Mathematics of **CLIFFORD** - A Maple Package for
Clifford and Graßmann Algebras

Rafał Ablamowicz\(^1\)     Bertfried Fauser\(^2\)

February 17, 2005

**Abstract**

CLIFFORD performs various computations in Graßmann and Clifford algebras. It can compute with quaternions, octonions, and matrices with entries in $\mathbb{C}\ell(B)$ - the Clifford algebra of a vector space $V$ endowed with an arbitrary bilinear form $B$. Two user-selectable algorithms for the Clifford product are implemented: cmulNUM - based on Chevalley’s recursive formula, and cmulRS - based on a non-recursive Rota-Stein sausage. Graßmann and Clifford bases can be used. Properties of reversion in undotted and dotted wedge bases are discussed.

**Keywords:** Quantum Clifford algebra, contraction, dotted wedge product, grade involution, Graßmann algebra, Hopf algebra, multivector, octonions, quaternions, reversion, wedge product

1 Introduction

As many programs CLIFFORD emerged from a practical problem. Relatively complicated algebraic manipulations with octonions, which can be performed in $\text{spin}(7)$, started a thread which has now developed into a multi purpose algebra tool. It is the basic structure of a vector space $V$ endowed with a quadratic form $Q$ which is common to a vast host of mathematical, physical and engineering problems, and which allows one to build naturally –i.e., in a categorial sense ‘for free’– an algebra structure, the Clifford algebra $\mathbb{C}\ell(V,Q)$.

AMS Subject Classification: 15A66, 68W30

1) DEPARTMENT OF MATHEMATICS, BOX 5054, TENNESSEE TECHNOLOGICAL UNIVERSITY, COOKEVILLE, TN 38505, USA
rablamowicz@tntech.edu,
http://math.tntech.edu/rafal/

2) MAX PLANCK INSTITUT FÜR MATHEMATIK, INSELSTRASSE 22-26, D-04103 LEIPZIG, GERMANY,
Bertfried.Fauser@uni-konstanz.de,
http://clifford.physik.uni-konstanz.de/~fauser/
While in a conventional vector calculus one makes a good use of the vector space structure, one does not have yet a vector algebra since vector multiplication is missing. Having established a Clifford algebra structure provides one with an entirely new formalism that now can be applied to solving completely different problems.

In this sense, CLIFFORD is a basic tool for all such investigations and applications which can be carried in finite dimensional vector spaces equipped with a quadratic form or, equivalently, with a symmetric bilinear form commonly referred to as an inner or a scalar product. The intrinsic abilities of Maple even allow one to use CLIFFORD in projective and affine geometries while visualizing complicated incidence relations that is helpful, e.g., for image processing, visual perception and robotics.

The authors of CLIFFORD have been interested in fundamental questions about q-deformed symmetries and quantum field theory. A reasonable number of new results has been derived by using systematically the ability to compute with a Computer Algebra System (CAS) at hand. Moreover, just asking questions such as “What is the most general element fulfilling . . .?” has led to unexpected results and new insights. Testing of theorems, usually to check the consistency of the software, has led from time to time to counter examples that have made a rethinking and a more careful restatement of those theorems necessary. However, the most striking ability of CLIFFORD is that it is unique in being able to handle Clifford algebras of an arbitrary bilinear form not restricted by symmetry and not directly related to any quadratic form. Since it is now well known that such structures are related to Hopf algebraic twists, later versions of CLIFFORD make an extensive use of a process called Rota-Stein cliffordization, which turns out to be a Drinfeld twist of a Graßmann Hopf algebra [9].

The present paper introduces the reader to the package. It is assumed that she is already familiar with Maple [22], a general purpose CAS; if not please consult e.g., [23]. Of course, such a paper cannot be a user guide but may only be a demonstration of the usability and strength of the package. The interested reader is invited to download the package and take a closer look at the online documentation in the Maple help browser or to download a pdf file with over 500 help pages. Therein we provide also further mathematical background and references and a detailed description for every function. However, the present article provides also a quick –and dirty– introduction which is sufficient to get started.

A list of goals behind an experimental mathematics, which are the guiding beacons for this work, is given in [5] that describes the supplementary package BIGEBA for computations with tensor and Hopf algebras.

2 Notations and basic computations

CLIFFORD uses as default a standard Graßmann basis (Graßmann multivectors) in $\Lambda V$ where $V = \text{span} \{ e_i \mid 1 \leq i \leq n \}$ for $1 \leq n \leq 9$. Then $\Lambda V = \text{span} \{ e_i \wedge e_j \wedge \ldots \wedge e_k \mid 0 \leq i < j < \ldots < k \leq n \}$. In CLIFFORD these basis monomials
are written as strings \( \{ Id, e_1, \ldots, e_9, e_1we2, e_1we3, \ldots, e_1we2we3, \ldots \} \) although they can be aliased to shorten input. Here \( e_1we2 \) is a string that denotes \( e_1 \wedge e_2 \) and \( Id \) denotes the identity \( 1 \) in \( \wedge V \). However, CLIFFORD can also use one-character long symbolic indices as in \( e_iwe \). Thus, in principle, it can compute with Clifford algebras in dimensions higher than 9. For example, when \( n = 3 \), Grassmann basis monomials are:

\[
W = [Id, e_1, e_2, e_3, e_1we2, e_1we3, e_2we3, e_1we2we3]
\]

but aliases can also be used to shorten input/output:

\[
\text{eval(makealiases(3));}
\]

\[
I, e12, e21, e31, e23, e32, e123, e121, e231, e312, e321
\]

In the above, \( eijk = e_iwejwek \) is the wedge product of three 1-vectors: \( e_i, e_j, e_k \). Thus, the most general element in the Grassmann algebra \( \wedge V \) is a Grassmann polynomial which is just a linear combination of Grassmann basis monomials with real coefficients. Notice that symbolic indices are allowed:

\[
p1 := Id + 4.5 \cdot e_i - \alpha e_123
\]

Reordering of Grassmann monomials can be explicitly accomplished with a procedure \text{reorder}. CLIFFORD procedures ordinarily return their results in the standard (reordered) basis.

\[
p2 := -e_3we2we1-x0*Id+x12*e_2we1+a*ejwei; \text{reorder(p2)};
\]

\[
p2 := -x321 - x0 Id + x12 e21 + a ejwei
\]

\[
e123 - x0 Id = x12 e12 - a eiwej
\]

The wedge product \( \wedge \) is computed with a procedure \text{wedge} or its ampersand counterpart \&w:

\[
\text{wedge(e1,e2),e1 &w e2; wedge(ea,eb,ec),ea &w eb &w ec; p1 &w p2;}
\]

Following Chevalley’s recursive definition, a Clifford product can be introduced in \( \wedge V \) by means of a left \( J_B \) (or right \( L_B \)) contraction dependent on an arbitrary bilinear form \( B : V \times V \rightarrow \mathbb{R} \) (see Sect. 3). This leads to elements of the Clifford algebra \( C(B) \) expanded into multivectors and makes the Clifford multiplication implicitly dependent on \( B \). The associative Clifford product is given by a procedure \text{cmul} or its infix form \&c.

\[
cmul(e1,e2),&c(e1,e2); \text{cmul(ea,eb,ec)};
\]

\[
e12 + B_{1,2} Id, e12 + B_{1,2} Id
\]

\[
\text{cawebwec, cawebwec}
\]

\[
e123 - x0 Id = -4.500000000 x0 e1 + \alpha x0 e123 - x12 e12
\]

\[
e12 + B_{1,2} Id, e12 + B_{1,2} Id
\]

\[
\text{cawebwec + B_{b,c} ea - B_{a,c} eb + B_{a,b} ec}
\]

\[
e12 + B_{1,2} Id, e12 + B_{1,2} Id
\]

\[
\text{cawebwec + B_{b,c} ea - B_{a,c} eb + B_{a,b} ec}
\]
Computations in $\mathbb{C}\ell(K)$ and $\mathbb{C}\ell(B)$ can be performed in the same worksheet since the name of a bilinear form can be passed to `cmul` as a parameter. For example,

> `cmul[K](e1,e2),&c[K](e1,e2); cmul[K](e1,ej,ek);`

\[
e_{12} + K_{1,2} I d, e_{12} + K_{1,2} I d \\
e_{iwjke} + K_{j,k} e_i - K_{i,k} e_j + K_{i,j} e_k
\]

Of course, the forms $B$ and $K$ can be numeric or symbolic. For example, when

> `B:=matrix(2,2,[1,a,a,1]);`

\[
B := \begin{bmatrix}
1 & a \\
0 & a
\end{bmatrix}
\]

then the Grassmann basis for $\mathbb{C}\ell(B)$ or $\wedge V$ will be:

> `cbas:=cbasis(2);`

\[
\text{cbas} := [I d, e_1, e_2, e_{12}]
\]

while the Clifford multiplication table of the basis Grassmann monomials will look as follows:

> `MultTable:=matrix(4,4,(i,j)->cmul(cbas[i],cbas[j]));`

\[
\text{MultTable} := 
\begin{bmatrix}
I d & e_1 & e_2 & e_{12} \\
e_1 & I d & e_{12} + a I d & e_2 - a e_1 \\
e_2 & -e_{12} + a I d & I d & a e_2 - e_1 \\
e_{12} & a e_1 - e_2 & e_1 - a e_2 & (-1 + a^2) I d
\end{bmatrix}
\]

Of course, irrespective of the bilinear form chosen, the Grassmann multiplication table will always remain as:

> `wedgetable:=matrix(4,4,(i,j)->wedge(cbas[i],cbas[j]));`

\[
\text{wedgetable} := 
\begin{bmatrix}
I d & e_1 & e_2 & e_{12} \\
e_1 & 0 & e_{12} & 0 \\
e_2 & -e_{12} & 0 & 0 \\
e_{12} & 0 & 0 & 0
\end{bmatrix}
\]

Let $B = g + F$ where $g$ and $F$ are the symmetric and the antisymmetric parts of $B$:

> `g,F:=matrix(2,2,[g11,g12,g12,g22]),matrix(2,2,[0,F12,-F12,0]);`

> `B:=evalm(g+F);`

\[
g, F := 
\begin{bmatrix}
g_{11} & g_{12} \\
g_{12} & g_{22}
\end{bmatrix}, 
\begin{bmatrix}
0 & F_{12} \\
-F_{12} & 0
\end{bmatrix}
\]
\[
B := \begin{bmatrix}
g_{11} & g_{12} + F_{12} \\
g_{12} - F_{12} & g_{22}
\end{bmatrix}
\]

Then, the Clifford multiplication table of the basis monomials in \(\mathbb{C}L(B)\) will be as follows:

\[
\text{MultTable} := \text{matrix}(4,4,(i,j)\rightarrow\text{cmul(cbas}[i],\text{cbas}[j]));
\]

\[
\text{MultTable} := \\
\begin{bmatrix}
\text{Id} \\
e_1 \text{Id} \\
e_2 \\
e_{12}
\end{bmatrix}
\]

\[
\begin{bmatrix}
e_{12} \text{Id} \\
(g_{12} - F_{12}) \text{Id} - e_{12} \\
(g_{12} - F_{12}) e_1 - g_{11} e_2, g_{22} e_1 - (g_{12} + F_{12}) e_2 \\
g_{12}^2 - F_{12}^2 - g_{22} g_{11} \text{Id} - 2 e_{12} F_{12}
\end{bmatrix}
\]

Observe, that the “standard” anticommutation relations

\[
e_i e_j + e_j e_i = (B_{i,j} + B_{j,i}) \text{Id} = 2g_{i,j} \text{Id}
\]

are satisfied by the generators \(e_i, i = 1,2,\ldots,n\), irrespective of the presence of the antisymmetric part \(F\) in \(B\). For example,

\[
\text{cmul}[g](e_1,e_2)+\text{cmul}[g](e_2,e_1);
\]

\[
2 \text{Id} g_{12}
\]

\[
\text{cmul}[B](e_1,e_2)+\text{cmul}[B](e_2,e_1);
\]

\[
(g_{12} + F_{12}) \text{Id} + (g_{12} - F_{12}) \text{Id}
\]

\[
\Rightarrow \text{clisort(simplify(%)};
\]

\[
2 g_{12} \text{Id}
\]

It is well known [16, 19] that real Clifford algebras \(\mathbb{C}L(V,Q) = \mathbb{C}L_{p,q}\) are classified in terms of the signature \((p,q)\) of \(Q\) and the dimension \(\text{dim} V = n = p + q\). Information about all Clifford algebras \(\mathbb{C}L_{p,q}\), \(1 \leq n \leq 9\), for any signature \((p,q)\) has been pre-computed and stored in \text{CLIFFORD}, and it can be retrieved with a procedure \text{clidata}. For example, for the Clifford algebra \(\mathbb{C}L_{2,0}\) (also denoted as \(\mathbb{C}L_2\)) of the Euclidean plane \(\mathbb{R}^2\) we find:

\[
\text{clidata}([2,0]); \#\text{Clifford algebra of the Euclidean plane}
\]

\[
[\text{real}, 2, \text{simple}, \frac{1}{2} \text{Id} + \frac{1}{2} e_1, [\text{Id}, e_2], [\text{Id}], [\text{Id}, e_2]]
\]

The meaning of the first three entries in the above output list is that \(\mathbb{C}L_2\) is a simple algebra isomorphic to \(\text{Mat}(2,\mathbb{R})\). The 4th entry in the list gives a primitive idempotent \(f\) that has been used to generate a minimal left spinor ideal \(S = \mathbb{C}L_2 f\) and, subsequently, the left spinor (lowest dimensional and faithful) representation of \(\mathbb{C}L_2\) in \(S\). In general it is known that, depending on \((p,q)\) and \(n = \text{dim} V\), the spinor ideal \(S = \mathbb{C}L_{p,q} f\) is a right \(K\)-module where \(K\) is either \(\mathbb{R}\), \(\mathbb{C}\), or \(\mathbb{H}\) for simple Clifford algebras when \((p-q) \neq 1 \mod 4\), or \(\mathbb{R} \oplus \mathbb{R}\) and \(\mathbb{H} \oplus \mathbb{H}\) for semisimple algebras when \((p-q) = 1 \mod 4\) [14, 17]. Elements in
the 5th entry (here \([\text{Id}, e^2]\)) generate a real basis in \(S\) with respect to \(f\), that is, \(S = \text{span}\{\text{Id} \& c f, e^2 \& c f\} = \text{span}\{f, e^2 \& c f\}\). Elements in the 6th entry span a subalgebra \(F\) of \(\mathbb{C}l(Q)\) that is isomorphic to \(K\). In the case of \(\mathbb{C}l_2\) we find that \(F = \text{span}\{\text{Id}\} \cong \mathbb{R}\). The last entry in the output gives \(2^k\) generators of \(S\) (with respect to \(f\)) viewed as a right module over \(K\) where \(k = q - r_{q-p}\) and \(r\) is the Radon-Hurwitz number.\(^1\)

Number \(k\) is the number of factors \(\frac{1}{2}(1 + T_i)\), where \(\{T_i\}, i = 1, \ldots, k\), is a set of commuting basis Grassmann monomials squaring in \(\mathbb{C}l(Q)\) to 1, whose product gives a primitive idempotent \(f\) in \(\mathbb{C}l(Q)\). Spinor representation for all Clifford algebras \(\mathbb{C}l_n(Q)\), \(1 \leq n = p + q \leq 9\), and for any signature \((p, q)\) has been pre-computed \([1]\) and can be retrieved from CLIFFORD with a procedure matKrepr. For example, 1-vectors \(e_1\) and \(e_2\) in \(\mathbb{C}l_2\) have the following spinor representation in the basis \(\{f, e^2 \& c f\}\) of \(S = \mathbb{C}l_2 f:\)

\[
\begin{bmatrix}
1 & 0 \\
0 & -1
\end{bmatrix},
\begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix}
\]

In another example, Clifford algebra \(\mathbb{C}l_3\) of \(\mathbb{R}^3\) is isomorphic with \(\text{Mat}(2, \mathbb{C})\):

\[
\begin{bmatrix}
\text{complex}, & 2, \text{simple}, & \frac{1}{2}\text{Id} + \frac{1}{2}e_1, & [\text{Id}, e_2, e_3, e_23], & [\text{Id}, e_23], & [\text{Id}, e_2]\n\end{bmatrix}
\]

and its spinor representation is given in terms of Pauli matrices:

\[
\begin{bmatrix}
1 & 0 \\
0 & -1
\end{bmatrix},
\begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix},
\begin{bmatrix}
0 & -e_23 \\
e_23 & 0
\end{bmatrix}
\]

Notice that \(F = \text{span}\{\text{Id}, e_23\}\) \((e_23 = e_2we_3)\) is a subalgebra of \(\mathbb{C}l_3\) isomorphic to \(\mathbb{C}\). Since Pauli matrices belong to \(\text{Mat}(2, \mathbb{F})\), it is necessary for CLIFFORD to compute with Clifford matrices, that is, matrices of a type climatrix with entries in a Clifford algebra.

