Book Title
Book Editors
IOS Press, 2003

An introduction to Combinatorial Hopf Algebras

— Examples and realizations —

Florent Hivert
LIFAR – Université de Rouen – Faculté des Sciences et des Techniques – Avenue de l'université – 76801 SAINT ETIENNE DU ROUVRAY – FRANCE
e-mail: Florent.Hivert@univ-rouen.fr

Abstract.

Keywords. Combinatorics, combinatorial Hopf algebras, symmetric functions, sorting algorithms, search trees.

1. introduction – some motivations

In the past recent years, several very different people coming from various part of science were naturally led to the same kind of objects, namely "The Combinatorial Hopf Algebras". A Hopf algebra is roughly speaking both an algebra and the dual of an algebra which are compatible. Consequently there is a product rule which allows one to *compose* the objects and a co-product rule which *decompose* the objects. Though there is for the moment no good definition of what is a "combinatorial" Hopf algebra, there are plenty of examples each of them having a very rich combinatorics. It's seems to be a field where algebraists, computer scientists and physicists can understand each other more or less and managing to work together as shown during the first meeting on this subjects in Montreal (2001) and Banff (2003).

One of the precursor of the subject is certainly Rota [27,28,14] which was the first to emphasizes the importance of Hopf algebra in combinatorics. He was inspired by the work of Mac-Mahon [24] on symmetric functions. Indeed, the hopf algebra of symmetric function is the prototype of what is a combinatorial Hopf algebra. This was deeply used by Zelevinsky [30] is his study of the representation of the classical groups. The representation theory is one very important motivation in the study of combinatorial Hopf algebra. They appears as character ring or Grothendieck ring of tower of algebras. This point of view has been further extended to various extension of the Hecke algebras (Hecke-Clifford, Cyclotomic Hecke algebras or Ariki-Koike algebras) in [16,17,6,2,13].

Another strong motivation coming from algebra is the study of operads [19, 20]. One of the main character of the present paper was defined in this setting namely the algebra of binary trees of Loday and Ronco [21,22]. The motivation was the study of a certain class of algebra called Dendriform algebras. These are algebras where the product decompose into the sum of two operations verifying some relations.

The physicists were led to the study of combinatorial Hopf algebras trough the quantum field theory. The main problem here is to give a meaning to some complicated diverging series. The classical approach use Feynman's Diagrams is more a receipt than a well founded theory. A systematic approach has been initiated by Connes and Kreimer [5] using some combinatorial Hopf algebras on planar trees. Their algebra is very close to the algebra of Loday and Ronco. This approach is extended by Brouder and Frabetti in [3,4].

In this paper, we present a contribution coming from theoretical computer science to the subject. It appears that the most important examples arise naturally in the study of some very basic searching and sorting algorithms and that this knowledge allows us to very simply construct these objects. All the axioms of Hopf algebras, which are usually very tedious to check using the standard way to define a Hopf algebras, are here mostly obvious coming from simple facts.

The key idea is not to define, as usual, a combinatorial object together with a product, a coproduct and then to show the various axioms like the associativity, the co-associativity, and the compatibility, but rather to *realize* the hopf algebra as a vector space of some maybe complicated non-commutative polynomials verifying some "symmetry" invariance. When this invariance is appropriate the product is inherited from the product of polynomials and the coproduct is defined using the so-called "alphabet doubling trick". Then the only things to prove is that the space is stable by the product and the co-product. This is usually done by *proving* the rules that are usually given as a definition. The main ingredient of the construction of a Hopf algebra is then an algorithm which compute the "symmetry class" $S(w)$ of a word. Then the definition

$$F_\sigma := \sum_{w \mid S(w)=\sigma} w \tag{1}$$

where σ is a "symmetry class" defines all what is needed, provided the algorithm verifies certain compatibility properties which are very easy to check. Here the word "symmetry" is to be understood in a very large sense and I must confess that it took several years to my research team to realize how wide is this sense.

The goal of the present paper is to introduce interested people to this beautiful subject and certainly not to build a formal base for the theory of Hopf algebras. Though we gives all the formal definitions, the reader is strongly advised to refer to basic textbooks on this subject as [29,1,26].

2. A very simple example

We start by giving a simple example namely the non-commutative polynomials hoping that this introduce smoothly the formalism and the key ideas.

2.1. The free algebra

One of the simplest combinatorial object is the word, that is finite sequences of letters coming from a set $\mathbb{X}$ called the alphabet. For example, $ababd$ is a word over the alphabet $\mathbb{X} = \{a, b, c, d, e\}$. There exists an empty word which is denoted ϵ, and the concatenation (gluing) of to word w and w is denoted by $w \cdot w'$ or simply by ww'. The word $ababd$ concatenated with bab gives $ababdbab$. The empty word is a neutral element for the concatenation which is associative: for all words w, w' and w'', one has

$$\epsilon \cdot w = w \cdot \epsilon = w \quad \text{and} \quad (w \cdot w') \cdot w'' = w \cdot (w' \cdot w'') =: w \cdot w' \cdot w'' \tag{2}$$

On resume all these properties by saying that the set of words $\mathbb{X}^*$ endowed with the unit ϵ and the product $\cdot$ is a monoïd. If one wants to count some multiplicities, one are naturally led to consider linear combinations of words with integer coefficients, that is element of the free algebra generated by $\mathbb{X}$ with scalar in $\mathbb{Z}$ denoted by $\mathbb{Z}\langle\mathbb{X}\rangle$. A typical element of $\mathbb{Z}\langle\mathbb{X}\rangle$ is of the form $\sum_w c_w w$, for example $3abc + 5baba + 2\epsilon$. The product is naturally extended by linearity

$$\left(\sum_w c_w w\right)\left(\sum_{w'} c_{w'} w'\right) := \sum_{w,w'} c_w c_{w'} w \cdot w' \,. \tag{3}$$

For example, $(3abc + 5baba + 2\epsilon)(3a + bb) = 9abca + 15babaa + 6a + 3abcbb + 5bababb + 2bb$. Note that the product symbol $\cdot$ is here implicit. It should be noticed that each word w have a length denoted by $\ell(w)$ and that the concatenation is compatible to this length in the sense that

$$\ell(w \cdot w') = \ell(w) + \ell(w') \,. \tag{4}$$

Hence the space $\mathbb{Z}\langle\mathbb{X}\rangle$ decompose into

$$\mathbb{Z}\langle\mathbb{X}\rangle = \bigoplus_{n\in\mathbb{N}} \mathbb{Z}^n\langle\mathbb{X}\rangle \tag{5}$$

where $\mathbb{Z}^n\langle\mathbb{X}\rangle$ is the linear combination of words of length n. Such a linear combination is called *homogeneous of degree* n and the compatibility property can be restated as the inclusion:

$$\mathbb{Z}^n\langle\mathbb{X}\rangle \;\; \mathbb{Z}^m\langle\mathbb{X}\rangle \subset \mathbb{Z}^{n+m}\langle\mathbb{X}\rangle \,. \tag{6}$$

All these facts (unit, associativity, homogeneous decomposition compatible with the product) is resumed by saying that $\mathbb{Z}\langle\mathbb{X}\rangle$ is a *graded algebra.*

2.2. A first co-product

In order to defines Hopf algebra we now want to introduce the notion of co-product. Recall that a coproduct is a notion dual to the notion of a product. A product is a *composition law* and consequently a coproduct is a *decomposition law.* A very simple but powerful way to define coproduct is the "alphabet doubling" trick presented here on an example.

