DEPARTMENT OF MATHEMATICS
TECHNICAL REPORT

CLIFFORD AND GRASSMANN HOPF ALGEBRAS VIA THE BIGEBRA PACKAGE FOR MAPLE(R)

RAFAL ABLAMOWICZ

AND

BERTFRIED FAUSER

DECEMBER 2002

No. 2002-5

TENNESSEE TECHNOLOGICAL UNIVERSITY
Cookeville, TN 38505

Article Submitted to Journal of Symbolic Computation

Clifford and Graßmann Hopf algebras via the BIGEBRA package for Maple(R)*

Rafał Abłamowicz[1] and Bertfried Fauser[2]

[1] *Department of Mathematics, Box 5054, Tennessee Technological University, Cookeville, TN 38505, USA, E-mail:* rablamowicz@tntech.edu

[2] *Universität Konstanz, Fachbereich Physik, Fach M678, 78457 Konstanz, Germany, E-mail:* Bertfried.Fauser@uni-konstanz.de

Abstract

Hopf algebraic structures will replace groups and group representations as the leading paradigm in forthcoming times. K-theory, co-homology, entanglement, statistics, representation categories, quantized or twisted structures as well as more geometric topics of invariant theory, e.g., the Graßmann-Cayley bracket algebra are all covered by the Hopf algebraic framework. The new branch of *experimental mathematics* allows one to easily enter these fields through direct calculations using symbolic manipulation and computer algebra system (CAS). We discuss problems which were solved when building the BIGEBRA package for Maple and CLIFFORD† to handle tensor products, Graßmann and Clifford algebras, coalgebras and Hopf algebras. Recent results showing the usefulness of CAS for investigating new and involved mathematics provide us with examples. An outlook on further developments is given.

1. Aim of the paper

The first aim of this paper is to present the features of BIGEBRA, a Maple V rel. 5 package. BIGEBRA was built to deal with tensored Clifford and Graßmann algebras, and it relies on CLIFFORD (Abłamowicz and Fauser, 1999-2002, 1996-2002). While the emphasis here is more on the package, we nevertheless provide novel results in the last section on quantum Yang-Baxter equations derived from a Clifford bi-convolution. CLIFFORD and BIGEBRA are meanwhile available for Maple 6, 7 and Maple 8.

*Package homepage is located at http://math.tntech.edu/rafal/ while CLIFFORD and BIGEBRA are available for Maple V, Maple 6, 7, and 8. Maple(R) is available from http://www.maplesoft.com.

†The CLIFFORD package is described in a companion paper by Abłamowicz and Fauser (2003).

Secondly, we will not pass an opportunity to raise the flag for the subject of *experimental mathematics*. We strongly believe that experimental mathematics based on algorithmic approach has already been been changing research and teaching of mathematics and theoretical physics (Fauser, 2002). The main aspects of experimental mathematics, in our view, are as follows:

Computations

Various results have been achieved by brute force calculations. CAS simplifies difficult computations with non-commutative algebras, and it allows one to tackle problems impossible for hand calculations. It makes fewer errors and its results can be validated. A CAS *cannot* and *does not* replace knowledge of the mathematics behind the problem and it requires that all concepts be well defined and sound.

Checking Assertions

Having a CAS at hand, one can easily check one's own assertions. A single counter-example uncovers one's misconceptions and leads to a more sound understanding of the subject. Indeed, when checking a *theorem* on examples, one can occasionally find that it is not valid or that it has not been correctly formulated. This in turn helps with finding errors in the computer code (*Never trust any result generated by a computer!*) and/or with coming up with a proper formulation of such ill-stated theorem.

Developing New Mathematics

It has proved to be necessary to develop new mathematics for solving problems in physics. For example, we have introduced the *Hopf Gebra* as a weakened form of a *Hopf Algebra*. Using a CAS this was done by first exploring the set of principles which yielded correct physical results and then by fixing the mathematics of the new structure.

Teaching

Having a well developed CAS can be useful in teaching! Students can check their own prejudices by exemplifying them directly on a computer. This requires a stable code which does not allow for illegal input, etc. `CLIFFORD`'s code is stable while `BIGEBRA` still needs knowledge of the user because to gain speed, type and input checking is hardly done since it is time consuming. Hence, `CLIFFORD` (and with restrictions `BIGEBRA`) can be effectively used for a fast access to the mathematical field of Graßmann and Clifford algebras and cogebras in teaching. Students can make their own experiments and develop –given a CAS which is stable under silly input– a sound understanding of the mathematical structures in question. Teaching mathematics or physics using a CAS allows one to abstract from technical details on the *first* approach. However, if the goal is that students develop an understanding of the topic, they should be encouraged to recalculate by hand and to recode themselves. Otherwise no deep understanding will ensue. A good example how this can be achieved is Wright (2002).

Experimental Mathematics

Having the opportunity to deal with a CAS opens the field of *experimental mathematics*. This includes partly the other topics given above but it should not be underestimated due to its own dynamics. Exploring mathematics *by doing particular experiments* and by strengthening or weakenning once's *own* assumptions is of extreme value as it allows one to enter a new mathematical field quickly and in a secure way.

Algorithmic Understanding

We share a growing belief (Gruel and Pfister, 2002) that computer usage not only allows one to successfully complete non-trivial computations but also, through the developments of algorithms, contributes to a better understanding of the problem. The so called *algorithmic approach* leads to achieving a computational proficiency that in turn leads to a deeper mastery of the subject on the theoretical level.

2. The BIGEBRA package

We concentrate in this article on BIGEBRA and its features which generally go beyond those of CLIFFORD and Maple. We assume that the reader is somewhat familiar with the syntax and features of Maple, see, e.g., Wright (2002), while CLIFFORD is described in (Abłamowicz and Fauser, 2003).

The package loads in a non verbose (silent) mode with the following commands while the linear algebra package linalg is loaded for convenience only. BIGEBRA automatically loads CLIFFORD that is needed for its functionality:

```
1 >restart:_CLIENV[_SILENT]:=true:with(linalg):with(Bigebra);
```

[&*cco*, &*gco*, &*gco_d*, &*gpl_co*, &*map*, &*v*, *EV*, *VERSION*, *bracket*, *cco_monom*, *contract*, *do_compose*, *drop_t*, *eps*, *gantipode*, *gco_d_monom*, *gco_monom*, *gco_unit*, *gpl_co_monom*, *gpl_co_monom2*, *gswitch*, *init*, *linop*, *linop2*, *lists2mat*, *lists2mat2*, *make_BI_Id*, *mapop*, *mapop2*, *meet*, *op2mat*, *op2mat2*, *pairing*, *peek*, *poke*, *remove_eq*, *switch*, *tcollect*, *tsolve1*, *type/tensorbasmonom*, *type/tensormonom*, *type/tensorpolynom*]

The output is a list with available functions; some of them are for internal use only. CLIFFORD and BIGEBRA come with an extensive online help page system which is included in the Maple online help and can be searched. It contains not only the syntax of the procedures but even a good deal of unpublished mathematics. It is meant to present to the user the mathematical concepts at hand. Help topics may be reached in Maple fashion with the command ?Bigebra followed by the enter key.