Of course Pauli matrices satisfy the same defining relations as the basis vectors \(e_{1,2,3}\). For example:

\[
'M1 &cm M2 + M2 &cm M1' = evalm(M1 &cm M2 + M2 &cm M1);
'e1 & cm e2 + e2 & cm e1' = e1 & cm e2 + e2 & cm e1;
\]

\(^1\)Type ?Rnumber in a Maple session when CLIFFORD is installed for more help.

\(^2\)We use the sloppy notation \(1 \equiv 1\) in Clifford algebra valued matrices which produces a simpler display.

\(^3\)Here &cm is a matrix product where Clifford multiplication is applied to the matrix entries. See ?&cm for more information.
of a single matrix over a double field \( \mathbb{R} \).

In this latter case, it is customary to represent an element in \( \mathbb{R} \) matrices.

In general, \( \mathbb{C} \ell \) Clifford algebra \( Q \) matKrepr gives the linear isomorphism \( \mathbb{C} \ell(Q) \cong \text{Mat}(2^k, K) \), where \( K = \mathbb{R}, \mathbb{C}, \mathbb{H} \), for simple algebras and \( \mathbb{C} \ell(Q) \cong \text{Mat}(2^k, K) \oplus \text{Mat}(2^k, K) \), where \( K = \mathbb{R}, \mathbb{H} \), for semisimple algebras. In this latter case, it is customary to represent an element in \( \mathbb{C} \ell(Q) \) in terms of a single matrix over a double field \( \mathbb{R} \oplus \mathbb{R} \) or \( \mathbb{H} \oplus \mathbb{H} \) rather than as pair of matrices.

One can easily list signatures of the quadratic form \( Q \) for which \( \mathbb{C} \ell(Q) \) is simple or semisimple. For more information, type \texttt{?all_sigs}. For example, \( \mathbb{C} \ell_{1,3} \) has a spinor representation given in terms of 2 by 2 quaternionic matrices whose entries belong to a subalgebra \( F \) of \( \mathbb{C} \ell_{1,3} \) spanned by \( \{1d, e2, e3, e2we3\} \):

\[
\begin{array}{c}
B := \text{linalg[diag]}(1, -1, -1, -1): \text{clidata}([1, 3]);
\end{array}
\]

\[
\begin{array}{c}
\text{quat}\text{er}\text{nionic, 2, simple, } \frac{1}{2} \text{Id} + \frac{1}{2} e1we4, [\text{Id}, e1, e2, e3, e12, e13, e23, e123], [\text{Id}, e2, e3, e23], [\text{Id}, e1]
\end{array}
\]

\[
\text{matKrepr}([1, 3]); \#\text{quaternionic matrices}
\]

\[
\begin{array}{c}
e1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},
\end{array}
\begin{array}{c}
e2 = \begin{bmatrix} e2 & 0 \\ 0 & -e2 \end{bmatrix},
\end{array}
\begin{array}{c}
e3 = \begin{bmatrix} e3 & 0 \\ 0 & -e3 \end{bmatrix},
\end{array}
\begin{array}{c}
e4 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}
\end{array}
\]

\texttt{CLIFFORD} includes several special-purpose procedures to deal with quaternions and octonions (type \texttt{?quaternion} and \texttt{?octonion} for help). In particular, following [18], octonions are treated as para-vectors in \( \mathbb{C} \ell_{0,7} \) while their non-associative multiplication, defined via Fano triples, is related to the Fano projective plane \( \mathbb{P}_2 \) (see \texttt{?omultable}, or \texttt{?Fano_triples} for more information). Since the bilinear form \( B \) can be degenerate, one can use \texttt{CLIFFORD} to perform computations in Clifford algebras \( \mathbb{C} \ell_{p,q,d} \) of degenerate quadratic form \( Q \) of signature \( (p, q) \) and the dimension of the radical \( d \). For example, the Clifford algebra \( \mathbb{C} \ell_{0,3,1} \) of the quadratic form \( Q(x) = -x_1^2 - x_2^2 - x_3^2 \) where

---

4Procedures \texttt{addmatrix} and \texttt{mdfmatrix} add and multiply matrices of type \texttt{dfmatrix} over such double fields. For more information see \texttt{matKrepr}.

5When \( B \equiv 0 \) then \( \mathbb{C} \ell(V, B) = \mathbb{C} \ell_{0,0,n} = \bigwedge V \) and computations in the Grassmann algebra \( \bigwedge V \) can then be done with \texttt{CLIFFORD}.
\[ x = x_1 e_1 + x_2 e_2 + x_3 e_3 + x_4 e_4 \in \mathbb{R}^4 \] is used in robotics to represent rigid motions in \( \mathbb{R}^3 \) and screw motions in terms of dual quaternions [21].

Thus, CLIFFORD is a repository of mathematical knowledge about Clifford algebras of a quadratic form in dimensions 1 through 9. Together with a supplementary package BIGEBRA [3] it can be extended to graded tensor products of Clifford algebras in higher dimensions. The BIGEBRA package is described in [5]. For more information about any CLIFFORD or BIGEBRA procedure, type \?Clifford or \?Bigebra to see its help page through the Maple browser. For a computation of spinor representations with CLIFFORD we refer to [1].

3 Clifford product

Since the Clifford product provides the main functionality of the CLIFFORD package, the best available mathematics has been used to program it. Two internal user-selectable functions \texttt{cmulRS} and \texttt{cmulNUM} encode the Clifford product but the user normally does not use either one. Instead, she uses a wrapper function \&\texttt{K}[\texttt{arg1}, \texttt{arg2}, \ldots] or \texttt{cmul}[\texttt{K}][\texttt{arg1}, \texttt{arg2}, \ldots] that passes the name of a bilinear form \( K \) to either \texttt{cmulRS} or \texttt{cmulNUM}, whichever one has been selected. The wrapper function can also act on any number of arguments of type \texttt{clipolynom} (this makes sense since the Clifford product is associative) and on a much wider class of types including Clifford matrices of type \texttt{climatrix}. It can also accept Clifford polynomials in other bases such as the Clifford basis \{1, e_i, e_i \& C e_j, \& C(e_i, e_j, e_k), \ldots\} where \&\texttt{C} denotes the unevaluated Clifford product. Clifford basis differs from the Graßmann exterior basis when \( B \) is not a diagonal matrix.\(^6\)

\(^6\)Procedures converting between Graßmann and Clifford bases belong to a supplementary package \texttt{Cliplus} [3] while Clifford polynomials expressed in the Clifford basis are of type \texttt{cliprod}. Type \?cliprod for more information.

The two internal Clifford multiplication procedures are appropriate for different purposes. While \texttt{cmulNUM} is fast on sparse numeric matrices and on numeric matrices in general when \( \dim V \geq 5 \), procedure \texttt{cmulRS} was designed for symbolic calculations. Since \texttt{cmulRS} computes reasonably well in the numeric sparse case up to \( \dim V = 5 \), it was chosen as the default product of the package.\(^7\) Both procedures take as input two Clifford monomials of type \texttt{clibasmon} along with a third argument \( K \) of a type \texttt{name}, \texttt{symbol}, \texttt{matrix} or \texttt{array} which contains the chosen bilinear form for \( Cl(K) \).

Via a procedure \texttt{useproduct}, the user can select either \texttt{cmulRS} or \texttt{cmulNUM} depending which procedure is expected to give a better performance for the bilinear form in use. In addition, the user can also supply a new product function (not necessarily a Clifford product) acting on two basis monomials. The wrapper function uses then the selected algorithm to compute the Clifford product.

\(^7\)When CLIFFORD is loaded, it is setup to use \texttt{cmulRS} by default. Type \?useproduct for help how to change to \texttt{cmulNUM}. 
3.1 Recursive procedure `cmulNUM`

The evaluation of Clifford products in the Grassmann basis is quite involved and it is normally done by a recursive process that involves Chevalley deformation. This algorithm is encoded in `cmulNUM`. Unfortunately, during the recursive evaluation many superfluous terms appear that later cancel out at the next recursive call. When the bilinear form is sparse numeric, many branches of the recursion are cut out by Maple quite early due to automatic evaluation that takes precedence over the recursion. In this case, the superfluous terms disappear and are not passed on to the next recursive step. However, in the symbolic case, in general, all these terms might be non-zero which prevents fast completion of the recursion. Fortunately, Hopf combinatorial methods free of the drawbacks of the recursion can also be applied and have been encoded in `cmulRS`. Thus, the two ways to evaluate the Clifford product in `CLIFFORD` have emerged.