Let w be a word. We compute the coproduct $\Delta(w)$ by the following sequence of operation. First we compute the sum of all possible replacement of the letters of w by any of two different copy of it, say a small one and a capital one. The letter a is replaced by a or A, the letter b is replaced by b or B and so on. Then we decide that the small letters and the capital one commute with each other, thus we can put the small one on the left and the capital one on the right. Finally, the concatenation of the two words wW is replaced by a tensor product $w \otimes w'$ where w' is the word in small letters associated to W. Here is a example:

$$\begin{aligned}\Delta(aba) &= aba + Aba + aBa + abA + ABa + AbA + aBA + ABA \\ &= aba\,\epsilon + ba\,A + aa\,B + ab\,A + a\,AB + b\,AA + a\,BA + \epsilon\,ABA \\ &= aba \otimes \epsilon + ba \otimes a + aa \otimes b + ab \otimes a + a \otimes ab + b \otimes aa + a \otimes ba + \epsilon \otimes aba\end{aligned}$$

Recall now that the tensor product symbol is bilinear that is for any scalar c and objects U and V one has

$$c\,(U \otimes V) = (cU) \otimes V = U \otimes (cV)\,. \tag{7}$$

It is then possible to collect the preceding example to get the multiplicities:

$$\Delta(aab) = aab \otimes 1 + 2\,(ab \otimes a) + aa \otimes b + 2\,(a \otimes ab) + b \otimes aa + 1 \otimes aab\,.$$

Moreover, we decide to extends this operation by linearity, hence defining a linear operation

$$\Delta \;:\; \mathbb{Z}\langle\mathbb{X}\rangle \longmapsto \mathbb{Z}\langle\mathbb{X}\rangle \otimes \mathbb{Z}\langle\mathbb{X}\rangle\;. \tag{8}$$

This operation goes in the reverse way of a product: indeed, a product on A is usually defined as a *bilinear operation* $A \times A \mapsto A$, but it can consequently be seen as a *linear operation* $A \otimes A \mapsto A$. This can be formalized using the duality, but for the moment we prefer to concentrate on some properties of this operations.

2.3. Co-associativity

If w is a word, the coproduct $\Delta(w)$ is a linear combination $\sum c(w_1 \otimes w_2)$ of symbol $w_1 \otimes w_2$. We can therefore decide to compute the linear combination $\sum c(\Delta(w_1) \otimes w_2)$ that is $(\Delta \otimes \mathrm{Id})(\Delta(w))$ where Id is the identity map and

$$(f \otimes g)(x \otimes y) := f(x) \otimes g(y)\,. \tag{9}$$

If we do the same on the right we get the same result:

$$\sum c(\Delta(w_1) \otimes w_2) = \sum c(w_1 \otimes \Delta(w_2))\,. \tag{10}$$

This is not surprising because if we go back to the definition of the coproduct we realize that both of these expression are computed by taking three copies of the original alphabet: In the left hand side each letter a is replaced by a or A and then by a or (A or A') whereas in the right hand side the letter a is replaced by a or A and then by (a or a') or A'. If we have used three different colors, there would have been no differences.

Thus we have the following identity

$$(\Delta \otimes \mathrm{Id}) \circ \Delta = (\mathrm{Id} \otimes \Delta) \circ \Delta \tag{11}$$

where $\circ$ is the composition of functions. This property is called *co-associativity* because it is dual to the associativity property. Indeed, if the product $A \times A \mapsto A$ is considered as a linear operation $\mu : A \otimes A \mapsto A$, then the associativity property can be rewritten has

$$\mu \circ (\mu \otimes \mathrm{Id}) = \mu \circ (\mathrm{Id} \otimes \mu) \tag{12}$$

The word dual is here to be understood as "reversing" (transpose the matrices) the operation. In this paper we decide not to emphasize on duality, therefore the reader is strongly encouraged to have a look at [1,29,26] for a more developed theory including the diagrammatic notation together with the "reversing the arrows" principle.

If we denote c the operation which send all the word to 0 except the empty one sent to 1 we have the following identity

$$(c \otimes \mathrm{Id})(\Delta(w)) = (\mathrm{Id} \otimes c)(\Delta(w)) = w \tag{13}$$

through the identification $s \otimes w = w \otimes s = w$ if s is a scalar. Again, this is obvious in our example since $(\mathrm{Id} \otimes c)\Delta(w)$ amounts to do the doubling alphabet tricks and then to erase the terms $w \otimes W$ where W is not empty. This property is dual to the property

$$\mu(\epsilon \otimes w) = \mu(w \otimes \epsilon) = w \tag{14}$$

and c is therefore called a *co-unit*.

We resume these properties in the following definition

Definition 1. *A* co-algebra *over the field $\mathbb{K}$ is a $\mathbb{K}$-vector space H together with two maps*

$$\Delta : H \mapsto H \otimes H \quad \textit{and} \quad c : H \mapsto \mathbb{K} \tag{15}$$

such that Δ is a co-associative coproduct (Equation (11)) admitting c for co-unit (Equation (13)).

Before going further in should be noted that the co-product is compatible with the length is the sense that

$$\Delta(\mathbb{Z}^m\langle\mathbb{X}\rangle) \subset \bigoplus_{i+j=m} \mathbb{Z}^i\langle\mathbb{X}\rangle \otimes \mathbb{Z}^j\langle\mathbb{X}\rangle \ . \tag{16}$$

One says that the co-algebra is *graded*.

2.4. Compatibility between the product and the co-product

We want here to introduce on our example the main axiom of the theory of Hopf algebras. First of all, we naturally extend the multiplication to tensors as

$$(u \otimes v)(u' \otimes v') = uu' \otimes vv \, . \tag{17}$$

Let v and w to words. Their co-product can be writen as linear combinations $\sum s(v_1 \otimes v_2)$ and $\sum t(w_1 \otimes w_2)$ where s and t are scalars. We want to compare $\Delta(v)\Delta(w)$, that is

$$\Big(\sum s(v_1 \otimes v_2)\Big)\Big(\sum t(w_1 \otimes w_2)\Big) = \sum st(v_1w_1 \otimes v_2w_2) \tag{18}$$

with the co-product $\Delta(v \cdot w)$. Let us do it first on an example. We start with

$$\begin{aligned}\Delta(a) &= a \otimes \epsilon + \epsilon \otimes a \\ \Delta(ab) &= ab \otimes \epsilon + a \otimes b + b \otimes a + \epsilon \otimes ab\end{aligned}$$

First we expand the product of these two expressions:

$$\begin{aligned}&\Delta(ab)\Delta(\mathbf{a}) = (ab \otimes \epsilon + a \otimes b + b \otimes a + \epsilon \otimes ab)(\mathbf{a} \otimes \epsilon + \epsilon \otimes \mathbf{a}) \\ &= ab\mathbf{a} \otimes \epsilon + a\mathbf{a} \otimes b + b\mathbf{a} \otimes a + \mathbf{a} \otimes ab + ab \otimes \mathbf{a} + a \otimes b\mathbf{a} + b \otimes a\mathbf{a} + \epsilon \otimes ab\mathbf{a} \, .\end{aligned}$$

The direct computation of the coproduct of $a \cdot ba$ gives

$$\Delta(aba) = aba \otimes \epsilon + ba \otimes a + aa \otimes b + ab \otimes a + a \otimes ab + b \otimes aa + a \otimes ba + \epsilon \otimes aba \, ,$$

which is the same linear combination. Actually, this is easily seen to be always true since the concatenation clearly commute with the "doubling alphabet trick". Moreover, the co-product of a word $v = l_1 l_2 \dots l_r$ is noting but the tensor notation of the product

$$(l_1 + L_1)(l_2 + L_2) \dots (l_r + L_r) \tag{19}$$

where the small l's and the capital L's commutes with the other. Thus it is clear that if $v = l'_1 \dots l'_s$ both $\Delta(v)\Delta(w)$ and $\Delta(vw)$ are the tensor notation for

$$(l_1 + L_1)(l_2 + L_2) \dots (l_r + L_r)\ (l'_1 + L'_1)(l'_2 + L'_2) \dots (l'_s + L'_s) \, , \tag{20}$$

and thus are equal.