2.1. Tensor product

Given the functionality of CLIFFORD to compute with Clifford algebras, we need two more key features to enter the realm of bi- and Hopf algebras. The first

one is the tensor product and some functions to manipulate it with while the second is the coproduct. The latter, however, follows naturally from the algebra structure on the linear dual space (Milnor and Moore, 1965).

Maple comes with a `define` facility which allows to introduce ampersand operators `&<name>` that are associative (flat), commutative (orderless), linear or multilinear. This facility unfortunately has two drawbacks:

- It cannot deal with user-defined scalars (ring elements). This is mathematically insensitive since any definition of a linear or multilinear function *needs* linearity with respect to "scalars" that one wants to compute with. We want to consider, e.g., tensor products (multilinear associative) over arbitrary rings, e.g., the integers, polynomial rings etc.
- It produces code which gives wrong output.

Remembering that Graßmann basis multivectors are denoted as `Id`, `e1`, `e2`, `ei`, `e1we2`, `e1we3`, `eiwej,...`, let us explore properties of the associative (flat) multilinear operator `&r` (comments in the code are separated by `#`):

```
> restart:                              # unload BIGEBRA !
> define(`&r`,flat,multilinear):        # produce the operator `&r`
> out[1]:=&r(2*e1,2*e2+4*a*e3):         # a is not known to be a ring element
> out[2]:=e1+e2 &r e3, &r(e1+e2,e3):# infix form is dangerous
> out[1];out[2];                        # we delete output for typographical reasons
```

$$4\,(e1\ \&\mathrm{r}\ e2) + 8\,(e1\ \&\mathrm{r}\ a\ e3)$$

$$e1 + (e2\ \&\mathrm{r}\ e3),\ (e1\ \&\mathrm{r}\ e3) + (e2\ \&\mathrm{r}\ e3)$$

Notice that Maple does not know how to deal with a so, effectively, the multilinearity is established over the integers only. If one wants to compute over a polynomial ring or, more generally, with K-modules, this function is useless.

```
> constants:=constants,a:
> out[1]:=&r(2*e2,2*e1+4*a*e1): # a is known to be a constant now
> out[2]:=&r(true*e1,false*e2): # true and false are 'constants'
> constants;out[1];out[2];
```

$$false,\ \gamma,\ \infty,\ true,\ Catalan,\ FAIL,\ \pi,\ a$$

$$4\,(e2\ \&\mathrm{r}\ e1) + 8\,a\,(e2\ \&\mathrm{r}\ e1)$$

$$true\ false\ (e1\ \&\mathrm{r}\ e2)$$

The last line does not make sense at all. Furthermore we find these peculiarities:

```
> out[1]:=&r(-e1,-e2):          # expected output -> &r(e1,e2)
> out[2]:=&r(-e1):              # expected output -> -1*&r(e1)
> out[3]:=&r(2.5*e1):           # expected output -> 2.5*&r(e1)
> out[4]:=&r(&r(-e1)):          # expected output -> &r(e1)
> out[1],out[2],out[3],out[4];
```

$$(-e1)\,\&\mathrm{r}\,(-e2),\ -(1\,\&\mathrm{r}\,e1),\ 2.5\,(1\,\&\mathrm{r}\,e1),\ -\&\mathrm{r}(1,\ 1,\ e1)$$

While the first expression is not even simplified, the last three results are just plain wrong. This error was detected in Maple 5 and is partially still present up to Maple 7. Hence, BIGEBRA includes a patched define facility.

In CLIFFORD and BIGEBRA almost everything can be treated as scalars, hence one can compute over function spaces etc. The generic tensor product of BIGEBRA is given by &t.

```
16 >#reload BIGEBRA, defined will be patched
17 >restart:_CLIENV[_SILENT]:=true:with(linalg):with(Bigebra):
18 >out[1]:=e1+e2 &t e3, &t(e1+e2,e3):   # infix form is still dangerous!
19 >out[2]:=&t(&t(-e1)):                 # fixed
20 >out[3]:=&t(2.5*e1):
21 >out[4]:=&t(a*e1+2*e2,sin(phi)*e3):
22 >out[1];out[2],out[3];out[4];
```

$$e1 + (e2\,\&\mathrm{t}\,e3),\ (e1\,\&\mathrm{t}\,e3) + (e2\,\&\mathrm{t}\,e3)$$

$$-\&\mathrm{t}(e1),\ 2.5\,\&\mathrm{t}(e1)$$

$$a\sin(\phi)\,(e1\,\&\mathrm{t}\,e3) + 2\sin(\phi)\,(e2\,\&\mathrm{t}\,e3)$$

The patched define that ships with BIGEBRA can handle tensor products over quite general rings. We give a final example where we define a tensor product &r over the polynomial ring $\mathbb{Z}[[x]]$ with integer coefficients.

```
23 >define(`&r`,flat,multilinear):
24 >unprotect(`type/cliscalar`):             # we want to change this type
25 >`type/cliscalar`:=proc(expr) type(expr,polynom(integer,x)) end:
26 >out[1]:=type( 2*x+3*x^5+5,cliscalar):    # true, since in Z[[x]]
27 >out[2]:=type( x/2        ,cliscalar):    # false, fraction
28 >out[3]:=type( 2*y^2+3*y-1,cliscalar):    # false, wrong indeterminant
29 >out[1],out[2],out[3];
```

$$true,\ false,\ false$$

Now we are ready to use the new tensor product &r over $\mathbb{Z}[[x]]$:

```
30 >out[1]:=&r(2.5*x,4*y):
31 >out[2]:=&r(3*x^2-x+5,x-x^4):
32 >out[3]:=&r(3*y*x^2-x-5,x*y-x^4):
33 >out[1];out[2];out[3];
```

$$4\,x\,(2.5\,\&\mathrm{r}\,y)$$

$$3\,x^3\,(1\,\&\mathrm{r}\,1) - 3\,x^6\,(1\,\&\mathrm{r}\,1) - x^2\,(1\,\&\mathrm{r}\,1) + x^5\,(1\,\&\mathrm{r}\,1) + 5\,x\,(1\,\&\mathrm{r}\,1) - 5\,x^4\,(1\,\&\mathrm{r}\,1)$$

$$3\,x^3\,(y\,\&\mathrm{r}\,y) - 3\,x^6\,(y\,\&\mathrm{r}\,1) - x^2\,(1\,\&\mathrm{r}\,y) + x^5\,(1\,\&\mathrm{r}\,1) - 5\,x\,(1\,\&\mathrm{r}\,y) + 5\,x^4\,(1\,\&\mathrm{r}\,1)$$

Observe, that real numbers like 2.5 or the variable y are still not treated as scalars but integers and the variable x are.

