

Combinatorial Hopf algebras in particle physics I



Erik Panzer
Scribed by Iain Crump

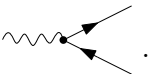
May 25

1 Combinatorial Hopf Algebras

Quantum field theory (QFT) describes the interactions of elementary particles. There are many different QFTs. These have different particles with different interactions.

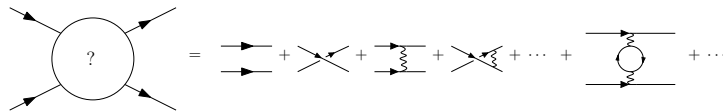
Example. Quantum electrodynamics (QED) has two particles;

- electron \rightarrow 
- photon \sim ,

and only one allowed interaction; .

From these building blocks, we can build arbitrary graphs. The problem is that quantum field theory is very difficult to solve, and even difficult to give a well-defined mathematical meaning. What you can do in practice is compute in perturbation theory. In perturbation theory we compute by an approximation or asymptotic expansion, where each process is described by an infinite sum of Feynman graphs.

Example. In QED, consider an experiment where we throw in two electrons and at the end we get two electrons back,



In perturbation theory we assume the coupling (the interaction strength) is small, so we assume graphs with no vertices are the most important graphs, and graphs with more vertices come with a coupling constant. There are infinitely many diagrams, and all of them contribute to the process. In a perturbative QFT computation, you define a particular process and a particular theory, and then you have to draw all the diagrams corresponding to these rules, and then you have to each probability as a contribution to the process.

The Feynman rules assign to each graph G a Feynman integral $\Phi(G, p, m)$, where p is the momenta and m the masses of the particles. Computing these integrals and adding them up gives an estimate of the energy the physical process should give during the experiment. This theoretical prediction can be compared to experimental measurements, and if you are lucky this can result in the discovery of the Higgs boson, for example. The problems with this are twofold, and will be discussed in the following subsections.

1.1 Renormalization

Usually $\Phi(G) = \infty$, and they do not make any sense when written down. This is a well-known problem which slowed the advancement of QFT for a long time, but was solved decades ago (and will be explained in greater detail in the next lecture).

To fix this, we redefine constants (g, m) , g the coupling constant (the probability of an interaction; if the coupling is high an interaction is more likely to occur, if the coupling is zero then we have a free theory), to cancel divergences. The idea is that while these integrals are infinite they did not depend on the masses or momenta so we can subtract infinity and make it a well-defined function. This is not so easy to do, as in order to give this physical meaning you be sure that the things that you change your parameters by are constants. For example, you cannot take a different constant to subtract for each momentum. Hence, the change must work for each diagram. This is called *locality of counterterms*, and will be examined next lecture.

This requires nested, recursive subtractions. In physics, there is a forest formula that describes this. To do this with mathematical rigor we use the Connes- Kreimer Hopf algebraic formulation. The renormalization can be formulated completely combinatorially, forgetting about the integrals.

1.2 Computing $\Phi(G)$

This is known to be difficult and extremely complicated. For example,

$$\Phi \left(\text{---} \left(\begin{array}{c} m_1 \\ \circ \\ m_2 \\ \circ \\ m_3 \end{array} \right) \text{---} \right)$$

was computed only two years ago.

In many cases (though not this previous one), $\Phi(G)$ are multiple polylogarithms (MPL). This will be discussed on Wednesday. There are infinitely many interesting (non-trivial) graphs with Feynman integral that can be computed algorithmically using MPL (actually hyperlogarithms). These algorithms are very nicely expressible using Hopf algebras. Again, this separates the analytics from the combinatorics. We try to compute a function on many variables, but in this case we can formulate the algorithm on a purely combinatorial level; each of the MPL can be written as a word over a certain alphabet, and things like integration or limits of functions represented by the word can be written as combinatorial operations on this algebra of words. We will take a look at a simple version of this on Friday.

2 Hopf algebras

Definition. An (associative, unital) *algebra* over a field k is a k -vector space \mathcal{A} with a linear map multiplication $m : \mathcal{A} \otimes \mathcal{A} \rightarrow \mathcal{A}$ such that

1. $m(m \otimes \text{id}) = m(\text{id} \otimes m)$ (associativity)
2. $\exists \mathbf{1} \in \mathcal{A}$ such that $m(\text{id} \otimes \mathbf{1}) = \text{id} = m(\mathbf{1} \otimes \text{id})$.