This lead us to the following definition

Definition 2. *A* bi-algebra *is a vector space H endowed with a structure of* algebra $(\cdot, 1)$ *together with a structure of* co-algebra (Δ, c) *satisfying the* compatibility *relation*

$$\Delta(xy) = \Delta(x)\Delta(y)\,. \tag{21}$$

A bi-algebra is said to be graded *for a degree ℓ if both the algebra and the co-algebra are graded. If moreover the homogeneous component of degree* 0 *is the line spanned by the unit the bi-algebra is said to be* connected.

Note 1. *All the bi-algebras considered in this paper are graded and connected. In the usual definition of a Hopf algebra one more ingredient is required called the* antipode. *The graded-connected hypothesis ensure that the antipode is defined and thus all the bi-algebras considered here are Hopf algebras. Therefore we will stick to the name Hopf algebra without speaking about the antipode.*

3. The bubble sort algorithm and the free quasi-symmetric functions

3.1. Basic properties

The goal of this section is to construct the Hopf algebra of permutations first defined by Malvenuto-Reutenauer [25] and called here the algebra of free quasi-symmetric functions. This algebra arise naturally is the study of a very simple (and inefficient) sorting algorithm called the bubble sort. Let us recall this algorithm

Algorithm 3 (Bubble sort).
INPUT : a word
OUTPUT : the associated sorted word.

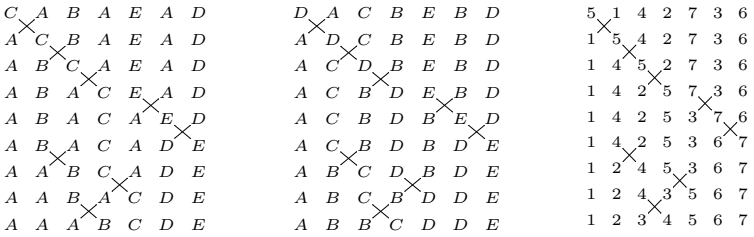
- *Given a word $w = l_1 l_2 \dots l_n$, find two adjacent letters l_i, l_{i+1}, in the wrong order $l_i > l_{i+1}$ and exchange them.*
- *Repeat the procedure until the word is sorted.*

Of course there is a need for a strategies of searching for the two letters. Actually there are several one, for example:

- left-to-right scanning strategy;
- right-to-left scanning strategy;
- go-and-back scanning strategy;
- try the leftmost one, etc.

The chosen strategy is irrelevant for our study, we therefore stick to the left-to-right scanning strategy.

Here are examples of the execution of the bubble sort on three words:



As it can be seen, though they are difference, the bubble sort behave the same. We want to formalize this notion of *execution of the bubble sort.*

Definitions 4. *An* inversion *of a word $w = l_1 l_2 \dots l_n$ is a pair $(i < j)$ of integers such that*

$$i < j \quad \textit{and} \quad l_i > l_j \,. \tag{22}$$

A descent *of a word $w = l_1 l_2 \dots l_n$ is an integer i such that*

$$l_i > l_{i+1} \,. \tag{23}$$

Clearly at each step, the bubble sort choose a descent and remove it. Moreover, doing that is removes only one inversion and thus, the number of inversions is the number of steps of the bubble sort algorithm.

Let $\mathfrak{S}_n$ denote the symmetric group of size n, that is the group of the $n!$ permutations of the set $\{1,\dots \mathrm{n}\}$. A permutations σ is identified with the word $\sigma(1)\sigma(2)\cdots\sigma(n)$. Such a word is called *standard*. Among the permutations, we are interested in the *elementary transposition* $\sigma_i = (i, i+1)$ which exchange i and $i+1$ and leave all the integer unchanged. There is an action of the symmetric group on words from the *right*:

$$(a_1 a_2 \cdots a_n) \cdot \sigma = a_{\sigma(1)} a_{\sigma(2)} \cdots a_{\sigma(n)} \,. \tag{24}$$

For example $abbac \cdot 31452 = baacb$

Definition 5. *An* execution *of the bubble sort is a sequence of transpositions $[\sigma_{i_1}, \dots, \sigma_{i_m}]$, sorting the word in increasing order. Actually, one easily sees that the resulting permutation*

$$\mathrm{exec}(w) := \sigma = \sigma_{i_1} \circ \cdots \circ \sigma_{i_m} \tag{25}$$

encodes the whole information.

Hence the goal of the bubble sort is to compute the smallest (using the minimal number of elementary transpositions) permutation $\sigma := \mathrm{exec}(w)$ such that $w \cdot \sigma$ is sorted. A natural question is to describe the set of the words which have the same bubble sort execution.

The answer is provided by the following simple notion:

Definition 6. *Given a word $w = l_1 l_2 \dots l_n$ of length n, there exists a unique permutation* $\mathrm{Std}(w) := \sigma$ *of* $\mathfrak{S}_n$ *which have the same inversion as* w*:*

$$\textit{for } i < j \quad \textit{then} \qquad \sigma(i) > \sigma(j) \quad \textit{iff} \quad l_i > l_j \,. \tag{26}$$

The permutation $\mathrm{Std}(w) \in \mathfrak{S}_n$ *is called the* standardized *of* w*.*

The simplest way to prove this is to give a effective algorithm. $\mathrm{Std}(w)$ can also be defined as the permutation obtained by iteratively scanning w from left to right, and labelling $1, 2, \dots$ the occurrences of its smallest letter, then numbering the occurrences of the next one, and so on. For example $\mathrm{Std}(abcadbcaa) = 157296834$ as seen on the following picture:

$$\begin{array}{cccccccccc} a & b & c & a & d & b & c & a & a \\ a_1 & b_5 & c_7 & a_2 & d_9 & b_6 & c_8 & a_3 & a_4 \\ 1 & 5 & 7 & 2 & 9 & 6 & 8 & 3 & 4 \end{array}$$

Here is the link between the notion of standardization and bubble sort executions.

Proposition 7. *For any word w, the word $w \cdot \mathrm{Std}(w)^{-1}$ is sorted. Moreover,*

$$\mathrm{exec}(w) = \mathrm{Std}(w)^{-1} \,. \tag{27}$$

Proof. Clearly, $\mathrm{exec}(w) = \mathrm{exec}(\mathrm{Std}(w))$, because $\mathrm{exec}(w)$ depends only on the inversion of w and by definition w and $\mathrm{Std}(w)$ have the same inversions. Now, for any permutation σ, the result of the sorting is $\sigma \cdot \mathrm{exec}(\sigma)$ and is the identity permutation. But for permutations, the action and the composition coincide $\sigma \cdot \mu = \sigma \circ \mu$. Consequently $\mathrm{exec}(\sigma) = \sigma^{-1}$. □

3.2. Execution and concatenation

We are now interested in the following natural question: Let u and v two words of execution σ and μ. What are the possible executions of the sorting of the word uv ? More formally, let $\mathbb{A}$ be a totally ordered alphabet. Define the language (set of word) $L_\sigma(\mathbb{A})$:

$$L_\sigma(\mathbb{A}) := \{w \in A^* \mid \mathrm{exec}(w) = \sigma\} \tag{28}$$

For example:

$$\begin{aligned} L_{12} &= \{\text{sorted words of length } 2\} \\ L_{123\dots n} &= \{\text{sorted words of length } n\} \\ L_{nn-1\dots 21} &= \{\text{strictly decreasing words of length } n\} \\ L_{2143} &= \{bacb, badc, cadc, cbdc, \dots\} = \{yxtz \mid x < y \leq z < t\} \end{aligned}$$

The question is now restated as: describe the language $L_\alpha(\mathbb{A})L_\beta(\mathbb{A})$ for any permutations α and β.