2.2. Basic tools

Having defined a tensor product, we need to have tools for operating on such structures. Let us fix notation as follows: A single term having no prefactors `&t(b1,...,bn)`, where $b_1, \ldots, b_n$ are Graßmann basis multivectors, will be called a *tensor basis monom* or a *word* that is composed from letters of the Graßmann multivector alphabet. *A tensor monom* is a tensor basis monom with a "scalar" prefactor where "scalar" means a ring element, a function, a polynomial, or just a number. *A tensor polynom*, or just a tensor, is a sum of tensor monoms. It is the multilinearity that guarantees that every tensor can be written as a linear combination of some tensor basis monoms (Hilbert basis theorem). The i-th place in a list of arguments of a tensor will be called the i-th *slot*.

Acting on a tensor means:

- *Removing or grafting of terms from or into a tensor*
- *Rearranging the tensor*
- *Acting with operators on n-slots producing m-slots*

The first set of operations is given by `peek` and `poke`. They work as expected: `peek` takes as an argument a number of the slot whose contents it intends to remove and returns a sequence of lists of the removed terms and the remaining tensor. Procedure `poke` needs three arguments: a tensor, a Graßmann multivector which will be put into the i-th place, and the slot number i. For example,

```
34 >f:=x-> 'peek'(x,2):x:=&t(e1,a*e2+b*e3,e3):f(x)=eval(f(x));
```

$$\mathrm{peek}(a\,\&\mathrm{t}(e1,\ e2,\ e3)+b\,\&\mathrm{t}(e1,\ e3,\ e3),\ 2)=([a\,e2,\ e1\ \&\mathrm{t}\ e3],\ [b\,e3,\ e1\ \&\mathrm{t}\ e3])$$

```
35 >g:=x-> 'poke'(x,e2,3):x:=&t(e1,a*e2+b*e3):g(x)=eval(g(x));
```

$$\mathrm{poke}(a\,(e1\ \&\mathrm{t}\ e2)+b\,(e1\ \&\mathrm{t}\ e3),\ e2,\ 3)=a\,\&\mathrm{t}(e1,\ e2,\ e2)+b\,\&\mathrm{t}(e1,\ e3,\ e2)$$

The second group of operations consists of switches –also called *crossings* or *braids*– which allow to reorder the tensors. From a CAS point, it would be easy, say, to multiply the i-th and the j-th slots of a tensor and place the output into the k-th slot. However, mathematical reasons disallow such a brute method. If one demands that any reordering process is to be generated from the reordering rules of the generators, one has to assume that a crossing is a *natural transformation* in a functorial sense and obeys coherence, see Kelly and Laplaza (1980); Lyubashenko (1995b,a). In fact these two conditions imply the quantum Yang Baxter equation which is discussed below. The Graßmann Hopf algebraic graded crossing and the ordinary switch (swap) of two elements fulfill these properties.

The `switch` procedure swaps two *adjacent* tensor slots in a general tensor polynomial, while the graded switch `gswitch` respects the grading of the factors. Both functions take as a second argument i, the number of the tensor slot to act on, that is, to switch the i-th and the $(i+1)$-st tensor entries.

```
36 >f1:=x->'switch'(x,2):x:=&t(e1,e2,e3,e4):f1(x)=eval(f1(x));
```

$$\mathrm{switch}(\&t(e1,\ e2,\ e3,\ e4),\ 2) = \&t(e1,\ e3,\ e2,\ e4)$$

```
37 >f2:=x->'gswitch'(x,2):x:=&t(e1,e2,e3,e4):f2(x)=eval(f2(x));
```

$$\mathrm{gswitch}(\&t(e1,\ e2,\ e3,\ e4),\ 2) = -\&t(e1,\ e3,\ e2,\ e4)$$

```
38 >x:=&t(e1,e2,e3we4,e5):f2(x)=eval(f2(x));
```

$$\mathrm{gswitch}(\&t(e1,\ e2,\ e3we4,\ e5),\ 2) = \&t(e1,\ e3we4,\ e2,\ e5)$$

Notice the different signs for `switch` and `gswitch` depending in the latter morphism on the grading of the Graßmann multivector elements.

Furthermore, `BIGEBRA` allows to act with any operator on tensors. We distinguish such operations by the number of input and output slots. An ordinary endomorphism is a $1 \to 1$ map while, e.g., a product is a $2 \to 1$ map. Functions which are currently available are `mapop` and `mapop2` for $1 \to 1$ and $2 \to 2$ operators, `&map` for $2 \to 1$ operators and `contract` for $2 \to 0$ operators. Let us first show a linear operator $proj1$ acting on a certain tensor slot:

```
39 >proj1:=proc(x) vectorpart(x,1) end:
40 >L:=x->'mapop'(x,2,proj1):x:=&t(Id,e1+e1we2,e3):L(x)=eval(L(x));
```

$$\mathrm{mapop}(\&t(Id,\ e1,\ e3) + \&t(Id,\ e1we2,\ e3),\ 2,\ proj1) = \&t(Id,\ e1,\ e3)$$

where `vectorpart`(x, n) projects on the n-vector part of x in the Graßmann basis. Let us define a general element x in a Graßmann or Clifford algebra over a two dimensional vector space and a general operator `Rop` given by its matrix R in a co-contravariant canonical basis:

```
41 >restart:_CLIENV[_SILENT]:=true:with(linalg):with(Bigebra):
42 >dim_V:=2:                                  # set dimension to 2
43 >bas:=cbasis(dim_V):                        # generate a list of basis monoms
44 >X:=add(x[i]*bas[i],i=1..2^dim_V):          # general element
45 >R:=matrix(2^dim_V,2^dim_V,(i,j)->r[i,j]): # matrix with entries r[i,j]
46 >Rop:=proc(x) linop(x,R) end:               # the operator Rop
47 >X;evalm(R);
```

$$x_1\, Id + x_2\, e1 + x_3\, e2 + x_4\, e1we2$$

$$\begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & r_{1,4} \\ r_{2,1} & r_{2,2} & r_{2,3} & r_{2,4} \\ r_{3,1} & r_{3,2} & r_{3,3} & r_{3,4} \\ r_{4,1} & r_{4,2} & r_{4,3} & r_{4,4} \end{bmatrix}$$

