**Description and Functions of a Package eClifford**
**for Computations in Clifford Algebras Cl(p,q,r)**

**by**

**Rafal Ablamowicz, Ph.D.**
**Department of Mathematics**
**Tennessee Technological University**
**Cookeville, TN 38501**
**rablamowicz@tntech.edu, USA (931) 372-3622**

This worksheet describes a new package eClifford for computations in universal Clifford algebras Cl(p, q,r) over a real quadratic space (V,Q) of dimension dim V = n = p+q+r and endowed with a quadratic form Q of signature (p,q,r). Let {e_1,e_2,...,e_n} be an orthonormal basis in (V,Q). Then, upon embedding of (V,Q) into Cl(p,q,r), the first p vectors e_1, e_2, ..., e_p square to +1; the next q vectors e_{p+1}, ..., e_{p+q} square to -1; and the last r vectors e_{p+q+1, ..., e_{p+q+r} square to 0. That is, the form Q can be degenerate with degeneracy in r dimensions. The dimension of the algebra Cl(p,q,r) is 2^{p+q+r}.

-- When r = 0, then the quadratic form Q is nondegenerate and the Clifford algebra Cl(p,q,0) is denoted as Cl(p,q).
-- When p=q=0 and r > 0, then the quadratic form Q = 0 and the Clifford algebra Cl(0,0,r) becomes the Grassmann algebra /\ V.

Here are a few major design assumptions implemented in eClifford compared to the original package Clifford/Bigebra:

1. A new data structure is implemented: instead of strings like Id, e1we2, e1we3, etc. used in Clifford/Bigebra, this package uses indexed variable e[], e[1,2], e[1,2,3] to represent basis monomials. This way, the limitation of dimension dim(V,Q) = 9 from Clifford/Bigebra is removed. That is, this new package gives the Clifford algebra Cl(p,q,r) for the quadratic space (V,Q) for any dimension dim V = n = p+q+r.

2. Like in Clifford/Bigebra, the basis of Cl(p,q,r) is the Grassmann basis

B = {e[], e[1],..., e[n], e[1,2], ..., e[n-1,n], ... e[1,2,3], ..., e[1,2,3,...n]}

or, in general, eI where I is a multi-index I = (i_1, i_2,..., i_k) with i_1 < i_2 < ... < i_k of length |I| = k where 0 <= k <= n. Thus, in particular, the basis monomials eI with |I| = k provide a basis for the vector subspace /\^k V of k-vectors in /\ V.

3. Upon loading the package eClifford, the basis monomials, including the identity element e[], become protected Maple names. This means, that the user cannot assign any values to e or e[1], e[1,2,3], etc. or an error will result.

4. This package does not require that Clifford/Bigebra package be loaded, and it does not require Maple's linear algebra packages 'linalg' or 'LinearAlgebra'. It only requires a set of procedures from 'Walshpackage.m' for computations with Walsh functions. This data file 'Walshpackage.m' is already built into the Clifford library because procedure 'cmulWalsh3' based on Walsh functions is used in Clifford/Bgebra to compute the Clifford product of two basis monomials in algebras Cl(p,q) over a non-degenerate quadratic form Q. However, the original procedure 'cmulWalsh3' implements the Clifford product only when Q is non degenerate. It uses the algorithm described in P. Lounesto's "Clifford

Algebras and Spinors", 2ed, Cambridge U. Press, 2001, on page 284.

A new procedure in eClifford called 'cmulWalsh3pqr' is a modification of the code for 'cmulWalsh3' in that it accepts any degenerate quadratic form of signature (p,q,r) whene computing the Clifford product of two basis monomials eI and eJ. Namely, it is called as cmulWalsh3pqr(eI,eJ,[p,q,r])..

5. eClifford can work in the same Maple worksheet as Clifford/Bigebra because its procedures have different, but similar, names than those of Clifford/Bigebra. As a rule, I have added an extra prefix 'e' to the names of those procedures from Clifford/Bigebra which have been implemented in 'eClifford' thus far. For example, instead of 'cbasis' we have 'ecbasis'; instead of 'cmul' we have 'ecmul'; instead of 'wedge' we have 'ewedge', etc. So far, only a handful of procedures have been implemented in eClifford only to test the feasibility of the design.

6. Two important global variables, whose values can be defined and changed by the user usually at the start of a Maple session, are:

-- _Bsignature_pqr needs to be assigned the list [p,q,r] where p,q,r are the non-negative integers which give the signature of Q. Thus, setting the values of p, q, and r in _Bisgnature_pqr defines the Clifford algebra Cl(p,q,r) in which computations are to take place. This list is used only by 'ecmul' which implements the Clifford product in Cl(p,q,r). 'ecmul' interanlly uses 'cmulWalsh3pqr' to compute the Clifford product of two basis monomials in the signature (p,q,r).

-- _scalartypes is a set of Maple names which are meant to be treated as scalars in the Clifford algebra Cl(p,q,r). When eClifford is loaded, the value of _scalartypes is initialized to include these maple names: ^, RootOf, complex, indexed, numeric, symbol, constant, function, mathfunc, rational, algebraic. This is the same global variable as the one used by Clifford/Bigebra.

NOTE 1: For eClifford to work properly, among the scalar names saved in _scalatypes, one must have 'symbol', 'indexed', and 'algebraic', among others. If not, this sample worksheet could not be executed.
NOTE 2: By adding or removing names in this list, user can modify what is considered to be a Clifford scalar. Usually, it is better to add more names than to remove any. If something goes wrong, either restart eClifford or add the Maple type 'anything' to the list of scalars. Then, anything could be considered a scalar in the algebra.

7. I have designed two useful conversion procedures:

-- `convert/clibas_to_eclibas` which converts any element of Cl(p,q,r) expressed in the 'old' string data type from Clifford/Bigebra to the 'new' indexed data type in eClifford;
-- `convert/eclibas_to_clibas` which converts any element of Cl(p,q,r) expressed in the 'new' indexed data type in eClifford to the 'old' string data type in Clifford/Bigebra.

The procedure `convert/eclibas_to_clibas` has a built-in error message that is displayed when an index greater than 9 is encountered in its input: this is because Clifford/Bigebra can only work when dim V <= 9. For example, this string e2we8we10 is not allowed in Clifford/Bigebra because an index k of e_k can only be one character long.

This way, one can compare computations in Cl(p,q,r) done with the 'old' Clifford/Bigebra package and the 'new' eClifford package as far as accuracy and speed.

8. Since the identity element in Clifford/Bigebra is denoted by 'Id', it cannot be used in eClifford to denote the identity element in Cl(p,q,r) if both packages are used in the same Mple worksheet, or an error will result. This is of course due to the different data structures used in both packages: 'Id' in

Clifford/Bigebra is a string whereas the identity element in eClifford represented by e[] is an indexed name. Thus, e[] cannot be aliased as 'Id' unless only the eClifford package is used. As a default, at the time of loading and initalizing eClifford, the ModuleLoad procedure defines an alias 'epsilon = e[]' where e[] is a generic name for e indexed by an empty list, that is, the identity element in Cl(p,q,r). User of course can change that alias to another one following standard Maple rules for aliases. If Clifford/Bigebra is not loaded, then of course the alias 'epsilon=e[]' could be redefined as 'Id=e[]'. For more information how to do that, seek Maple help ?alias.

In the following, we have four sections:

-- Section 1 describes how to start eClifford and begin simple computations in Cl(p,q,r).
-- Section 2 descibes in detail all data types and procedures in eClifford.
-- Section 3 describes how to convert data types between eClifford and Clifford/Bigebra, and how to simultaneously compute with both packages in the same worksheet.
-- Section 4 describes some benchmarks and speed testing.

9. Please address all comments, bugs, questions, and suggestions for the improvement of this package to Rafal Ablamowicz at rablamowicz@tntech.edu.

Cookeville, November 15, 2015

**Section 1: A quick start in eClifford**

```
> restart:with(eClifford);
```

$$\begin{aligned}
&_known_types = [eclibasmon, eclimon, eclipolynom]\\
&_known_sortings = [sort/byegrade, sort/byegradenum, sort/byegradesymb]\\
&_known_conversions = [convert/binarytuple_to_eclibasmon, convert/clibas_to_eclibas,\\
&\quad convert/eclibas_to_clibas, convert/eclibasmon_to_binarytuple]\\
&\quad _revisions = [eClifford:\text{-}eclicollect, eClifford:\text{-}ecliterms, eClifford:\text{-}esplit, eClifford:\text{-}split]\\
&[ModuleLoad, ModuleUnload, byegrade, byegradenum, byegradesymb, cmulWalsh3pqr, \qquad \textbf{(1)}\\
&\quad eCliffordversion, ecbasis, eclicollect, ecliscalarpart, ecliterms, ecmul, econjugation,\\
&\quad eextract, egrade, egradeinv, emaxgrade, emaxindex, ereversion, esplit, evectorpart, ewedge,\\
&\quad split]
\end{aligned}$$

Upon loading the package above, lists of known types, known sortings, known conversions, as well as a list of exported commands available in the package are displayed. This is done so that in the absence of Maple built-in help pages, user will know names of all procedures in the package. The procedure 'eCliffordversion' displays the current version of the package:

```
> eCliffordversion();
```

$$\begin{aligned}
&++++++++++++++++++++++++++++++++++++++++++++++++++\\
&+\\
&\qquad eClifford - A\ Maple\ 17\ Package\ for\ Clifford\ Algebras\ Cl(p,q,r)
\end{aligned}$$