The answer is provided by the shuffle product

Definition 8. *The shuffle product* $\sqcup\!\sqcup$ *of two words is the element of* $\mathbb{Z}\langle\mathbb{A}\rangle$ *defined recursively by*

$$w \sqcup\!\sqcup \epsilon = \epsilon \sqcup\!\sqcup w = w$$
$$xu \sqcup\!\sqcup yv = x(u \sqcup\!\sqcup yv) + y(xu \sqcup\!\sqcup v) \qquad x, y \in \mathbb{A}, \qquad u, v \in \mathbb{A}^*.$$

Alternatively, $u \sqcup\!\sqcup v$ is defined as the sum of all ways of building a word w together with two complementary subwords equal to u and v. Here is an example:

$$\begin{aligned} aba \sqcup\!\sqcup \mathbf{cb} &= aba\mathbf{cb} + ab\mathbf{c}a\mathbf{b} + ab\mathbf{cb}a + a\mathbf{c}ba\mathbf{b} + a\mathbf{c}b\mathbf{b}a + a\mathbf{cb}ba + \mathbf{c}aba\mathbf{b} + \mathbf{c}ab\mathbf{b}a + \mathbf{ca}b\mathbf{b}a + \mathbf{cb}aba \\ &= abacb + abcab + abcba + acbab + acbba + acbba + cabab + 2\,cabba + cbaba \end{aligned}$$

Of course the sum of the coefficients of $u \sqcup\!\sqcup v$ is the binomial coefficient $\binom{\ell(u)+\ell(v)}{\ell(u)}$.

Then the main result of this section is the following theorem

Theorem 9 (Duchamp-H.-Thibon [6]). *For any permutations* $\alpha \in \mathfrak{S}_m$ *and* $\beta \in \mathfrak{S}_n$*, the language* $L_\alpha L_\beta$ *is a disjoint union of languages* L_μ*:*

$$L_\alpha L_\beta = \bigsqcup_{\mu \in \alpha \sqcup\!\sqcup \beta[m]} L_\mu \,, \tag{29}$$

where $\beta[m] := \beta_1 + m \ldots \beta_n + m \in \mathfrak{S}(n+1, n+2, \ldots, n+m)$.

Here are some examples of products:

$$\begin{aligned} L_{12}L_{123} &= L_{34512} \sqcup L_{34152} \sqcup L_{34125} \sqcup L_{31452} \sqcup L_{31425} \\ &\quad \sqcup L_{31245} \sqcup L_{13452} \sqcup L_{13425} \sqcup L_{13245} \sqcup L_{12345} \\ L_{21}L_{123} &= L_{34521} \sqcup L_{34251} \sqcup L_{34215} \sqcup L_{32451} \sqcup L_{32415} \\ &\quad \sqcup L_{32145} \sqcup L_{23451} \sqcup L_{23415} \sqcup L_{23145} \sqcup L_{21345} \end{aligned}$$

It is remarkable that Equation (29) is independent of the underlying language $\mathbb{A}$, the only difference is that if $\mathbb{A}$ is too small, then some $L_\mu(\mathbb{A})$ are empty, for instance $L_{3,2,1}(\{a, b\})$ is empty. For this reason, it is easier to work with an infinite language.

A formal proof of this theorem can be found in [6]. It essentially rely on the following equivalence for any permutation $\sigma_1\sigma_2 \ldots \sigma_{n+m}$:

- $\mathrm{Std}(\sigma_1\sigma_2 \ldots \sigma_n) = \alpha$ and $\mathrm{Std}(\sigma_{n+1} \ldots \sigma_{n+m}) = \beta$ is equivalent to
- σ^{-1} occur in the shuffle of α^{-1} and β^{-1}.

3.3. Free Quasi symmetric functions

At this stage a simple remark is required. To be able to later take care of multiplicities, instead of working with languages, it is better to work with their characteristic series. The language $L_\sigma(\mathbb{A})$ is then replaced by the noncommutative formal series

$$\mathbf{F}_\sigma(\mathbb{A}) = \sum_{\mathrm{exec}(w)=\sigma} w \in \mathbb{Z}\langle\mathbb{A}\rangle \ . \tag{30}$$

Thus we work in a sub-algebra of the free algebra.

Definition 10. *The subalgebra of* $\mathbb{C}\langle\mathbb{A}\rangle$

$$\mathbf{FQSym}(\mathbb{A}) = \bigoplus_{n\geq 0} \bigoplus_{\sigma\in\mathfrak{S}_n} \mathbb{C}\,\mathbf{F}_\sigma(\mathbb{A}) \tag{31}$$

is called the algebra of free quasi-symmetric functions.

It is convenient as this point to take an infinite alphabet $\mathbb{A}$. Indeed, if $\mathbb{A}$ is infinite then the structure of $\mathbf{FQSym}(\mathbb{A})$ is independent of $\mathbb{A}$, the resulting algebra is denoted **FQSym**. Note that there is an empty permutations () and that $F_{()} = \epsilon$ which can be identified with the scalar 1.

Thus the theorem 9 is now seen as the product rule of **FQSym**:

Proposition 11 (Duchamp-H.-Thibon [6]). *$\alpha \in \mathfrak{S}_m$ and $\beta \in \mathfrak{S}_n$. Then,*

$$\mathbf{F}_\alpha\mathbf{F}_\beta = \sum_{\sigma\in\alpha \,⧢\, \beta[m]} \mathbf{F}_\sigma \tag{32}$$

This was the original definition of Malvenuto-Reutenauer [25]. For example:

$$\begin{aligned}\mathbf{F}_{132}\mathbf{F}_{21} &= \mathbf{F}_{13265} + \mathbf{F}_{13625} + \mathbf{F}_{13652} + \mathbf{F}_{16325} + \mathbf{F}_{16352} \\ &+ \mathbf{F}_{16532} + \mathbf{F}_{61325} + \mathbf{F}_{61352} + \mathbf{F}_{61532} + \mathbf{F}_{65132}\,.\end{aligned}$$

3.4. The co-product of **FQSym**

We want now to give **FQSym** a structure of a Hopf algebra, namely we need to define a coproduct. We will use an adapted alphabet doubling trick. From now on all alphabet are supposed infinite.

Definition 12. *Let $\mathbb{A}$ and $\mathbb{B}$ be two infinite, totally ordered, mutually commuting alphabets. The* ordered sum *$\mathbb{A}\hat{+}\mathbb{B}$ of $\mathbb{A}$ and $\mathbb{B}$ is the union of $\mathbb{A}$ and $\mathbb{B}$ where the variables of $\mathbb{A}$ are smaller than the variables of $\mathbb{B}$.*

Then the coproduct of F_σ defined is defined as follows: We expand F_σ over the alphabet $\mathbb{A}\hat{+}\mathbb{B}$. Since the variable of $\mathbb{A}$ and $\mathbb{B}$ commute mutually, we reorder the resulting expression to put the letters from $\mathbb{A}$ on the left and those from $\mathbb{B}$ on the right. We have now an expression in $\mathbb{K}\langle\mathbb{A}\rangle\,\mathbb{K}\langle\mathbb{B}\rangle$. But it happens, and this is the only thing to prove, that this expression actually belongs to the sub-algebra $\mathbf{FQSym}(\mathbb{A})\mathbf{FQSym}(\mathbb{B})$. Then we use the tensor notation to get an element of $\mathbf{FQSym}(\mathbb{A}) \otimes \mathbf{FQSym}(\mathbb{B}) \approx \mathbf{FQSym} \otimes \mathbf{FQSym}$. To summarize the coproduct is defined by

$$\mathbf{F}_\sigma \longmapsto \mathbf{F}_\sigma(\mathbb{A}\hat{+}\mathbb{B}) \longmapsto \sum \mathbf{F}_\alpha(\mathbb{A})\mathbf{F}_\beta(\mathbb{B}) \longmapsto \sum \mathbf{F}_\alpha \otimes \mathbf{F}_\beta \tag{33}$$