The action of `Rop` on Graßmann multivectors and tensors is computed as

```
48 >out[1]:=Rop(Id):out[2]:=Rop(X):out[3]:=mapop(&t(e1,Id,e2),2,Rop):
49 >out[1];out[2];out[3];
```

$$r_{1,1}\,Id + r_{2,1}\,e1 + r_{3,1}\,e2 + r_{4,1}\,e1we2,$$

$$(r_{1,1}\,x_1 + r_{1,2}\,x_2 + r_{1,3}\,x_3 + r_{1,4}\,x_4)\,Id + (r_{3,4}\,x_4 + r_{3,1}\,x_1 + r_{3,2}\,x_2 + r_{3,3}\,x_3)\,e2 + (r_{2,2}\,x_2 + r_{2,3}\,x_3 + r_{2,4}\,x_4 + r_{2,1}\,x_1)\,e1 + (r_{4,1}\,x_1 + r_{4,2}\,x_2 + r_{4,3}\,x_3 + r_{4,4}\,x_4)\,e1we2,$$

$$r_{1,1}\,\&\mathrm{t}(e1,\,Id,\,e2) + r_{2,1}\,\&\mathrm{t}(e1,\,e1,\,e2) + r_{3,1}\,\&\mathrm{t}(e1,\,e2,\,e2) + r_{4,1}\,\&\mathrm{t}(e1,\,e1we2,\,e2)$$

Operators may be defined as any Maple procedures also.

The next class of operations is the application of product maps, i.e., $2 \to 1$, that have two *adjacent* entry slots and one output slot. Examples of such operations which are predefined in `BIGEBRA` are the Graßmann and the Clifford products, and the contractions. User defined functions may be applied also. They are mapped onto tensors via the `&map` function which takes a tensor, the slot number and the (product) function as input parameters:

```
> trm:=&t(Id,e1,e2,Id):
> out[1]:=`&map(&t(Id,e1,e2,Id),2,wedge)`,` --> `,&map(trm,2,wedge):
> out[2]:=`&map(&t(Id,e1,e2,Id),2,cmul[K])`,` --> `,&map(trm,2,cmul[K]):
> out[3]:=`&map(&t(Id,e1,e2,Id),2,LC,K)`,` --> `, &map(trm,2,LC,K):
> out[1];out[2];out[3];
```

$$\&map(\&t(Id,\,e1,\,e2,\,Id),\,2,\,wedge),\,-->,\,\&\mathrm{t}(Id,\,e1we2,\,Id)$$

$$\&map(\&t(Id,\,e1,\,e2,\,Id),\,2,\,cmul[K]),\,-->,\,\&\mathrm{t}(Id,\,e1we2,\,Id) + K_{1,2}\,\&\mathrm{t}(Id,\,Id,\,Id)$$

$$\&map(\&t(Id,\,e1,\,e2,\,Id),\,2,\,LC,\,K),\,-->,\,K_{1,2}\,\&\mathrm{t}(Id,\,Id,\,Id)$$

Note that the Clifford product `cmul[K]` and the left contraction $\mathtt{LC}(...,K)$ depend on an arbitrary bilinear form K which is then passed on as an additional parameter to the procedure `&map`. This mechanism can then be used to deal with different Clifford algebras in different tensor slots.

As a last example of actions on tensors we examine the evaluation and other $2 \to 0$ mappings with values in the base ring. A very important case is given by the *evaluation* which employs the action of dual elements w.r.t. the canonical basis on multivectors. In fact one would need a new kind of basis vectors. However, for technical reasons, in the current version of `BIGEBRA` it is the *user* who is responsible for keeping track which tensor slots are to contain co(multi)vectors.

```
> out[1]:=[EV(Id,Id),EV(e1,e2),EV(e1,e1)]: # eval = Kronecker delta
> out[2]:=`contract(&t(Id,e1,e1+e2+e3,Id),2,EV)`,
>         `  -->  `,contract(&t(Id,e1,e1+e2+e3,Id),2,EV):
> out[1];out[2];
```

$$[1,\,0,\,1]$$

$$contract(\&t(Id,\,e1,\,e1 + e2 + e3,\,Id),\,2,\,EV),\quad -->\ ,\ Id\,\&\mathrm{t}\,Id$$

2.3. Graßmann Hopf algebra

A Graßmann Hopf algebra is a vector space that is a Graßmann algebra and a Graßmann coalgebra fulfilling certain compatibility laws (Sweedler, 1969). Since the algebra side is well known, we give some explanation on the coalgebra side. A coproduct on an algebra can be defined as the categorial dual of a product via a linear form defined on the vector space. This law is called *product coproduct duality* (Milnor and Moore, 1965). Now, it is easy to check that covectors may form a covector Graßmann algebra $V^\vee$, where $\vee$ is the Graßmann exterior product on co-multivectors. Using duality via the evaluation map $\langle . \mid . \rangle$ we find

$$\begin{aligned} \langle w \mid x \wedge y \rangle &= \langle w_{(1)} \mid y \rangle \langle w_{(2)} \mid x \rangle, \\ \langle w \vee w' \mid x \rangle &= \langle w \mid x_{(2)} \rangle \langle w' \mid x_{(1)} \rangle \end{aligned} \tag{1}$$

where we have used Sweedler's notation for the coproduct $\Delta(x) = \sum_{(x)} x_{(1)} \otimes x_{(2)}$ (the summation symbol is usually dropped). The evaluation map is given by the function `EV` from the package. The coproduct is seen to be a split of the homogeneous multivectors into pairs where the sign of permutation is taken into account. We give one example of the Graßmann coproduct acting on a Graßmann basis monomial $\mathbf{e}_a \wedge \mathbf{e}_b$ with symbolic indices a and b :

```
59 >&gco(eaweb);
```

$$(Id \,\&\mathrm{t}\, eaweb) + (ea \,\&\mathrm{t}\, eb) - (eb \,\&\mathrm{t}\, ea) + (eaweb \,\&\mathrm{t}\, Id)$$