*Last revised: November 15, 2015 (Source file: eClifford_01_M17.mw)*



*(*) Department of Mathematics, Box 5054*
*Tennessee Technological University, Cookeville, TN 38505*
*tel: USA (931) 372-3622, fax: USA (931) 372-6353*
*rablamowicz@tntech.edu*
*http://math.tntech.edu/rafal/*
*(**) University of Konstanz*
*Fachbereich Physik, Fach M678*
*Universitätsstrasse 10, D-78457 Konstanz, Germany*
*Phone: +49 (0)7531 693491*
*E-mail: Bertfried.Fauser@uni-konstanz.de*
*http://clifford.physik.uni-konstanz.de/~fauser/*

*+++++This is eClifford ver. 01 for Maple 17+++++* **(2)**

To start quickly computations in Cl(p,q,r), define first the signature of Q via the global variable _Bsignature_pqr. The command 'ecbasis' displays a basis for Cl(p,q,r) used. Note that the identity element is displayed as e[] but later the alias epsilon can be used. That is, user can enter the identity element by typing the name 'epsilon'.

```
> p,q,r:=3,0,0;
  dim_V:=p+q+r;
  dim_Clpqr:=2^dim_V;
  _Bsignature_pqr:=[p,q,r];
  ecbas:=ecbasis(dim_V);
```

$$p, q, r := 3, 0, 0$$

$$dim\_V := 3$$

$$dim\_Clpqr := 8$$

$$\_Bsignature\_pqr := [3, 0, 0]$$

$$ecbas := [e_{[\,]}, e_1, e_2, e_3, e_{1,2}, e_{1,3}, e_{2,3}, e_{1,2,3}] \quad \textbf{(3)}$$

The Clifford product in Cl(p,q,r) is implemented via the procedure 'ecmul'. The following displays the matrix of all Clifford products of the basis monomials in Cl(p,q,r) for the above-defined signature (p,q,r).

```
> M300:=matrix(dim_Clpqr,dim_Clpqr,(i,j)->ecmul(ecbas[i],ecbas[j]))
  ;
```

$$M300 := \begin{bmatrix} \varepsilon & e_1 & e_2 & e_3 & e_{1,2} & e_{1,3} & e_{2,3} & e_{1,2,3} \\ e_1 & \varepsilon & e_{1,2} & e_{1,3} & e_2 & e_3 & e_{1,2,3} & e_{2,3} \\ e_2 & -e_{1,2} & \varepsilon & e_{2,3} & -e_1 & -e_{1,2,3} & e_3 & -e_{1,3} \\ e_3 & -e_{1,3} & -e_{2,3} & \varepsilon & e_{1,2,3} & -e_1 & -e_2 & e_{1,2} \\ e_{1,2} & -e_2 & e_1 & e_{1,2,3} & -\varepsilon & -e_{2,3} & e_{1,3} & -e_3 \\ e_{1,3} & -e_3 & -e_{1,2,3} & e_1 & e_{2,3} & -\varepsilon & -e_{1,2} & e_2 \\ e_{2,3} & e_{1,2,3} & -e_3 & e_2 & -e_{1,3} & e_{1,2} & -\varepsilon & -e_1 \\ e_{1,2,3} & e_{2,3} & -e_{1,3} & e_{1,2} & -e_3 & e_2 & -e_1 & -\varepsilon \end{bmatrix} \quad \textbf{(4)}$$

Note that in the above matrix the identity element is displayed as 'epsilon' whereas in the list of the basis monomials 'ecbas' above returned by 'ecbasis', the identity element is displayed as e[].

The wedge product in Cl(p,q,r) is implemented via the procedure 'ewedge'. The following displays the matrix of all wedge products of the basis monomials in Cl(p,q,r):

```
> M300wedge:=matrix(dim_Clpqr,dim_Clpqr,(i,j)->ewedge(ecbas[i],
  ecbas[j]));
```

$$M300wedge := \begin{bmatrix} \varepsilon & e_1 & e_2 & e_3 & e_{1,2} & e_{1,3} & e_{2,3} & e_{1,2,3} \\ e_1 & 0 & e_{1,2} & e_{1,3} & 0 & 0 & e_{1,2,3} & 0 \\ e_2 & -e_{1,2} & 0 & e_{2,3} & 0 & -e_{1,2,3} & 0 & 0 \\ e_3 & -e_{1,3} & -e_{2,3} & 0 & e_{1,2,3} & 0 & 0 & 0 \\ e_{1,2} & 0 & 0 & e_{1,2,3} & 0 & 0 & 0 & 0 \\ e_{1,3} & 0 & -e_{1,2,3} & 0 & 0 & 0 & 0 & 0 \\ e_{2,3} & e_{1,2,3} & 0 & 0 & 0 & 0 & 0 & 0 \\ e_{1,2,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \textbf{(5)}$$

In order to compute the Clifford product in another signature [p',q',r'], one has two options:

1. change the value of the global variable _Bsignature_pqr by assigning

[> _Bsignature_pqr:=[p',q',r'],

or,

2. call the procedure 'ecmul' with the new list [p',q',r'] used as an optional parameter. For example, the matrix of the Clifford products of the basis monomials in Cl(1,1,1) can be computed this way:

```
> M111:=matrix(dim_Clpqr,dim_Clpqr,(i,j)->ecmul[[1,1,1]](ecbas[i],
  ecbas[j]));
```

$$M111 := \begin{bmatrix} \varepsilon & e_1 & e_2 & e_3 & e_{1,2} & e_{1,3} & e_{2,3} & e_{1,2,3} \\ e_1 & \varepsilon & e_{1,2} & e_{1,3} & e_2 & e_3 & e_{1,2,3} & e_{2,3} \\ e_2 & -e_{1,2} & -\varepsilon & e_{2,3} & e_1 & -e_{1,2,3} & -e_3 & e_{1,3} \\ e_3 & -e_{1,3} & -e_{2,3} & 0 & e_{1,2,3} & 0 & 0 & 0 \\ e_{1,2} & -e_2 & -e_1 & e_{1,2,3} & \varepsilon & -e_{2,3} & -e_{1,3} & e_3 \\ e_{1,3} & -e_3 & -e_{1,2,3} & 0 & e_{2,3} & 0 & 0 & 0 \\ e_{2,3} & e_{1,2,3} & e_3 & 0 & e_{1,3} & 0 & 0 & 0 \\ e_{1,2,3} & e_{2,3} & e_{1,3} & 0 & e_3 & 0 & 0 & 0 \end{bmatrix} \quad \textbf{(6)}$$

Likewise, for Cl(0,2,1):

```
> M021:=matrix(dim_Clpqr,dim_Clpqr,(i,j)->ecmul[[0,2,1]](ecbas[i],
  ecbas[j]));
```

$$M021 := \begin{bmatrix} \varepsilon & e_1 & e_2 & e_3 & e_{1,2} & e_{1,3} & e_{2,3} & e_{1,2,3} \\ e_1 & -\varepsilon & e_{1,2} & e_{1,3} & -e_2 & -e_3 & e_{1,2,3} & -e_{2,3} \\ e_2 & -e_{1,2} & -\varepsilon & e_{2,3} & e_1 & -e_{1,2,3} & -e_3 & e_{1,3} \\ e_3 & -e_{1,3} & -e_{2,3} & 0 & e_{1,2,3} & 0 & 0 & 0 \\ e_{1,2} & e_2 & -e_1 & e_{1,2,3} & -\varepsilon & e_{2,3} & -e_{1,3} & -e_3 \\ e_{1,3} & e_3 & -e_{1,2,3} & 0 & -e_{2,3} & 0 & 0 & 0 \\ e_{2,3} & e_{1,2,3} & e_3 & 0 & e_{1,3} & 0 & 0 & 0 \\ e_{1,2,3} & -e_{2,3} & e_{1,3} & 0 & -e_3 & 0 & 0 & 0 \end{bmatrix} \quad \textbf{(7)}$$

Obviously, the matrix for Cl(0,0,3) is the same as M300wedge because Cl(0,0,3) = /\ V:

```
> M003:=matrix(dim_Clpqr,dim_Clpqr,(i,j)->ecmul[[0,0,3]](ecbas[i],
  ecbas[j]));
  'M300wedge'=evalm(M300wedge);
```

$$M003 := \begin{bmatrix} \varepsilon & e_1 & e_2 & e_3 & e_{1,2} & e_{1,3} & e_{2,3} & e_{1,2,3} \\ e_1 & 0 & e_{1,2} & e_{1,3} & 0 & 0 & e_{1,2,3} & 0 \\ e_2 & -e_{1,2} & 0 & e_{2,3} & 0 & -e_{1,2,3} & 0 & 0 \\ e_3 & -e_{1,3} & -e_{2,3} & 0 & e_{1,2,3} & 0 & 0 & 0 \\ e_{1,2} & 0 & 0 & e_{1,2,3} & 0 & 0 & 0 & 0 \\ e_{1,3} & 0 & -e_{1,2,3} & 0 & 0 & 0 & 0 & 0 \\ e_{2,3} & e_{1,2,3} & 0 & 0 & 0 & 0 & 0 & 0 \\ e_{1,2,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$M300wedge = \begin{bmatrix} \varepsilon & e_1 & e_2 & e_3 & e_{1,2} & e_{1,3} & e_{2,3} & e_{1,2,3} \\ e_1 & 0 & e_{1,2} & e_{1,3} & 0 & 0 & e_{1,2,3} & 0 \\ e_2 & -e_{1,2} & 0 & e_{2,3} & 0 & -e_{1,2,3} & 0 & 0 \\ e_3 & -e_{1,3} & -e_{2,3} & 0 & e_{1,2,3} & 0 & 0 & 0 \\ e_{1,2} & 0 & 0 & e_{1,2,3} & 0 & 0 & 0 & 0 \\ e_{1,3} & 0 & -e_{1,2,3} & 0 & 0 & 0 & 0 & 0 \\ e_{2,3} & e_{1,2,3} & 0 & 0 & 0 & 0 & 0 & 0 \\ e_{1,2,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \textbf{(8)}$$