We introduce the Chevalley deformation and the Clifford map to explain the algorithm used in `cmulNUM`. The Clifford map $\gamma_x$ is defined on $u \in \bigwedge V$ as

(i) $\gamma_x(u) = LC(x, u, B) + \text{wedge}(x, u) = x \, J_B \, u + x \wedge u$

(ii) $\gamma_x \gamma_y = \gamma_x \wedge y + B(x, y) \gamma_1$

(iii) $\gamma_{ax + by} = a \gamma_x + b \gamma_y$

where $x, y \in V$ (see, for example, [19]). One knows how to compute with the wedge $x \wedge u$ and the left contraction $x \, J_B \, u$ with respect to the bilinear form $B$ (in `CLIFFORD`, the left contraction $J_B$ is given by the procedure $LC(x, u, B)$). Following Chevalley, the left contraction has the following properties:

(i) $x \, J_B \, y = B(x, y)$

(ii) $x \, J_B \, (u \wedge v) = (x \, J_B \, u) \wedge v + \hat{u} \wedge (x \, J_B \, v)$

(iii) $(u \wedge v) \, J_B \, w = u \, J_B \, (v \, J_B \, w)$

where $x \in V$, $u, v \in \bigwedge V$ and $\hat{u}$ is the Grassmann grade involution. Hence we can use the Clifford map $\gamma_x$ (Chevalley deformation of the Grassmann algebra) to define a Clifford product of a one-vector $x$ and a multivector $u$ as

$xu = x \, J_B \, u + x \wedge u.$

Analogous formula can also be given for a right Clifford map using the right contraction $L_B$ implemented as the procedure $\text{RC}$.

The Clifford product `cmul` or its ampersand form `&c` of two Grassmann basis monomials can now be defined as follows: We have to split off recursively a single element from the first factor of the product and then use the Chevalley’s Clifford map. Namely,

\[
\begin{align*}
(e_a \wedge \ldots \wedge e_b \wedge e_c) \&c (e_f \wedge \ldots \wedge e_g) &= \\
&= (e_a \wedge \ldots \wedge e_b) \&c (e_c \, J_B \, (e_f \wedge \ldots \wedge e_g) + e_c \wedge e_f \wedge \ldots \wedge e_g) \\
&\quad - ((e_a \wedge \ldots \wedge e_b) \, L_B \, e_c) \&c (e_f \wedge \ldots \wedge e_g). \quad (1)
\end{align*}
\]
Specifically, for \((e_1 \wedge e_2) \& c \ (e_3 \wedge e_4)\) we have

\[
(e_1 \wedge e_2) \& c \ (e_3 \wedge e_4) = (e_1 \& c \ e_2) \& c \ (e_3 \wedge e_4) - B(e_1, e_2) 1 \& c \ (e_3 \wedge e_4) \\
= e_1 \& c \ (B(e_2, e_3)e_4 - B(e_2, e_4)e_3 + e_2 \wedge e_3 \wedge e_4) - B(e_1, e_2) 1 \& c \ (e_3 \wedge e_4)
\]

and a second recursion of the process gives now

\[
= B(e_2, e_3)B(e_1, e_4) - B(e_2, e_4)B(e_1, e_3) + B(e_2, e_3)(e_1 \wedge e_4) \\
- B(e_2, e_4)(e_1 \wedge e_3) + B(e_1, e_2)(e_3 \wedge e_4) - B(e_1, e_3)(e_2 \wedge e_4) \\
+ B(e_1, e_4)(e_2 \wedge e_3) + e_1 \wedge e_2 \wedge e_3 \wedge e_4 - B(e_1, e_2)(e_3 \wedge e_4)
\]

with the bolded terms canceling out. Note that the last term in the r.h.s. was superfluously generated in the first step of the recursion.

The Clifford product can be derived from the above recursion by linearity and associativity. The induction starts with a left factor of grade one or grade zero which is trivial, i.e., \(1 \& c \ e_0 \ldots \wedge e_0 = e_0 \wedge \ldots \wedge e_0\). In the case when the left factor is of grade one, we use the Clifford product expressed by the Clifford map of Chevalley, i.e., \(e_a \& c \ e_b \wedge \ldots \wedge e_c = e_a \& B(e_b \wedge \ldots \wedge e_c) + e_a \wedge e_b \wedge \ldots \wedge e_c\).

We make a complete induction in the following way: If the left factor is of higher grade, say \(n\), one application of the recursion yields Clifford products where the new left factor is of grade either \(n - 1\) or \(n - 2\), hence the recursion stops after at most \(n - 1\) steps.

A disadvantage of the recursive approach is that additional terms are produced by shifting Grassmann wedge products into Clifford products in order to swap one factor to the right. While these terms eventually cancel out, their computation increases unnecessarily the total computing time. More importantly, they may easily exhaust any computer memory available and prevent Maple from completing the computation of the product.

An advantage of the recursive approach is realized when the bilinear form \(B\) is numeric and sparse, that is, with many zeros. In this case, after each recursive step many terms drop out since Maple automatically simplifies expressions as soon as they are computed so that only a few remaining terms enter into the next step of the recursion. If the dimension of \(V\) is large, i.e., \(\dim V \geq 6\), computation with a sparse matrix \(B\) benefits dramatically from this simplification process. It is then more efficient than the Hopf combinatorial algorithm encoded in \texttt{cmulRS} which computes all terms in the final output without taking advantage of the sparseness of \(B\) except at the final evaluation step.

One could try to shift factors from the right to the left, however this works the same way and requires the same number of steps. Moreover, if the grade of the left factor \(n\) is greater than the grade \(m\) of the right factor, then the recursion stops also (since the terms evaluate to zero) after at most \(n - 1\) steps, so no increase in performance can be gained this way. For example, the above computation in Maple will be performed as follows:

```maple
> cmul(e1w2,e3w4);
(B2_3 B1_4 - B2_4 B1_3) ld + B2_3 e14 - B2_4 e13 - B1_3 e24 + B1_4 e23 + e1234
```
Notice also that \texttt{cmul} accepts an arbitrary bilinear form $K$ as its argument:

\begin{verbatim}
> cmul[K](e1we2,e3we4);
\end{verbatim}

\[(K_{2,3} K_{1,4} - K_{2,4} K_{1,3}) \text{Id} + K_{2,3} e_{14} - K_{2,4} e_{13} - K_{1,3} e_{24} + K_{1,4} e_{23} + e_{1234}\]

and likewise its ampersand form\(^8\)

\begin{verbatim}
> &c[K](ei,ejwekwel);
\end{verbatim}

\[eiwejwekwel + K_{i,j} ejwel - K_{i,k} ejwel + K_{i,1} ejwek\]

where we have also shown the ability of \textsc{Clifford} to use symbolic indices. For clarity and to show our approach we display the algorithm of \texttt{cmulNUM} in Appendix A.

### 3.2 Procedure \texttt{cmulRS} based on the Rota-Stein combinatorial process

The procedure \texttt{cmulRS} is computed using the non-recursive Rota-Stein cliffordization. See \cite{4, 5, 12, 20} and \textsc{Bigebra} help pages for additional references. The cliffordization process is based on the Hopf algebra theory. The Clifford product is obtained from the Grassmann wedge product and its Grassmann co-product as shown by the following tangle:

\begin{equation}
\Delta \wedge \Delta \wedge B \wedge \wedge \wedge (2)
\end{equation}

Here $\wedge$ is the Grassmann exterior wedge product and $\Delta \wedge$ is the Grassmann exterior co-product which is obtained from the wedge product by a categorial duality: To every algebra over a linear space $A$ with a product we find a co-algebra with a co-product over the same space by reversing all arrows in all axiomatic commutative diagrams. Note that the co-product splits each input ‘factor’ $x$ into a sum of tensor products of ordered pairs $x_{(1)i}, x_{(2)i}$. The main requirement is that every such pair multiplies back to the input $x$ when the dual operation of multiplication is applied, i.e., $x_{(1)i} \wedge x_{(2)i} = x$ for each $i$-th pair. The ‘cup’ like part of the tangle decorated with $B \wedge$ is the bilinear form $B$ on the generating space $V$ extended to the whole Grassmann algebra: It is a map $B \wedge : \wedge V \times \wedge V \to k$ with $B : V \times V \to k$ evaluating to $B(x,y)$ on vectors in $V$.

\(^8\)Procedures \texttt{cmulNUM} and \texttt{cmulRS} do not have their special ampersand forms. Procedure $&c$ uses internally \texttt{cmulNUM} or \texttt{cmulRS} depending on the current value of an environmental variable \_default\_Clifford\_product. Current value of this and other environmental variables can be displayed by a procedure \texttt{CLIFFORD\_ENV}.  