The first compatibility law valid in a Hopf algebra is that the product is a coalgebra homomorphism and the coproduct is an algebra homomorphism, see Fauser (2002). Furthermore, an antipode exists, which in the Graßmann case turns out to be isomorphic to the grade involution. We check this in dimension 2:

```
60 >dim_V:=2:
61 >`S^`=matrix(2^dim_V,2^dim_V,(i,j)->EV(bas[i],gantipode(bas[j])));
62 >`Grade_Inv`=matrix(2^dim_V,2^dim_V,(i,j)->EV(bas[i],gradeinv(bas[j])));
```

$$S\hat{} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad Grade\_Inv = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The above described tools allow us to perform all algebraic manipulations in a Graßmann Hopf algebra, also when using symbolic indices.

2.4. Clifford bi-convolution

A *Clifford bi-convolution* is a space endowed with a Clifford algebra structure and a Clifford coalgebra structure. Both products are unital and associative but not all such convolutions posses an antipode. However, if an antipode exists then one can prove that one deals then with a Clifford Hopf algebra (Oziewicz,

2001a; Fauser, 2002). We are interested in Clifford algebras over a general bilinear form, hence we can introduce the Clifford product via the Chevalley deformation (Chevalley, 1997). Let $\mathbf{x}$ be an element in V, the space of generators, and let u be a general element in $V^{\wedge}(=\bigwedge V)$. Then, one defines the action of $\mathbf{x}$ on U as

$$\gamma_{\mathbf{x}} \circ u = i_{\mathbf{x}}(u) + \mathbf{x} \wedge u$$

where $i_{\mathbf{x}}$ is the *inner product* or *contraction*. The action is then extended to a map $\circ : V \otimes V \mapsto V$ by demanding that

$$i_{\mathbf{x}}(\mathbf{y}) = B(\mathbf{x}, \mathbf{y}), \qquad i_{u\wedge v}(w) = i_u(i_v(w))$$

$$i_{\mathbf{x}}(u \wedge v) = i_{\mathbf{x}}(u) \wedge v + \hat{u} \wedge i_{\mathbf{x}}(v)$$

with $\hat{u} = (-1)^{\partial u} u$ being the grade involution, e.g., Fauser (2002). It is a remarkable fact that this product can be directly defined via a deformation called *cliffordization* by Rota and Stein (Rota and Stein, 1994). It may also be called a Drinfeld twist, see below. Using the undeformed Graßmann product $\wedge$ and coproduct Δ one can write

$$u \circ v = B^{\wedge}(u_{(2)}, v_{(1)})\, u_{(1)} \wedge v_{(2)}$$

Here $B^{\wedge}$ is the scalar valued bilinear form tied to the inner product action via the counit ϵ as $B^{\wedge}(a, b) = \epsilon(i_a(b))$. We exemplify this in the BIGEBRA package as

```
> restart:_CLIENV[_SILENT]:=true:with(Bigebra):
> unprotect(gamma):
> Gamma:=proc(U) local x; x:=op(procname); LC(x,U)+wedge(x,U) end:
> out[1]:=gamma[e1](Id)=Gamma[e1](Id):
> out[2]:=gamma[e1](e2)=Gamma[e1](e2):
> out[3]:=gamma[e1](e2we3)=Gamma[e1](e2we3):
> out[1],out[2],out[3];
```

$$\gamma_{e1}(Id) = e1,\ \gamma_{e1}(e2) = B_{1,2}\, Id + e1we2,\ \gamma_{e1}(e2we3) = B_{1,2}\, e3 - B_{1,3}\, e2 + e1we2we3$$

If we do the same using the cliffordization process, which we will make explicit by defining a function `cliff` as to avoid using the internal function `cmul`, we get

```
> cliff:=proc(x,y) clicollect(simplify(drop_t(&map(contract(
>                           &t(&gco(x),&gco(y)),2,scalarpart@LC),1,wedge)))) end:
> out[1]:=e1    &C e2we3 = cliff(e1   ,e2we3):
> out[2]:=e1we2 &C e2we3 = cliff(e1we2,e2we3):
> out[1];out[2];
```

$$e1\ \&\mathrm{C}\ e2we3 = B_{1,2}\, e3 - B_{1,3}\, e2 + e1we2we3$$

$$e1we2\ \&\mathrm{C}\ e2we3 = B_{2,2}\, e1we3 - B_{1,2}\, e2we3 - B_{2,3}\, e1we2 - (-B_{2,2}\, B_{1,3} + B_{2,3}\, B_{1,2})\, Id$$

The most interesting structure which we can now access computationally with the BIGEBRA package is that of a Clifford bi-convolution. In any pair (U, V) of

structures having a product on $V \otimes V$ and a coproduct on U, which need not to be a bialgebra or Hopf algebra at all, one can define a convolution product between morphisms $f, g, ... : U \to V$, see Fauser (2002). We consider here the Graßmann Hopf algebra with endomorphisms, and define a *star* or *convolution product* of endomorphisms according to Sweedler, (Sweedler, 1969) as

$$(f \star g)(x) = m \circ (f \otimes g) \circ \Delta(x)$$

where $\circ$ is the composition of maps. The element x can be dropped safely. Re-examining the definition of the antipode shows that the map is the convolutive inverse of the identity morphism id_V on V. In the following steps we will compute the antipodes of the Graßmann Hopf algebra and of a twisted Graßmann Hopf algebra where only the product is deformed, and of the Clifford bi-convolution, where both structure maps, product and coproduct, have been deformed. The last possibility is beyond current deformation theory where only one structure map is deformed. We choose dimension 2 for the generating space which gives a 4 dimensional Graßmann resp. Clifford algebra. The output is suppressed in most steps since it is very tedious and long. The defining equations for the three antipodes are stored in `eq_gr`, `eq_cl` and `eq_bc`.

```
75 >dim_V:=2:bas:=cbasis(dim_V):
76 >B :=matrix(dim_V,dim_V,(i,j)->b[i,j]):        # scalar product
77 >BI:=matrix(dim_V,dim_V,(i,j)->p[i,j]):        # coscalar product
78 >make_BI_Id():                            ## <== initialize the Clifford coproduct
79 >S   := matrix(2^dim_V,2^dim_V,(i,j)->s[i,j]): # antipode template
80 >Sop := proc(x) linop(x,S) end:                # operator using S
81 >X   := add(x[i]*bas[i],i=1..2^dim_V):        # general element
82 >eq_gr :=drop_t(&map(mapop(&gco(X),1,Sop),1,wedge)-gco_unit(&t(X),1)):
83 >eq_cl :=drop_t(&map(mapop(&gco(X),1,Sop),1,cmul )-gco_unit(&t(X),1)):
84 >eq_bc :=drop_t(&map(mapop(&cco(X),1,Sop),1,cmul )-gco_unit(&t(X),1)):
```