The procedure 'ecbasis' can display a k-basis for /\^k V, a subspace of k-vectors in /\ V, or, for the even/odd part of the Clifford algebra as follows:

```
> ecbasis(4,0);
  ecbasis(4,1);
  ecbasis(4,2);
  ecbasis(4,3);
  ecbasis(4,4);
  ecbasis(3,'even');
  ecbasis(4,'odd');
```

$$[e_{[\,]}]$$

$$[e_1, e_2, e_3, e_4]$$

$$[e_{1,2}, e_{1,3}, e_{1,4}, e_{2,3}, e_{2,4}, e_{3,4}]$$

$$[e_{1,2,3}, e_{1,2,4}, e_{1,3,4}, e_{2,3,4}]$$

$$[e_{1,2,3,4}]$$

$$\left[e_{[\,]},\ e_{1,\,2},\ e_{1,\,3},\ e_{2,\,3}\right]$$

$$\left[e_1,\ e_2,\ e_3,\ e_4,\ e_{1,\,2,\,3},\ e_{1,\,2,\,4},\ e_{1,\,3,\,4},\ e_{2,\,3,\,4}\right] \tag{9}$$

In particular, let's illustrate computation of a product of two or three general elements in Cl(1,1,1):

```
> p1:=add(x[i]*ecbas[i],i=1..nops(ecbas));
  p2:=add(y[i]*ecbas[i],i=1..nops(ecbas));
  p3:=add(z[i]*ecbas[i],i=1..nops(ecbas));
```

$$p1 := e_1 x_2 + e_2 x_3 + e_3 x_4 + e_{[\,]} x_1 + e_{1,\,2} x_5 + e_{1,\,3} x_6 + e_{2,\,3} x_7 + e_{1,\,2,\,3} x_8$$

$$p2 := e_1 y_2 + e_2 y_3 + e_3 y_4 + e_{[\,]} y_1 + e_{1,\,2} y_5 + e_{1,\,3} y_6 + e_{2,\,3} y_7 + e_{1,\,2,\,3} y_8$$

$$p3 := e_1 z_2 + e_2 z_3 + e_3 z_4 + e_{[\,]} z_1 + e_{1,\,2} z_5 + e_{1,\,3} z_6 + e_{2,\,3} z_7 + e_{1,\,2,\,3} z_8 \tag{10}$$

```
> p1p2:=ecmul(p1,p2);
```

$$\begin{aligned} p1p2 := {} & \left(x_1 y_1 + x_2 y_2 + x_3 y_3 + x_4 y_4 - x_5 y_5 - x_6 y_6 - x_7 y_7 - x_8 y_8\right) \varepsilon + \left(x_1 y_2 + x_2 y_1 - x_3 y_5\right. \\ & \left. - x_4 y_6 + x_5 y_3 + x_6 y_4 - x_7 y_8 - x_8 y_7\right) e_1 + \left(x_1 y_3 + x_2 y_5 + x_3 y_1 - x_4 y_7 - x_5 y_2 + x_6 y_8\right. \\ & \left. + x_7 y_4 + x_8 y_6\right) e_2 + \left(x_1 y_4 + x_2 y_6 + x_3 y_7 + x_4 y_1 - x_5 y_8 - x_6 y_2 - x_7 y_3 - x_8 y_5\right) e_3 \\ & + \left(x_1 y_5 + x_2 y_3 - x_3 y_2 + x_4 y_8 + x_5 y_1 - x_6 y_7 + x_7 y_6 + x_8 y_4\right) e_{1,\,2} + \left(x_1 y_6 + x_2 y_4 - x_3 y_8\right. \\ & \left. - x_4 y_2 + x_5 y_7 + x_6 y_1 - x_7 y_5 - x_8 y_3\right) e_{1,\,3} + \left(x_1 y_7 + x_2 y_8 + x_3 y_4 - x_4 y_3 - x_5 y_6 + x_6 y_5\right. \\ & \left. + x_7 y_1 + x_8 y_2\right) e_{2,\,3} + \left(x_1 y_8 + x_2 y_7 - x_3 y_6 + x_4 y_5 + x_5 y_4 - x_6 y_3 + x_7 y_2 + x_8 y_1\right) e_{1,\,2,\,3} \end{aligned} \tag{11}$$

```
> p1p2p3:=ecmul(p1,p2,p3);
```

$$\begin{aligned} p1p2p3 := {} & \left(z_3 \left(x_1 y_3 + x_2 y_5 + x_3 y_1 - x_4 y_7 - x_5 y_2 + x_6 y_8 + x_7 y_4 + x_8 y_6\right) - z_6 \left(x_1 y_6 + x_2 y_4\right.\right. \\ & \left. - x_3 y_8 - x_4 y_2 + x_5 y_7 + x_6 y_1 - x_7 y_5 - x_8 y_3\right) + z_4 \left(x_1 y_4 + x_2 y_6 + x_3 y_7 + x_4 y_1 - x_5 y_8\right. \\ & \left. - x_6 y_2 - x_7 y_3 - x_8 y_5\right) - z_7 \left(x_1 y_7 + x_2 y_8 + x_3 y_4 - x_4 y_3 - x_5 y_6 + x_6 y_5 + x_7 y_1 + x_8 y_2\right) \\ & + z_1 \left(x_1 y_1 + x_2 y_2 + x_3 y_3 + x_4 y_4 - x_5 y_5 - x_6 y_6 - x_7 y_7 - x_8 y_8\right) + z_2 \left(x_1 y_2 + x_2 y_1 - x_3 y_5\right. \\ & \left. - x_4 y_6 + x_5 y_3 + x_6 y_4 - x_7 y_8 - x_8 y_7\right) - z_8 \left(x_1 y_8 + x_2 y_7 - x_3 y_6 + x_4 y_5 + x_5 y_4 - x_6 y_3\right. \\ & \left. + x_7 y_2 + x_8 y_1\right) - z_5 \left(x_1 y_5 + x_2 y_3 - x_3 y_2 + x_4 y_8 + x_5 y_1 - x_6 y_7 + x_7 y_6 + x_8 y_4\right)\Big) \varepsilon \\ & + \left(z_1 \left(x_1 y_2 + x_2 y_1 - x_3 y_5 - x_4 y_6 + x_5 y_3 + x_6 y_4 - x_7 y_8 - x_8 y_7\right) - z_5 \left(x_1 y_3 + x_2 y_5 + x_3 y_1\right.\right. \\ & \left. - x_4 y_7 - x_5 y_2 + x_6 y_8 + x_7 y_4 + x_8 y_6\right) + z_3 \left(x_1 y_5 + x_2 y_3 - x_3 y_2 + x_4 y_8 + x_5 y_1 - x_6 y_7\right. \\ & \left. + x_7 y_6 + x_8 y_4\right) - z_6 \left(x_1 y_4 + x_2 y_6 + x_3 y_7 + x_4 y_1 - x_5 y_8 - x_6 y_2 - x_7 y_3 - x_8 y_5\right) - z_7 \left(x_1 y_8\right. \\ & \left. + x_2 y_7 - x_3 y_6 + x_4 y_5 + x_5 y_4 - x_6 y_3 + x_7 y_2 + x_8 y_1\right) - z_8 \left(x_1 y_7 + x_2 y_8 + x_3 y_4 - x_4 y_3\right. \\ & \left. - x_5 y_6 + x_6 y_5 + x_7 y_1 + x_8 y_2\right) + z_4 \left(x_1 y_6 + x_2 y_4 - x_3 y_8 - x_4 y_2 + x_5 y_7 + x_6 y_1 - x_7 y_5\right. \\ & \left. - x_8 y_3\right) + z_2 \left(x_1 y_1 + x_2 y_2 + x_3 y_3 + x_4 y_4 - x_5 y_5 - x_6 y_6 - x_7 y_7 - x_8 y_8\right)\Big) e_1 + \left(z_1 \left(x_1 y_3\right.\right. \\ & \left. + x_2 y_5 + x_3 y_1 - x_4 y_7 - x_5 y_2 + x_6 y_8 + x_7 y_4 + x_8 y_6\right) + z_4 \left(x_1 y_7 + x_2 y_8 + x_3 y_4 - x_4 y_3\right. \end{aligned} \tag{12}$$

