, effectively, all this work is packed into the definition of  $\Phi$  for a given pair of vector spaces, plus a proof of the universal property, which we can then just use without messing around with the details. Tensoring linear maps are rather frequently encountered in expositions of Hopf algebras.

A simple example that also introduces us to the use of diagrams in category theory is a statement of the associativity of the algebra product, which is written as ‘ $\sqcup$ ’ in MBC. So here is the diagram:

$$(26) \quad \begin{array}{ccc} A \otimes A \otimes A & \xrightarrow{\sqcup \otimes \text{Id}_A} & A \otimes A \\ \downarrow \text{Id}_A \otimes \sqcup & & \downarrow \sqcup \\ A \otimes A & \xrightarrow{\sqcup} & A \end{array}$$

In spite of the fact that we already know what associativity is, quite a few things about this might require explanation.

The easiest place to start is probably the two items on the right, connected by a downward pointing arrow labelled with ‘ $\sqcup$ ’. This posits a linear map from  $A \otimes A$  to  $A$ , which is what the product is, according to the definition of an algebra. In general, arrows in diagrams designate functions or ‘function-like things’; for us, they will only designate functions. The horizontal arrow along the bottom is the same story. The more challenging bits start at the upper left corner. Here we have a triple tensor, which seems

odd because we have introduced tensoring as a binary operation, so might expect to see either  $(A \otimes A) \otimes A$  or  $A \otimes (A \otimes A)$ , or perhaps even both. I suspect that the latter would be what would happen in a formally fussier Category Theory presentation, but it is not really necessary, because the two ways of building a triple tensor produce results that are isomorphic, so the normal practice with (Hopf) algebras at least is to leave out the parentheses. But they are in a sense implicitly present with the label on the upper arrow heading right: since  $\sqcup$  applies to a tensor, it takes the first two  $A$ 's, while the third is processed by the identity function on  $A$ , designated  $\mathbf{Id}_A$ , which does nothing, leaving  $A$  unaltered. This produces  $A \otimes A$ , which the rightmost downward arrow takes onto  $A$ . Then we have the same story with the left downward arrow, except that  $\mathbf{Id}_A$  applies to the first  $A$ ,  $\sqcup$  to the second. Then, what the entire diagram says is that whichever path we take, the results are the same, that is, both paths through the diagram describe the same function. Such diagrams are widely employed in discussion of Hopf algebras, including MCB.

Diagrams like this can be presented as rigorous formal objects, but also understood as informal presentations of equations, in this case:

$$(27) \quad \sqcup \circ (\sqcup \otimes \mathbf{Id}_A) = \sqcup \circ (\mathbf{Id}_A \otimes \sqcup)$$

Diagrams of this nature are a major feature of Category Theory, which we won't go into here, but a slowish introduction to basic category theory for people who aren't highly capable in math would be Barr & Wells (1999a), while more capable people will probably prefer Awodey (2006).

### 3 Coalgebras

A Hopf algebra is two different species of mathematical system combined into one, one of them is an algebra of the kind we have been discussing, the other a different kind of system called a 'coalgebra'. In general, in my experience so far, co-things are always harder than mere things, and, unfortunately coalgebras are no exception to this. Their critical feature is a 'co-product', which is a kind of opposite, technically called a 'dual', to the product, obeying a compatibility constraint.<sup>9</sup> In this section we consider coalgebras. The treatment will be heavily based on Foissy (undated), which I think is a clearer presentation, although of a different system than that used in MCB.

One basic characteristic of cothings is that their diagrams have arrows going in the opposite direction from the corresponding 'non-co's, and

---

<sup>9</sup>My wild and probably ignorant conjecture as to why cothings are harder than things is that, being easier, the things are discovered first, and then somebody notices the cothing and the duality relationship between them. Duality is a very interesting topic, but beyond the scope of this exposition.

when considering an isolated specimen, it is customary to change the letters used. For the coproduct, we have (28) below to compare with (26), the associativity diagram:

$$(28) \quad \begin{array}{ccc} C & \xrightarrow{\Delta} & C \otimes C \\ \downarrow \Delta & & \downarrow \mathbf{Id} \otimes \Delta \\ C \otimes C & \xrightarrow{\Delta \otimes \mathbf{Id}} & C \otimes C \otimes C \end{array}$$

After considerable discussion of the coproduct, we will give a much brief consideration of the considerably simpler counit.

