Network Rewriting II: Bi- and Hopf Algebras

Lars Hellström

Division of Applied Mathematics, The School of Education, Culture and Communication
Mälardalen University, Box 883, 721 23 Västerås, Sweden
`lars.hellstrom@residenset.net`

Abstract

Bialgebras and their specialisation Hopf algebras are algebraic structures that challenge traditional mathematical notation, in that they sport two core operations that defy the basic functional paradigm of taking zero or more operands as input and producing one result as output. On the other hand, these peculiarities do not prevent studying them using rewriting techniques, if one works within an appropriate network formalism rather than the traditional term formalism. This paper restates the traditional axioms as rewriting systems, demonstrating confluence in the case of bialgebras and finding the (infinite) completion in the case of Hopf algebras. A noteworthy minor problem solved along the way is that of constructing a quasi-order with respect to which the rules are compatible.

1998 ACM Subject Classification F.4.2 Grammars and Other Rewriting Systems

Keywords and phrases confluence, network, PROP, Hopf algebra

Digital Object Identifier 10.4230/LIPIcs.RTA.2015.194

1 Introduction

Bialgebras and Hopf algebras are rarely mentioned in first (or second) abstract algebra courses, but many familiar algebraic and combinatorial [5] structures possess a Hopf algebra structure, which may be viewed as giving a more complete picture of the basic thing than the mere algebra would. For polynomials in one variable x over a field $\mathcal{K}$, one may define the coproduct $\Delta\colon \mathcal{K}[x] \longrightarrow \mathcal{K}[x] \otimes \mathcal{K}[x]$, the counit $\varepsilon\colon \mathcal{K}[x] \longrightarrow \mathcal{K}$, and the antipode $S\colon \mathcal{K}[x] \longrightarrow \mathcal{K}[x]$ as the linear maps which satisfy

$$\Delta(x^n) = \sum_{k=0}^{n} \binom{n}{k} x^k \otimes x^{n-k}, \qquad \varepsilon(x^n) = \begin{cases} 1 & \text{if } n = 0, \\ 0 & \text{otherwise,} \end{cases} \qquad S(x^n) = (-1)^n x^n$$

for all $n \geqslant 0$; this turns $\mathcal{K}[x]$ into a Hopf algebra. For any group G, the corresponding group algebra $\mathcal{K}[G]$ is similarly endowed with coproduct $\Delta\colon \mathcal{K}[G] \longrightarrow \mathcal{K}[G] \otimes \mathcal{K}[G]$, counit $\varepsilon\colon \mathcal{K}[G] \longrightarrow \mathcal{K}$, and antipode $S\colon \mathcal{K}[G] \longrightarrow \mathcal{K}[G]$ defined by

$$\Delta(g) = g \otimes g, \qquad \varepsilon(g) = 1, \qquad S(g) = g^{-1}$$

for all $g \in G$ and then extended to the whole of $\mathcal{K}[G]$ by linearity, that turn $\mathcal{K}[G]$ into a Hopf algebra. If G is finite, then the linear dual of $\mathcal{K}[G]$ will moreover also be a Hopf algebra. Hopf algebras are thus close at hand, but they can for *syntactic* reasons be awkward to work with abstractly.

The simplest way to fully formalise the Hopf algebra concept of coproduct in a classical computational context would be that it is a function which returns a generator object for a finite sequence of pairs of algebra elements, because the basic way to encode a general

26th International Conference on Rewriting Techniques and Applications (RTA'15).
Editor: Maribel Fernández; pp. 194–208
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

element in a vector space tensor product $V \otimes V$ is as a finite sum $a_1 \otimes b_1 + \cdots + a_n \otimes b_n$ where $a_i, b_i \in V$ for all $i = 1, \ldots, n$; note however that neither the length of this sum nor any particular term of it is uniquely determined by the element that the sum encodes. For example, in the polynomial Hopf algebra one can express $\Delta(x)$ as $1 \otimes x + x \otimes 1$, but equally well as $1 \otimes (x+1) + (x-1) \otimes 1$ or $(x+1) \otimes (x+1) - 1 \otimes 1 - x \otimes x$, since these are all the same element of $\mathcal{K}[x] \otimes \mathcal{K}[x]$. That the overall result of a computation should be independent of how such a tensor product element happened to get encoded places far-reaching constraints on what one may do to the a_is and b_is; in particular, all the a_i must be processed in the same way, and all the b_i must be processed in the same way, although a_is need *not* be processed in the same way as the b_i. Hence a more intuitive syntactic interpretation of the coproduct Δ is that it is like a subroutine with one in-parameter and two out-parameters, one of which is the "sequence" of a_i and the other being the corresponding "sequence" of b_i. In a composite expression, the left and the right results of a coproduct may then be used in quite separate places of the expression as a whole.

The counit ε is syntactically even stranger, as it also takes one operand as input, but produces *no* result as output, although it contributes a global factor to the final result of any composite expression of which it is part. Getting a grip on Δ and ε is difficult, and the main reason for this is precisely that they in the natural interpretation go beyond one of the fundamental principles of mathematical notation, namely that each expression is either atomic or a combination of *independent* subexpressions that each contributes one intermediate result to the final combining operation, thus giving every expression an underlying rooted tree structure. The two output results of a coproduct can instead create a syntactic dependence between what from the root looks like separate subexpressions, and the no output results of a counit can leave an expression syntactically disconnected, in both cases invalidating the traditional presumption that an expression is structured like a tree.

One approach for working with bialgebras has been to device special notational extensions to traditional notation, such as the Sweedler [10] notation which however has the drawback of having the bialgebra axioms built in; it cannot be used if one wishes to study the bialgebra axioms themselves. Another approach has been to give up on traditional expressions for equational reasoning, to rather work in the formalism of category theory: instead of an equational proof, one has a huge commutative diagram, where the various paths correspond to expressions, and the facets correspond to applications of axioms. An awkward trait of this approach is that it places considerable emphasis on such elementary issues as the domains of intermediate results at the expense of more structural aspects (like saying

$$\mathbb{R} \xrightarrow{\sin} [-1,1] \xrightarrow{\mathrm{sqr}} [0,1] \xrightarrow{t \mapsto 1-t} [0,1] \xrightarrow{\mathrm{sqrt}} [0,1]$$

instead of $\sqrt{1 - \sin^2 x}$ while aiming to do basic calculus) and a significant disadvantage is that it requires many steps for trivial rearrangements of parentheses.