$$-x_5 y_6 + x_6 y_5 + x_7 y_1 + x_8 y_2) + z_3 (x_1 y_1 + x_2 y_2 + x_3 y_3 + x_4 y_4 - x_5 y_5 - x_6 y_6 - x_7 y_7$$
$$- x_8 y_8) + z_6 (x_1 y_8 + x_2 y_7 - x_3 y_6 + x_4 y_5 + x_5 y_4 - x_6 y_3 + x_7 y_2 + x_8 y_1) - z_2 (x_1 y_5 + x_2 y_3$$
$$- x_3 y_2 + x_4 y_8 + x_5 y_1 - x_6 y_7 + x_7 y_6 + x_8 y_4) + z_8 (x_1 y_6 + x_2 y_4 - x_3 y_8 - x_4 y_2 + x_5 y_7$$
$$+ x_6 y_1 - x_7 y_5 - x_8 y_3) + z_5 (x_1 y_2 + x_2 y_1 - x_3 y_5 - x_4 y_6 + x_5 y_3 + x_6 y_4 - x_7 y_8 - x_8 y_7)$$
$$- z_7 (x_1 y_4 + x_2 y_6 + x_3 y_7 + x_4 y_1 - x_5 y_8 - x_6 y_2 - x_7 y_3 - x_8 y_5)) e_2 + (-z_8 (x_1 y_5 + x_2 y_3$$
$$- x_3 y_2 + x_4 y_8 + x_5 y_1 - x_6 y_7 + x_7 y_6 + x_8 y_4) + z_6 (x_1 y_2 + x_2 y_1 - x_3 y_5 - x_4 y_6 + x_5 y_3$$
$$+ x_6 y_4 - x_7 y_8 - x_8 y_7) + z_4 (x_1 y_1 + x_2 y_2 + x_3 y_3 + x_4 y_4 - x_5 y_5 - x_6 y_6 - x_7 y_7 - x_8 y_8)$$
$$+ z_7 (x_1 y_3 + x_2 y_5 + x_3 y_1 - x_4 y_7 - x_5 y_2 + x_6 y_8 + x_7 y_4 + x_8 y_6) - z_5 (x_1 y_8 + x_2 y_7 - x_3 y_6$$
$$+ x_4 y_5 + x_5 y_4 - x_6 y_3 + x_7 y_2 + x_8 y_1) - z_3 (x_1 y_7 + x_2 y_8 + x_3 y_4 - x_4 y_3 - x_5 y_6 + x_6 y_5$$
$$+ x_7 y_1 + x_8 y_2) + z_1 (x_1 y_4 + x_2 y_6 + x_3 y_7 + x_4 y_1 - x_5 y_8 - x_6 y_2 - x_7 y_3 - x_8 y_5) - z_2 (x_1 y_6$$
$$+ x_2 y_4 - x_3 y_8 - x_4 y_2 + x_5 y_7 + x_6 y_1 - x_7 y_5 - x_8 y_3)) e_3 + (z_5 (x_1 y_1 + x_2 y_2 + x_3 y_3$$
$$+ x_4 y_4 - x_5 y_5 - x_6 y_6 - x_7 y_7 - x_8 y_8) + z_6 (x_1 y_7 + x_2 y_8 + x_3 y_4 - x_4 y_3 - x_5 y_6 + x_6 y_5$$
$$+ x_7 y_1 + x_8 y_2) + z_3 (x_1 y_2 + x_2 y_1 - x_3 y_5 - x_4 y_6 + x_5 y_3 + x_6 y_4 - x_7 y_8 - x_8 y_7) + z_1 (x_1 y_5$$
$$+ x_2 y_3 - x_3 y_2 + x_4 y_8 + x_5 y_1 - x_6 y_7 + x_7 y_6 + x_8 y_4) + z_4 (x_1 y_8 + x_2 y_7 - x_3 y_6 + x_4 y_5$$
$$+ x_5 y_4 - x_6 y_3 + x_7 y_2 + x_8 y_1) - z_7 (x_1 y_6 + x_2 y_4 - x_3 y_8 - x_4 y_2 + x_5 y_7 + x_6 y_1 - x_7 y_5$$
$$- x_8 y_3) - z_2 (x_1 y_3 + x_2 y_5 + x_3 y_1 - x_4 y_7 - x_5 y_2 + x_6 y_8 + x_7 y_4 + x_8 y_6) + z_8 (x_1 y_4 + x_2 y_6$$
$$+ x_3 y_7 + x_4 y_1 - x_5 y_8 - x_6 y_2 - x_7 y_3 - x_8 y_5)) e_{1,2} + (-z_2 (x_1 y_4 + x_2 y_6 + x_3 y_7 + x_4 y_1$$
$$- x_5 y_8 - x_6 y_2 - x_7 y_3 - x_8 y_5) + z_6 (x_1 y_1 + x_2 y_2 + x_3 y_3 + x_4 y_4 - x_5 y_5 - x_6 y_6 - x_7 y_7$$
$$- x_8 y_8) + z_1 (x_1 y_6 + x_2 y_4 - x_3 y_8 - x_4 y_2 + x_5 y_7 + x_6 y_1 - x_7 y_5 - x_8 y_3) - z_5 (x_1 y_7 + x_2 y_8$$
$$+ x_3 y_4 - x_4 y_3 - x_5 y_6 + x_6 y_5 + x_7 y_1 + x_8 y_2) - z_3 (x_1 y_8 + x_2 y_7 - x_3 y_6 + x_4 y_5 + x_5 y_4$$
$$- x_6 y_3 + x_7 y_2 + x_8 y_1) + z_7 (x_1 y_5 + x_2 y_3 - x_3 y_2 + x_4 y_8 + x_5 y_1 - x_6 y_7 + x_7 y_6 + x_8 y_4)$$
$$+ z_4 (x_1 y_2 + x_2 y_1 - x_3 y_5 - x_4 y_6 + x_5 y_3 + x_6 y_4 - x_7 y_8 - x_8 y_7) - z_8 (x_1 y_3 + x_2 y_5 + x_3 y_1$$
$$- x_4 y_7 - x_5 y_2 + x_6 y_8 + x_7 y_4 + x_8 y_6)) e_{1,3} + (z_2 (x_1 y_8 + x_2 y_7 - x_3 y_6 + x_4 y_5 + x_5 y_4$$
$$- x_6 y_3 + x_7 y_2 + x_8 y_1) + z_1 (x_1 y_7 + x_2 y_8 + x_3 y_4 - x_4 y_3 - x_5 y_6 + x_6 y_5 + x_7 y_1 + x_8 y_2)$$
$$- z_6 (x_1 y_5 + x_2 y_3 - x_3 y_2 + x_4 y_8 + x_5 y_1 - x_6 y_7 + x_7 y_6 + x_8 y_4) + z_8 (x_1 y_2 + x_2 y_1 - x_3 y_5$$
$$- x_4 y_6 + x_5 y_3 + x_6 y_4 - x_7 y_8 - x_8 y_7) + z_5 (x_1 y_6 + x_2 y_4 - x_3 y_8 - x_4 y_2 + x_5 y_7 + x_6 y_1$$
$$- x_7 y_5 - x_8 y_3) + z_4 (x_1 y_3 + x_2 y_5 + x_3 y_1 - x_4 y_7 - x_5 y_2 + x_6 y_8 + x_7 y_4 + x_8 y_6) + z_7 (x_1 y_1$$
$$+ x_2 y_2 + x_3 y_3 + x_4 y_4 - x_5 y_5 - x_6 y_6 - x_7 y_7 - x_8 y_8) - z_3 (x_1 y_4 + x_2 y_6 + x_3 y_7 + x_4 y_1$$
$$- x_5 y_8 - x_6 y_2 - x_7 y_3 - x_8 y_5)) e_{2,3} + (-z_3 (x_1 y_6 + x_2 y_4 - x_3 y_8 - x_4 y_2 + x_5 y_7 + x_6 y_1$$
$$- x_7 y_5 - x_8 y_3) + z_7 (x_1 y_2 + x_2 y_1 - x_3 y_5 - x_4 y_6 + x_5 y_3 + x_6 y_4 - x_7 y_8 - x_8 y_7) + z_1 (x_1 y_8$$

$$+x_2 y_7 - x_3 y_6 + x_4 y_5 + x_5 y_4 - x_6 y_3 + x_7 y_2 + x_8 y_1) + z_2 (x_1 y_7 + x_2 y_8 + x_3 y_4 - x_4 y_3$$
$$- x_5 y_6 + x_6 y_5 + x_7 y_1 + x_8 y_2) - z_6 (x_1 y_3 + x_2 y_5 + x_3 y_1 - x_4 y_7 - x_5 y_2 + x_6 y_8 + x_7 y_4$$
$$+ x_8 y_6) + z_4 (x_1 y_5 + x_2 y_3 - x_3 y_2 + x_4 y_8 + x_5 y_1 - x_6 y_7 + x_7 y_6 + x_8 y_4) + z_8 (x_1 y_1 + x_2 y_2$$
$$+ x_3 y_3 + x_4 y_4 - x_5 y_5 - x_6 y_6 - x_7 y_7 - x_8 y_8) + z_5 (x_1 y_4 + x_2 y_6 + x_3 y_7 + x_4 y_1 - x_5 y_8$$
$$- x_6 y_2 - x_7 y_3 - x_8 y_5)) e_{1,2,3}$$