### 3.1 The Coproduct

Linguistically, the function of the coproduct  $\Delta$  is to remove subtrees from lower positions, making them available for Merge to attach them to other trees, thereby implementing Internal Merge. It is an interesting feature of this kind of approach that the extracted subtrees will have no memory of their previous position, except as encoded in the features of their terminal elements (case, for example). The image that I suggest is pruning in an orchard. Suppose a competent orchardist, who does not make useless cuts, prunes a tree. We then have a collection of cut off branches (with their sub-branches), and the original tree. Including the possibility that they did nothing (no branches, tree unaltered), or removed the entire tree. And, being competent, they did not cut off a branch, and then decide that they had to remove more and cut off some branch that the first removed branch was a sub-branch of, thereby making the first cut redundant. And it makes no difference whether we think of the pruning as applied to individual trees, with the branches and remainders of the trees gathered together later, or of it as applying to the entire orchard or subdivisions of the orchard which are then gathered together later. So that is not too hard, but the next step is harder: all the different ways the orchardist can do it, presented as a formal sum (adding up the scalar coefficients gives the number of different ways that the pruning could happen).<sup>10</sup>

So what the coproduct does is apply to a forest, work out all the ways of removing nonoverlapping (sub) trees from the trees in that forest, and present the results as a sum of tensors, where a term with scalar coefficient 1 represents a single way of doing this, other than 1, that number of ways of doing it that produce results that look the same.<sup>11</sup> The tensors themselves

---

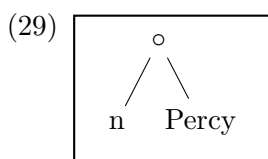
<sup>10</sup>There is however an interesting difference between this application of formal sums and the application to collections and inventories, which is that the latter are things that are all present at once, whereas the former are alternative results from a body of resources, which suggests to me that linear logic might have an application here.

<sup>11</sup>This suggests to me that fractional coefficients will not arise, that is, there is no use for the field  $\mathbb{Q}$ , and that consequently we want our algebra to be a module over the integers, not an actual vector space.



present the removed subtrees as a forest (a heap of cut off branches), and the pruned trees as another forest (which can be empty, in the case of the ‘total cut’, where every tree is fully removed). In Foissy’s version, nothing more happens, but in MCB’s, there is a ‘cleanup’ phrase, whereby nodes left with only one daughter are ‘contracted’ with their mothers, leaving a strictly binary tree behind (with at least one potentially problematic residue, to be discussed below). Before defining it formally, I will first suggest a concrete image, and then some simple linguistic examples.

For an ultrasimple example, consider the following workspace:



The basic conceptualisation of how  $\Delta$  works with trees (in both MCB and Foissy) is that it cuts links connecting a subtree to the whole, with the product of the subtrees on under the cut links put on the left of the tensor, and the remainder, with or without cleanup (‘the quotient’) on the right. So in terms of the orchard-pruning image, the tensor expresses the organizational difference between the prunings and the trees with the prunings removed. The cleanup, used in MCB but not Foissy, consists of ‘contracting’<sup>12</sup> the now single-daughtered node that used to immediately dominate the removed node, with its surviving daughter taking over its position in the tree. So three of the things we get from (29) are:

- (30) a.  $\boxed{n} \otimes \boxed{\text{Percy}}$  (n’s link is snipped, leaving Percy as the quotient)
- b.  $\boxed{\text{Percy}} \otimes \boxed{n}$  (the other way around)
- c.  $\boxed{n \text{ Percy}} \otimes \boxed{o}$  (snip both links, producing two disconnected nodes and leaving behind a single node.<sup>13</sup>)

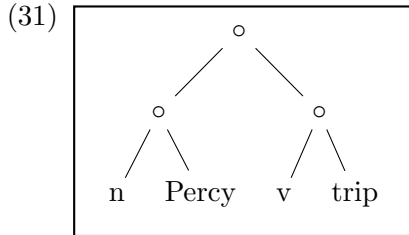
But there are two more options consistent with the definition (MCB Lemma 2.7, page 7), the ‘total cut’ in which the entire tree is removed, so that (29)

<sup>12</sup>MCB are not clear about exactly what ‘contraction’ consists of, but if it is mapping the old workspace onto a new one such that contracted nodes are mapped onto the same thing in the new space as what they are contracting with, things should work out as long as the disappearing nodes have no properties based on labelling, which appears to be the case.