For example, let $\mathbb{A} = \{a < b < \cdots\}$ and $\mathbb{B} = \{A < B < \cdots\}$. In $\mathbb{A}\hat{+}\mathbb{B}$, one has $z < A$. then by definition

$$\mathbf{F}_{312} = \sum_{x<y\leq z} yzx = \sum_{x<y<z} yzx + \sum_{x<y} yyx$$

This given

$$\begin{aligned}
\mathbf{F}_{312}(\mathbb{A}\hat{+}\mathbb{B}) &= \; bba + bca \;+\; bAa \;+ AAa + ABa + BBA + \cdots \\
&= bba\,1 + bca\,1 + \; ba\,A \;+ a\,AA + a\,AB + 1\,BBA + \cdots \\
\Delta(\mathbf{F}_{312}) &= \; \mathbf{F}_{312}\otimes 1 \;+ \mathbf{F}_{21}\otimes\mathbf{F}_1 + \;\mathbf{F}_1\otimes\mathbf{F}_{12} \;+\; 1\otimes\mathbf{F}_{312}
\end{aligned}$$

The precise rule is given by

Proposition 13 (Duchamp-H.-Thibon [6]). *The coproduct in* **FQSym** *is given by*

$$\Delta(\mathbf{F}_\sigma) = \sum_{k=0}^{n} \mathbf{F}_{\mathrm{Std}(w_1\ldots w_k)} \otimes \mathbf{F}_{\mathrm{Std}(w_{k+1}\ldots w_n)}\,, \tag{34}$$

for all permutation $\sigma = w_1 \ldots w_n$.

The proof is very similar to Theorem 9 and can be found in [6].

At this stage some remarks are in order. By construction, the co-product is co-associative because $(\mathbb{A}\hat{+}\mathbb{B})\hat{+}\mathbb{C} = \mathbb{A}\hat{+}(\mathbb{B}\hat{+}\mathbb{C})$. Moreover the compatibility relation holds because the expansion of $\mathbf{F}_\alpha(\mathbb{A}\hat{+}\mathbb{B})\mathbf{F}_\beta(\mathbb{A}\hat{+}\mathbb{B})$ is the same as the expansion of $\mathbf{F}_\alpha(\mathbb{A})\mathbf{F}_\beta(\mathbb{A})$ provided $\mathbb{A}$ and $\mathbb{B}$ are infinite. Thus the only non-trivial thing is that

$$\mathbf{FQSym}(\mathbb{A}\hat{+}\mathbb{B}) \subset \mathbf{FQSym}(\mathbb{A})\mathbf{FQSym}(\mathbb{B})\,. \tag{35}$$

which is precisely the result of the previous propositions. Hence we have proved for free that

Theorem 14. **FQSym** *is a Hopf algebra.*

This illustrate the strategy of the construction of Hopf algebra by realizations. Let us summarize it. The element of the Hopf algebra are defined (realized) as a vector space H of some kind of complicated polynomials (commutative or not). The product is inherited from the product of polynomials. The co-product is defined using an alphabet doubling trick. Then one must prove that the space H is stable by the product and the coproduct. This is usually done by proving an explicit formula. Then for free we have the associativity, the co-associativity and the compatibility.

This is a interesting exercise to prove the compatibility relation using the explicit formulas for the product and co-product.

Going back to the bubble sort, The coproduct answers the following question: If a word w is sorted by a permutation $\sigma = \mathrm{exec}(w)$, what is the execution of the sort of the sub-word obtained from the k biggest ($n-k$ smallest) letters of the word w ?

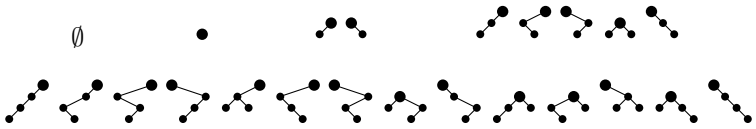
4. The binary search trees

We want to apply the preceding strategy to define a Hopf algebra on binary rooted trees. This algebra has been first defined by Loday and Ronco [21] and is closely related to the renormalisation Hopf algebra of Connes and Kreimer [5]. As in the previous section, we start with a simple and powerful algorithm from computer science, namely the binary-searching algorithm. The reader interested in this algorithm can refer to [15]. The construction given here is fully described in [9,10,12].

4.1. The binary search insertion algorithm

Through this paper, by *binary tree* we means (un-complete) planar rooted binary tree, that is a binary tree is either void $\emptyset$ or a pair of (possibly void) binary trees grafted on a node. The size of a binary tree is it's number of node. The number of binary trees of size n is the Catalan number $C_n := \frac{\binom{2n}{n}}{n+1}$. Here are the first value together with the associated trees:

$$1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796$$



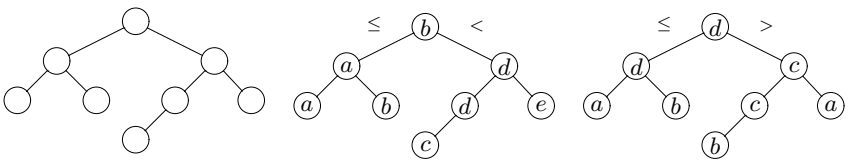
By labeled trees we means a tree with a label attached to each node, the label are taken either from the alphabet $\mathbb{A}$ or from $\mathbb{N}$.

Definition 15. *A* binary search tree *T is a labeled binary tree such that for each node n the label of n is greater or equal than all the labels of the left subtree and strictly smaller than all the label of the right one.*

A decreasing trees *T is a labeled binary tree such that for each node n the label of n is greater or equal than all the labels of the left subtree and strictly greater than all the label of the right one.*

A labeled trees which is labeled by the first natural number $1, \ldots, n$ where each number appear only once is called standard.

Here are some examples:



binary tree (BT) search tree (BST) decreasing tree (DT)

The fundamental properties of binary search trees is given by the following proposition.

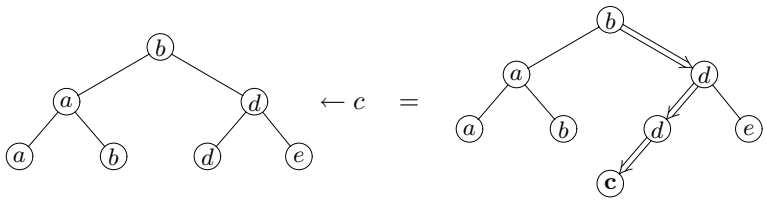
Proposition 16. *If T is a binary search tree and a is a letter, there exists one and only one position to add a leaf labeled a such that the resulting tree $T \leftarrow a$ is a binary search tree.*

It is proved by the following algorithm.

Algorithm 17 (Binary search tree insertion).
INPUT: a binary search tree T and an letter a
OUTPUT: the binary search tree $T \leftarrow a$.

- *if T is empty then* return *the tree* Ⓐ*;*
- *compare a with the root of T;*
- *if it's smaller or equal insert recursively a in the* left *subtree;*
- *if it's greater insert recursively a in the* right *subtree.*

Here is an example:



Then the insertion of a word w is the result of the consecutive insertion of its letters the resulting tree is denoted by $\mathbf{Tree}(w)$. The reader have to be careful that for compatibility with the convention of [21], we decided to read the letters of the word from *right to left*. The figure 1 shows the insertion of the word *cadbaedb*. The meaning of the second tree will be explained later.

4.2. The sylvester monoid

In this subsection, we first focus on the following question. Given a binary search tree T, how to describe the set of words that inserts to T ? We are lead to a monoïd structure on binary search trees called the sylvester monoïd.

We start be reading some words from a tree

Definitions 18.
- *The* infix reading *of a labeled tree is the word w obtained by reading recursively the left subtree, the root and the right subtree;*
- *the* (left to right) postfix reading *of a tree T is the word w_T obtained by reading the left subtree, then the right and finally the root.*

It is clear that the infix reading gives the sorting of the word. This is the binary search tree sorting algorithm.