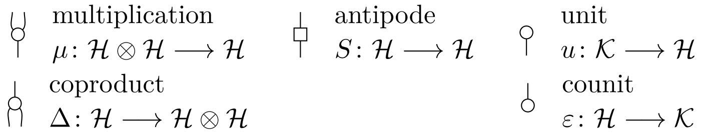
The approach followed here, to the end of examining bi- and Hopf algebras using techniques of rewriting, is instead to adopt a more general expression (formal term) concept, where the underlying structure is a DAG rather than a simple tree. This kind of generalisation is known from the works of for example Hasegawa [2], Lafont [6], and Mimram [8], but the exact realisation of it that will be used here is that of [3]. This *network* concept of more general expressions can be transcribed in terms of the categorical primitives of morphism composition, tensor product, and component permutation, but it is graphical (primarily as in graph, only secondarily as in graphics) and thus more accessible to the human eye. Even better, the matter of whether two categorical expressions are equal modulo the axioms of a symmetric monoidal category (the "rearrangement of parentheses" mentioned above) turns out to be exactly the same as whether the corresponding networks are isomorphic (as graphs with some extra structure). There is much formal nonsense, but once the *many* minutiae of establishing the more general expression concept have been taken care of [3, Secs. 4, 5, 7], rewriting behaves *very much* as we're used to, even if some new phenomena pop up.

What is not a new phenomenon, but deserves to be stated explicitly, is that the rewriting operates on a higher level of abstraction than is usual in applications of rewriting to abstract algebra; the objects being rewritten do not represent/evaluate to elements of the algebra in question, but are rather expressions that could be applied to some tuple of algebra elements. More technically, the objects being rewritten represent (could be taken as evaluating to) multilinear maps $\mathcal{H}^{\otimes n} \longrightarrow \mathcal{H}^{\otimes m}$ which live in the PROP (symmetric monoidal category) of such maps that is generated by the five Hopf algebra operations. Rewrite theories that describe more specific Hopf algebras, for example "the free Hopf algebra generated by a given coalgebra", can be had by extending the generic system described below with extra constant operations representing the generating elements and extra rewrite rules representing how the coproduct and counit act on these generators; the resulting Hopf algebra is then the no-input-one-output component of the generated PROP.

Section 2 gives an introduction to the network formalism for bi- and Hopf algebras. Section 3 presents the axiom system for bialgebras and shows that it constitutes a confluent system of rewrite rules. Section 4 presents the axiom system for Hopf algebras and derives the additional rules needed to make the rewrite system complete. Section 5 shows that the system of the previous section indeed is confluent. Section 6 takes care of a technical detail left aside in the earlier critical pairs/completion oriented sections, namely that of how to construct a compatible order on the set of networks, to ensure termination.

2 Network formalism for bi- and Hopf algebras

In a Hopf algebra $\mathcal{H}$ over a field $\mathcal{K}$, there are five multilinear operations:


multiplication $\mu\colon \mathcal{H}\otimes\mathcal{H} \longrightarrow \mathcal{H}$

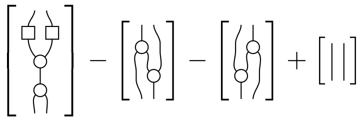
antipode $S\colon \mathcal{H} \longrightarrow \mathcal{H}$

unit $u\colon \mathcal{K} \longrightarrow \mathcal{H}$

coproduct $\Delta\colon \mathcal{H} \longrightarrow \mathcal{H}\otimes\mathcal{H}$

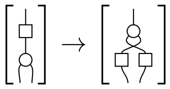
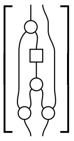
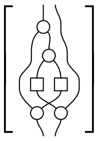
counit $\varepsilon\colon \mathcal{H} \longrightarrow \mathcal{K}$

A bialgebra is not required to have an antipode. The graphic symbols shown are used to denote these operations in network notation expressions (see [3, Sec. 5] for the formal definition of network notation). These networks will be directed acyclic graphs where each inner vertex is one of the above five, and all edges by convention are directed downwards; no arrowheads are drawn. Edges beginning at the top of the network correspond to inputs and edges ending at the bottom correspond to outputs; together, these constitute the *legs* of the network. A network may be interpreted as a "circuit" performing Hopf algebra operations; any antichain k-edge-cut separating input side from output side is then the location of an intermediate result of the circuit; technically such an intermediate result is an element of the tensor power $\mathcal{H}^{\otimes k}$. When occurring as parts of a larger mathematical formula, network expressions are for clarity framed in brackets, like so:



The rightmost of these networks is the identity map $\mathrm{id}^{\otimes 2} = \mathrm{id}\otimes\mathrm{id} : \mathcal{H}^{\otimes 2} \longrightarrow \mathcal{H}^{\otimes 2}$, whereas the first three in categorical notation rather would be $\Delta\circ\mu\circ(S\otimes S)$, $(\mathrm{id}\otimes\mu)\circ(\Delta\otimes\mathrm{id})$, and $(\mu \otimes \mathrm{id}) \circ (\mathrm{id} \otimes \Delta)$. The rewriting formalism applied operates on linear combinations of networks, but the bialgebra and Hopf algebra axioms are all binomial, so the reader may for this paper ignore that aspect. (Rewrite rules always have simple networks, as opposed to formal linear combinations of networks, as left hand sides. Critical pairs/ambiguities thus only arise at simple networks, even though their resolutions might involve linear combinations if there are rules introducing such.)

Rewrite rules act on networks in the pictorially intuitive way of removing a subnetwork isomorphic to the left hand side and instead splicing in a subnetwork isomorphic to the right hand side, making sure that corresponding legs of the left and right hand sides are spliced into the same edge of the network being rewritten. Thus the rule

 can change  into  ;

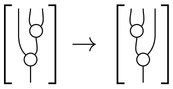
it makes no mathematical difference that the crossing of two edges is shown above the two antipodes in the right hand side of the rule but below them in the spliced network (rightmost), as in this formalism the crossing of two edges is merely a presentational artifact that arises when the abstract network (a graph not given with an embedding) has to be depicted on a two-dimensional page.

Critical pairs (the formal term used in [3] is *decisive ambiguities*) arise when the left hand sides of two rules occur as overlapping subnetworks of some network that they cover completely, at least in the case of the rewrite systems considered here. (In some more general cases, there can be an ambiguity even when there is not an overlap.)