<sup>13</sup>There is actually an interpretive issue here. MCB motivate cleanup on the basis of problems with labels, but here we have a node with no label. As it happens, given the way the coproduct is actually used, this situation will not arise in a normal derivation. A bit more will be said about this below.

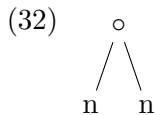
appears on the left of the tensor and 1 on the right, and the ‘null cut’, where we do nothing, and so the same things appear in the opposite order. Then the result of this is the (formal) sum of all five ways of cutting up the workspace.

There is a further constraint, which requires a more complex tree to illustrate usefully:

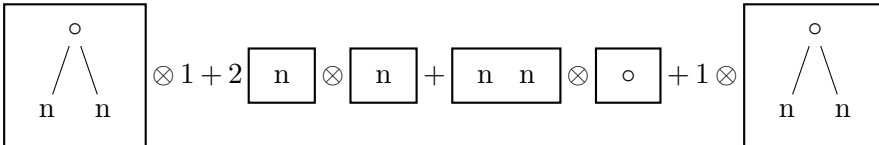


It says that the subtrees pulled out by  $\Delta$  should not overlap. In terms of link removal, this means that no path from a terminal to the root should have to cross more than one link targeted for removal.<sup>14</sup> In the orchardry image, this is represented by the stipulation that the orchardist is competent and does not make unnecessary cuts any particular pruning of the trees. This restriction is encoded into MCB’s Lemma 2.6, pg 6.

The definition of  $\Delta$  then says that we take all possible ways of cutting up the workspace, and then add them up as a formal sum of tensors. A final example that might help solidify a useful point is what  $\Delta$  does to this, a branching tree with two identical terminals (this represents nothing that occurs naturally in syntax):



The sum consists of the total and null cuts plus 3 ‘interesting’ terms, two of which come up looking as identical, i.e. are isomorphic, and so produce the coefficient 2 (or, we could, equivalently, just add the term in twice; here, as in grade school,  $1+1=2$ ). With contraction, the total sum is:

(33) 

<sup>14</sup>To my mind, this has a slightly eerie resemblance to the notion of ‘analysability’ in classic TG. However, it is found in Foissy (undated), which does not draw on generative grammar as a source at all, but is absent from Ardila’s discussion of Hopf algebras for general graphs. So something to do with trees is presumably behind it, but what?

The term with scalar coefficient 2 arises because there is no difference between the results of cutting the right and the left branch. But if the two terminals carried unique indexes that were ‘visible’ to whatever is assessing isomorphism, then there would be three different terms in the sum, all with scalar coefficient 1, and no possibility of scalar multiplication because they would be different.

As a final example, it might be useful to have look at the equality (a) below, and the non-equality (b):

$$(34) \text{ a. } \boxed{n} + \boxed{n} = 2 \boxed{n}$$

$$\text{ b. } \boxed{n \ n} \neq 2 \boxed{n}$$

Although (b) as equation is false, the right hand side will appear in some of the terms of what the coproduct will produce from it.

Before pushing on to a more formal presentation, I’ll make two more general remarks. First, there is a possible glimmer of an actual empirical issue in the cleanup process, a major difference between MCB and Foissy, which is that there is also a recent idea that Internal Merge is driven by failure of labelling of of the mother: if the mother gets no label, it seems more reasonable to contract it with something else than if it does, due to the possibility of a label conflict. But I think there is more work to be done here. (And we need more guidance as to how the other things needed to make syntax actually work are to be integrated into the system.)

Another observation is that in spite of large range of outputs produced by  $\Delta$ , the only ones that are used is when a single subtree is extracted, for later application of Merge. The discrepancy between the power of the means and the limited range of what is used reminds me a bit of Sapir’s observation “It is somewhat as though a dynamo capable of generating enough power to run an elevator were operated almost exclusively to feed an electric doorbell.” (Sapir, 1921, 14).