Since this is a vector space we can of course multiply by scalars and add vectors. We will always take $k = \mathbb{Q}$. Usually, we have the distributive law;

$$\begin{aligned} m(v + w, u) &= m(v, u) + m(w, u) \\ (v + w)u &= v \cdot u + w \cdot u. \end{aligned}$$

This is already encoded in saying that multiplication is a linear map. Actually, the distributive laws tell you that the multiplication is a bilinear map.

Remark 1. Say $\{v_i : i \in I\}$ is a basis of vector space V and $\{w_j : j \in J\}$ is a basis of vector space W . Then $\{v_i \otimes w_j : (i, j) \in I \times J\}$ is a basis of $V \otimes W$.

Universal property: For any bilinear map $b : V \times W \rightarrow Z$, there exists a unique linear map $\tilde{b} : V \otimes W \rightarrow Z$ (linear) such that

$$\begin{array}{ccc}
 b: V \times W & \longrightarrow & Z \\
 \downarrow \otimes & \curvearrowright & \nearrow \tilde{b} \\
 V \otimes W & &
 \end{array}
 ,$$

where $\otimes : V \times W \rightarrow V \otimes W$ by $(\sum_i \lambda_i v_i, \sum_j \mu_j w_j) \mapsto \sum_{i,j} \lambda_i \mu_j v_i \otimes w_j$.

Associativity:

$$\begin{array}{ccc}
 & \mathcal{A} \otimes \mathcal{A} \otimes \mathcal{A} & \\
 m \otimes \text{id} \swarrow & & \searrow \text{id} \otimes m \\
 \mathcal{A} \otimes \mathcal{A} & & \mathcal{A} \otimes \mathcal{A} \\
 m \swarrow & & \searrow m \\
 & \mathcal{A} &
 \end{array}$$

Example. To give an example that we have all seen before, polynomials in any number of variables $\mathbb{Q}[x_1, \dots, x_n]$. Here, $\mathbf{1} = 1x_1^0 \cdots x_n^0$.

The tensor algebra is a little more complicated.

Definition. Let V be a vector space. The tensor algebra is

$$T(V) := \bigoplus_{n \geq 0} V^{\otimes n} = \mathbb{Q} \cdot \mathbf{1} \oplus V \oplus V \otimes V \oplus V \otimes V \otimes V \oplus \dots$$

It has the concatenation product

$$v_1 \otimes \cdots \otimes v_n \cdot v_{n+1} \otimes \cdots \otimes v_{n+m} = v_1 \otimes \cdots \otimes v_{n+m} \in V^{\otimes(n+m)} \subseteq T(V).$$

Hence, T has an associative product, we have a neutral element $\mathbf{1}$, and it is obviously not commutative.

The tensor algebra is a general construction and appears often. For example it is the Hopf algebra we will need for iterated integrals; when we want to compute a Feynman integral, this is what appears naturally.

Remark 2. If we choose some basis of our vector space, the tensor algebra is actually the linear span of all these elementary tensors; if $\{v_i \mid i \in I\}$ is a basis of V , then $T(V) = \lim_{\mathbb{Q}} \{v_{i_1} \otimes \cdots \otimes v_{i_n} \mid n \in \mathbb{N}_0, i_1, \dots, i_n \in I\}$. We write v_{i_1}, \dots, v_{i_n} for this, where these are words over the alphabet I .

Example. For a finite basis $V = \lim_{\mathbb{Q}} \{v_0, v_1\}$,

$$T(V) = \{\lambda \mathbf{1} + \lambda_0 v_0 + \lambda_1 v_1 + \lambda_{12} v_1 v_2 + \lambda_{21} v_2 v_1 + \cdots \mid \text{only finitely many } \lambda \text{ are nonzero}\}.$$

We see here a combinatorial structure; we have identified the basis with words, and this gives the algebra itself additional structure. Because our algebra is composed of these strings we can do things other than multiplication. This is a general concept of combinatorial Hopf algebras; the objects making up the algebra is combinatorial and they have substructures. These substructures can be exploited to give more information about the elements. Multiplication takes two elements and gives a bigger element. Conversely, the coproduct;

$$\Delta : T(V) \rightarrow T(V) \otimes T(V),$$

$$v_{i_1} \cdots v_{i_n} \mapsto \sum_{j=0}^n v_{i_1} \cdots v_{i_j} \otimes v_{i_{j+1}} \cdots v_{i_n}.$$

Further, $(\Delta \otimes \text{id})\Delta = \sum_{j_1 \leq j_2} v_{i_1} \cdots v_{i_{j_1}} \otimes v_{i_{j_1+1}} \cdots v_{i_{j_2}} \otimes v_{i_{j_2+1}} \cdots v_{i_n}$.