The above shows that 'ecmul' returns answer collected and simplified, and that it can accept more than two arguments: then, ecmul(args[1],args[2],...,args[n]) is computed recursively as ecmul(ecmul(args[1], args[2]),args[3]...,args[n]), etc.Similarly for 'ewedge':

```
> p1p2w:=ewedge(p1,p2);
```

$$p1p2w := x_1 y_1 \varepsilon + (x_1 y_2 + x_2 y_1) e_1 + (x_1 y_3 + x_3 y_1) e_2 + (x_1 y_4 + x_4 y_1) e_3 + (x_1 y_5 + x_2 y_3 \quad \textbf{(13)}$$
$$- x_3 y_2 + x_5 y_1) e_{1,2} + (x_1 y_6 + x_2 y_4 - x_4 y_2 + x_6 y_1) e_{1,3} + (x_1 y_7 + x_3 y_4 - x_4 y_3$$
$$+ x_7 y_1) e_{2,3} + (x_1 y_8 + x_2 y_7 - x_3 y_6 + x_4 y_5 + x_5 y_4 - x_6 y_3 + x_7 y_2 + x_8 y_1) e_{1,2,3}$$

```
> p1p2p3w:=ewedge(p1,p2,p3);
```

$$p1p2p3w := x_1 y_1 z_1 \varepsilon + (x_1 y_1 z_2 + z_1 (x_1 y_2 + x_2 y_1)) e_1 + (x_1 y_1 z_3 + z_1 (x_1 y_3 + x_3 y_1)) e_2 \quad \textbf{(14)}$$
$$+ (z_1 (x_1 y_4 + x_4 y_1) + x_1 y_1 z_4) e_3 + (z_1 (x_1 y_5 + x_2 y_3 - x_3 y_2 + x_5 y_1) + z_3 (x_1 y_2 + x_2 y_1)$$
$$- z_2 (x_1 y_3 + x_3 y_1) + x_1 y_1 z_5) e_{1,2} + (z_4 (x_1 y_2 + x_2 y_1) + x_1 y_1 z_6 + z_1 (x_1 y_6 + x_2 y_4$$
$$- x_4 y_2 + x_6 y_1) - z_2 (x_1 y_4 + x_4 y_1)) e_{1,3} + (-z_3 (x_1 y_4 + x_4 y_1) + x_1 y_1 z_7 + z_1 (x_1 y_7$$
$$+ x_3 y_4 - x_4 y_3 + x_7 y_1) + z_4 (x_1 y_3 + x_3 y_1)) e_{2,3} + (-z_3 (x_1 y_6 + x_2 y_4 - x_4 y_2 + x_6 y_1)$$
$$+ z_7 (x_1 y_2 + x_2 y_1) + z_1 (x_1 y_8 + x_2 y_7 - x_3 y_6 + x_4 y_5 + x_5 y_4 - x_6 y_3 + x_7 y_2 + x_8 y_1)$$
$$+ z_4 (x_1 y_5 + x_2 y_3 - x_3 y_2 + x_5 y_1) + z_2 (x_1 y_7 + x_3 y_4 - x_4 y_3 + x_7 y_1) + x_1 y_1 z_8 + z_5 (x_1 y_4$$
$$+ x_4 y_1) - z_6 (x_1 y_3 + x_3 y_1)) e_{1,2,3}$$

Let us illustrate computations in dimensions higher than 9, say in Cl(8,2):

```
> p,q,r:=8,2,0;
  _Bsignature_pqr:=[p,q,r];
  dim_Clpqr:=2^(p+q+r);
```

$$p, q, r := 8, 2, 0$$
$$\_Bsignature\_pqr := [8, 2, 0]$$
$$dim\_Clpqr := 1024 \quad \textbf{(15)}$$

Obviously, we won't display a basis of 1,024 elements, but let us define two elements in Cl(8,2) = Cl(8, 2,0):

```
> p1:=epsilon+a*e[1]+e[4]+e[8]+b*e[6,10];
  p2:=epsilon+b*e[1]+e[8]+c*e[1,2,10];
```

$$p1 := a\,e_1 + b\,e_{6,\,10} + \varepsilon + e_4 + e_8$$

$$p2 := b\,e_1 + c\,e_{1,\,2,\,10} + \varepsilon + e_8 \qquad (16)$$

```
> ecmul(p1,p2);
  ewedge(p1,p2);
```

$$(a\,b+2)\,\varepsilon + (a+b)\,e_1 + e_4 + 2\,e_8 - b\,e_{1,\,4} + (a-b)\,e_{1,\,8} + a\,c\,e_{2,\,10} + b^2\,e_{1,\,6,\,10}$$
$$- b\,c\,e_{1,\,2,\,6} + b\,e_{6,\,10} - b\,e_{6,\,8,\,10} + c\,e_{1,\,2,\,10} + c\,e_{1,\,2,\,4,\,10} + c\,e_{1,\,2,\,8,\,10} + e_{4,\,8}$$

$$\varepsilon + (a+b)\,e_1 + e_4 + 2\,e_8 - b\,e_{1,\,4} + (a-b)\,e_{1,\,8} + b^2\,e_{1,\,6,\,10} + b\,e_{6,\,10} - b\,e_{6,\,8,\,10} + c\,e_{1,\,2,\,10} \qquad (17)$$
$$+ c\,e_{1,\,2,\,4,\,10} + c\,e_{1,\,2,\,8,\,10} + e_{4,\,8}$$

In the above, coefficients 'b' and 'c' are treated as Maple symbols.

We can change the signature and compute the Clifford product say in Cl(4,4,2) and Cl(0,1,9) using the same elements p1 and p2: by usign signature [p,q,r] as a parameter in 'ecmul':

```
> ecmul[[4,4,2]](p1,p2);
  ecmul[[0,1,9]](p1,p2);
```

$$a\,b\,\varepsilon + (a+b)\,e_1 + e_4 + 2\,e_8 - b\,e_{1,\,4} + (a-b)\,e_{1,\,8} + a\,c\,e_{2,\,10} + b^2\,e_{1,\,6,\,10} + b\,e_{6,\,10}$$
$$- b\,e_{6,\,8,\,10} + c\,e_{1,\,2,\,10} + c\,e_{1,\,2,\,4,\,10} + c\,e_{1,\,2,\,8,\,10} + e_{4,\,8}$$

$$(-a\,b+1)\,\varepsilon + (a+b)\,e_1 + e_4 + 2\,e_8 - b\,e_{1,\,4} + (a-b)\,e_{1,\,8} - a\,c\,e_{2,\,10} + b^2\,e_{1,\,6,\,10} + b\,e_{6,\,10} \qquad (18)$$
$$- b\,e_{6,\,8,\,10} + c\,e_{1,\,2,\,10} + c\,e_{1,\,2,\,4,\,10} + c\,e_{1,\,2,\,8,\,10} + e_{4,\,8}$$

```
> ecmul[[4,4,2]](p1,p1);
  ecmul[[0,1,9]](p1,p1);
```

$$(a^2+1)\,\varepsilon + 2\,b\,a\,e_{1,\,6,\,10} + 2\,a\,e_1 + 2\,b\,e_{6,\,10} + 2\,b\,e_{4,\,6,\,10} - 2\,b\,e_{6,\,8,\,10} + 2\,e_4 + 2\,e_8$$

$$(-a^2+1)\,\varepsilon + 2\,b\,a\,e_{1,\,6,\,10} + 2\,a\,e_1 + 2\,b\,e_{6,\,10} + 2\,b\,e_{4,\,6,\,10} - 2\,b\,e_{6,\,8,\,10} + 2\,e_4 + 2\,e_8 \qquad (19)$$

Here is another example of computation in Cl(8,3):

```
> p1:=epsilon+e[1]+e[8]+b*e[6,10]+7*e[2,3,10,11];
  p2:=epsilon+e[1]+e[8]+c*e[1,2,10];
```

$$p1 := b\,e_{6,\,10} + \varepsilon + e_1 + e_8 + 7\,e_{2,\,3,\,10,\,11}$$

$$p2 := c\,e_{1,\,2,\,10} + \varepsilon + e_1 + e_8 \qquad (20)$$

```
> ecmul[[8,3,0]](p1,p2);
```

$$-b\,c\,e_{1,\,2,\,6} + b\,e_{6,\,10} + b\,e_{1,\,6,\,10} - b\,e_{6,\,8,\,10} + c\,e_{2,\,10} + c\,e_{1,\,2,\,10} - 7\,c\,e_{1,\,3,\,11} + c\,e_{1,\,2,\,8,\,10} \qquad (21)$$

$$+3\,\varepsilon+2\,e_1+2\,e_8+7\,e_{2,3,10,11}+7\,e_{1,2,3,10,11}+7\,e_{2,3,8,10,11}$$

```
> ewedge(p1,p2);
```

$$b\,e_{6,10}+b\,e_{1,6,10}-b\,e_{6,8,10}+c\,e_{1,2,10}+c\,e_{1,2,8,10}+\varepsilon+2\,e_1+2\,e_8+7\,e_{2,3,10,11}+7\,e_{1,2,3,10,11}+7\,e_{2,3,8,10,11} \quad (22)$$