3 The bialgebra axioms and rewriting system

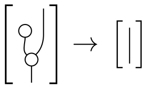
The bialgebra axioms are straightforward to state as network rewrite rules. First, there are the axioms for an associative unital algebra

associativity

$s_1\colon$ 

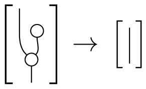
$\mu \circ (\mathrm{id} \otimes \mu) \to \mu \circ (\mu \otimes \mathrm{id})$

left unit

$s_2\colon$ 

$\mu \circ (u \otimes \mathrm{id}) \to \mathrm{id}$

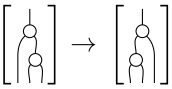
right unit

$s_3\colon$ 

$\mu \circ (\mathrm{id} \otimes u) \to \mathrm{id}$

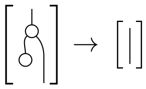
then the dual axioms (obtained from the above by exchanging the roles of inputs and outputs) for a coassociative counital coalgebra

coassociativity

$s_4\colon$ 

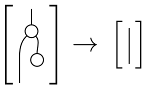
$(\mathrm{id} \otimes \Delta) \circ \Delta \to (\Delta \otimes \mathrm{id}) \circ \Delta$

left counit

$s_5\colon$ 

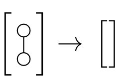
$(\varepsilon \otimes \mathrm{id}) \circ \Delta \to \mathrm{id}$

right counit

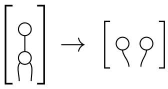
$s_6\colon$ 

$(\mathrm{id} \otimes \varepsilon) \circ \Delta \to \mathrm{id}$

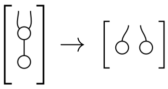
and finally the axioms relating co- and non-co operations

$s_7\colon$ 

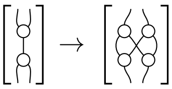
$\varepsilon \circ u \to \mathrm{id}^{\otimes 0}$

$s_8\colon$ 

$\Delta \circ u \to u \otimes u$

$s_9\colon$ 

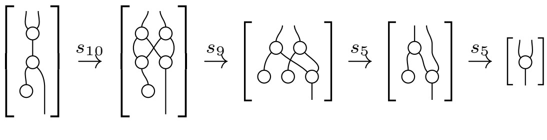
$\varepsilon \circ \mu \to \varepsilon \otimes \varepsilon$

$s_{10}\colon$ 

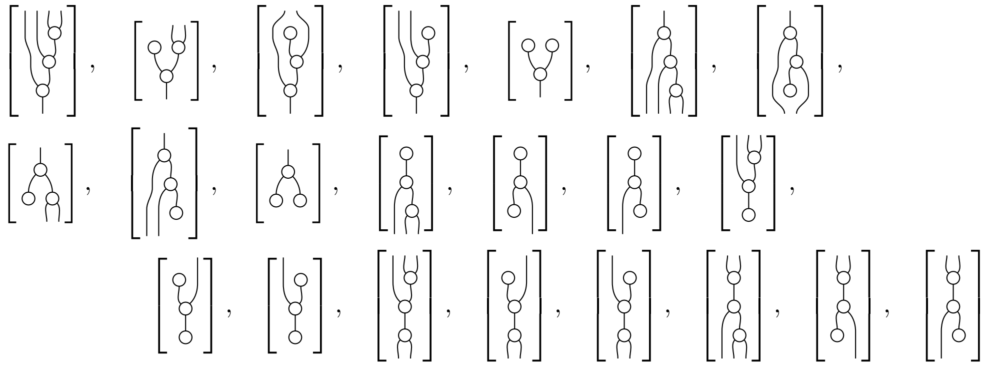
$\Delta \circ \mu \to \mu \otimes \mu \circ \mathrm{id} \otimes \tau \otimes \mathrm{id} \circ \Delta \otimes \Delta$

where $\mathrm{id}^{\otimes 0}$ is the neutral element with respect to the tensor product operation $\otimes$ and τ is a "twist" map defined by $\tau(x \otimes y) = y \otimes x$ for all x and y; it is usually when reaching expressions the size of the right hand side of s_{10} that one starts to appreciate the network notation (and its less formalised kin, such as 'shorthand diagrams' [7] and Penrose's pictorial notation [9]) as an improvement over the raw categorical notation. In more traditional algebraic presentations, axioms s_7 and s_9 are often combined into the claim that ε is a unital algebra homomorphism, whereas axioms s_8 and s_{10} combine into the same claim about Δ. The crossing of edges (or τ twist) in the right hand side of s_{10} is then swept under the rug as a detail of how the multiplication operation of a tensor product algebra $\mathcal{H} \otimes \mathcal{H}$ is defined, but it is an important feature which deserves to be made explicit.

A sequence of rewriting steps modulo the system $\{s_1, s_2, \dots, s_{10}\}$ is



and that is also half of the resolution of the critical pair formed by rules s_{10} and s_5; the other half amounts to just one application of s_5. The full list[1] of networks being sites of critical pairs for this rewriting system is



and these all resolve in a quite straightforward manner. This list was compiled by enumerating all networks that satisfy the conditions of [3, Lemma 10.15]. Together with the quasi-order discussed in Section 6, this meets the conditions of the network rewriting diamond lemma [3, Th. 10.24], and so it follows that:

▶ **Theorem 1.** *The rewriting system* $\{s_k\}_{k=1}^{10}$ *is terminating and confluent.*

Remark on proof. This may seem abrupt, but proofs of confluence using a diamond lemma admit a degree of stylisation that almost render them redundant. Recall that a diamond lemma is a theorem of the form that if certain prerequisites are met, then various claims are equivalent; one of these claims is that a rewriting system is confluent and another that the rewriting system is locally confluent at each critical pair. Proofs relying upon it therefore tends to have two parts: first check that the prerequisites are met, which among other things

---

[1] This list of critical pairs, resolutions of these critical pairs, ditto for the extensions of the rewriting system treated below, and all drawings of networks shown in this paper, were computed using a utility for completion in network rewriting that was written by the author. The homepage of that utility, where its sources are available for download, is currently `http://www.mdh.se/ukk/personal/maa/lhm03/sw/rewriting`

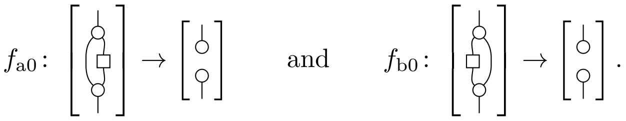
establishes termination, second check the local confluence. But when termination holds, the matter of local confluence becomes algorithmically decidable, so recording the details of those calculations is not essential for the proof. The second part can therefore be abbreviated pretty much to the point of being omitted entirely, and the first part is often completely standard. (For this particular theorem, it is not so standard; the argument underlying termination can be found in Section 6, since the same argument would be used for all rewriting systems in this paper.)