So, Δ is called coassociative; $(\Delta \otimes \text{id})\Delta = (\text{id} \otimes \Delta)\Delta$. Note that $\Delta w = \sum w_{(1)} \otimes w_{(2)} = \mathbf{1} \otimes w + w \otimes \mathbf{1} + \sum w' \otimes w''$, where this last sum is all non-trivial terms.

Define $\mathcal{E} : T(V) \rightarrow \mathbb{Q}$ by $\mathcal{E}(\mathbf{1}) = 1, \mathcal{E}(v_{i_1} \cdots v_{i_n}) = 0$ if $n > 0$. Then, $(\mathcal{E} \otimes \text{id})\Delta(w) = w = (\text{id} \otimes \mathcal{E})\Delta(w)$. This is called the *counit property*.

So, we have an analogue of multiplication and the unit in the coproduct case; a counit which is dual to the unit. The question is if these structures have something in common with each other.

For a Hopf algebra, we want to have

1. $\Delta(a \cdot b) = \Delta(a) \cdot \Delta(b)$
2. $\mathcal{E}(a \cdot b) = \mathcal{E}(a) \cdot \mathcal{E}(b)$.

Checking,

$$\Delta(v_0 v_1) = \mathbf{1} \otimes v_0 v_1 + v_0 \otimes v_1 + v_0 v_1 \otimes \mathbf{1}$$

$$\Delta(v_0)\Delta(v_1) = (\mathbf{1} \otimes v_0 + v_0 \otimes \mathbf{1})(\mathbf{1} \otimes v_1 + v_1 \otimes \mathbf{1}) = \Delta(v_0 v_1) + v_1 \otimes v_0.$$

Hence, we must define a new product or coproduct to rectify this. For our purposes, it makes more sense to define a new product.

Definition. The *shuffle product* $\sqcup : T(V) \otimes T(V) \rightarrow T(V)$ is

$$v_{i_1} \cdots v_{i_n} \sqcup v_{i_{n+1}} \cdots v_{i_{n+m}} = \sum_{\sigma \in S_{n,m}} v_{i_{\sigma(1)}} \cdots v_{i_{\sigma(n+m)}}$$

where the (n, m) shuffles are

$$S_{n,m} = \{\sigma \in S_{n+m} \mid \sigma^{-1}(1) < \cdots < \sigma^{-1}(n), \sigma^{-1}(n+1) < \cdots < \sigma^{-1}(n+m)\}.$$

Example. $v_0 v_1 \sqcup v_2 = v_0 v_1 v_2 + v_0 v_2 v_1 + v_2 v_0 v_1$

There is also a recursive definition of the shuffle product. If $w_1, w_2 \in T(V)$ and $v_1, v_2 \in V$, then

$$(v_1 w_1) \sqcup (v_2 w_2) = v_1 (w_1 \sqcup v_2 w_2) + v_2 (v_1 w_1 \sqcup w_2).$$

We see that the shuffle product has the properties we want.

Lemma 3. 1. \sqcup is associative

2. $a \sqcup \mathbf{1} = a$

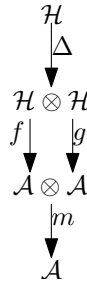
3. $\Delta(w_1 \sqcup w_2) = \Delta(w_1) \sqcup \Delta(w_2)$, the latter shuffle being $\sqcup \otimes \sqcup$

4. \sqcup is commutative.

The bialgebra $(T(V), \sqcup, \Delta)$ is called the *shuffle algebra*.

We also have the ability to multiply to linear maps. This is unusual, as normally if you multiply two linear maps you get a quadratic map.

Definition. Let \mathcal{H} be a bialgebra and some algebra \mathcal{A} . For $f, g : \mathcal{H} \rightarrow \mathcal{A}$, $f \star g := m \cdot (f \otimes g) \cdot \Delta_{\mathcal{H}}$, the *convolution product*.