Now we move on to a more formal presentation, also extending our treatment from single trees to workspaces in general. I find Foissy’s presentation clearer than MCB’s, so will mostly follow that. First (Foissy p4), we define the notion of a ‘non-total cut’, which is a choice of edges to be deleted, ‘admissible’ if no path from any leaf to the root crosses at most one chosen edge. We have already discussed this requirement.<sup>15</sup> To these admissible cuts is added one more admissible cut, the ‘total cut’, in which the entire forest is removed, and the empty one left behind. Given an admissible cut  $c$ , the product of all the removed subtrees from a tree  $t$  is denoted by  $P^c(t)$  ( $P$  for product/pieces/prunings?), the residue/remainder  $R^c(T)$ . So if  $c$  is the

---

<sup>15</sup>Foissy also suggests regarding the links as oriented upward, consistent with my thought that we want them to be directed, from daughter to mother.

total cut, then the resulting term is  $t \otimes 1$ .<sup>16</sup> So for individual trees we get the following definition, where, in general,  $P^c$  is a forest, while  $R^c$  is always a tree:

$$(35) \quad \Delta(t) = \sum_{c \in \text{Adm}(t)} P^c(t) \otimes R^c(t)$$

The fact that we have a product on the left side of the tensor here, which is simply the disjoint union of all the removed pieces, indicates that there is no ‘tracking’ of what part of the residue the original pieces came from.<sup>17</sup> This defines  $\Delta$  for the ‘generators’, but not the basis, of the algebra (and, therefore, not the whole algebra).

But this definition applies only to the generators of the coalgebra, that is, individual trees, whereas a forest can have multiple trees, and  $\Delta$  also has to work on formal sums of forests. The requirement that  $\Delta$  be linear extends (35) to formal sums of trees without effort, but to get the multitree forests, we need to extend it through the multiplication. One thing we need for this is to construct from the algebra  $A$  the tensor algebra  $A \otimes A$ . We already have the vector space  $A \otimes A$ ; to make it an algebra we need a unit, which will come later, and a product, which is what we are doing now. The requirement that must be satisfied for  $\Delta$  to be an algebra homomorphism is (36) below, where the product symbols are subscripted for which algebra they are the product of:

$$(36) \quad \Delta(w \sqcup_A z) = \Delta(w) \sqcup_{A \otimes A} \Delta(z)$$

In the case where  $w$  and  $z$  are single trees, in accordance with (35), this will expand out to:

$$(37) \quad \Delta(w \sqcup_A z) = \sum_{c \in \text{Adm}(w)} P^c(w) \otimes R^c(w) \sqcup_{A \otimes A} \sum_{d \in \text{Adm}(z)} P^d(z) \otimes R^d(z)$$

And then, because a product must obey the distributive law, this is equivalently:

---

<sup>16</sup>There is a bit of puzzling notation whereby the set of admissible cuts is denoted by  $\text{Adm}_*(t)$ , this set minus the total and null cut,  $\text{Adm}(t)$ . Seems a bit perverse to me, but there is probably a good explanation for the reversal of where I would expect the  $*$  subscript to be.

<sup>17</sup>Contrary to what I would have thought as the author of Andrews (1971) back in the day, I now think that no memory of the past is a highly desirable property of a derivational system: if such memory seems to be required, that suggests that some different kind of system architecture is called for. See Quicoli (1982) for a different account of that material.

(38)

$$\Delta(w \sqcup_A z) = \sum_{c \in \text{Adm}(w), d \in \text{Adm}(z)} (P^c(w) \otimes R^c(w)) \sqcup_{A \otimes A} (P^d(z) \otimes R^d(z))$$

So how do we define  $\sqcup_{A \otimes A}$ ? The story that works is:

(39)

$$(a \otimes b) \sqcup (c \otimes d) = (a \sqcup c) \otimes (b \sqcup d)$$

Applied to the coproduct, this fits in with the orchard imagery. We want to think of the prunings and remainder of two one-tree orchards as being those those of a two-tree orchard. Therefore, we think of the prunings all together as one thing, and there remainders all together as another thing, and, because we are representing these things as tensors with the prunings first and the remainders second, the result is:

(40)

$$\Delta(w \sqcup z) = \sum_{c \in \text{Adm}(w), d \in \text{Adm}(z)} (P^c \sqcup P^d) \otimes (R^c \sqcup R^d)$$

where the sum is ranging over all combinations of prunings from  $w$  and those from  $z$ .

One thing I'm not sure of is whether there is any alternative to (39) for making the tensor algebra at all, or whether (39) is just the one that works properly with  $\Delta$ . And maybe want to think a bit more about why it is well-defined.