What, on the other hand, may require a great deal of ingenuity and calculations is the construction of the confluent rewrite system. But the rewrite system must be included already in the statement of the theorem, so in a sense such claims tell the reader how to prove them. ◀

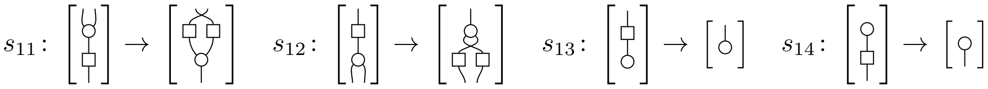
Networks which are built from μ, u, Δ, and ε vertices and moreover are on normal form with respect to $\{s_k\}_{k=1}^{10}$ can be fully characterised. Let $M_0 = u$, $M_1 = \mathrm{id}$, $M_{i+2} = M_{i+1} \circ (\mu \otimes \mathrm{id}^{\otimes i})$ for $i \geqslant 0$. Dually let $D_0 = \varepsilon$, $D_1 = \mathrm{id}$, and $D_{i+2} = (\Delta \otimes \mathrm{id}^{\otimes i}) \circ D_{i+1}$ for $i \geqslant 0$. Then the networks on normal form consist of three layers and have the overall form $A \circ B \circ C$, where $A = \bigotimes_{k=1}^{m} M_{p_k}$ for some numbers $\{p_k\}_{k=1}^{m} \subseteq \mathbb{N}$, the middle B part is a permutation, and $C = \bigotimes_{k=1}^{n} D_{q_k}$ for some numbers $\{q_k\}_{k=1}^{n} \subseteq \mathbb{N}$. In other words, the A part contains all the μ and u, whereas the C part contains all the Δ and ε, and both the A part and the C part are written on left-leaning form.

4 The Hopf algebra axioms and rewriting system

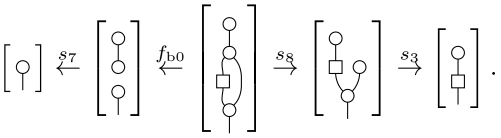
The situation for Hopf algebras is far more complicated. The traditional axiom system for these adds just two axioms to the ten of a bialgebra, namely



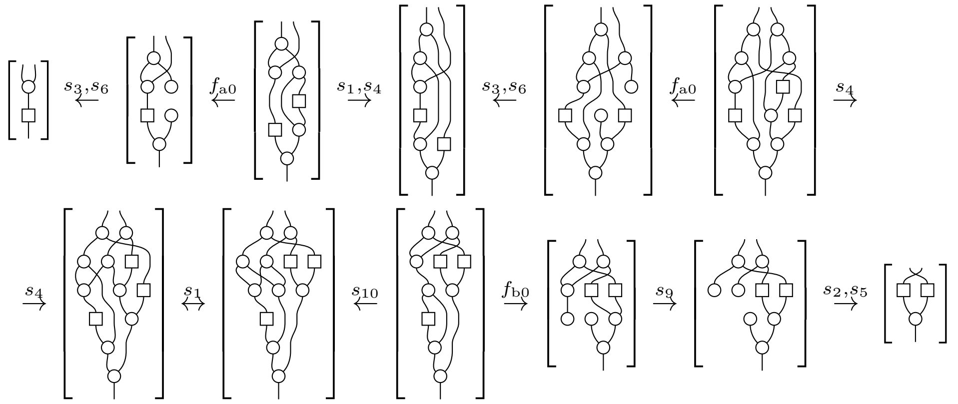
Logically, these two are all that is needed, but in practical calculations one needs to employ a number of derived rules. In particular, there are four rules describing interaction of an antipode with one of the four bialgebra operations:



The rules s_{13} and s_{14} for how an antipode interacts with a counit and unit are fairly straightforward; they are among the first things an automated completion procedure discovers when given the Hopf algebra axioms as input, and the derivation of s_{14} is merely



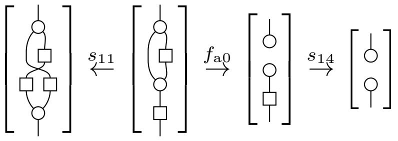
The rules s_{11} and s_{12} for how an antipode interacts with the multiplication and coproduct are on the other hand among the last spurious rules such a procedure discovers; a derivation of s_{11} with some steps combined is



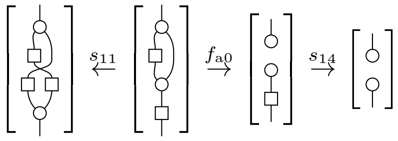
(1)

and the derivation of s_{12} is just the vertical flip (exchanging inputs and outputs, multiplication and coproduct, and unit and counit) of this one.

Given rules s_{11} and s_{12}, it is easy to see that these will form critical pairs with the axioms f_{a0} and f_{b0} that lead to the failed resolutions



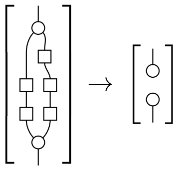
and



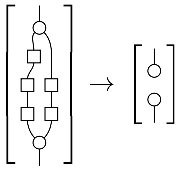
.

This thus calls for the introduction of two derived rules f_{c0} and f_{d0}, which are themselves involved in similar critical pairs, that in turn call for another two derived rules with an extra pair of antipodes and an extra crossing. Since crossing twice takes one back to the original uncrossed state, this second pair of derived rules may be called f_{a1} and f_{b1} as they look just like f_{a0} and f_{b0}, except with two extra antipodes on each of the two paths between coproduct and multiplication:

f_{a1}:

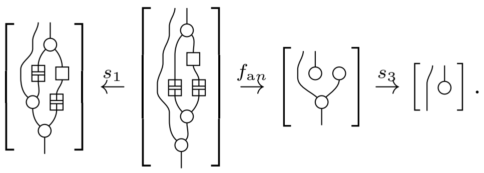


f_{b1}:



Continuing this way, one will generate four infinite families of rules, where the members of a family differ only in how many extra antipodes are inserted between the coproduct and the multiplication in the left hand side, but all rules in a family have the same right hand side. To present this succinctly in network notation, it becomes convenient to introduce a special *double antipode sequence* vertex ⊞, that denotes a path of some $2n$ antipode vertices; the number n will appear as an index in the rule name. Note especially that all double antipode sequence vertices in a single network denote the *same* even number of antipodes. Using this, the f_{an}, f_{bn}, f_{cn}, and f_{dn} families of rules are what is shown in the top row of Figure 1.