Lemma 4. 1. $f \star (g \star h) = (f \star g) \star h$

2. $e_{(\mathcal{A})} := \mathbf{1}_{(\mathcal{A})} \cdot \mathcal{E}$ is the neutral element.

This is a powerful way of encoding combinatorial operations. An equation where you write down the convolution product can mean a lot of computation in practice; you compute the coproducts of all objects involved, and sometimes this greatly simplifies complicated formula.

Question 1. Are there inverses? In particular, does id have an inverse?

Definition. The bialgebra \mathcal{H} is called *Hopf* if and only if the id has an inverse with respect to \star . If so, this is called the *antipode* $S := \text{id}^{\star^{-1}}$. That is, $S \star \text{id} = e$.

Lemma 5. *The tensor algebra with shuffle product $(T(V), \sqcup, \Delta)$ is Hopf, and $S(v_{i_1}, \dots, v_{i_n}) = (-1)^n v_{i_n} \cdots v_{i_1}$.*

Example. To check that this is in fact the antipode,

$$\begin{aligned} (S \star \text{id})(v_1 v_2) &= S(v_1 v_2) + S(v_1) \sqcup v_2 + v_1 v_2 \\ &= v_2 v_1 + (-v_1) \sqcup v_2 + v_1 v_2 = 0. \end{aligned}$$

3 Hopf algebras of rooted trees

Definition. A *rooted tree* (t, r) is a tree t together with a distinguished vertex, the root, $r \in V(t)$.

Remark 6. We consider isomorphic classes (ie. there is no labeling on the trees and no distinguishing between left and right, hence not plane trees).

$$\mathcal{T}_1 = \{\bullet\}, \mathcal{T}_2 = \left\{ \begin{array}{c} \bullet \\ | \\ \bullet \end{array} \right\}, \mathcal{T}_3 = \left\{ \begin{array}{c} \bullet \\ | \\ \bullet \\ | \\ \bullet \end{array}, \begin{array}{c} \bullet \\ \diagdown \quad \diagup \\ \bullet \quad \bullet \end{array} \right\}, \dots$$

We will distinguish the root by drawing it at the top of the tree.

Definition. *Rooted forests* are disjoint unions of rooted trees.

$$\mathcal{F}_0 = \{1\}, \mathcal{F}_1 = \{\bullet\}, \mathcal{F}_2 = \left\{ \begin{array}{c} \bullet \\ | \\ \bullet \end{array}, \bullet\bullet \right\}, \mathcal{F}_3 = \mathcal{T}_3 \cup \left\{ \begin{array}{c} \bullet \\ | \\ \bullet \\ | \\ \bullet \end{array}, \bullet\bullet\bullet \right\}, \dots$$

With the definitions of trees and forests, we can immediately make an algebra out of it.

Definition. $\mathcal{H}_R := \mathbb{Q}[\mathcal{T}] = \lim_{\mathbb{Q}} \mathcal{F}$. We give \mathcal{H}_R the product from \mathcal{T} , disjoint union;

$$\left(2\bullet + \begin{array}{c} \bullet \\ | \\ \bullet \end{array} \right) \cdot (\bullet\bullet) = 2\bullet\bullet\bullet + \begin{array}{c} \bullet \\ | \\ \bullet\bullet \end{array}$$

To make this a Hopf algebra, we need a coproduct.

Definition. Recall that a rooted forest induces a partial order on its vertices, “below”; \preceq . Hence,

$$\Delta(f) = \sum_{\substack{C \subseteq V(f) \\ C \text{ incomparable}}} P^C(f) \otimes R^C(f),$$

where $P^C(f) = \prod_{v \in C} \{w \mid w \preceq v\}$ and $R^C(f) = f \setminus P^C(f)$. By incomparable, we mean subsets such that there are no two vertices connected by an upwards-only path in the forest.

Example.

$$\begin{aligned} \Delta(\bullet) &= \mathbf{1} \otimes \bullet + \bullet \otimes \mathbf{1} \\ \Delta\left(\begin{array}{c} \bullet \\ | \\ \bullet \end{array}\right) &= \begin{array}{c} \bullet \\ | \\ \bullet \end{array} \otimes \mathbf{1} + \mathbf{1} \otimes \begin{array}{c} \bullet \\ | \\ \bullet \end{array} + \bullet \otimes \bullet \\ \Delta\left(\begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array}\right) &= \mathbf{1} \otimes \begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array} + \begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array} \otimes \mathbf{1} + 2\bullet \otimes \begin{array}{c} \bullet \\ | \\ \bullet \end{array} + \dots \otimes \bullet \end{aligned}$$

Note that for trees, the right hand side of Δ is always trees. Hence, Δ is not co-commutative (symmetric about the tensor).