If an index encountered in the input is larger than the dimension of the current vector space V equal to p+q+r, then 'ecmul' returns an error:

```
> _Bsignature_pqr;
```

$$[8,2,0] \quad (23)$$

```
> p0:=epsilon+e[10,11];
```

$$p0 := \varepsilon + e_{10,11} \quad (24)$$

```
> ecmul(p0,p0);
Error, (in eClifford:-cmulWalsh3pqr) maximum index 11 found in
the arguments e[10, 11] and e[] is larger then dim_V = 10
derived from the last argument [8, 2, 0]
> ecmul[[1,2,3]](p1,p2);
Error, (in eClifford:-cmulWalsh3pqr) maximum index 8 found in
the arguments e[8] and e[] is larger then dim_V = 6 derived from
the last argument [1, 2, 3]
```

**Section 2: All remaining procedures in eClifford**

Let us show how the remaining procedures work:

Procedures 'ereversion', 'egradeinv', 'econjugation' apply reversion, grade involution, and Clifford conjugation to an element of Cl(p,q,r);

```
> p1:=7*epsilon+2*e[1]-e[2]+3*e[10]+2*e[1,2,3]-10*e[1,7,10]+e[1,3,
  6,8,9,10]+a*e[1,2,3];
  ereversion(p1);
  egradeinv(p1);
  econjugation(p1);
```

$$p1 := a\,e_{1,2,3}+7\,\varepsilon+2\,e_1-e_2+3\,e_{10}+2\,e_{1,2,3}-10\,e_{1,7,10}+e_{1,3,6,8,9,10}$$

$$7\,\varepsilon+3\,e_{10}-e_2+2\,e_1+10\,e_{1,7,10}+(-a-2)\,e_{1,2,3}-e_{1,3,6,8,9,10}$$

$$7\,\varepsilon-3\,e_{10}+e_2-2\,e_1+10\,e_{1,7,10}+(-a-2)\,e_{1,2,3}+e_{1,3,6,8,9,10}$$

$$7\,\varepsilon-3\,e_{10}+e_2-2\,e_1-10\,e_{1,7,10}+(a+2)\,e_{1,2,3}-e_{1,3,6,8,9,10} \quad (25)$$

Procedure 'evectorpart' returns the k-vector part of an element of Cl(p,q,r) while the procedure 'ecliscalarpart' returns the coefficient of the 0-vector part. Procedure 'emaxgrade' returns tha maximum

grade in the input while the procedure 'emaxindex' returns the maximum index:

```
> 'p1'=p1;
```

$$p1 = a\, e_{1,2,3} + 7\,\varepsilon + 2\, e_1 - e_2 + 3\, e_{10} + 2\, e_{1,2,3} - 10\, e_{1,7,10} + e_{1,3,6,8,9,10} \tag{26}$$

```
> emaxgrade(p1);
  emaxindex(p1);
```

$$6$$
$$10 \tag{27}$$

```
> for k from 0 to emaxgrade(p1) do
      evectorpart(p1,k); ##<<-- return the k-vector part of p1
  end do;
```

$$7\,\varepsilon$$
$$3\, e_{10} - e_2 + 2\, e_1$$
$$0$$
$$-10\, e_{1,7,10} + (a+2)\, e_{1,2,3}$$
$$0$$
$$0$$
$$e_{1,3,6,8,9,10} \tag{28}$$

```
> ecliscalarpart(p1);
```

$$7 \tag{29}$$

Procedure 'ecliterms' returns a list of monomial terms in its input:

```
> 'p1'=p1;
```

$$p1 = a\, e_{1,2,3} + 7\,\varepsilon + 2\, e_1 - e_2 + 3\, e_{10} + 2\, e_{1,2,3} - 10\, e_{1,7,10} + e_{1,3,6,8,9,10} \tag{30}$$

```
> ecliterms(p1);
```

$$[\varepsilon, e_1, e_2, e_{10}, e_{1,2,3}, e_{1,7,10}, e_{1,3,6,8,9,10}] \tag{31}$$

Procedure 'eclicollect' collects terms in its input:

```
> 'p1'=p1;
```

$$p1 = a\, e_{1,2,3} + 7\,\varepsilon + 2\, e_1 - e_2 + 3\, e_{10} + 2\, e_{1,2,3} - 10\, e_{1,7,10} + e_{1,3,6,8,9,10} \tag{32}$$

```
> p2:=8*epsilon+2*e[1,2]-13*e[10]+12*e[1,2,3]-3*e[1,3,6,8,9,10]-b*e
  [1,7,10]+c*epsilon+t*e[1,2,3];
```

$$p2 := -b\, e_{1,7,10} + c\,\varepsilon + t\, e_{1,2,3} + 8\,\varepsilon - 13\, e_{10} + 2\, e_{1,2} + 12\, e_{1,2,3} - 3\, e_{1,3,6,8,9,10} \tag{33}$$

The sum p1 + p2 is not quite collected by Maple so a procedure 'eclicollect' is needed. Note that Maple automatically collects terms if their coefficients are numeric, but not when they are symbolic or mixed:

```
> p1+p2; ##<<-- some terms remain uncollected, for example, -10*e
  [1,7,10] and -b*e[1,7,10]
```

$$a\,e_{1,2,3} - b\,e_{1,7,10} + c\,\varepsilon + t\,e_{1,2,3} + 15\,\varepsilon + 2\,e_1 - e_2 - 10\,e_{10} + 2\,e_{1,2} + 14\,e_{1,2,3} - 10\,e_{1,7,10} - 2\,e_{1,3,6,8,9,10} \quad (34)$$

```
> eclicollect(p1+p2); ##<<-- all terms are collected
```

$$(c+15)\,\varepsilon + 2\,e_1 - e_2 - 10\,e_{10} + 2\,e_{1,2} + (a+t+14)\,e_{1,2,3} + (-b-10)\,e_{1,7,10} - 2\,e_{1,3,6,8,9,10} \quad (35)$$

The remaining procedures are more for internal use:

Procedure 'esplit' splits the given polynomial into two lists: the first list contains scalar coefficients and the second list contains monomials:

```
> p11:=p1+7*e[]+aa*e[]-ss*e[10];
```

$$p11 := a\,e_{1,2,3} + aa\,\varepsilon - ss\,e_{10} + 14\,\varepsilon + 2\,e_1 - e_2 + 3\,e_{10} + 2\,e_{1,2,3} - 10\,e_{1,7,10} + e_{1,3,6,8,9,10} \quad (36)$$

```
> esplit(p11);
```

$$[aa+14,\ -ss+3,\ -1,\ 2,\ -10,\ a+2,\ 1],\ [\varepsilon,\ e_{10},\ e_2,\ e_1,\ e_{1,7,10},\ e_{1,2,3},\ e_{1,3,6,8,9,10}] \quad (37)$$

Procedure 'eextract' extracts the list of indices from a monomial term:

```
> ecbas:=ecbasis(3);
```

$$ecbas := [e_{[\,]},\ e_1,\ e_2,\ e_3,\ e_{1,2},\ e_{1,3},\ e_{2,3},\ e_{1,2,3}] \quad (38)$$

```
> map(eextract,ecbas);
```

$$[[\,],\ [1],\ [2],\ [3],\ [1,2],\ [1,3],\ [2,3],\ [1,2,3]] \quad (39)$$

```
> eextract(p1); ##<<-- eextract works only on monomials
Error, invalid input: eClifford:-eextract expects its 1st
argument, x, to be of type eclibasmon, but received a*e[1, 2, 3]
+7*e[]+2*e[1]-e[2]+3*e[10]+2*e[1, 2, 3]-10*e[1, 7, 10]+e[1, 3,
6, 8, 9, 10]
```

Procedure 'egrade' returns the grade of a monomial term:

```
> map(egrade,ecbas);
```

$$[0,\ 1,\ 1,\ 1,\ 2,\ 2,\ 2,\ 3] \quad (40)$$

```
> egrade(p1); ##<<-- egrade works only on monomials
Error, invalid input: eClifford:-egrade expects its 1st
argument, a, to be of type eclibasmon, but received a*e[1, 2, 3]
```

```
+7*e[]+2*e[1]-e[2]+3*e[10]+2*e[1, 2, 3]-10*e[1, 7, 10]+e[1, 3,
6, 8, 9, 10]
```

Procedure 'split' is actually meant to split expressions in Cl(p,q,r) when they are expressed in the 'old' string basis. Otherwise it works like 'esplit'. This procedure is used internally by the conversion procedure `convert/clibas_to_eclibas' descibed below. It requires that the lackage Clifford/Bigebra be loaded in.

```
> pp:=2*Id-3*e1we2we3+10*e1we5we8;
```

$$pp := 2\,Id - 3\,e1we2we3 + 10\,e1we5we8 \tag{41}$$

```
> split(pp);  ##<<-- package Clifford/Bigebra needs to be loaded in
Error, type `climon` does not exist
```

Finally, we describe the three basic types in eClifford:

(a) `type/eclibasmon` -- these are the Grassmann basis monomials;
(b) `type/eclimon` -- these are ordinary monomials which are products a*m where a is one of the types stored in _scalartypes and m is of type 'elibasmon';
(c) `type/eclipolynom` -- this is the most general element in Cl(p,q,r) which is a linear combination of the basis monomials, i.e., it is a sum of monomial terms of type 'eclimon'

```
> type(e[1,2,3],eclibasmon);
  type(e[1,2,3],eclimon);
  type(e[1,2,3],eclipolynom);
```

$$true$$
$$false$$
$$false \tag{42}$$

```
> type(7*e[1,2,3],eclibasmon);
  type(7*e[1,2,3],eclimon);
  type(7*e[1,2,3],eclipolynom);
```

$$false$$
$$true$$
$$false \tag{43}$$

```
> 'p1'=p1;
  type(p1,eclibasmon);
  type(p1,eclimon);
  type(p1,eclipolynom);
```

$$p1 = a\,e_{1,\,2,\,3} + 7\,\varepsilon + 2\,e_1 - e_2 + 3\,e_{10} + 2\,e_{1,\,2,\,3} - 10\,e_{1,\,7,\,10} + e_{1,\,3,\,6,\,8,\,9,\,10}$$
$$false$$
$$false$$
$$true \tag{44}$$

There are three sorting procedures that sort a list of monomials of type 'eclibasmon'
:
(a) `sort/byegrade` -- sorts a list of monomials of type 'eclibasmon' by the k-grade
(b) `sort/byegradenum` -- sorts a list of monomials of type 'eclibasmon by the k-grade and within the grade lexicographically assuming that all indices are numeric
(c) `sort/byegradesymb` --- sorts a list of monomials of type 'eclibasmon by the k-grade and within the grade lexicographically assuming that all indices are symbolic

```
> ecbas:=ecbasis(4);  ##<<-- this list is sorted by the k-grade
```

$$ecbas := [e_{[\,]}, e_1, e_2, e_3, e_4, e_{1,2}, e_{1,3}, e_{1,4}, e_{2,3}, e_{2,4}, e_{3,4}, e_{1,2,3}, e_{1,2,4}, e_{1,3,4}, e_{2,3,4}, e_{1,2,3,4}] \quad (45)$$

```
> L:=convert(convert([seq(ecbas[-k],k=1..nops(ecbas))],set),list);
  ##<<-- unsorted list
```

$$L := [e_1, e_2, e_3, e_4, e_{[\,]}, e_{1,2}, e_{1,3}, e_{1,4}, e_{2,3}, e_{2,4}, e_{3,4}, e_{1,2,3}, e_{1,2,4}, e_{1,3,4}, e_{2,3,4}, e_{1,2,3,4}] \quad (46)$$

```
> sort(L,byegrade); ##<<-- sort list L by grade but not within each
  grade
```

$$[e_{[\,]}, e_4, e_3, e_2, e_1, e_{3,4}, e_{2,4}, e_{2,3}, e_{1,4}, e_{1,3}, e_{1,2}, e_{2,3,4}, e_{1,3,4}, e_{1,2,4}, e_{1,2,3}, e_{1,2,3,4}] \quad (47)$$

```
> sort(L,byegradenum);  ##<<-- sort list L by grade and within each
  grade
```

$$[e_{[\,]}, e_1, e_2, e_3, e_4, e_{1,2}, e_{1,3}, e_{1,4}, e_{2,3}, e_{2,4}, e_{3,4}, e_{1,2,3}, e_{1,2,4}, e_{1,3,4}, e_{2,3,4}, e_{1,2,3,4}] \quad (48)$$

```
> sort(L,byegradesymb);
Error, (in eClifford:-byegradesymb) byegradesymb can only
compare eclibasmons with all distinct symbolic indices
```

```
> i:='i';j:='j':k:='k':l:='l': ##<<-- unassigning values of i,j,k,l
  to make them symbols
  L:=[e[j,k,l],e[j,k],e[],e[i,j]];
```

$$i := i$$

$$L := [e_{j,k,l}, e_{j,k}, \varepsilon, e_{i,j}] \quad (49)$$

```
> sort(L,byegradesymb);
```

$$[\varepsilon, e_{i,j}, e_{j,k}, e_{j,k,l}] \quad (50)$$

Finally, we have two conversions that convert monomials of the type 'eclibasmon' to binary tuples, needed in Walsh functions, and back:

(a) `convert/eclibasmon_to_binarytuple` -- converts a basis monomial of type 'eclibasmon' to a binary tuple of type 'binarytuple' in Cl(p,q,r) where dim_V = n = p+q+r. This procedure needs an extra argument dim_V to know how long the binary tuple must be;
(b) `convert/binarytuple_to_eclibasmon` -- converts a binary tuple of type 'binarytuple' to a basis monomial of type 'eclibasmon'
.

```
> dim_V:=3:
  ecbas:=ecbasis(dim_V);
```

$$ecbas := [e_{[\,]}, e_1, e_2, e_3, e_{1,2}, e_{1,3}, e_{2,3}, e_{1,2,3}] \quad (51)$$

```
> L:=map(convert,ecbas,eclibasmon_to_binarytuple,dim_V); ##<<--
  binary tuples representing elements in ecbas
```

$$L := [[0, 0, 0], [1, 0, 0], [0, 1, 0], [0, 0, 1], [1, 1, 0], [1, 0, 1], [0, 1, 1], [1, 1, 1]] \quad (52)$$

```
> map(convert,L,binarytuple_to_eclibasmon);  ##<<-- conversion back
  to the basis monomials
```

$$[\varepsilon, e_1, e_2, e_3, e_{1,2}, e_{1,3}, e_{2,3}, e_{1,2,3}] \quad (53)$$

**Section 3: Simultaneous computation in eClifford and Clifford/Bigebra**

In this section we show how to compute simultaneously with eClifford and Clifford/Bigebra packages, and convert from one data type to another. We will need these two conversion procedures:

(a) `convert/eclibas_to_clibas` -- converts any element of Cl(p,q,r) expressed in the 'new' indexed basis used in eClifford to the 'old' string basis used in Cliffor/Bigebra
(b) `convert/clibas_to_eclibas` -- converts any element of Cl(p,q,r) expressed in the 'old' string basis used in Cliffor/Bigebra to the 'new' indexed basis used in eClifford

```
> with(Clifford):with(linalg):
> ecbas:=ecbasis(3); ##<<-- basis in the new index format
```

$$ecbas := [e_{[\,]}, e_1, e_2, e_3, e_{1,2}, e_{1,3}, e_{2,3}, e_{1,2,3}] \quad (54)$$

```
> cbas:=map(convert,ecbas,eclibas_to_clibas); ##<<-- basis in the
  old string format
```

$$cbas := [Id, e1, e2, e3, e1we2, e1we3, e2we3, e1we2we3] \quad (55)$$

```
> map(convert,cbas,clibas_to_eclibas); ##<<-- basis in the new
  indexed format
Cliplus ver. 17.1 of 9/18/2015 has been loaded. Definitions for type/climon
and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.
```

$$[\varepsilon, e_1, e_2, e_3, e_{1,2}, e_{1,3}, e_{2,3}, e_{1,2,3}] \quad (56)$$

However, an error is returned when an index larger than 9 is found in the input expression:

```
> convert(e[1,2,10],eclibas_to_clibas); ##<<-- error is returned
  when index greater than 9 is found
Error, (in convert/eclibas_to_clibas) Sorry, can't convert as
index 10 greater than 9 has been found in the input. Expressions
of the type eclibasmon, eclimon, and eclipolynom can be
converted to the type clibasmon, climon, and clipolynom,
respectively, used in CLIFFORD/Bigebra provided the maximum
index in the input is 9 or less due to the restriction of the
CLIFFORD/Bigebra package to quadratic spaces (V,Q) of dimension
dim(V) <= 9.
```

Procedure 'split' will now work:

```
> pp:=add(x[i]*cbas[i],i=1..nops(cbas));
```

$$pp := Id\,x_1 + e1\,x_2 + e1we2\,x_5 + e1we3\,x_6 + e2\,x_3 + e2we3\,x_7 + e3\,x_4 + e1we2we3\,x_8 \quad (57)$$

```
> split(pp);
```

$$[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8], [Id, e1, e2, e3, e1we2, e1we3, e2we3, e1we2we3] \quad (58)$$

```
> pp1:=pp+2*e1+a*e1;
```

$$pp1 := Id\,x_1 + a\,e1 + e1\,x_2 + e1we2\,x_5 + e1we3\,x_6 + e2\,x_3 + e2we3\,x_7 + e3\,x_4 + e1we2we3\,x_8 + 2\,e1 \quad (59)$$

```
> split(pp1);
```

$$[x_1, a + x_2 + 2, x_3, x_4, x_5, x_6, x_7, x_8], [Id, e1, e2, e3, e1we2, e1we3, e2we3, e1we2we3] \quad (60)$$