11
Hence, \texttt{cmulRS} computes the Clifford product on Grassmann basis monomials \(x\) and \(y\) for the given \(B\), which is later extended to Clifford polynomials by bilinearity, as follows:

\[
\text{cmulRS}(x, y, B) = \sum_{i=1}^{n} \sum_{j=1}^{m} (\pm) x_{(1)i} \wedge y_{(2)j} B(x_{(2)i}, y_{(1)j})
\]  

where \(n\) and \(m\) give the cardinalities of the required splits and the sign is due to the parity of a permutation needed to arrange the factors.

A simplified algorithm of \texttt{cmulRS} looks as follows:

\texttt{cmulRS}(x,y,B) [x, y two Grassmann monomials, B - bilinear form]

\begin{verbatim}
begin
  lstx <- list of indices from x
  lsty <- list of indices from y
  NX <- length of lstx
  NY <- length of lsty
  funx <- function maps integers 1..NX onto elements of lstx keeping their order
  funy <- function maps integers 1..NY onto elements of lsty keeping their order
    (this is to calculate with arbitrary indices and to compute necessary signs)
  psetx <- power set of 1..NX (actually a list in a certain order)
    (the i-th and (2ˆNX+1-i)-th element are disjoint adding up to the set \{1..NX\})
  psety <- power set of 1..NY (actually a list in a certain order)
    (the i-th and (2ˆNY+1-i)-th element are disjoint adding up to the set \{1..NY\})
    (for faster computation we sort this power sets by grade)
    (we compute the sign for any term in the power set)
  psetx <- sort psetx by grade
  psety <- sort psety by grade
  pSgx <- sum_{i in psetx} (-1)^sum_{j in psetx[i]} (psetx[i][j]-j)
  pSgy <- sum_{i in psety} (-1)^sum_{j in psety[i]} (psety[i][j]-j)
    (we need a subroutine for cup tangle computing the bilinear form cup(x,y,B))
  begin cup
    if |x| <> |y| then return 0 end if
    if |x| = 0 then return 1 end if
    if |x| = 1 then return B[x[1],y[1]] end if
    return sum_{j in 1..|x|}(1-(-1)^j)*B(x[1],y[j])\*cup(x[2..-1],y/y[j],B)
  end cup
  (now we compute the double sum, to gain efficiency we do this grade wise)
  (note that there are r over NX r-vectors in psetx, analogously for psety)
  max_grade - |lstx <- convert_to_set union lsty <- convert_to_set|
  res <- 0, pos1 <- 0
  for j from 0 to NX (iterate over all j-vectors of psetx)
    begin
      F1 <- N1!/(N1-j)!\*j!  (number of terms (N1 over j))
      pos2 <- 0
      for i from 0 to min(N2,max_grade-j)
        begin
          \end{verbatim}
(iterate over all i-vectors of psety not exceeding max_grade while others are zero)
begin
F2 <- N2!/( (N2-i)! * i! ) (number of terms (N2 over i))
for n from 1 to F1 (for all j-vectors)
begin
for m from 1 to F2 (for all i-vectors)
begin
res <- res + pSgmx[pos1+n]*pSgmy[pos2+m]*
*cup(fun1(psetx[PN1-pos1-n]),fun2(psety[pos2+m]),lname)*
makeclibasmon -> ([fun1 -> psetx[pos1+n],fun2 -> psety[PN2-pos2-m]])
end
end
pos2 <- pos2 + F2
end
pos1 <- pos1 + F1
end
reorder -> res (reorder basis elements in res into standard order)
end cmulRS

It is clear from this algorithm that only those terms are considered which might be non-zero: If all $B_{i,j}$ are non-zero and different so that no cancelation takes place between them, all these terms will survive. The combinatorial power of the Hopf algebraic approach is clearly demonstrated with this algorithm and its superior behavior shows up in benchmarks [4].

4 Dotted and undotted Graßmann bases in quantum Clifford algebras

4.1 On the meaning of the dotted wedge

We are going to show that Graßmann algebras with the dotted and the undotted wedge products are isomorphic. From a mathematical point of view, isomorphic objects are identical. The importance of studying otherwise isomorphic but not identical algebras comes from the fact that one is interested in various situations when one needs pairs of such algebras. This leads to a relative isomorphism, which is then mathematically and physically relevant. We just mention two places where the dotted wedge appears.

- In quantum field theory one needs to study various orderings of field operator products and/or correlation functions. In fermionic quantum field theory, a normal ordered product is expressed in terms of graded-commutative wedge products. A transition to time ordered products resp. correlation functions is equivalent to a transition to the dotted wedge products. The antisymmetric bilinear form in this case is called a Wightman bilinear form, see [9, 10, 11].
In the theory of group representations one wants to deduce characters of subgroups of a given group by branching laws. If one derives the branching \( U(n) \downarrow U(n-1) \) one encounters a pair of products which are related to the transition from the undotted to the dotted wedge, see [13].

In general, one can use Hopf algebra cohomology to classify maps which connect the various products. From this analysis it is known that algebra isomorphisms are related to 1-cocycles. The 1-cocycle condition guarantees that the transition is an algebra homomorphism. Below, we investigate in which way the wedge product –related to the creation operators– and the contraction –related to the annihilation operators– is affected by the algebra isomorphism induced by the antisymmetric part \( F \) of a bilinear form \( B \). This analysis can be extended to symmetric algebras [13] and to superspaces [9].

It was shown above that CLIFFORD uses the Grassmann algebra \( \Lambda V \) as the underlying vector space of the Clifford algebra \( C\ell(V, B) \). Thus, the Grassmann wedge basis of monomials is the standard basis used in CLIFFORD. A general element \( u \) in \( C\ell(V, B) \) can be therefore viewed as a Grassmann polynomial.

When the bilinear form \( B \) has an antisymmetric part \( F = -F^T \), it is convenient to split it as \( B = g + F \), where \( g \) is the symmetric part of \( B \), and to introduce the so called “dotted Grassmann basis” [6] and the dotted wedge product \( \land_d \). The original Grassmann basis will be referred to here as the “undotted Grassmann basis”. In CLIFFORD, the wedge product is given by the procedure \( \wedge \) and \( \&w \) while the dotted wedge product is given by \( \wedge_d \) and \( \&dw \).

According to the Chevalley definition of the Clifford product &c, we have

\[
x &c u = x \downarrow_B u + x \&w u = \text{LC}(x, u, B) + \wedge(x, u)
\]  

for a 1-vector \( x \) and an arbitrary element \( u \) of \( C\ell(B) \). As before, \( \text{LC}(x, u, B) \) denotes the left contraction of \( u \) by \( x \) with respect to the bilinear form \( B \).

However, when \( B = g + F \) then

\[
x \downarrow_B u = \text{LC}(x, u, B) = x \downarrow_g u + x \downarrow_F u = \text{LC}(x, u, g) + \text{LC}(x, u, F)
\]  

and

\[
x &c u = \text{LC}(x, u, B) + x \&w u = \text{LC}(x, u, g) + \text{LC}(x, u, F) + x \&w u
\]

\[
= \text{LC}(x, u, g) + \text{dedge}(F)(x, u) = \text{LC}(x, u, g) + x \&dw u
\]

where \( x \&dw u = x \&w u + \text{LC}(x, u, F) \). That is, the wedge and the dotted wedge “differ” by the contraction term(s) with respect to the antisymmetric part \( F \) of \( B \). This dotted wedge \( \&dw \) can be extended to elements of higher grades. Its properties are discussed next.

### 4.2 Indexing \( \wedge_d \) and \( \&dw \)

Procedure \( \text{dedge} \) (and its infix form \( \&dw \)) requires an index which can be a symbol or an antisymmetric matrix. That is, \( \text{dedge} \) computes the dotted
wedge product of two Graßmann polynomials and expresses its answer in the undotted basis. Special procedures exist which convert polynomials between the undotted and dotted bases. When no index is used, the default is $F$:

\[
\begin{align*}
&\text{d}wedge[\mathbb{K}](e_1+2e_2we_3,e_4+3e_1we_2);&\text{d}wedge(e_i+2e_jwe_k,e_i+2e_jwe_k); \\
&-(K_{1,4}+6K_{2,3}K_{1,2})\text{Id} - 6K_{1,2}e2we3 - 6K_{2,3}e1we2 - 2K_{2,4}e3 + \\
&2K_{3,4}e2 - 3K_{1,2}e1 + e1we4 + 2e2we3we4 \\
&4e1wejwek - 4F_{i,k}ej + 4F_{i,j}ek - 8F_{j,k}ejwek - 4F_{j,k}^2\text{Id}
\end{align*}
\]

Observe that conversion from the undotted wedge basis to the dotted wedge basis using antisymmetric form $F$ and $\text{d}wedge[F]$ are related through the following convert function:

\[
\begin{align*}
&\text{d}wedge[F](e_1,e_2,...,e_n) = \text{convert}(e_1we2we...wemen,\text{wedge_to_dwedge},F)
\end{align*}
\]

which can be shown as follows:

\[
\begin{align*}
&F:=\text{array}(1..9,1..9,\text{antisymmetric}); \\
&\text{d}wedge[F](e_1,e_2)=\text{convert}(\text{wedge}(e_1,e_2),\text{wedge_to_dwedge},F); \\
&e1we2 + F_{1,2}\text{Id} = e1we2 + F_{1,2}\text{Id}
\end{align*}
\]

\[
\begin{align*}
&\text{d}wedge[F](e_1,e_2,e_3)=\text{convert}(\text{wedge}(e_1,e_2,e_3),\text{wedge_to_dwedge},F); \\
&e1we2we3 + F_{2,3}e1 - F_{1,3}e2 + F_{1,2}e3 = e1we2we3 + F_{2,3}e1 - F_{1,3}e2 + F_{1,2}e3
\end{align*}
\]

\[
\begin{align*}
\text{wedge_to_dwedge} \\
\text{d}wedge \to \text{wedge}
\end{align*}
\]

Diagram 1: Isomorphisms between $\text{Cl}(B)_\wedge$ and $\text{Cl}(B)_\dot\wedge$.