Families a–d also form critical pairs with rules s_1 and s_4, that do not resolve using the rules mentioned so far; one example is



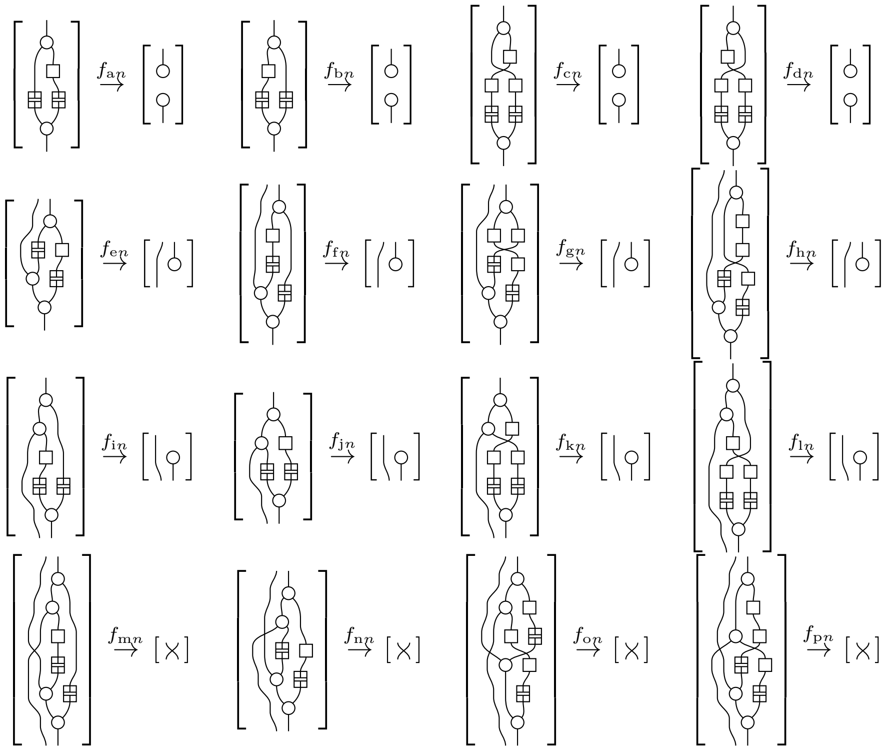
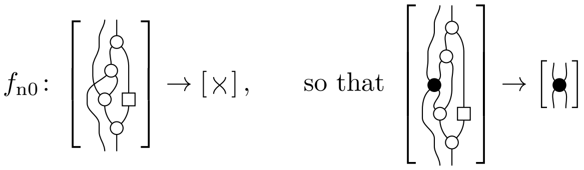


Figure 1 The sixteen infinite families of rewrite rules for Hopf algebras. The letters a–p in the index correspond to the subequation labels in [3, Eq. 1.2], where this system of rewrite rules was first announced.

These failed resolutions thus give rise to additional families of derived rules; with s_1 one gets $f_{\mathrm{e}n}$ through $f_{\mathrm{h}n}$ and with s_4 one gets $f_{\mathrm{i}n}$ through $f_{\mathrm{l}n}$, also defined in Figure 1. Furthermore families e–h form critical pairs with s_4 that give rise to another four families m–p, and those same four families also arise from critical pairs of s_1 and a member of families i–l. But after these last four that bring the total up to sixteen families $\{f_{\mathrm{a}n}\}_{n=0}^{\infty}$ through $\{f_{\mathrm{p}n}\}_{n=0}^{\infty}$, there are no more derived rules to discover; the rewrite system is, as shall be shown in Section 5, complete.

The last four families, the simplest member of which is



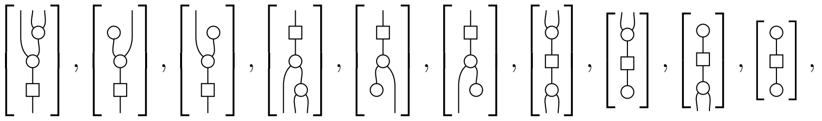
(the filled vertex in that formula is to be read as a placeholder for an arbitrary network expression) do however exhibit an interesting property: they apply even in places where the first input is reachable from the first output. Note that since networks are by definition DAGs, a rewrite formalism for networks may not perform any surgery that would introduce cycles. A simple condition to that effect would be that left hand sides of rules may only be identified with networks in such a way that no directed path exists from a left hand side output to a left hand side input, because that ensures the result of applying the rule is also acyclic no matter what the right hand side looks like. Though simple, this *convexity* condition turns out to be a bit too restrictive in practice, and a more appropriate condition is that a rewrite rule should not introduce any *new* connections; the right hand side should not have a path from input j to output i unless there was such a path in the left hand side.

Remember that a derived rule can be considered a bottled sequence of proof steps exercising more elementary rules; any application of a derived rule can, as far as equational reasoning is concerned, be replaced by a sequence of more elementary steps. When considering such a sequence of steps in the context of a larger network, there is no a priori reason why the union of the subnetworks operated upon in the more elementary steps should always be convex, and in general it will indeed not be. What matters for the validity of a derived rule is merely that the elementary steps it is a parcel for can be carried out in every context where the rule is claimed to apply, and that is certainly the case with rule families m–p.

Experience with completing a modification of the Hopf axiom system suggests that rules will typically become nonconvex as soon as they grow complicated enough. An open problem in the more general case is however that there may exist more connections in the intermediate steps of a rule derivation than there are in neither the final left or right hand sides. In this case the rule is *non-sharp* [3, Def. 10.3], and it may be involved in critical pairs other than the decisive ambiguities, a matter which requires further research [4]. The rewrite rules considered in this paper are however all sharp, so that is not a concern for the results stated here.