### 3.2 Counit (and unit)

An algebra has a unit, so it is not a surprise that a coalgebra has a counit, and neither would it be a surprise that this is a bit harder than the algebra unit, although it is easier than the coproduct. What the counit does in a tree algebra is map every forest but the empty one onto 0, and the latter onto the 1 of the field (the  $\delta$  is the Kronecker  $\delta$ , which is a test for equality).

The above satisfies the general condition for the counit, of correct interaction with the coproduct, formulated by Ardila at some length,<sup>18</sup> and more concisely by Foissy:5. Note that  $K$  (often in a fancier font), is a popular choice of letter to designate the field:

$$(41) \quad \begin{array}{ccccc} & & C \otimes C & & \\ & \swarrow \epsilon \otimes \text{Id} & \uparrow \Delta & \searrow \text{Id} \otimes \epsilon & \\ K \otimes C & & & & H \otimes K \\ & \swarrow & & \searrow & \\ & & C & & \end{array}$$

<sup>18</sup>Notes pg 1 gives a diagram for the counit, but he goofed and wrote  $\epsilon$  instead of  $\Delta$  on the central arrow going from  $H$  to  $H \times H$ . This is fixed in Lecture 5:54:00.

From the upper part of the diagram we see that  $\epsilon$  goes from the algebra to its field (and, unsurprisingly, the unit, when we get to it, goes in the opposite direction), but what about the unlabelled arrows going up from  $H$  at the bottom? This is, as discussed by Ardila in Lec 5:54, the trivial map (‘natural isomorphism’) that takes an element  $h$  of the algebra to  $1 \otimes h$ , so perhaps it would be better to draw these arrows going both ways.

Staring at this in light of what has already been said about  $\epsilon$  for the tree coalgebra, we can grasp why it’s true.  $\Delta$  produces a sum of terms of the form  $H \otimes H$  with perhaps some scalar coefficient but  $\epsilon$  maps all of the tensor factors but 1 into 0 (of the field), so that they disappear from the sum, and we are left, effectively, with the original from  $H$  on either side. Foissy presents exactly this as an equation.

By what we’ve said about cothings, the unit ought have a diagram with arrows going in the opposite direction, and here it is, with the lower arrows bidirecional:

$$(42) \quad \begin{array}{ccccc} & & A \otimes A & & \\ & v \otimes \mathbf{Id} \nearrow & \downarrow & \nwarrow \mathbf{Id} \otimes v & \\ K \otimes A & & \sqcup & & A \otimes K \\ & \nwarrow & \downarrow & \nearrow & \\ & & A & & \end{array}$$

### 3.3 A Coassociativity Proof (optional, or maybe for later)

Here I will try to provide some hints to understand a coassociativity proof. I so far find the one in MCB to be completely unintelligible, but the one in Foissy seems more manageable. This unfortunately is for a somewhat different system, with no binarity restriction on the trees, and therefore, also, no cleanup. This will hopefully not make any serious difference, but even if it does, perhaps working through it will improve one’s facility with this kind of material.

First, there is an important Lemma 1 on page 4, which defines the operator  $B^+$  (which also appears in MCB), which converts a forest  $t_1, \dots, t_n$  to a tree by creating a new mother node and setting all the trees (maximal ones, not any subtrees they might have) to be its daughters. The Lemma then says that  $\Delta$  almost commutes with  $B^+$ , except for a slight discrepancy, so that if  $x$  is the forest, then:

$$(43) \quad \Delta \circ B^+(x) = B^+(x) \otimes 1 + (\mathbf{Id} \otimes B^+) \circ \Delta(x)$$

The discrepancy is that all the different ways of extracting subtrees from the forest  $x$  do not include the result of extracting the entire new tree  $B^+(x)$ , requiring the first term of (43). Then the first component of the second term will cover all the bits removable from the trees of  $x$  (including entire ones),

passed through unchanged by  $\mathbf{Id}$ , and then the second is the remainder trees put together with  $B^+$ . I won't go through the actual proof on Foissy pg 5, but it is an exercise in shuffling sums and products.

The issue of exactly how cleanup works is relevant here: my impression is that if it was deletion rather than contraction that is used in MCB, Lemma 1 would not hold there, because it would be possible for some of the trees in the forest to disappear. Then, because of the role of the lemma in the proof, the proof would then definitely not apply to the MCB system.

