cmulGTensor

File: cmulGtensor.mw

Date: June 10, 2012

Authors: Rafal Ablamowicz & Bertfried Fauser (to accompany "Using Periodicity Theorems for Computations in Higher Dimensional Clifford Algebras")

Description: cmulGtensor.mw describes how to set up and calculate with tensor product Clifford algebras using a graded tensor product. The graded tensor product is exported from the Bigebra package, computations in Clifford algebras for different bilinear forms are done using Clifford.

Provides:
-- cmulGTensor : Clifford product on a graded tensor product Clifford algebra
-- GTgradeinv: grade involution for the graded tensor product Clifford algebra
-- GTreversion: (bilinear form dependent) grade reversion on the tensor product Clifford algebra
-- bas2GTbas: graded algebra morphisms from a Grassmann basis over \( \Lambda(V_1 + V_2) \) to the tensor basis of \( \Lambda(V_1) \times \Lambda(V_2) \)
-- GTbas2bas: inverse graded algebra morphisms to bas2GTbas

Benchmarks: We provide some benchmarks to show that the Clifford product on the tensor algebra is roughly (in the symbolic case) as fast (or even a bit faster) than the Clifford product directly computed (in dim \( V = \dim V_1 + \dim V_2 \leq 9 \)).

Setup Clifford & Bigebra and Helper function:

First we need to load Clifford and Bigebra. Maple packages are loaded using the >with(<package>); command. Functions in packages are also available via their long form >packagename:-function (<args>); Clifford and Bigebra come with a large set of help pages, which can be accessed for example by >Clifford; or ?command; Each command comes with an explanation who to use it, as well as some examples. The help pages also contain test cases so that migrating the packages to new Maple versions undergoes detailed testing.

The definition of the actual Clifford algebras in use will be done below when testing/proving some facts of the procedures we are going to define.

> restart:with(Clifford);with(Bigebra);

Increase verbosity by infolevel[`function`] = val -- use online help > ?Bigebra[help]
We use a helper routine to merge two bilinear forms of two Clifford algebras to form a bilinear form of the ambient (graded tensor) Clifford algebra.

> #
# mergeBs takes two bilinear forms and returns their direct sum
# usage: B:=mergeBs(B1,B2);
#
mergeBs:=proc(b1,b2)
  local row,col,Zero;
  row:=linalg[coldim](b1);
  col:=linalg[rowdim](b2);
  Zero:=linalg[matrix](row,col,(i,j)->0);
  linalg:-blockmatrix(2,2,[b1,Zero,linalg[transpose](Zero),b2]);
end proc:

Vector space isomorphisms: bas2GTbas and GTbas2bas

We define two mutually inverse graded vector space maps (isomorphisms)

(i) bas2GTbas: \( CL_{p+r,q+s} \longrightarrow CL_{p,q} \times CL_{r,s} \)
(ii) GTbas2bas: \( CL_{p,q} \times CL_{r,s} \longrightarrow CL_{p+r,q+s} \)

To show that these maps are isomorphisms, we need to show that their compositions fulfill

\[ GTbas2bas \circ bas2GTbas = \text{Id}_{bas} \]
\[ bas2GTbas \circ GTbas2bas = \text{Id}_{GTbas} \]

as we work in finite dimensions, to show one of these equations is sufficient.

The map bas2GTbas is defined as follows on generators

\( \text{Id} \mapsto \langle \text{Id}, \text{Id} \rangle \)
\( e_i \mapsto \langle \text{Id}, e_i \rangle \) if \( e_i \) in \( V_1 \) (that is \( 1 \leq i \leq (p+q) \))
\( e_j \mapsto \langle \text{Id}, e_j \rangle \) if \( e_j \) in \( V_2 \) (that is \( p+q < j \leq (p+q)+(r+s) \))

and is extended multiplicatively and linearly to the whole vectors space \( \Lambda(V) \) underlying \( CL_{p+r,q+s} \).