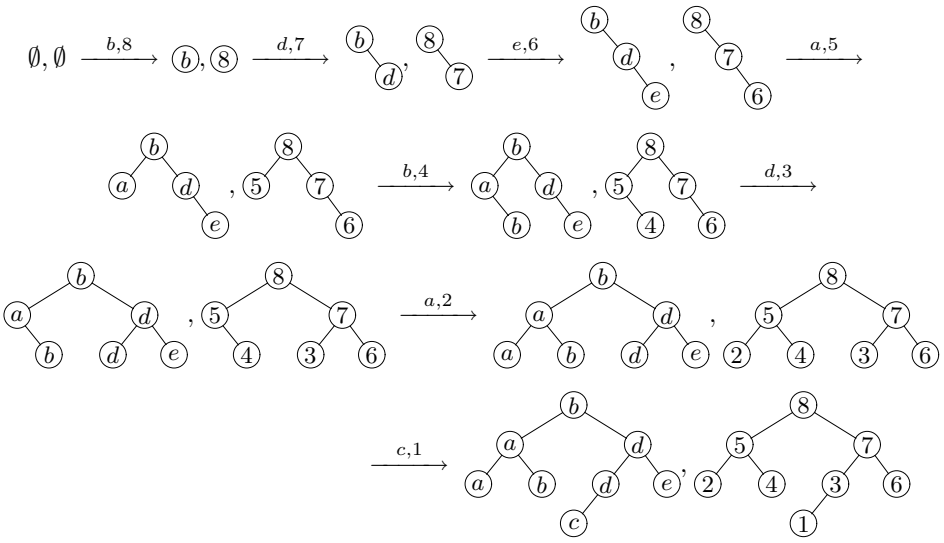


Figure 1. sylvester insertion of the word *cadbaedb*.

Proposition 19. *Let T be a binary search tree and w_T its postfix reading. Then* $\mathbf{Tree}(w_T) = T$. *Moreover w_T is the smallest word (for the lexicographic order) w such that* $\mathbf{Tree}(w) = T$.

The word corresponding to the previous tree is the word $w_{BT} = abacdedb$. Note that the biggest word w such that $\mathbf{Tree}(w) = T$ is obtained by a right to left postfix reading, in our example we get $ecddbaab$.

Let us first define the sylvester monoid by means of congruencies without using any trees.

Definition 20. *Let w_1 and w_2 two words. They are said to be sylvester adjacent if there exists three words u, v, w and three letters $A \leq B < C$ such that*

$$w_1 = u\,AC\,v\,B\,w \quad \textit{and} \quad w_2 = u\,CA\,v\,B\,w\,. \tag{36}$$

The sylvester equivalence is the transitive closure of the sylvester adjacency. That is, two words u, v are sylvester equivalent if there exists a chain

$$u = w_1, w_2, \ldots, w_k = v \tag{37}$$

of words such that w_i and w_{i+1} are adjacent for a all i. In this case we write $u \equiv_{sylv} v$.

Definition 21. *The* sylvester monoid $\mathrm{Sylv}(A)$ *is the quotient of the free monoid A^* by the sylvester equivalence:* $\mathrm{Sylv}(A) := A^*/\equiv_{sylv}$.

Note that $\equiv_{sylv}$ is a congruence on A^*, that is for any words u, v_1, v_2, w such that $v_1 \equiv_{sylv} v_2$ then $uv_1w \equiv_{sylv} uv_2w$. For example the class of the word 21354 is the set

$$\{52134, 25134, 21534, 21354\}.$$

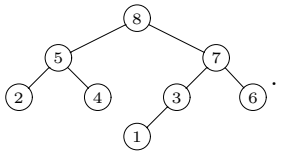
The fundamental theorem is the following:

Theorem 22. *Two words u and v are equivalent if and only if they corresponds to the same binary search tree:* $\mathbf{Tree}(u) = \mathbf{Tree}(v)$.

Consequently the postfix readings $(w_T)_T$ of the binary search trees gives a section of the sylvester monoid, that is there is one and only one tree-word in each equivalence class of $\equiv_{sylv}$.

This is very similar to a classical construction in algebraic combinatoric called the plactic monoïd [18,23]. The plactic monoïd is related to an algorithm called Schensted algorithm the same way the Sylvester monoïd is related to the binary search algorithm.

Actually, it in possible to have not only the analogue of Schensted insertion, but also the full Robinson-Schensted correspondence. Suppose that $\sigma \in \mathfrak{S}_n$ is a permutation of $\{1, 2, \ldots, n\}$. To σ we associate a decreasing tree $\mathrm{DT}(\sigma)$ as follows. The root is labeled by the biggest letter n of σ and if as a word $\sigma = unv$, then the left subtree is $\mathrm{DT}(u)$ and the right subtree is $\mathrm{DT}(v)$. For each $w \in A^*$, we set $\mathbf{Q}(w) = \mathrm{DT}((\mathrm{Std}\, w)^{-1})$. For example, with $w = cadbaedb$, one has $\mathrm{Std}(w) = 51632874$, the inverse is $\mathrm{Std}(w)^{-1} = 25481376$, the decreasing tree is therefore

$\mathbf{Q}(cadbaedb) =$ 

.

The following theorem is the analogue of the Robinson-Schensted correspondence $w \longmapsto (\mathbf{P}(w), \mathbf{Q}(w))$, where $\mathbf{P}(w)$ and $\mathbf{Q}(w)$ are two young tableaux of the same shape ($\mathbf{Q}(w)$ is standard). The trees play here the role of the tableaux [18,23].

Theorem 23. *For each word $w \in A^*$, the tree $\mathbf{Tree}(w)$ is obtained by replacing in $\mathbf{Q}(w)$ each label i by the i-th letter of w.*

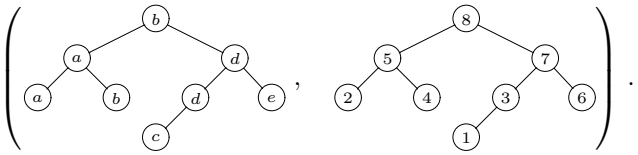
The tree $\mathbf{Q}(w)$ records the reverse order of creation of the node in the binary insertion of w.

The map $w \longmapsto (\mathbf{P}(w), \mathbf{Q}(w))$ is a bijection between A^* and the set of pairs of binary search trees and standard decreasing tree of the same shape (unlabeled tree).

The following theorem is the analogue of the Robinson-Schensted correspondence $w \longmapsto (\mathbf{P}(w), \mathbf{Q}(w))$, where $\mathbf{P}(w)$ and $\mathbf{Q}(w)$ are two young tableaux

of the same shape ($\mathbf{Q}(w)$ is standard). The trees play here the role of the tableaux [18,23].

The two first points are easy to see and can be checked on the following example:

cadbaedb $\longmapsto$ 

The third point is proved by giving the converse bijection: the word corresponding to a pair (BST, DT) is obtained by reading the labels of BST in the order of the label of DT. Note that, in opposition to the plactic insertion the labels does never move until they reach a leaf.

4.3. The algebra of trees

We want now to investigate the following question. When we project the $F_\sigma(\mathbb{A})$ into the Sylvester algebra

$$\mathbf{F}_\sigma(\mathbb{A}) = \sum_{\mathrm{exec}(w)=\sigma} w \longmapsto \mathbf{F}_\sigma(\mathbb{A}/\equiv_{sylv}) := \sum_{\mathrm{exec}(w)=\sigma} \mathbf{Tree}(w)\,, \tag{38}$$

what remains of the Hopf-algebra structure ? Let us denote $\mathbf{G}_\sigma = \mathbf{F}_{\sigma^{-1}}$ so that $\mathbf{G}_\sigma(\mathbb{A}) = \sum_{\mathrm{Std}(w)=\sigma} w$. It is clear that $\mathbf{Tree}(std(w))$ and $\mathbf{Tree}(w)$ have the same shape. Moreover, there is only one BST of a given shape so that

Proposition 24 (compatibility with standardization). *Let u, v be two words. Then the following are equivalent*

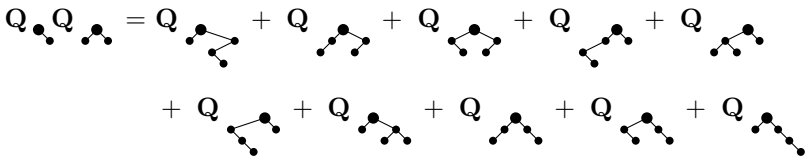
- $u \equiv_{sylv} v$*;*
- $\mathrm{Std}(u) \equiv_{sylv} \mathrm{Std}(v)$ *and for all letter a one has $|u|_a = |v|_a$,*
- $\mathrm{Sh}(\mathrm{Std}(u)) = \mathrm{Sh}(\mathrm{Std}(u))$ *and for all letter a one has $|u|_a = |v|_a$,*

where $|w|_a$ denotes the number of a in the word w.