### 4.3 Associativity of $\text{d}wedge$

Operation $\text{d}wedge$ is associative with the unity $1 = \text{Id}$ as a unit:

\[
\begin{align*}
&\text{evalb}(\text{d}wedge[F](\text{d}wedge[F](e_1,e_2),e_3)=\text{d}wedge[F](e_1,\text{d}wedge[F](e_2,e_3))); \\
&\text{true}
\end{align*}
\]

Associativity can be encoded in a commutative diagram, see diagram 2. It was checked with CLIFFORD up to dimension 5.\(^9\)

For some arbitrary random Clifford polynomials\(^10\) $u,v,z$ expressed in Graßmann undotted basis we can show associativity as follows:

\(^9\)There is a proof making use of Hopf algebra cohomology that any 2-cocycle deformation comes up with an associative product. The deformation by $F$ is such a 2-cocycle and associativity can be proved for the general case.

\(^10\)In CLIFFORD ver. 6 and higher there are three procedures useful for testing that return a random Graßmann basis monomial, a random monomial and a random polynomial, respectively. See ?rd_clibasmon, ?rd_climon, ?rd_clipolynom.
\[
\begin{array}{ccc}
\text{Cl}(B)^\wedge \otimes \text{Cl}(B)^\wedge & \xrightarrow{\text{dedge}[F] \otimes 1} & \text{Cl}(B)^\wedge \otimes \text{Cl}(B)^\wedge \\
\downarrow 1 \otimes \text{dedge}[F] & & \downarrow \text{dedge}[F] \\
\text{Cl}(B)^\wedge \otimes \text{Cl}(B)^\wedge & \xrightarrow{\text{dedge}[F]} & \text{Cl}(B)^\wedge
\end{array}
\]

Diagram 2: Associativity of \text{dedge}[F] in \text{Cl}(B)^\wedge.

\[
\text{true}, \text{true}
\]

\[
\text{evalb}(\text{dedge}[F](\text{Id},u)=u), \text{evalb}(\text{dedge}[F](u,\text{Id})=u);
\]

\[
\text{true}
\]

We have, therefore, the following identity that expresses an isomorphism between two Graßmann algebras: dotted and undotted. For any two elements \(u\) and \(v\) in \(\text{Cl}(B)\), \(B = g + F\), that are, by default, expressed in terms of the undotted Graßmann basis, we find:

\[
u \wedge v = (u_F \wedge v_F)_{-F}. \quad (9)
\]

Here \(u_F\) and \(v_F\) are the elements \(u\) and \(v\) expressed in the dotted basis with respect to the form \(F\) while \((...)_{-F}\) denotes conversion back from the dotted basis to the undotted basis w.r.t. \(-F = F^T\). \(\text{Cl}(B)^\wedge\) and \(\text{Cl}(B)^\dot\wedge\) denote the modules w.r.t. the two filtrations in use.

\[
\begin{array}{ccc}
\text{Cl}(B)^\wedge \otimes \text{Cl}(B)^\wedge & \xrightarrow{(\ldots)F \otimes (\ldots)F} & \text{Cl}(B)^\wedge \otimes \text{Cl}(B)^\wedge \\
\downarrow \wedge & & \downarrow \dot\wedge \\
\text{Cl}(B)^\wedge & \xleftarrow{(\ldots)_{-F}} & \text{Cl}(B)^\wedge
\end{array}
\]

Diagram 3: Relation between \(\wedge\) and \(\dot\wedge\) products.

This can be illustrated in CLIFFORD as follows:

\[
> \text{uu} := \text{convert}(u, \text{wedge\_to\_dedge}, F); \text{vv} := \text{convert}(v, \text{wedge\_to\_dedge}, F);
\]

\[
uu := e1 - 3 e2we3 - 3 F_{2,3} \text{Id} + 2 \text{Id}
\]

\[
vv := 3 \text{Id} - 4 e1we3 - 4 F_{1,3} \text{Id} + e7
\]
> out1:=dwedge[F](uu,vv): #dwedge computed w.r.t. F
> out2:=convert(out1,dwedge_to_wedge,-F); #back to undotted basis

\[
\text{out2} := 3 e_1 - 9 e_2 e_3 + 6 \text{Id} - 8 e_1 e_3 + e_1 e_7 - 3 e_2 e_3 e_7 + 2 e_7
\]

> out3:=wedge(u,v); #direct computation of wedge product

\[
\text{out3} := 3 e_1 - 9 e_2 e_3 + 6 \text{Id} - 8 e_1 e_3 + e_1 e_7 - 3 e_2 e_3 e_7 + 2 e_7
\]

and it can be seen that \(\text{out2} = \text{out3}\) establishing the relation:

### 4.4 Dotted and undotted wedge bases

The default Graßmann basis in \(\mathcal{C}(B)\) used in CLIFFORD is undotted. However, one can easily use the dotted basis. For example, we expand the basis of the original wedge into the dotted wedge, and back. For this purpose we choose \(\dim V = 3\) and consider \(\mathcal{C}(B)\) with the antisymmetric part \(F\). The undotted wedge basis for \(\bigwedge V\) is then:

\[
\text{w_bas} := [\text{Id}, e_1, e_2, e_3, e_1 e_2, e_1 e_3, e_2 e_3, e_1 e_2 e_3]
\]

Now we map the convert function onto this basis to get the dotted wedge basis:

\[
\text{d_bas} := \text{map(convert, w_bas, wedge_to_dwedge, F)}
\]

\[
\text{test_wbas} := \text{map(convert, d_bas, dwedge_to_wedge, -F)}
\]

\[
\text{d_bas} := [\text{Id}, e_1, e_2, e_3, e_1 e_2 + F_{1,2} \text{Id}, e_1 e_3 + F_{1,3} \text{Id}, e_2 e_3 + F_{2,3} \text{Id},
\]

\[
e_1 e_2 e_3 + F_{2,3} e_1 - F_{1,3} e_2 + F_{1,2} e_3]
\]

\[
\text{test_wbas} := [\text{Id}, e_1, e_2, e_3, e_1 e_2, e_1 e_3, e_2 e_3, e_1 e_2 e_3]
\]

Notice that only the unity 1 and the one vector basis elements \(e_i\) remain unaltered and that the other basis elements of higher grades pick up additional terms of lower grades (which preserves the filtration). It is possible to define aliases in CLIFFORD for the dotted wedge basis “monomials” similar to the Graßmann basis monomials. For example, we could denote the element \(e_1 e_2 + F[1,2] \text{Id}\) by \(e_1 W e_2 (= e_1 \wedge e_2)\) and similarly for other elements:

\[
\text{alias(e1We2=e1we2 + F[1,2]*Id, e1We3=e1we3 + F[1,3]*Id,}
\]

\[
e_2 W e_3=e2we3 + F[2,3]*Id, e_1 W e_2 e_3=e1we2we3+F[2,3]*e1-F[1,3]*e2+F[1,2]*e3);
\]

\[I, e_1 W e_2, e_1 W e_3, e_2 W e_3, e_1 W e_2 W e_3\]

and then Maple will automatically display dotted basis in terms of the aliases:

\[
\text{d_bas};
\]

\[
[\text{Id}, e_1, e_2, e_3, e_1 W e_2, e_1 W e_3, e_2 W e_3, e_1 W e_2 W e_3]
\]
That is, as linear spaces we find the isomorphism:

\[ \mathcal{C}l(B) \cong \langle 1, e_1, e_2, e_3, e_1 \wedge e_2, e_1 \wedge e_3, e_2 \wedge e_3, e_1 \wedge e_2 \wedge e_3 \rangle \cong \langle 1, e_1, e_2, e_3, e_1 \wedge e_2, e_1 \wedge e_3, e_2 \wedge e_3, e_1 \wedge e_2 \wedge e_3 \rangle \]

where \( e_1 \wedge e_2 = e_1 \text{We} e_2 \), etc.