5 Confluence of the Hopf system

For the matter of proving confluence of the system of Hopf algebra rules derived in the previous section, one may begin with the system of the fourteen spurious rules s_1 through s_{14}. Since this is a superset of the bialgebra system, all the critical pairs of that system arise again, but they can also be resolved in exactly the same way as there. The additional critical pairs that arise are at the sites



and these also resolve through straightforward calculations. Again putting aside until Section 6 the technical details concerning the construction of a compatible order on the set of networks, it may now be claimed that:

▶ **Theorem 2.** *The rewriting system* $\{s_k\}_{k=1}^{14}$ *is terminating and confluent.*

The normal form modulo $\{s_k\}_{k=1}^{14}$ of a Hopf algebra expression is a three-layered $A \circ B \circ C$ as in the case of a bialgebra, but with the difference that B in this case may contain antipodes. Hence rather than being a simple matching (as permutations are), the B network is in general a disjoint union of paths where each path may contain any number (including 0) of antipode vertices. Adding the sixteen infinite families to the system will reduce that slightly, but not very much.

Continuing with that full rewriting system $F = \{s_k\}_{k=1}^{14} \cup \{f_{\mathrm{a}n}\}_{n=0}^{\infty} \cup \cdots \cup \{f_{\mathrm{p}n}\}_{n=0}^{\infty}$, one may first observe that the spurious rules s_3, s_6, s_7, s_{13}, and s_{14} do not form any critical pairs with the family rules. Rules s_8 and s_9 form critical pairs, but these all resolve very easily as a unit or counit will effectively gobble any vertex to which it becomes adjacent. Rules s_1, s_2, s_4, s_5, s_{11}, and s_{12} are another matter, as they get involved in a rather complicated

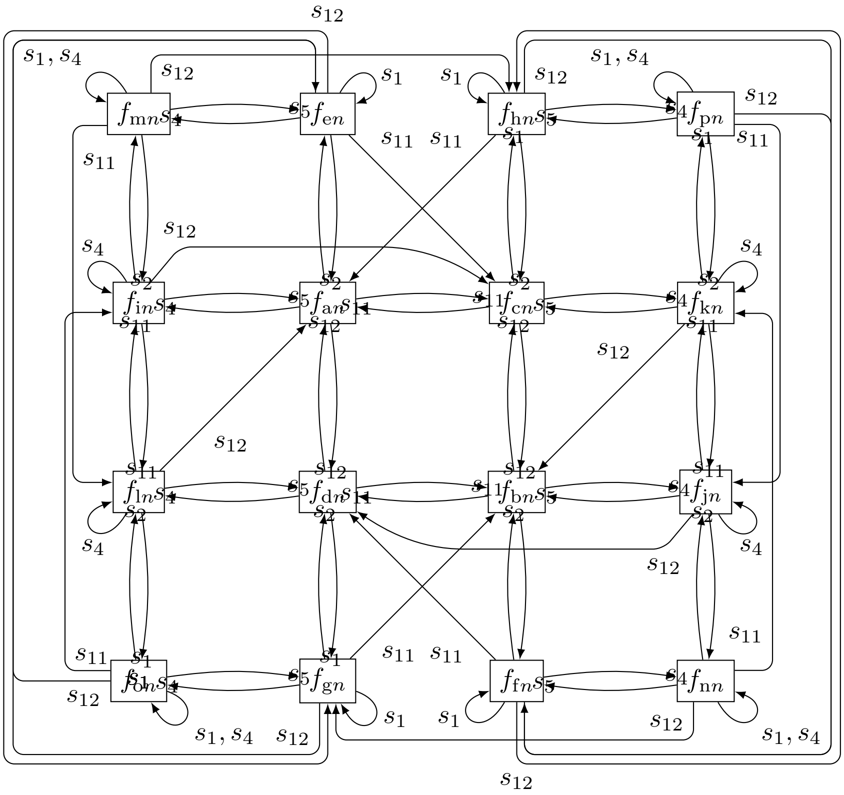


Figure 2 The effect of spurious rules on family rules. Critical pairs formed by one spurious rule from $\{s_1, s_2, s_4, s_5, s_{11}, s_{12}\}$ and some rule f_{xn} from one of the sixteen infinite families resolve in a number of spurious rule steps and one family rule step; in several but not all cases, these resolutions can alternatively be used as derivations of that family rule. The head of an arrow point at the family of which a member might be derived, whereas the family at the tail and the arrow label correspond to the rules that would be involved in the critical pair.

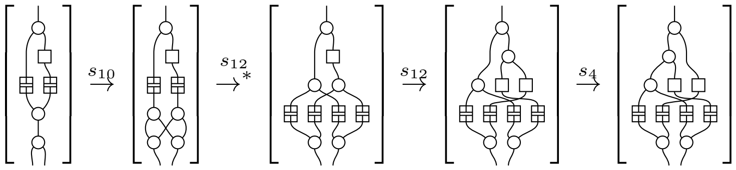
dance of transforming rules of one family into rules of another family (or sometimes the same family); Figure 2 gives an overview of how the families connect. If not for the fact that all rules in a family form the same kind of critical pair with a spurious rule, and also that the resolutions are (within each family) all trivial variations on each other, it would be very hard to verify that the resolutions all succeed. The sheer volume of calculations that are needed is such that one appreciates also having obtained a computer verification[2] of them (for the first couple of rules from each family), even though it remains within the realm of what can be carried out manually.

The final spurious rule s_{10} is not only the one most prolific in forming critical pairs with family rules (once for the coproduct at the top, once for the multiplication at the bottom), but also the one which creates the most complicated intermediate steps in their resolutions.

---

[2] Using the utility mentioned in a previous footnote.

A typical sequence is



where now a rule from family $f_{\mathrm{j}n}$ applies on the subnetwork containing the right multiplication, and then a rule from family $f_{\mathrm{a}n}$ resolves the rest. In the middle step it looks unlikely that any family rule can apply, because the crossing introduced by rule s_{10} means that two paths of antipodes that are adjacent on the coproduct side are not adjacent on the multiplication side, but what makes everything fit together is that one of the original paths between coproduct and multiplication has an even number of antipodes whereas the other has an odd number, and thus there is an extra twist at the end which makes one pair of paths adjacent on both sides, after which the resolution becomes straightforward. So even though rule s_{10} creates a lot of noise as far as critical pairs are concerned, it does not really contribute anything interesting here. (Note however that rule s_{10} plays a crucial role at one point in the derivation (1) of rule s_{11}.)