On to the proof. Coassociativity means that this identity holds:

$$(44) \quad (\Delta \otimes \mathbf{Id}) \circ \Delta = (\mathbf{Id} \otimes \Delta) \circ \Delta$$

The proof starts by defining  $A$  is the set of forests for which (44) holds. It is then observed that whatever  $A$  is, it's a subalgebra (closed under the algebra operations). Then we have two equations, each converting one side of (44) to an identical expression, provided that  $x \in A$ . For me, the key to understanding the first equation was as follows:

- (45) a. The first line of the first equation gets its first term by applying  $(\Delta \otimes \mathbf{Id})$  to the first term of Lemma 1, while not going beyond formal function concatenation for the second term.
- b. In the second line, the first two terms are produced by application of Lemma 1 to the first term of the first line, while the last term is justified by the fact that to do nothing to something and then apply  $\Delta$  to it produces the same result as first apply  $\Delta$  and then do nothing to the resulting components (it is a  $\Delta$ -elaborated version of the 'interchange law' from baby category theory).

And for the second equation:

- (46) a. For the first line, apply Lemma 1 to expand  $\Delta(B^+(x))$ . Then observe that the first component of the first term on the right hand side of that equation is the first component of the first term of the result of that application,, and that the other two components are what  $\Delta$  does to 1, and the second term results from the fact that  $\mathbf{Id} \otimes \Delta$  will do nothing to the first component of the result of applying  $\Delta$  to  $x$ , and apply  $B^+$  and apply  $B^+$  and then  $\Delta$  again to the second.
- b. Then the second line, equal to the second line of the first equation, has the first term from the first line unaltered. The second two terms are basically Lemma 1 again, but with  $\Delta(x)$  in the place of  $x$ , and  $\mathbf{Id}$  stuck onin the front. So the second term of this line is the first term of Lemma 1 sitting in this environment, the third term the second of Lemma 1.

Well this may or may not help; this may be one of those cases where everyone has to find their own way. But the rest of the proof is a straightforward induction to the effect that  $A$  is in fact the entire workspace algebra.

## 4 Bialgebras and Hopf Algebras

We have now amassed most of the stuff we need to make a Hopf algebra, but there are a few more pieces and a general plan of organization that we need to complete this. A Hopf algebra is a Bialgebra that has a gadget called an ‘antipode’, which our system already has, but what is a Bialgebra? It is a system that is both an algebra and a coalgebra, meaning that it has a product, a unit, a coproduct and a co-unit, all of which we have looked at. But there is an additional requirement, that the algebra and coalgebra be ‘compatible’ in the following sense, which can be formulated either as (a) or (b) below:

- (47) a. The coproduct and counit must be algebra homomorphisms  
 b. The product and unit must be coalgebra homomorphisms.

Since the first is easier to understand, and we have already started on it, we will take that path; the second is covered in Ardila’s lectures. We have already shown that the coproduct is an algebra homomorphism into the tensor algebra, so what’s left is to show that the counit also is one, and provide a unit for the tensor algebra. This latter will be (using  $H$  to designate the algebra):

$$(48) \quad u(1_K) = 1_H \otimes 1_H$$

Then, the coproduct to be an algebra homomorphism, it must take the algebra unit onto the tensor algebra unit, which it does:

$$(49) \quad \Delta(1_H) = 1_H \otimes 1_H$$

And then we also have the requirement that the counit be an algebra homomorphism:

- (50) a.  $\epsilon(1) = 1$   
 b.  $\epsilon(w \sqcup z) = \epsilon(w) \sqcup \epsilon(z)$

Both of these are true for the product and counit as defined, and not too hard to check.

So far, we have worked through the criteria for being a bialgebra; a Hopf algebra requires one more thing, an ‘antipode’, which is a function from the algebra to algebra satisfying yet another condition. The detailed specification of the antipode seems to be rather complex, but its presence turns out



to follow from the existence of a simpler property, which actually has some relevance to the syntactic application, a grading. A grading is a classification of the elements of the algebra in terms of some kind of complexity, such that:

- (51) a. Each grade is a vector space (the product will tend to produce something in a different grade)
- b. The grades are disjoint (a partition of the vector space)
- c. The union of all the grades constitutes the entire algebra

Foissy uses the number of nodes in the forest as the grade, while MCB use the number of terminals; I am aware of no significant difference between these choices (but could have missed something later in MCB).

But the happy result is that a graded Bialgebra whose lowest grade is the underlying field has an antipode, and is therefore a Hopf algebra. I'll point out that although the antipode does not appear to play a direct role in syntax (so far), its definition does make use of negative scalar coefficients.