4.5 Contraction in dotted and undotted bases

The contraction \( J_K \) w.r.t. any bilinear form \( K \) works on both undotted and dotted bases in the same manner which can be seen if we re-convert the dotted basis after the computation into the wedge (undotted) basis. In a reasonable setting, the antisymmetric bilinear form \( F \) would be the antisymmetric part of \( B \). To read more about the left contraction \( LC \) in \( \mathcal{C}l(B) \) check the help page for \( LC \) or see [6]. We have the following identity for any two elements \( u \) and \( v \) in \( \mathcal{C}l(B) \) expressed in the undotted Grassmann basis:

\[ v \cdot J_B \cdot u = (v \cdot J_B \cdot u_F)_{-F} \]  

Diagram 4: Contraction w.r.t. wedge and dotted wedge.

As before, \( u_F \) is the element \( u \) expressed in the dotted basis while \( (\ldots)_{-F} \) accomplishes conversion back to the undotted basis. To illustrate this fact, we first contract from the left an arbitrary element \( u \) in \( \mathcal{C}l(B) \) by \( 1, e_i, e_i \wedge e_j, e_i \wedge e_j \wedge e_k, 1 \leq i, j, k \leq 3 \) (here we limit our example to \( \dim V = 3 \)) and then we extend it to a left contraction by an arbitrary element \( v \) in \( \mathcal{C}l(B) \).

As before, \( u_F \) is the element \( u \) expressed in the dotted basis while \( (\ldots)_{-F} \) accomplishes conversion back to the undotted basis. To illustrate this fact, we first contract from the left an arbitrary element \( u \) in \( \mathcal{C}l(B) \) by \( 1, e_i, e_i \wedge e_j, e_i \wedge e_j \wedge e_k, 1 \leq i, j, k \leq 3 \) (here we limit our example to \( \dim V = 3 \)) and then we extend it to a left contraction by an arbitrary element \( v \) in \( \mathcal{C}l(B) \).
Contraction with respect to $e_i \wedge e_j \wedge e_k$:

\[
\text{evalb}(\text{LC}(e_i \wedge e_j \wedge e_k, u, B) = \text{convert}(\text{LC}(e_i \wedge e_j \wedge e_k, uF, B), \text{dwedge}_\text{to}_\wedge, -F));
\]

true

Finally, contraction with respect to an arbitrary element $v$:

\[
\text{evalb}(\text{LC}(v, u, B) = \text{convert}(\text{LC}(v, uF, B), \text{dwedge}_\text{to}_\wedge, -F));
\]

true

### 4.6 Clifford product in dotted and undotted bases

We can build a Clifford algebra $\mathbb{C}\ell(B)$ over each basis set, that is, over the undotted or dotted Graßmann basis, but with different bilinear forms: $B = g$ or $B = g + F$ respectively (following notation from [6]). Let us compute various Clifford products with respect to the symmetric form $g$ and with respect to the full form $B$ using procedure cmul that takes a bilinear form as its index. As an example, we will use two most general elements $u$ and $v$ in $\Lambda V$ when $\dim V = 3$. Most output will be eliminated.

\[
u := \text{add}(x.k \cdot w_{\text{bas}}[k+1], k = 0..7); v := \text{add}(y.k \cdot w_{\text{bas}}[k+1], k = 0..7);
\]

We can then define in $\Lambda V$ a Clifford product cmul$[g]$ with respect to the symmetric part $g$ and another Clifford product cmul$[B]$ with respect to the entire form $B$:

\[
\text{cmulg} := \text{proc}(\text{args}) \text{ return cmul}[g](\text{args}) \text{ end proc};
\]

\[
\text{cmulB} := \text{proc}(\text{args}) \text{ return cmul}[B](\text{args}) \text{ end proc};
\]

Thus, we are ready to perform computations around our next commutative diagram, however most output will be eliminated to save space.

\[
\begin{array}{c}
\text{cmul}[g] \\
\text{cmul}[B]
\end{array}
\]

\[
\begin{array}{ccc}
\mathbb{C}\ell(g)_{\wedge} \otimes \mathbb{C}\ell(g)_{\wedge} & \stackrel{(\ldots)F \otimes (\ldots)F}{\longrightarrow} & \mathbb{C}\ell(g)_{\wedge} \otimes \mathbb{C}\ell(g)_{\wedge} \\
\downarrow & & \downarrow \\
\mathbb{C}\ell(g)_{\wedge} & \stackrel{(\ldots)_{-F}}{\longrightarrow} & \mathbb{C}\ell(g)_{\wedge}
\end{array}
\]

Diagram 5: Clifford multiplications cmul$[g]$ and cmul$[B]$ w.r.t. dotted and undotted basis.

First, we compute the Clifford product cmul$[g](u, v)$ in $\mathbb{C}\ell(g)$ in undotted Graßmann basis.

\[
uv := \text{cmulg}(u, v); \#\text{Clifford product w.r.t. } g \text{ in } \mathbb{C}\ell(g) \text{ in wedge basis}
\]

Now, we convert $u$ and $v$ to $u_F$ and $v_F$, respectively, expressed in the dotted wedge basis:

\[
uF := \text{convert}(u, \text{wedge}_\text{to}_\text{dwedge}, F); vF := \text{convert}(v, \text{wedge}_\text{to}_\text{dwedge}, F);
\]

We now compute the Clifford product of $u_F$ and $v_F$ in $\mathbb{C}\ell(B)$ in the dotted wedge basis:

\[
uFvF := \text{cmulB}(uF, vF); \#\text{Clifford product in } \mathbb{C}\ell(B) \text{ in dwedge basis}
\]

convert back the above result back to the undotted wedge basis:

\[
\text{uv2} := \text{convert}(uFvF, \text{dwedge}_\text{to}_\wedge, -F); \#\text{convert result dwedge}\to\wedge
\]

and verify that the results are the same:
Thus, we have shown that the following identity involving \( cmul[g] \) and \( cmul[B] \) is true (at least when \( \dim V = 3 \)).\(^{11}\) The result is folklore, and may be found e.g. in \([7, 15]\).

\[
(u \cdot v)_g = u \wedge c_g v = (u_F \wedge c_B v_F)_F = ((u_F v_F)_B)_F \quad (11)
\]

This shows that the Clifford algebra \( \Cl(g) \) of the symmetric part \( g \) of \( B \) using the undotted exterior basis is isomorphic, as an associative algebra, to the Clifford algebra \( \Cl(B) \) of the entire bilinear form \( B = g + F \) spanned by the dotted wedge basis if the antisymmetric part \( F \) of \( B \) is exactly the same as \( F \) used to connect the two bases.

\[ (\ldots)_F \in \Hom_{\Lambda B}(\Cl(g), \Cl(B)), \quad B = g + F \]

### 4.7 Reversion in dotted and undotted bases

We proceed to show that the expansion of the Clifford basis elements into the dotted or undotted exterior products has also implications for other well known operations such as the Clifford reversion anti-automorphism \( \tilde{\cdot} : \Cl(B) \rightarrow \Cl(B) \), \( uv \mapsto \tilde{v} \tilde{u} \), which preserves the grades in \( \bigwedge V \) [but not in \( \bigwedge V \) unless \( B \) is symmetric.] Only when the bilinear form is symmetric, we find that the reversion is grade preserving, otherwise it reflects only the filtration: That is, reversed elements are in general sums of terms of the same and lower degrees.

\[ \text{reversion}(e_1 \wedge e_2, B); \quad \text{reversion}(e_1 \wedge e_2, g); \quad \text{reversion}(e_1 \wedge e_2, g) \]

\[ -e_1 \wedge e_2 - 2F_{1,2} \text{Id} \]

\[-e_1 \wedge e_2 \]

To provide a clear idea how the various reversions are related, we consider the following commuting diagram:

\[
\begin{array}{ccc}
\Cl(B)_\Lambda & \xrightarrow{\text{reversion}[B](\ldots)} & \Cl(B)_\Lambda \\
\downarrow{\text{reversion}[g](\ldots)} & & \downarrow{(\ldots)_2 F} \\
\Cl(B)_\Lambda & & \Cl(B)_\Lambda
\end{array}
\]

Diagram 6: Relation between \( \text{reversion}[g] \) and \( \text{reversion}[B] \) and the basis transformation \( (\ldots)_2 F \).