The final case of critical pair would be one formed by two family rules, and although that happens (for example between $f_{\mathrm{a}n}$ and $f_{\mathrm{m}n}$), it does not happen very often. The main reason is that the overlap has to take the form of a path starting in a coproduct, passing some number of antipodes, and ending in a multiplication; this places a strong restriction on the n values that might be involved, as both rules must have such paths with the same number of antipode vertices. Considering in addition that twisted families (c, d, g, h, k, l, o, and p) cannot form overlaps with the straight families (a, b, e, f, i, j, m, and n), one ends up with the conclusion that the n values of both rules involved must in fact be equal, and then it follows that all these critical pairs have trivial resolutions. Thus we have, again in anticipation of the technical details that will be dealt with in the next section, the main claim that:

▶ **Theorem 3.** *The full Hopf rewriting system F is terminating and confluent.*

As before, networks on normal form with respect to F can be written as $A \circ B \circ C$ where all the μ and u are in A, all the antipodes S are in B, and all Δ and ε are in C. What is new in the full system is that those arrangements of coproduct, antipodes, and multiplication upon which one of the family rules would act are forbidden, but which arrangements are those? Define a *mid-section path* of a network to be one that begins in a coproduct, have antipodes as inner vertices, and ends in a multiplication. Two mid-section paths are said to be *adjacent* on the coproduct or multiplication side if their outermost vertices on that side are adjacent or coincide. Clearly, the family rules may only apply to pairs of paths that are adjacent on both sides. Moreover, the number of antipodes on the paths in the pair must differ by 1, and the paths must cross (or not cross) depending on whether it is the path with the even number of antipodes that is the longer (or shorter, respectively).

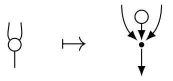
6 Compatible ordering of networks

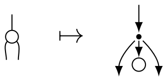
The diamond lemma in [3] is a descendant of Bergman's diamond lemma for associative algebras [1], so it requires a well-founded quasi-order P on the set of networks, that on one hand is compatible with the rules of the rewriting system, and on the other is strictly preserved under composition of networks. This turns out to not be entirely trivial to construct in this setting.

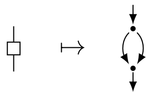
The main complication is the coproduct–multiplication rule s_{10}, since this has the unhelpful property of *increasing* the number of vertices in a network; were it not for this (and to a lesser extent rules s_{11} and s_{12}), one could have ensured well-foundedness simply by first ordering networks by the number of vertices in them. A generalisation to weighted vertex counts achieves nothing, since one would need $w_\mu + w_u \geqslant 0$ for rules s_2 and s_3, $w_\Delta + w_\varepsilon \geqslant 0$ for rules s_5 and s_6, $w_\mu \geqslant w_\varepsilon$ for rule s_9, $w_\Delta \geqslant w_u$ for rule s_8, and $0 \geqslant w_\mu + w_\Delta$ for rule s_{10}, all of which taken together merely imply that we have equality in all those inequalities. Beyond weighted vertex counts, it is not easy to come up with an ordering principle that is preserved under composition; most elementary suggestions of orders one can make up that do take the structure of an expression into account tend to fail at being preserved under composition.

Might it be better to orient some rules the other way? But no, this is the natural orientation; rules s_8, s_9, s_{10}, s_{11} and s_{12} expand things, whereas most of the others remove superfluous operations, and only the orientations of s_1 and s_4 are really arbitrary. It is natural that s_{11} changes expressions so that antipodes are applied before the multiplication rather than after, so how would one formalise this intuition? Obviously the order in which operations are performed matters, but how does one express that when comparing networks, as the structure of one network can be quite different from the structure of another? One possibility is to compare the sequence in which different vertex types occur along paths from input to output, because in the left hand side of s_{10} each path passes first a μ vertex and then a Δ vertex, whereas in the right hand side it is Δ first and μ second; the same kind of condition works for s_{11} and s_{12}. The only catch is that in order to be preserved under composition of networks, these comparisons must be performed separately for each pair of input and output, and also separately for paths that begin or end within a network.

In the end, it turns out to be sufficient to compare the *number* of paths through a network, provided that vertices are replaced by suitable gadgets, as follows:







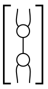
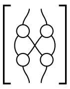
Unfilled circles (like for the unit u and counit ε) may serve as start or end point of a path, but the filled dots may only occur as inner vertices on a path; hence a multiplication μ counts as having three types of paths: paths from the left input passing through, paths from the right input passing through, and paths beginning here and continuing through the output. With these substitutions, it turns out that both the left and right hand sides of s_{10} contribute the same number of paths reaching the boundary of the network (despite the right hand side having 3 edges more), but the left hand side in addition has a path that both starts and ends within the network (starting at the μ, ending at the Δ), which the right hand side does not (since every μ there, where such a path might start, comes after the Δs where it would have to end), and *therefore* the left hand side achieves a greater number of paths than the right hand side. Similarly in rule s_{11}, the antipode doubles the number of paths passing through it, but it is only in the left hand side that the antipode also doubles the number of paths beginning at the multiplication. This is how these rules can be oriented from greater to smaller, even though there are more vertices in the right hand side than in the left hand side.

That is however the intuitive explanation. The formal nonsense is rather that a suitable quasi-order is constructed by pulling back the standard [3, Constr. 6.1] PROP quasi-order [3, Def. 3.1] on the biaffine PROP Baff($\mathbb{N}$) [3, Ex. 2.15] along a cleverly chosen PROP homomorphism, namely that g which satisfies

$$g(\mu) = \begin{pmatrix} 1 & \mathit{0} & \mathit{0} & \mathit{0} \\ 0 & 1 & 0 & 0 \\ 0 & \mathit{1} & \mathit{1} & \mathit{1} \end{pmatrix} \quad g(u) = \begin{pmatrix} 1 & \mathit{0} \\ 0 & 1 \\ 0 & \mathit{1} \end{pmatrix} \quad g(\Delta) = \begin{pmatrix} 1 & \mathit{0} & \mathit{1} \\ 0 & 1 & 0 \\ 0 & \mathit{0} & \mathit{1} \\ 0 & \mathit{0} & \mathit{1} \end{pmatrix} \quad g(S) = \begin{pmatrix} 1 & \mathit{0} & \mathit{0} \\ 0 & 1 & 0 \\ 0 & \mathit{0} & \mathit{2} \end{pmatrix}$$