In a second step we solve these equations using the BIGEBRA tangle solver `tsolve1`:

```
85 >sol_gr :=tsolve1(clicollect(eq_gr) ,[seq(seq(s[i,j],
86 >            i=1..2^dim_V),j=1..2^dim_V)],[seq(x[i],i=1..2^dim_V)]):
87 >sol_cl :=tsolve1(clicollect(eq_cl) ,[seq(seq(s[i,j],
88 >            i=1..2^dim_V),j=1..2^dim_V)],[seq(x[i],i=1..2^dim_V)]):
89 >sol_bc :=tsolve1(clicollect(eq_bc) ,[seq(seq(s[i,j],
90 >            i=1..2^dim_V),j=1..2^dim_V)],[seq(x[i],i=1..2^dim_V)]):
```

Before we display these morphisms in a matrix form, we compute a normalization factor for the antipode of the Clifford bi-convolution to simplify the output, which will be given as $S_{bc} = N * S'_{bc}$. Here the primed antipode is the actual one and the unprimed is the normalized one.

```
91 >N:=linalg[det](linalg[diag](1$dim_V)- B &* BI):
92 >S_GR := subs(sol_gr[1],evalm(S)):
```

```
93 >S_CL := subs(sol_cl[1],evalm(S)):
94 >S_BC := map(simplify@expand,subs(sol_bc[1],N*evalm(S))):
95 >S_GR=evalm(S_GR),S_CL=evalm(S_CL),S_bc=map(simplify,evalm(S_BC));
```

$$S\_GR = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad S\_CL = \begin{bmatrix} 1 & 0 & 0 & b_{1,2} - b_{2,1} \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$S\_bc = \begin{bmatrix} 1 - p_{1,2}\, b_{2,1} + p_{2,1}\, b_{2,1} + p_{1,2}\, b_{1,2} - p_{2,1}\, b_{1,2} & 0 & 0 & b_{1,2} - b_{2,1} \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ p_{1,2} - p_{2,1} & 0 & 0 & 1 \end{bmatrix}$$

PROPOSITION 2.1: *(*dim_V=2*) The antipode S_{bc} of a Clifford bi-convolution is up to a factor identical with the antipode $S^{\wedge}$ of the Graßmann Hopf algebra if and only if the cliffordization is performed with a symmetric scalar product and a symmetric coscalar product (both derivable from a quadratic and a coquadratic form via polarization in* char $\neq 2$).

COROLLARY 2.1: *The recursive formula for computing the antipode given by Milnor and Moore (Milnor and Moore, 1965) and currently used in the Connes-Kreimer renormalization procedure cannot be applied in general for Clifford bi-convolutions with antisymmetric part in the scalar and coscalar product*

$$S(x) = \epsilon(x) - x - S(x'_{(1)})x'_{(2)}, \quad S(Id) = Id$$

where the primed coproduct is over proper cuts only, i.e. $x'_{(i)} \neq Id$.

Note however that bilinear forms having antisymmetric parts are involved in the process of Wick normal ordering in QFT (Fauser, 2001). A CAS is a valuable help here to find explicit examples of such structures.

3. Deformation and quantum Yang-Baxter equation

In this section we want to use BIGEBRA to make the relation between cliffordized products, coproducts and standard deformation theory explicit. We keep the definitions from the previous section, i.e. dim_V=2, the settings for the scalar product B, the coscalar product BI, and the general element X. For further reference we need an explicit form of the scalar product extended to V and recall the matrix form of the Graßmann antipode, this time from the BIGEBRA built-in function gantipode. We compute the convolutive inverse of the bilinear form needed below:

```
96 >BW  :=matrix(2^dim_V,2^dim_V,(i,j)->EV(cmul(bas[i],bas[j]),Id)):
97 >S_Gr:=matrix(2^dim_V,2^dim_V,(i,j)->EV(bas[i],gantipode(bas[j]))):
98 >BS  :=evalm(BW &* S_Gr):
99 >BW=evalm(BW),S_gr=evalm(S_Gr),BS=evalm(BS);
```

$$BW = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & b_{1,1} & b_{1,2} & 0 \\ 0 & b_{2,1} & b_{2,2} & 0 \\ 0 & 0 & 0 & b_{2,1}\,b_{1,2} - b_{2,2}\,b_{1,1} \end{bmatrix}, \quad S\_gr = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$BS = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -b_{1,1} & -b_{1,2} & 0 \\ 0 & -b_{2,1} & -b_{2,2} & 0 \\ 0 & 0 & 0 & b_{2,1}\,b_{1,2} - b_{2,2}\,b_{1,1} \end{bmatrix}$$

Note that the convolutive inverse is just the scalar product w.r.t the negative bilinear form.

PROPOSITION 3.1: *Not every endomorphism has as the convolutive inverse* $B^S = B \circ S$. *However, exponentially generated morphisms and, more generally, graded morphisms, do.*

A quasi triangular structure is an element $R \in V \otimes V$ which satisfies, among others, the following condition:

$$\hat{\mathrm{sw}} \circ \Delta_{Cl}(x) = (R_{(1)} \otimes R_{(2)}) \circ \Delta_{Gr}(x) \qquad (*)$$

see, e.g., Majid (1995). Note that our definition is somewhat different due to our reversed ordering of tensor products of dual elements. The task is now to define a general element R and try to compute all possible solutions of equation $(*)$. Therefore we start to define R as

```
100 >Y:=add(y[i]*bas[i],i=1..2^dim_V):
101 >R:=proc(x,y) local bas,tr_tbl; option remember;
102 >   bas:=cbasis(dim_V):
103 >   tr_tbl:=table([seq(op(Cliff5[extract](bas[i]))=i,i=1..2^dim_V)]);
104 >   R[tr_tbl[op(Cliff5[extract](x))],tr_tbl[op(Cliff5[extract](y))]]
105 >end:
106 >RR:=add(add(contract(&t(bas[i],bas[j]),1,R)*&t(bas[i],bas[j]),
107 >   i=1..2^dim_V),j=1..2^dim_V):
```