And finally, another source for this material, heavily based on Ardila but at an overall more advanced level, is the earlier sections of Ebert (2014).

## Appendix A Typoes, Minor Unclarities, and Textual Explication

Here are a few typoes etc. in MCB and MBC:

1. MCB:2 has syntactic features and lexical items as the terminal elements of trees, while MBC:18 has lexical items and syntactic objects.
2. MCB:5 seems to have wrong definition for  $Acc(T)$ , I think it should be:  $Acc(T) = \{T_v | v \in V_{int}(T)\}$ .
3. MCB:8  $\delta_{S,S'}$  being a 'linear operator' means that it preserves the vector space operations, but not the product. 'matching' appears to be isomorphism. I don't know why they didn't just call it that.
4. MCB:9 Def 2.8 "labels labelled by the set ..."  $\rightarrow$  "leaves labelled by the set ..."
5. MBC:19 (3.9) a typo where the  $v$  under  $\Sigma$  wants an underscore, since it's a list of nodes, as seen by comparison with the corresponding formula (2.10) in MCB:7.

## References

Andrews, Avery D. 1971. Case agreement of predicate modifiers in Ancient Greek. *Linguistic Inquiry* 2. 127–152.

Awodey, Stephen. 2006. *Category theory*. Oxford University Press.

Barr, Michael & Charles Wells. 1999a. *Category theory for computing science*. Montréal: Centre de Recherches Mathématiques 3rd edn. URL: <http://www.math.mcgill.ca/triples/Barr-Wells-ctcs.pdf>. The most essential content for linguists, but no exercises, can be found in Barr & Wells (1999b).

Barr, Michael & Charles Wells. 1999b. Category theory lecture notes for ESSLLI. URL: <http://www.let.uu.nl/esslli/Courses/barr-wells.html>. a condensed version, without exercises, of Barr & Wells (1999a). It is impressively well targeted on material that seems likely to be useful to linguists.

Burris, Stanley & H.P. Sankappanavar. 1981. *A course in universal algebra*. Springer Verlag. Updated editions available online (latest 2012) at <http://www.thoralf.uwaterloo.ca/htdocs/ualg.html>.

Chomsky, Noam, Ángel Gallego & Dennis Ott. 2019. Generative grammar and the faculty of language: Insights, questions, and challenges. *Catalan Journal of Linguistics* special issue. <https://revistes.uab.cat/catJL/article/view/sp2019-chomsky-gallego-ott>.

Ebert, Becca. 2014. A basic introduction to Hopf algebras. URL: [http://mathcs.pugetsound.edu/~bryans/Current/Spring\\_2014/3\\_Becca\\_HopfAlgebraFinal](http://mathcs.pugetsound.edu/~bryans/Current/Spring_2014/3_Becca_HopfAlgebraFinal)

Foissy, Loïc. undated. An introduction to the hopf algebras of trees. <https://www2.mathematik.hu-berlin.de/~kreimer/wp-content/uploads/Foissy.pdf>.

Gowers, Tim. undated. How to lose your fear of tensor products. <https://www.dpmms.cam.ac.uk/~wtg10/tensors3.html>, very usefully LaTeXed version at [https://nessie.ilab.sztaki.hu/~kornai/2023/Hopf/Resources/gowers\\_](https://nessie.ilab.sztaki.hu/~kornai/2023/Hopf/Resources/gowers_) (LaTeXed by Blanka Kövér and ??).

Grätzer, G. 1978. *Universal algebra*, 2nd edition. New York: Springer.

Harley, Heidi. 2014. On the identity of roots. *Theoretical Linguistics* 40. 225–276. <http://heidiharley.com/heidiharley/wp-content/uploads/2016/09/Harley2014Identity0>

Marculli, Matilde, Robert Berwick & Noam Chomsky. 2023a. Old and new minimalism: a Hopf Algebra comparison. <https://arxiv.org/abs/2306.10270>.

Marcolli, Matilde, Noam Chomsky & Robert Berwick. 2023b. Mathematical structure of syntactic merge. <https://arxiv.org/abs/2305.18278>.

Quicoli, Carlos A. 1982. *The structure of complementation*. Ghent: Story-Scientia.

Sapir, Edward. 1921. *Language*. Harcourt, Brace and World.