The reader should note that the map, depicted by the diagonal arrow, involves a change of basis induced by the antisymmetric bilinear form \( 2F \) and not \( F \). The factor 2 is crucial and appears due to an asymmetry between the undotted and dotted bases.

\(^{11}\)Here, \( (u \cdot v)_g \) is the Clifford product with respect to \( g \) while \( u_F \wedge c_B v_F \) and \( (u_F v_F)_B \) are the Clifford products with respect to \( B \), that is, in \( \Cl(g) \) and \( \Cl(B) \), respectively.
This suggests to introduce a symmetrically related triple of bases w.r.t. $-F/2, F \equiv 0$ and $F/2$. In such a setting, $F$ (resp. $-F$) connects the two dotted bases induced by $\pm F/2$.

Observe in the pre-last display above that only when $B_{1,2} = B_{2,1}$, the result $-e_1 \wedge e_2$ known from the theory of classical Clifford algebras is obtained. Likewise,

```map(reversion, cbas, B);
```

If instead of $B$ we use a symmetric matrix $g = g^T$ (or the symmetric part of $B$), then

```map(reversion, cbas, g);
```

Convert now $e_1 \wedge e_2$ to the dotted basis to get $e_1 \dot \wedge e_2 = e_1 We_2$:

```convert(e1we2, wedge_to_dwedge, F);
```

Apply reversion to $e_1 We_2$ with respect to $F$ to get the reversed element in the dotted basis:

```reversed_e1We2 := reversion(e1We2, F);
```

Observe, that the above element is equal to the negative of $e_1 We_2$ just like reversing $e_1we2$ with respect to the symmetric part $g$ of $B$:

```reversed_e1We2 + e1We2;
```

Finally, convert reversed $e_1 We_2$ to the undotted standard Graßmann basis to get $-e_1we2$:

```convert(reversed_e1We2, dwedge_to_wedge, -F);
```

The above, of course, can be obtained by applying reversion to $e1we2$ with respect to the symmetric part $g$ of $B$:

```reversion(e1we2, g); #reversion w.r.t. the symmetric part g
```

- $e1we2$
This shows that the dotted wedge basis is the particular basis which is stable under the Clifford reversion computed with respect to $F$, the antisymmetric part of the bilinear form $B$. This requirement allows one to distinguish Clifford algebras $\mathcal{C}(g)$ which have a symmetric bilinear form $g$ from those which do not have such symmetric bilinear form but a more general form $B$ instead. We call the former classical Clifford algebras while we use the term quantum Clifford algebras for the general not necessarily symmetric case [2].

$$C\ell(X) \otimes C\ell(X) \rightarrow C\ell(X)$$

Diagram 7: Relation between the reversion of type $X \in \{g,F,B\}$ with the corresponding Clifford multiplication $\text{cmul}[X]$. The map called switch is the ungraded switch of tensor factors, that is, $\text{switch}(A \otimes B) = B \otimes A$.

5 Conclusions

This paper continues with the second part [5] about BIGEBRA where further aims and outlooks for the future applications of CLIFFORD and BIGEBRA are given.

Appendix A: Code of cmulNUM

Here is a shortened code of the recursive procedure cmulNUM.

\begin{verbatim}
cmulNUM(a1,a2,B) [a1, a2 - two Grassmann monomials, B - name of bilinear form] begin if nargs <>3 then error "exactly three arguments are needed" end if if has(0,map(simplify,[a1,a2])) then return 0 end if if a2='Id' then return a1 end if if a1='Id' then return a2 end if L <- indices from a1 N <- length of L coB,nameB <- coefficient of B, B [to handle -B] if N=0 then return coeff(a1,Id)*a2 elif N=1 then x1 <- substring(a1,1..2) return reorder(simplify(makeclibasmon([L[1],op(L2)]))) +add((-1)^(i-1)*coB*nameB[L[1],L2[i]]*makeclibasmon(subs(L2[i]=NULL,L2)),i=1..nops(L2))) # this line seems to be missing elif N=2 then x1 <- substring(a1,1,..2)
\end{verbatim}
x2 <- substring(a1,4..5)
p2 <- procname(x2,a2,B)
S <- clibilinear(x1,p2,procname,B)
return simplify(S-coB*nameB[op(L)]*a2)
end if:
x <- cat(e,L[-1])
p1 <- substring(a1,1..(3*N-4))
p2 <- procname(x,a2,B)
S <- clibilinear(p1,p2,procname,B)
- add((-1)^i*coB*nameB[L[-i],L[-1]]*
   procname(makeclibasmon(subs(L[-i]=NULL,L[1..-2])),a2,B),i=2..N)
return reorder(simplify(S))
end cmulNUM

Acknowledgment:
The first author, R.A., gratefully acknowledges financial support from the College of Arts and Sciences, Tennessee Technological University, to present a preliminary version of this paper at the ACA 2002 in Volos, Greece.

References

[1] Abłamowicz, R.: Spinor representations of Clifford algebras: A symbolic approach, Computer Physics Communications 115 (1998) 510–535
[2] Abłamowicz, R., and Fauser, B.: On the Decomposition of Clifford Algebras of Arbitrary Bilinear Form in: Clifford Algebras and their Applications in Mathematical Physics, R. Abłamowicz and B. Fauser, eds. (Birkhäuser, Boston, 2000) 341–366
[3] Abłamowicz, R., and Fauser, B.: CLIFFORD, Cliplus, and Bigebra for Maple, http://math.tntech.edu/rafal/ (2004)
[4] Abłamowicz, R., and Fauser, B.: Effective algorithms for evaluating Clifford products (In preparation)
[5] Abłamowicz, R., and Fauser, B.: Clifford and Grassmann Hopf algebras via the BIGEBRA package for Maple, Computer Physics Communications (To appear) (2005)
[6] Abłamowicz, R., and Lounesto, P.: On Clifford algebras of a bilinear form with an antisymmetric part, in: Clifford Algebras with Numeric and Symbolic Computations, R. Abłamowicz, P. Lounesto, and J. Parra, eds. (Birkhäuser, Boston, 1996) 167–188
[7] Anderson, J.L.: Green’s functions in quantum electrodynamics, Phys. Rev. 171(94) (1954) 703–11
[8] Bourbaki, N.: Algebra 1 (Springer Verlag, Berlin, 1989)
[9] Brouder, Ch., Fauser, B., Frabetti, A., and Oeckl, R.: Quantum field theory and Hopf algebra cohomology [formerly ‘Let’s twist again’] J. Phys. A: Math. Gen: 37(22) (2004) 5895–5927, (hep-th/0311253)
[10] Fauser, B.: On an easy transition from operator dynamics to generating functionals by Clifford algebras, *J. Math. Phys.* **39** (1998) 4928–47 (Preprint hep-th/9710186)

[11] Fauser, B.: Clifford geometric parameterization of Wick normal-ordering, *J. Phys. A: Math. Gen.* **34** (2001) 105–15

[12] Fauser, B.: *A Treatise on Quantum Clifford Algebras* (Habilitationsschrift, Universität Konstanz, Konstanz, 2002)

[13] Fauser, B., and Jarvis, P.D.: A Hopf laboratory for symmetric functions, *J. Phys. A: Math. Gen.* **37** (2004) 1633–1663

[14] Helmstetter, J.: Algèbres de Clifford et algèbres de Weyl, *Cahiers Math* **25** (1982)

[15] Helmstetter, J.: Monoïdes de Clifford et déformations d’algèbres de Clifford, *J. of Alg.* **111**(1) (1987) 14–48

[16] Lam, T.Y.: *The Algebraic Theory of Quadratic Forms* (The Benjamin Cummings Publishing Company, Reading, 1973)

[17] Lounesto, P.: Scalar products of spinors and an extension of Brauer-Wall groups, *Foundations of Physics* **11** (1981) 721–740

[18] Lounesto, P., Mikkola, R., and Vierros, V.: *CLICAL User Manual*, Helsinki University of Technology, Institute of Mathematics, Research Reports **A248** (1987)

[19] Lounesto, P.: *Clifford Algebras and Spinors*, 2nd. ed. (Cambridge University Press, Cambridge, 2001)

[20] Rota, G.-C., and Stein, J.A.: Plethystic Hopf algebras, *Proc. Natl. Acad. Sci. USA* **91** (1994) 13057–13061

[21] Selig, J.M.: *Geometrical Methods in Robotics* (Springer Verlag, New York, 1996)

[22] Waterloo Maple Incorporated, *Maple, a general purpose computer algebra system*, Waterloo, http://www.maplesoft.com (2004)

[23] Wright, F.: *Computing with Maple* (Chapman & Hall/CRC, Boca Raton, 2002)