Let us see a sample computation in both packages and conversions between results derived in eClifford and Clifford/Bigebra

```
> p,q,r:=1,1,1;
  _Bsignature_pqr:=[p,q,r];
  dim_V:=p+q+r;
  B:=linalg:-diag(1$p,(-1)$q,0$r);
```

$$p, q, r := 1, 1, 1$$

$$_Bsignature\_pqr := [1, 1, 1]$$

$$dim\_V := 3$$

$$B := \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (61)$$

```
> string_basis:=cbasis(dim_V); ##<<-- this is a basis in Cl(p,q,r)
  written as strings of type 'clibasmon'
```

$$string\_basis := [Id, e1, e2, e3, e1we2, e1we3, e2we3, e1we2we3] \quad (62)$$

```
> M1:=matrix(2^dim_V,2^dim_V,(i,j)->cmul(string_basis[i],
  string_basis[j]));
```

$$M1 := [[Id, e1, e2, e3, e1we2, e1we3, e2we3, e1we2we3], \quad (63)$$
$$[e1, Id, e1we2, e1we3, e2, e3, e1we2we3, e2we3],$$
$$[e2, -e1we2, -Id, e2we3, e1, -e1we2we3, -e3, e1we3],$$
$$[e3, -e1we3, -e2we3, 0, e1we2we3, 0, 0, 0],$$
$$[e1we2, -e2, -e1, e1we2we3, Id, -e2we3, -e1we3, e3],$$
$$[e1we3, -e3, -e1we2we3, 0, e2we3, 0, 0, 0],$$
$$[e2we3, e1we2we3, e3, 0, e1we3, 0, 0, 0],$$
$$[e1we2we3, e2we3, e1we3, 0, e3, 0, 0, 0]]$$

```
> M2:=map(convert,evalm(M1),clibas_to_eclibas);
```

$$M2 := \begin{bmatrix} \varepsilon & e_1 & e_2 & e_3 & e_{1,2} & e_{1,3} & e_{2,3} & e_{1,2,3} \\ e_1 & \varepsilon & e_{1,2} & e_{1,3} & e_2 & e_3 & e_{1,2,3} & e_{2,3} \\ e_2 & -e_{1,2} & -\varepsilon & e_{2,3} & e_1 & -e_{1,2,3} & -e_3 & e_{1,3} \\ e_3 & -e_{1,3} & -e_{2,3} & 0 & e_{1,2,3} & 0 & 0 & 0 \\ e_{1,2} & -e_2 & -e_1 & e_{1,2,3} & \varepsilon & -e_{2,3} & -e_{1,3} & e_3 \\ e_{1,3} & -e_3 & -e_{1,2,3} & 0 & e_{2,3} & 0 & 0 & 0 \\ e_{2,3} & e_{1,2,3} & e_3 & 0 & e_{1,3} & 0 & 0 & 0 \\ e_{1,2,3} & e_{2,3} & e_{1,3} & 0 & e_3 & 0 & 0 & 0 \end{bmatrix} \quad (64)$$

```
> M3:=map(convert,evalm(M2),eclibas_to_clibas);
```

$$M3 := [[Id, e1, e2, e3, e1we2, e1we3, e2we3, e1we2we3],$$
$$[e1, Id, e1we2, e1we3, e2, e3, e1we2we3, e2we3],$$
$$[e2, -e1we2, -Id, e2we3, e1, -e1we2we3, -e3, e1we3],$$
$$[e3, -e1we3, -e2we3, 0, e1we2we3, 0, 0, 0],$$
$$[e1we2, -e2, -e1, e1we2we3, Id, -e2we3, -e1we3, e3],$$
$$[e1we3, -e3, -e1we2we3, 0, e2we3, 0, 0, 0],$$
$$[e2we3, e1we2we3, e3, 0, e1we3, 0, 0, 0],$$
$$[e1we2we3, e2we3, e1we3, 0, e3, 0, 0, 0]] \quad (65)$$

Now, let's do a sample computation:

```
> pp:=add(x[i]*string_basis[i],i=1..nops(string_basis)); ##<<-- a
  general element in Cl(p,q,r) in old basis
  ppc:=convert(pp,clibas_to_eclibas);
```

$$pp := Id\,x_1 + e1\,x_2 + e1we2\,x_5 + e1we3\,x_6 + e2\,x_3 + e2we3\,x_7 + e3\,x_4 + e1we2we3\,x_8$$

$$ppc := \varepsilon\,x_1 + e_1\,x_2 + e_2\,x_3 + e_3\,x_4 + e_{1,2}\,x_5 + e_{1,3}\,x_6 + e_{2,3}\,x_7 + e_{1,2,3}\,x_8 \quad (66)$$

```
> out1:=cmul(pp,pp); ##<<-- Clifford product cmul applied to pp and
  pp
```

$$out1 := \left(x_1^2 + x_2^2 - x_3^2 + x_5^2\right) Id + 2\,x_2\,x_1\,e1 + 2\,x_5\,x_1\,e1we2 + 2\left(x_6\,x_1 + x_8\,x_3\right) e1we3$$
$$+ 2\,x_3\,x_1\,e2 + 2\left(x_7\,x_1 + x_8\,x_2\right) e2we3 + 2\left(x_4\,x_1 + x_8\,x_5\right) e3 + 2\left(x_8\,x_1 + x_7\,x_2 - x_3\,x_6$$
$$+ x_4\,x_5\right) e1we2we3 \quad (67)$$

```
> out2:=ecmul(ppc,ppc);  ##<<-- Clifford product ecmul applied to
  ppc and ppc
```

$$out2 := \left(x_1^2 + x_2^2 - x_3^2 + x_5^2\right)\varepsilon + 2\,x_2\,x_1\,e_1 + 2\,x_3\,x_1\,e_2 + \left(2\,x_1\,x_4 + 2\,x_5\,x_8\right) e_3 + 2\,x_5\,x_1\,e_{1,2} \quad (68)$$

$$+\left(2\,x_1\,x_6+2\,x_3\,x_8\right)e_{1,3}+\left(2\,x_1\,x_7+2\,x_2\,x_8\right)e_{2,3}+\left(2\,x_1\,x_8+2\,x_2\,x_7-2\,x_3\,x_6+2\,x_4\,x_5\right)e_{1,2,3}$$

```
> out11:=convert(out1,clibas_to_eclibas); ##<<-- conversion of out1
  to the new basis
```

$$out11 := \left(x_1^2+x_2^2-x_3^2+x_5^2\right)\varepsilon+2\,x_2\,x_1\,e_1+2\,x_3\,x_1\,e_2+\left(2\,x_1\,x_4+2\,x_5\,x_8\right)e_3+2\,x_5\,x_1\,e_{1,2}+\left(2\,x_1\,x_6+2\,x_3\,x_8\right)e_{1,3}+\left(2\,x_1\,x_7+2\,x_2\,x_8\right)e_{2,3}+\left(2\,x_1\,x_8+2\,x_2\,x_7-2\,x_3\,x_6+2\,x_4\,x_5\right)e_{1,2,3} \quad (69)$$

```
> simplify(out2-out11);  ##<<--confirming that out2=out11 in the
  new basis
```

$$0 \quad (70)$$

```
> out22:=convert(out2,eclibas_to_clibas);  ##<<-- conversion of
  out2 to the old basis
```

$$out22 := \left(x_1^2+x_2^2-x_3^2+x_5^2\right)Id+2\,x_2\,x_1\,e1+2\,x_5\,x_1\,e1we2+\left(2\,x_1\,x_6+2\,x_3\,x_8\right)e1we3+2\,x_3\,x_1\,e2+\left(2\,x_1\,x_7+2\,x_2\,x_8\right)e2we3+\left(2\,x_1\,x_4+2\,x_5\,x_8\right)e3+\left(2\,x_1\,x_8+2\,x_2\,x_7-2\,x_3\,x_6+2\,x_4\,x_5\right)e1we2we3 \quad (71)$$

```
> simplify(out22-out1);  ##<<--confirming that out22=out1 in the
  old basis
```

$$0 \quad (72)$$

**Section 4: Speed testing**

```
> _processor_type;
```

"Intel (R) Core (TM) i7-5500U CPU @ 2.40 GHz, 8.0 GB RAM" (73)

```
> _operating_system;
```

"Windows 10 64-bit" (74)

```
> p,q,r:=3,4,0: #[2,2,0]<1sec,[2,3,0]=1.3sec,[3,3,0]=5.3sec, [3,4,
  0]=23sec,[0,0,7]=15sec,[4,4,0]=101sec, [0,0,8]=61sec,[0,0,9]=
  148sec,[4,5,0]=496sec,[0,0,10]=677sec,[5,5,0]=2195sec in
  eClifford
  p,q,r:=3,4,0: #[2,2,0]<1sec,[2,3,0]=1.5sec,[3,3,0]=10.5sec,[3,4,
  0]=96sec,[0,0,7]=90sec,[4,4,0]=1143sec,[0,0,8]=1108sec,[0,0,9]=
  NA,[4,5,0]=NA,[0,0,10]=NA,[5,5,0]=NA in Clifford/Bigebra
  _Bsignature_pqr:=[p,q,r];
  B:=diag(1$p,-1$q,0$r):
  ecbas:=ecbasis(p+q+r):
  cbas:=cbasis(p+q+r):
```

(75)

$$\_Bsignature\_pqr := [3, 4, 0] \quad \textbf{(75)}$$

```
> start:=time():
  for m1 in ecbas do
      for m2 in ecbas do
          ecmul(m1,m2);
      end do;
  end do:
  eCliffordtime[[p,q,r]]:=time()-start;
```

$$eCliffordtime_{[3, 4, 0]} := 23.953 \quad \textbf{(76)}$$

```
> start:=time():
  for m1 in cbas do
      for m2 in cbas do
          cmul(m1,m2);
      end do;
  end do:
  Cliffordtime[[p,q,r]]:=time()-start;
```

$$Cliffordtime_{[3, 4, 0]} := 100.609 \quad \textbf{(77)}$$

Cookeville, November 15, 2015