Therefore $\mathbf{G}_\sigma(\mathbb{A}/\equiv_{sylv})$ depends only on the shape of $\mathbf{Tree}(\sigma)$. Let us define

$$\mathbf{Q}_T = \mathbf{G}_\sigma(\mathbb{A}/\equiv_{sylv}) \tag{39}$$

for any σ such that $\mathrm{Sh}(\mathbf{Tree}(\sigma)) = T$. Then the monoid structure of $\mathrm{Sylv}(A)$ makes it clear that the product $\mathbf{G}_\sigma\mathbf{G}_\mu(\mathbb{A}/\equiv_{sylv})$ depends only on the shape of σ and μ. Therefore the quotient $\mathbf{FQSym}(\mathbb{A}/\equiv_{sylv})$ inherits naturally an algebra structure. Here is an example of product



(40)

To deal with the co-algebra structure we need to investigate $\mathbf{G}_\sigma(\mathbb{A}\hat{+}\mathbb{B}/\equiv_{sylv})$. This is done by the following proposition. For any subset I of the alphabet A and any word w on A, let us denote w/I the word obtained from w by erasing the letters that are not in I.

Proposition 25 (compatibility with restriction to intervals). *Suppose that I is an interval of A. Then $u \equiv_{sylv} v$ implies $u/I \equiv_{sylv} v/I$.*

As a consequence, since $\mathbb{A}$ and $\mathbb{B}$ are intervals of the alphabet $\mathbb{A}\hat{+}\mathbb{B}$ on has that $\mathbf{G}_\sigma(\mathbb{A}\hat{+}\mathbb{B}/\equiv_{sylv})$ depends only on the shape of σ. Therefore the quotient $\mathbf{FQSym}(\mathbb{A}/\equiv_{sylv})$ inherits naturally a co-algebra structure. Here is an example of co-product

$$\begin{aligned}\Delta\mathbf{Q}_{\bullet(\bullet\bullet)} &= \mathbf{Q}_{\bullet(\bullet\bullet)} \otimes 1 + \mathbf{Q}_{\bullet(\bullet\bullet)} \otimes \mathbf{Q}_{\bullet} + \mathbf{Q}_{\bullet\bullet} \otimes \mathbf{Q}_{\bullet\bullet} \\ &+ \mathbf{Q}_{\bullet\bullet}\otimes \mathbf{Q}_{\bullet\bullet} + \mathbf{Q}_{\bullet\bullet} \otimes \mathbf{Q}_{(\bullet\bullet)\bullet} + \mathbf{Q}_{\bullet}\otimes \mathbf{Q}_{\bullet(\bullet\bullet)} \\ &+ 1 \otimes \mathbf{Q}_{(\bullet\bullet)\bullet}\end{aligned} \tag{41}$$

Hence we have proved that

Theorem 26. *The equivalence relation $\equiv_{sylv}$ is compatible with the Hopf-algebra structure on* **FQSym**. *The quotient Hopf algebra is a Hopf algebra whose basis are indexed by binary trees.*

This Hopf algebra is isomorphic to the algebra of Loday and Ronco.
Actually we could have worked in a dual way:

Theorem 27. *The space spanned by*

$$\mathbf{P}_T := \sum_{\mathrm{DT}(w)=T} w = \sum_{\mathrm{Sh}(\mathrm{BST}(\sigma))=T} \mathbf{F}_T \tag{42}$$

if a sub-hopf algebra of **FQSym** *which is dual to* $\mathbf{FQSym}(\mathbb{A}/\equiv_{sylv})$.

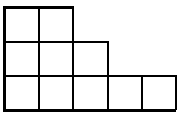
The compatibility with restriction to intervals shows that the vector space spanned by the $\mathbf{P}_T$ is stable under the product.

5. Toward a general theory

5.1. Free Schur functions

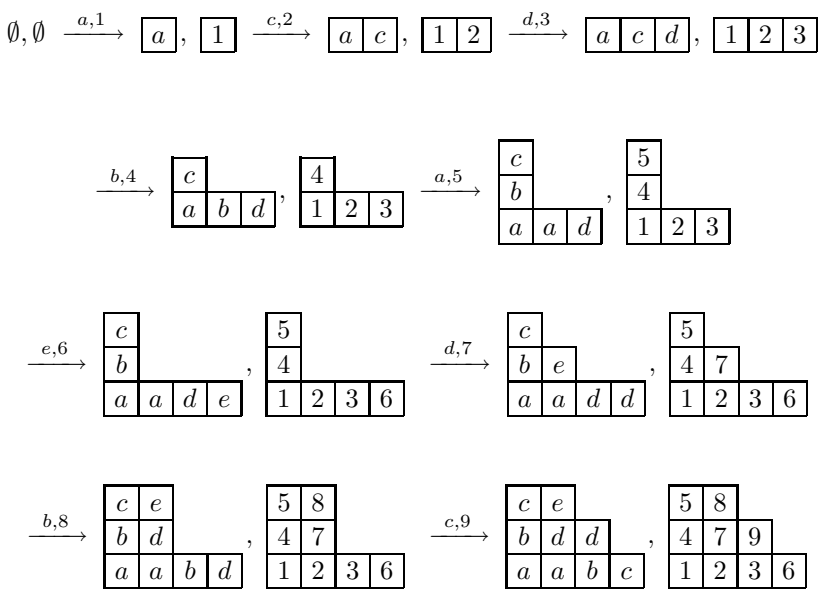
In the preceding section we used a certain relation on words to define a quotient Hopf algebra of **FQSym**. Actually, we only used very few properties of this congruences and there are several other well known examples.

Let's just describe without proof another very important one. It is related to Robinson-Schensted algorithm. We will not recall the algorithm, the reader should refer to [23] for the details. Recall a partition $\lambda = (\lambda_1 \geq \cdots \geq \lambda_k$ of n is a non-increasing sequence of positive numbers of sum n. A partition is depicted by a so called Ferrer's diagram as follows: the partitions $(5, 3, 2)$ is depicted as



Then a filling of the boxes of such a diagram is a tableau if and only if the content of the boxes are increasing along rows and strictly increasing along columns.

The Robinson-Schented correspondence is a bijection from the set of words to the set of pairs of tableaux and standard tableaux of the same shape. The first tableau corresponding to a word w in denoted $P(w)$ the second $Q(w)$. The following example shows the algorithm on the word *acdbaedbc*.



The quotient of the free monoïd by the relation "have the same $P(w)$ tableau" is a monoïd called the plactic monoïd. It can be equivalently defined by Knuth relations:

Theorem 28. *Let $\equiv_{plact}$ be the transitive closure of the two following relations*

$$\cdots xzy \cdots \equiv \cdots zxy \cdots \quad \textit{for} \quad x \leq y < z \tag{43}$$

$$\cdots yxz \cdots \equiv \cdots yzx \cdots \quad \textit{for} \quad x < y \leq z \tag{44}$$

Then, for two words w_1 and w_2, the equality $P(w_1) = P(w_2)$ holds if and only if $w_1 \equiv_{plact} w_2$.