Now the actual computation starts by evaluating the l.h.s. and r.h.s. of $(*)$ as

```
108 >Leq:=gswitch(&cco(X),1):
109 >Req:=&map(&map(switch(switch(&t(RR,&gco(X)),2),1),3,wedge),1,wedge):
```

Note that in the right hand side of the equation `Req`, two switches were used to put R in-between the $x_{(i)}$ of the coproduct and are not part of the formula $(*)$ while the graded switch in the l.h.s. equation `Leq` is generic. This time we show directly how to solve tangle equations:

```
110 >eq:=tcollect(Leq-Req):
111 >vars:=seq(seq(R[i,j],i=1..2^dim_V),j=1..2^dim_V):
112 >T:=seq(seq(&t(bas[i],bas[j]),i=1..2^dim_V),j=1..2^dim_V):
113 >CO:=seq(x[i],i=1..2^dim_V):
114 >sys:=:
```

```
> for t in T do
> for co in CO do
>   sys:=op(sys),coeff(coeff(eq,t),co);
> od:od:
> sol:=solve(sys,vars):
> matR:=matrix(2^dim_V,2^dim_V,(i,j)->R[i,j]):
> `R`=subs(sol,evalm(matR));
```

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -p_{1,1} & -p_{2,1} & 0 \\ 0 & -p_{1,2} & -p_{2,2} & 0 \\ 0 & 0 & 0 & p_{2,1}\,p_{1,2} - p_{2,2}\,p_{1,1} \end{bmatrix}$$

We have put the solution once more in a matrix form. Inspection of this result shows that our R is the convolutive inverse of the coscalar product, and hence it is exponentially generated. In fact it is an axiom of a quasi triangular structure that a convolutive inverse exists. We can check our assertion explicitly via

```
> eq1:=gco_unit(gco_unit(&t(X,Y),2),1):
> eq2:=simplify(drop_t(contract(contract(&t(&gco(X),&gco(Y))
>                               ,2,scalarpart@cmul[-K]),1,scalarpart@cmul[K])))*Id:
> is(eq1=eq2);
```

true

COROLLARY 3.1: *The (co-)cliffordizations with respect to (co-)bilinear forms $-K$ and K are convolutive inverse w.r.t. the Graßmann Hopf convolution.*

We are now ready to check that our quasi triangular structure fulfills the *quantum Yang Baxter equation*. We do this first for the graded switch of the Graßmann Hopf convolution, which is trivially a braid. We compute

```
> Z:=add(z[i]*bas[i],i=1..2^dim_V): eq0:=&t(X,Y,Z):
> eq1:=gswitch(gswitch(gswitch(eq0,1),2),1):
> eq2:=gswitch(gswitch(gswitch(eq0,2),1),2):
> is(eq1=eq2);
```

true

where we have not shown the cumbersome terms but simply checked that both sides evaluate to the same result, which establishes the assertion.

Since a quasi triangular structure is an exponentially generated endomorphism, we have to check that it fulfills

$$R_{12}R_{23}R_{12} = R_{23}R_{12}R_{23}.$$

To check this assertion we calculate with BIGEBRA :

```
> bw:=proc(x,y) local i,j;
>  add(add(BW[i,j]*EV(bas[i],y)*EV(bas[j],x),i=1..2^dim_V),j=1..2^dim_V)
> end:
```

```
> Bsw:=proc(x,i)
>   tcollect(gswitch(contract(&gco(&gco(x,i+1),i),i+1,bw),i))
> end:
> eq3:=Bsw(Bsw(Bsw(eq0,1),2),1):
> eq4:=Bsw(Bsw(Bsw(eq0,2),1),2):
> is(eq3=eq4);
```

$$true$$

which proves our claim in `dim_V=2`. As our last demonstration, we check the remaining properties of a quasi triangular structure to be valid for an exponentially generated endomap.

$$R(S(a),b) = R^S(a,b), \quad R^S(a,S(b)) = R(a,b), \quad R(S(a),S(b)) = R(a,b).$$

Hence we generate the endomap 'R' and apply to its arguments the Graßmann antipode:

```
> R:='R':
> out[1]:='R'=matrix(2^dim_V,2^dim_V,(i,j)->scalarpart(cmul[R](bas[i],bas[j]))):
> out[2]:='RS'=matrix(2^dim_V,2^dim_V,(i,j)->scalarpart(cmul[R]
>                                          (gantipode(bas[i]),bas[j]))):
> out[1];out[2];
```

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & R_{1,1} & R_{1,2} & 0 \\ 0 & R_{2,1} & R_{2,2} & 0 \\ 0 & 0 & 0 & R_{2,1}R_{1,2} - R_{2,2}R_{1,1} \end{bmatrix}$$

$$RS = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -R_{1,1} & -R_{1,2} & 0 \\ 0 & -R_{2,1} & -R_{2,2} & 0 \\ 0 & 0 & 0 & R_{2,1}R_{1,2} - R_{2,2}R_{1,1} \end{bmatrix}$$

```
> out[3]:='RSS'=matrix(2^dim_V,2^dim_V,(i,j)->scalarpart(cmul[R]
>                                    (gantipode(bas[i]),gantipode(bas[j])))):
> out[4]:='SR'=matrix(2^dim_V,2^dim_V,(i,j)->scalarpart(cmul[R]
>                                          (bas[i],gantipode(bas[j])))):
> out[3];out[4];
```

$$RSS = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & R_{1,1} & R_{1,2} & 0 \\ 0 & R_{2,1} & R_{2,2} & 0 \\ 0 & 0 & 0 & R_{2,1}R_{1,2} - R_{2,2}R_{1,1} \end{bmatrix}$$

$$SR = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -R_{1,1} & -R_{1,2} & 0 \\ 0 & -R_{2,1} & -R_{2,2} & 0 \\ 0 & 0 & 0 & R_{2,1}R_{1,2} - R_{2,2}R_{1,1} \end{bmatrix}$$

Comparing the matrix representations gives us the desired results when we recall that the convolutive inverse is given by the 'scalar product' `-R` on the generating space.

We can compute along the same lines as exemplified above the Yang Baxter matrix, a 16 by 16 matrix, which represents the action of R on $V \otimes V$. Since the output of these computations is quite lengthy we will give only some further results and not the explicit matrix. The actual worksheet is available from the home page of the second author.