$$g(\varepsilon) = \begin{pmatrix} 1 & \mathit{0} & \mathit{1} \\ 0 & 1 & 0 \end{pmatrix};$$

these conditions uniquely determine a homomorphism, because the set of all networks is the free PROP [3, Th. 8.4]. The italic matrix entries are those that are not fixed for elements of the biaffine PROP and thus may be chosen, although for the resulting PROP order to be strict it is necessary that there is at least one positive element in each row and at least one positive element in each column [3, Cor. 6.5]. The relevant interpretation of an element of $\mathrm{Baff}(\mathbb{N})$ is that the $(i+2, j+2)$ entry keeps track of the number of paths going from input j to output i of a network, whereas entry $(i+2, 2)$ keeps track of the number of paths which begin inside the network and reach output i, entry $(1, j+2)$ keeps track of the number of paths which come from input j but end inside the network, and entry $(1, 2)$ keeps track of the number of paths which both begin and end inside the network. Thus the above argument about counting paths in a gadgetified s_{10} corresponds to the observation that

$g\Big(\Big[$  $\Big]\Big) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} > \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} = g\Big(\Big[$  $\Big]\Big).$

What is not distinguished by the order pulled back over that g are the left and right hand sides of rules s_1 and s_4; since these impose a left–right asymmetry, they would interact poorly with the left–right swaps introduced by rules s_{11} and s_{12}. To orient also these, one introduces a secondary comparison criterion (technically makes a lexicographic composition [3, Constr. 3.7] of quasi-orders) by pulling back along a second cleverly chosen homomorphism g_2, for example that which has

$$g_2(\mu) = \begin{pmatrix} 1 & \mathit{0} & \mathit{0} & \mathit{1} \\ 0 & 1 & 0 & 0 \\ 0 & \mathit{1} & \mathit{1} & \mathit{1} \end{pmatrix} \qquad g_2(\Delta) = \begin{pmatrix} 1 & \mathit{0} & \mathit{1} \\ 0 & 1 & 0 \\ 0 & \mathit{0} & \mathit{1} \\ 0 & \mathit{1} & \mathit{1} \end{pmatrix}$$

In the path-counting interpretation, this amounts to adding the opportunity to end a path entering through the right input of a μ and begin a path leaving through the right output of a Δ. This causes the third input of a right-leaning $\mu \circ (\mathrm{id} \otimes \mu)$ to offer two chances for a path to end, whereas the third input of a left-leaning $\mu \circ (\mu \otimes \mathrm{id})$ only sees one; this suffices for making the left hand side of s_1 strictly larger than the right hand side.

Two additional results of [3] that are important in verifying that the quasi-order constructed as described above meet the conditions for the diamond lemma (Th. 10.24) are Corollary 9.16 and Lemma 9.18. Arguably also Lemma 3.5 on well-foundedness, but that result is on the other hand quite standard.

7 Final remarks

An anonymous reviewer has asked about the difficulty of finding redexes in network rewriting. Since network rewriting is nondeterministic, this ends up being a search problem, but the search is in practice fairly constrained: most of the time, a redex is uniquely determined by the correspondence of one vertex in the rule left hand side to one vertex in the network being reduced, because each edge incident with a vertex occupies a distinct "port" on that vertex (e.g. multiplication μ has a left incoming factor, a right incoming factor, and an outgoing result, no two of which are interchangeable). Therefore it is feasible to pick one vertex in the left hand side and try to match it against each vertex of the network to reduce; when things don't match, one tends to discover that early, and the only case in which the search might need to backtrack would be for a left hand side that is not connected. There are some additional complications related to keeping track of the *transference types* of rules [3, Defs. 6.14, 10.3], which can prevent something from being a redex even though the networks match, but that boils down to doing some extra bookkeeping.

A somewhat tougher problem is how to check which rules in a rewrite system might apply to a given network, especially when the rewrite system is large and in flux due to a (Knuth–Bendix style) completion being in progress. The author has implemented a system where each network is assigned a "signature" that counts occurrences of various small subgraphs therein, to the end of only considering rules whose signature is dominated by that of the target network. The performance has been good enough that reduction is not perceived as a problem when running large completions.

Acknowledgements. This work was begun in the spring of 2004, during my postdoc stay at the Mittag-Leffler institute as part of the NOG (Noncommutative Geometry) programme, funded by the European Science Foundation and the Royal Swedish Academy of Sciences. Much research has also been done while the author was associated with the Department of Mathematics and Mathematical Statistics at Umeå University. Final write-up was supported by The School of Education, Culture and Communication at Mälardalen University.

References

1 George M. Bergman. The diamond lemma for ring theory. *Adv. in Math.*, 29(2):178–218, 1978.

2 Masahito Hasegawa. *Models of sharing graphs.* CPHC/BCS Distinguished Dissertations. Springer-Verlag London, Ltd., London, 1999. A categorical semantics of let and letrec, Dissertation, University of Edinburgh, Edinburgh.

3 Lars Hellström. Network Rewriting I: The Foundation. *ArXiv e-prints*, 2012. arXiv:1204.2421 [math.RA].

4 Lars Hellström. Critical pairs in network rewriting. In Takahito Aoto and Delia Kesner, editors, *IWC 2014, 3rd International Workshop on Confluence*, pages 9–13, 2014. `http://www.nue.riec.tohoku.ac.jp/iwc2014/iwc2014.pdf`.

5 S. A. Joni and G.-C. Rota. Coalgebras and bialgebras in combinatorics. *Stud. Appl. Math.*, 61(2):93–139, 1979.

6 Yves Lafont. Towards an algebraic theory of Boolean circuits. *J. Pure Appl. Algebra*, 184(2-3):257–310, 2003.

7 Shahn Majid. Cross products by braided groups and bosonization. *J. Algebra*, 163(1):165–190, 1994.

8 Samuel Mimram. Computing critical pairs in 2-dimensional rewriting systems. In *RTA 2010: Proceedings of the 21st International Conference on Rewriting Techniques and Applications*, volume 6 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 227–241. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2010.

9 Roger Penrose. Applications of negative dimensional tensors. In *Combinatorial Mathematics and its Applications (Proc. Conf., Oxford, 1969)*, pages 221–244. Academic Press, London, 1971.

10 Moss E. Sweedler. *Hopf algebras.* Mathematics Lecture Note Series. W. A. Benjamin, Inc., New York, 1969.