Lemma 7. \mathcal{H}_R is a bialgebra.

Outline. That it is an algebra we have seen. We need to check that it is coassociative, that the coproduct is compatible with the product, and we have to define the counit.

$$\begin{aligned} \mathcal{E}(\mathbf{1}) &= 1 \\ \mathcal{E}(f) &= 0 \text{ for all } f \in \mathcal{F}_{>0} \\ (\mathcal{E} \otimes \text{id})\Delta &= (\text{id} \otimes \mathcal{E})\Delta = \text{id} \\ \Delta(t_1 t_2) &= \Delta(t_1)\Delta(t_2) \\ (\Delta \otimes \text{id})\Delta &= (\text{id} \otimes \Delta)\Delta \quad (\text{left as an exercise}). \end{aligned}$$

□

We also can define the coproduct with a recursive definition, which is useful for inductive proofs.

Definition. The linear map $B_+ : \mathcal{H}_R \rightarrow \mathcal{H}_R$ is defined by

$$B_+(t_1, \dots, t_n) := \begin{array}{c} \bullet^r \\ / \quad \backslash \\ t_1 \quad \dots \quad t_n \end{array}.$$

Note that $B_+ : \mathcal{F} \rightarrow \mathcal{T}$ is bijective.

Lemma 8. $\Delta B_+ = (\text{id} \otimes B_+) \Delta + B_+ \otimes \mathbf{1}$

Proof. Looking at the coproduct, if $r \in C$, $C = \{r\}$, so

$$P^C(B_+(\dots)) \otimes R^C(B_+(\dots)) = B_+(t_1 \dots t_n) \otimes \mathbf{1}.$$

If $r \notin C$, C is just a set of incomparable elements of t_1, \dots, t_n . What remains is not only the remainders of the individual trees, but glued together by B_+ . \square

As each forest is a product of trees and each tree comes through B_+ as a forest, we get a recursive definition for the coproduct.

Example.

$$\begin{aligned} \Delta \left(\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \end{array} \right) &= \Delta(B_+(\bullet\bullet)) = (\text{id} \otimes B_+)([\Delta(\bullet)]^2) + \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \end{array} \otimes \mathbf{1} \\ &= \left(\mathbf{1} \otimes \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \end{array} + 2 \bullet \otimes \begin{array}{c} \bullet \\ \diagup \\ \bullet \end{array} + \bullet \otimes \bullet \right) + \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \end{array} \otimes \mathbf{1} \end{aligned}$$

$$\begin{aligned} \Delta(\bullet) &= \Delta B_+(\mathbf{1}) = (\text{id} \otimes B_+)(\mathbf{1} \otimes \mathbf{1}) + \bullet \otimes \mathbf{1} \\ &= \mathbf{1} \otimes \bullet + \bullet \otimes \mathbf{1} \end{aligned}$$

Remark 9. Every forest can be constructed recursively by iteration of m and B_+ . So, imposing $\Delta B_+ = (\text{id} \otimes B_+) \Delta + B_+ \otimes \mathbf{1}$ uniquely determines Δ .

The following definitions are needed for the exercises.

Definition. A *primitive element* p is an element such that $\Delta p = \mathbf{1} \otimes p + p \otimes \mathbf{1}$.

Definition. For forest f , the *grading operator* Y is $Yf := f \cdot |V(f)|$.

$$\begin{aligned} \text{ie. } Y\bullet &= \bullet \\ Y \begin{array}{c} \bullet \\ \diagup \\ \bullet \end{array} &= 2 \begin{array}{c} \bullet \\ \diagup \\ \bullet \end{array} \end{aligned}$$

Definition. A *grading* of \mathcal{H} is a decomposition $\mathcal{H} = \bigoplus_{n \geq 0} \mathcal{H}_n$ such that;

1. $m(\mathcal{H}_n, \mathcal{H}_m) \subseteq \mathcal{H}_{n+m}$
2. $\Delta(\mathcal{H}_n) \subseteq \sum_{k+l=n} \mathcal{H}_k \otimes \mathcal{H}_l$.