Remark: The Yang Baxter matrix of our 2 dimensional example has determinant 1 and is a function of all four parameters of the quasi triangular structure R.

A further question is whether this structure is really quasi triangular or only triangular. Due to our reversed indexing in duals, we have to check for triangularity if the Yang Baxter matrix squares to unity.

Remark: The Yang Baxter matrix of our 2 dimensional example is triangular if and only if the exponentially generated endomap is derived from a symmetric bilinear form on the generating space. This has deep implications for ordering process in quantum field theory, see Fauser (2001).

We expect such assertions to be true in higher dimensions, though that needs an algebraic proof. However, having solutions in low dimensions allows in many cases to prove by induction a general hypothesis. This is the step from experimental mathematics to 'pure' mathematics.

4. Conclusions

The present paper was intended to show how a CAS allows to ask questions of research interest and to come up with low dimensional solutions. During this process we have worked out examples showing how to use the `BIGEBRA` package and how to attack more advanced problems.

We hope that it became obvious from our presentation of the `CLIFFORD` (Abłamowicz and Fauser, 2003) and `BIGEBRA` packages that the ability to compute, via a CAS, allows to build a sound knowledge about mathematical problems and that there is a need for *experimental mathematics* in education and research.

This article is, due to restrictions in space, obviously not able to give a complete review of all abilities of the `CLIFFORD` and `BIGEBRA` packages. Therefore the interested reader is invited to check the package home page[‡] for the documentation. Both packages come with a built-in online help where, for every function, syntax, synopsis and examples, and sometimes a more advanced mathematical background, are provided. A printed .ps and .pdf versions of approx. 500 pages are also available there. For those who wish to have a look at and a feel for the packages, there is a possibility to access them online over the web[§].

[‡]url: `http://math.tntech.edu/rafal/`

[§]url: `http://kaluza.physik.uni-konstanz.de/~ fauser`

There are tremendously many open problems coming with a bi-convolution of mathematical and physical type. It is not yet clear under what conditions antipodes exist in higher dimensions. In dimension 3 a Clifford bi-convolution has in general no antipode, hence further relations between scalar and coscalar product have to be found. Only very little is known about the nature of the crossings derived from antipodal bi-convolution, but see Fauser and Oziewicz (2001). The axiomatics, in terms of morphims and category theory of Clifford bi-convolutions, is under consideration but not yet finished (Oziewicz, 1998, 2001b). Clifford bi-convolutions are deeply connected to the renormalization process of the quantum field theory. However, there one has to go beyond the scheme presented in this paper, see Brouder (2002); Brouder et al. (2002). A CAS and especially CLIFFORD and BIGEBRA are ideally suited to explore this exciting field.

Finally we want to emphasize that the present article contains Maple output (with only a minor TEX cosmetics) which was generated directly by processing a Maple worksheet.

Acknowledgment: The second author, BF, acknowledges gratefully financial support from University of Konstanz, LS Prof. Heinz Dehnen.

References

Rafał Abłamowicz and Bertfried Fauser. CLIFFORD. Tennessee Technological University, http://math.tntech.edu/rafal/, 1996-2002. Maple V program package.

Rafał Abłamowicz and Bertfried Fauser. BIGEBRA. Tennessee Technological University and University of Konstanz, http://math.tntech.edu/rafal/, 1999-2002. Maple V program package.

Rafał Abłamowicz and Bertfried Fauser. Mathematics of CLIFFORD - A Maple Package for Clifford and Graßmann Algebras. *Journal of Symbolic Computation*, 2003. Submitted.

Christian Brouder. A quantum field algebra. *J. Phys. A: Math. Gen. submitted*, 2002. math-ph/0201033.

Christian Brouder, Bertfried Fauser, Alessandra Frabetti, and Robert Oeckl. Quantum groups and quantum field theory II. 2002. Preprint.

Claude Chevalley. *The Algebraic Theory of Spinors and Clifford Algebras*. Springer-Verlag, Berlin, 1997. Collected Works Vol. 2, Pierre Cartier, Catherine Chevalley Eds.

Bertfried Fauser. On the Hopf-algebraic origin of Wick normal-ordering. *Journal of Physics A: Mathematical and General*, 34:105–115, 2001. hep-th/0007032.

Bertfried Fauser. A Treatise on Quantum Clifford Algebras. Konstanz, 2002. Habilitationsschrift, arXiv:math.QA/0202059.

Bertfried Fauser and Zbigniew Oziewicz. Clifford Hopf gebra for two dimensional space. *Miscellanea Algebraicae*, 2(1):31–42, 2001. math.QA/0011263.

Gert-Martin Gruel and Gerhard Pfister. *A* Singular *Introduction to Commutative Algebra*. Springer-Verlag, New York, 2002.

Gregory Maxwell Kelly and Miguel L. Laplaza. Coherence for compact closed categories. *Journal of Pure and Applied Algebra*, 19:193–213, 1980.

Volodimir Lyubashenko. Modular transformations for tensor categories. *Journal of Pure and Applied Algebra*, 98:279–327, 1995a.

Volodimir Lyubashenko. Tangles and Hopf algebras in braided categories. *Journal of Pure and Applied Algebra*, 98:245–278, 1995b.

Shahn Majid. *Foundations of Quantum Group Theory*. Cambridge University Press, Cambridge, 1995.

John W. Milnor and John C. Moore. On the structure of Hopf algebras. *Annals of Mathematics*, 81:211–264, 1965.

Zbigniew Oziewicz. The Dirac operator as graph and the Clifford Hopf-gebra. In John Ryan and Daniele Struppa, Eds., *Analysis of Dirac Operators*, volume Pitmann Re, Harlow, Essex, 1998. Addison Wesley Longman Limited.

Zbigniew Oziewicz. Guest editor's note: Clifford algebras and their applications. *International Journal of Theoretical Physics*, 40(1):1–13, 2001a.

Zbigniew Oziewicz. Operad of graphs, convolution and Hopf gebra. *Contemporary Mathematics*, pages 1–17, 2001b. Submitted.

Gian-Carlo Rota and Joel A. Stein. Plethystic Hopf algebras. *Proc. Natl. Acad. Sci. USA*, 91:13057–13061, December 1994.

Moss E. Sweedler. *Hopf Algebras*. W. A. Benjamin, INC., New York, 1969.

Francis Wright. *Computing with Maple*. Chapman & Hall/CRC, Boca Raton, 2002.

Submitted: December 10, 2002; Revised: TBA.