Here are two examples of plactic rewriting.

$$aba\mathbf{acb}c \equiv aba\mathbf{cab}c \quad and \quad aca\mathbf{bdb}c \equiv aca\mathbf{dbb}c$$

We then define

$$\mathbb{S}_t := \sum_{\mathrm{Tab}(\sigma)=t} \mathbf{F}_\sigma = \sum_{Q(w)=t} w\,, \tag{45}$$

where $w \mapsto (P(w), Q(w))$ is the usual Robinson-Schensted map. Then we have the following analogue of Theorem 27

Theorem 29. *The space spanned by*

$$\mathbb{S}_t := \sum_{\mathrm{Tab}(\sigma)=t} \mathbf{F}_\sigma = \sum_{Q(w)=t} w\,, \tag{46}$$

if a sub-hopf algebra of **FQSym** *which is dual to* $\mathbf{FQSym}(\mathbb{A}/\equiv_{plact})$.

This provide a very simple proof of the Littlewood-Richardson rule for computing the multiplication of Schur functions or equivalently for computing the decomposition of the tensor product of representations of GL_N.

5.2. General theorems

The planar binary trees Hopf algebra and the free symmetric functions Hopf algebra are actually two examples of a more general construction. The main data is a plactic-like monoïd. Let summarize the following definitions.

Definitions 30. *A* congruence $\equiv$ *on the free monoïd* A^* *is an equivalence relation which is compatible with the concatenation, that is, for any words* u, v_1, v_2, w *such that* $v_1 \equiv v_2$ *then* $uv_1w \equiv uv_2w$.

A congruence $\equiv$ *is* generated by transpositions *if it is the transitive closure of a (not necessarily finite) set of relation of the type*

$$uabv \equiv ubav \tag{47}$$

where u, v *are words and* a, b *letters.*

We are now in position to state the main theorem.

Theorem 31. *Suppose that* $\equiv$ *is a congruence generated by transpositions which is compatible with the standardization and the restriction to intervals. For any class* C *of standard words* (i.e.: *permutations) for the congruence* $\equiv$, *let*

$$\mathbf{P}_t := \sum_{\sigma\in C} \mathbf{F}_\sigma\,. \tag{48}$$

Then the space spanned by $\mathbf{P}_t$ *is a sub Hopf algebra of* **FQSym** *which is dual to the quotient* $\mathbf{FQSym}(\mathbb{A}/\equiv)$.

The proof is essentially the same than in the case of the plactic monoïd and ca be found in [6]. There are more instances of the following construction. For example, the hypoplactic monoïd [16], leading to the pair quasi-symmetric/noncommutative symmetric functions.

References

[1] E. Abe, *Hopf algebras*, Cambridge tract in mathematics, Cambridge University Press 1980.

[2] N. Bergeron, F. Hivert and J.-Y. Thibon, *The peak algebra and the Hecke-clifford algebras at $q = 0$*, J. Combinatorial Theory A, **117** (2004), 1–19.

[3] C. Brouder and A. Frabetti, *Renormalization of QED with planar binary trees*, Europ. Phys. J. C **19** (2001), 715–741.

[4] C. Brouder and A. Frabetti, *QED Hopf algebras on planar binary trees*, J. Algebra, **267** (2003), no. 1, 298–322.

[5] A. Connes and D. Kreimer, *Hopf algebras, renormalization and noncommutative geometry*, Comm. Math. Phys. **199** (1998), no. 1, 203–242.

[6] G. Duchamp, F. Hivert and J.-Y Thibon, *Noncommutative symmetric functions. VI. Free quasi-symmetric functions and related algebras*, Internat. J. Algebra Comput. 12 (2002), no. 5, 671–717.

[7] I.M. Gelfand, D. Krob, B. Leclerc, A. Lascoux, V.S. Retakh and J.-Y. Thibon, *Noncommutative symmetric functions*, Adv. in Math., **112** (1995), 218–348.

[8] I. Gessel, *Multipartite P-partitions and inner products of skew Schur functions*, in *ńCombinatorics and algebraż*, C. Greene, ed., Contemporary Mathematics **34** (1984), 289–301.

[9] F. Hivert, J.-C. Novelli and J.-Y. Thibon, *Un analogue du monoïde plaxique pour les arbres binaires de recherche,* C. R. Acad. Sci. Paris, **335** (2002), 1–4.

[10] F. Hivert, J.-C. Novelli and J.-Y. Thibon, *Sur quelques propriétés de l'algèbre des arbres binaires,* C. R. Math. Acad. Sci. Paris, **337(9)** (2003), 565–568.

[11] F. Hivert and N. Thiéry, *Mupad-combinat, an open-source package for research in algebraic combinatorics,* Séminaire Lotharingien de Combinatoire, **51** (2003), 70 p. electronic.

[12] F. Hivert, J.-C. Novelli and J.-Y. Thibon, *The algebra of binary search trees*, Theoretical Computer Science, **339(1)** (2005), 129–165.

[13] F. Hivert, J.-C. Novelli and J.-Y. Thibon, *Yang-Baxter bases of 0-Hecke algebras and representation theory of 0-Ariki-Koike-Shoji algebras*, Advances in Mathematics, to appears.

[14] S. A. Joni and G.-C. Rota, *Coalgebra and bialgebra in combinatorics*, Stud. an Appl. Math. **61** (1979) 93–139.

[15] D. E. Knuth, *The art of computer programming, vol.3: Sorting and searching*, (Addison-Wesley, 1973).

[16] D. Krob and J.-Y. Thibon, *Noncommutative symmetric functions IV: Quantum linear groups and Hecke algebras at $q = 0$*, J. Alg. Comb., **6** (1997), no. 4, 339–376.

[17] D. Krob and J.-Y. Thibon, *Noncommutative symmetric functions V: A degenerate version of $U_q(gl_N)$*, Internat. J. Algebra Comput., **9** (1999), no. 3-4, 405–430.

[18] A. Lascoux and M.-P. Schützenberger, *Le monoïde plaxique* in *Noncommutative structures in algebra and geometric combinatorics* (Naples, 1978), pp. 129–156, Quad. Ricerca Sci., 109, CNR, Rome, 1981.

[19] J.-L. Loday, *Dialgebras and Related Operads*, Lecture Notes in Mathematics, **1763**, Springer-Verlag, 2001,7–66

[20] J.-L. Loday, *Realization of the Stasheff polytope*, math.AT/0212126, to appear in Arch. Math. (Basel).

[21] J.-L. Loday and M.O. Ronco, *Hopf algebra of the planar binary trees*, Adv. Math., **139** (1998) n. 2, 293–309.

[22] J.-L. Loday and M.O. Ronco, *Order structure on the algebra of permutations and of planar binary trees*, J. Algebraic Combin., **15** (2002) n. 3, 253–270.

[23] A. Lascoux, B. Leclerc and J.-Y. Thibon, *The plactic monoid*, Chapter 5 of M. Lothaire, *Algebraic Combinatorics on Words*, Cambridge University Press.

[24] P. A. MacMahon, *Combinatorial analysis*, Cambridge University Press, (1915) Chelsea reprint 1960.

[25] C. Malvenuto and C. Reutenauer, *Duality between quasi-symmetric functions and Solomon descent algebra*, J. Algebra, **177** (1995), 967–982.

[26] S. Montgomery, *Hopf algebras and their action on rings*, AMS 1994, 240p.

[27] G.-C. Rota, *Hopf algebra methods in combinatorics*, Colloques internationaux C.N.R.S, Orsay (1976) 363–365, reprinted in Gian-Carlo in combinatorrics, Birkhauser, 1995.

[28] G.-C. Rota, *Baxter algebras and combinatorial identities I, II*, Bull. A.M.S. **75** (1969) 325–334.

[29] M. Sweedler, *Hopf algebras*, Benjamin 1969.

[30] A. Zelevinsky, *Representations of Finite Classical Groups. A Hopf Algebra Approach*, Lecture Notes in Mathematics, **869**, Springer-Verlag, Berlin-New York, 1981, 184 pp.