> #
# bas2GTbas_mon : iso on monomials, handles Id correctly
# In a package, this routine would be internal
# and not exported
#
bas2GTbas_mon:=proc(x)
local N1, lst, lst1, lst2, i;
N1:=linalg:-rowdim(B1);
lst:=Clifford:-extract(x, 'integer');
lst1, lst2:=[], [];
for i from 1 to nops(lst) do
  if N1<lst[i] then
    lst2:=[op(lst2), lst[i]-N1];
  else
    lst1:=[op(lst1), lst[i]];
  end if;
end do;
&t(makeclibasmon(lst1), makeclibasmon(lst2));
end proc:
#
#  bas2GTbas: algebra isomorphism from CL_{p+1,q+1} -> CL_p,1 (x)
CL_1,1
#
bas2GTbas:=proc(x)
  local cf, term;
  if type(x, `+`) then
    return map(procname, x);
  elif type(x, `*`) then
    term, cf:=selectremove(type, x, clibasmon);
    return cf*bas2GTbas_mon(x);
  else
    return bas2GTbas_mon(x);
  end if;
end proc:
#
# GTbas2bas_mon : iso on monomials, handles Id correctly
#                 In a package, this routine would be internal
#                 and not exported
#
GTbas2bas_mon:=proc(x, y)
  local N1, lst, lst1, lst2, i;
  N1:=linalg:-rowdim(B1);
  lst1:=Clifford:-extract(x, 'integer');
  lst2:=map(n->n+N1, Clifford:-extract(y, 'integer'));
  makeclibasmon([op(lst1), op(lst2)]);
end proc:
#
# GTbas2bas: algebra isomorphism from CL_p,1 (x) CL_1,1 ->
Proof of vector space isomorphism:

We prove the isomorphism in (a) particular case(es), where we use arbitrary bilinear forms B1 on V1, and B2 on V2. We define first the basis for CL_p+q,r+s, stored in bas1. Then we apply bas2GTbas to obtain GTbas, a basis for CL_p,q (x) CL_r,s. Next we apply GTbas2bas to get bas2 in CL_p+r,q+s. The fact that bas1=bas2 proves the isomorphism property. We do not show output, but only the last statement showing the equality.

Before we can prove the statement, we need to set up the Clifford algebras and (Grassmann) bases properly. We use here general symbolic bilinear forms, that is we use an arbitrary basis generating the two subalgebras making up the ambient Clifford algebra:

\[ CL(B,V1+V2) \approx CL(V1,B1) (x) CL(V2,B2) \]

where \((x)\) is the graded tensor product. The grading determines what happens when two tensor factors are swapped or switched, and Bigebra exports functions to do this.

N1, N2 hold the dimensions of the Clifford algebras CL(V1,B1) and CL(V2,B2), the next two lines define the bilinear forms B1 and B2 using the (deprecated) Maple package linalg. Hence, calling procedures from linalg (which we have not loaded) requires the long form linalg:-procedure(<args>) (or linalg[procedure](<args>)). We use our helper function mergeBs to define the (symmetric) bilinear form B=B1+B2. Clifford can only compute with this form if the dimension of V is \(\leq 9\), we do this for testing that the two ways of computing the vector space isomorphisms gives equivalent results. (For some internal reasons Clifford loads an additional Cliplus package, which provides some functionality to deal with Clifford bases (standard bases used by Clifford are Grassmann bases).

The actual test that GTbas2bas and bas2GTbas are isomorphism is done as follows:

-- produce the Grassmann basis for CL(V,B), stored in bas1
-- map this basis to the graded tensor product Grassmann basis of CL(V1,B1) (x) CL(V2,B2) stored in GTbas
-- map the graded tensor basis back to the Grassmann basis of CL(V,B) and store it in bas2
-- compare if the two bases bas1 and bas2 are equal.

You may want to change the parameters (here N1,N2) to any pair of positive integers \(p,q\) such that \(p+q\leq 9\). (If \(p+q>9\) Clifford cannot form the algebra CL(V,B1) or its Grassmann basis, but you still can produce the Grassmann basis GTbas of the tensor product algebra as long as \(p\leq 9\), \(q\leq 9\)).

**Note:** As we use only properties of the Grassmann basis (vector space isomorphism) this example does not actually need the bilinear forms at all.
\[ N_1 := 3; N_2 := 2; \]
\[ B_1 := \text{linalg[matrix]}(N_1, N_1, (i, j) \rightarrow \text{if } i \leq j \text{ then } b[i, j] \text{ else } b[i, j] \text{ end if}); \]
\[ B_2 := \text{linalg[matrix]}(N_2, N_2, (i, j) \rightarrow \text{if } i = j \text{ then } g[i, j] \text{ else } g[i, j] \text{ end if}); \]
\[ \text{dim}_V := N_1 + N_2; \]
\[ B := \text{mergeBs}(B_1, B_2); \]
\[ B := \begin{bmatrix} b_{1,1} & b_{1,2} & b_{1,3} & 0 & 0 \\ b_{2,1} & b_{2,2} & b_{2,3} & 0 & 0 \\ b_{3,1} & b_{3,2} & b_{3,3} & 0 & 0 \\ 0 & 0 & 0 & g_{1,1} & g_{1,2} \\ 0 & 0 & 0 & g_{2,1} & g_{2,2} \end{bmatrix} \]  
\[ \text{evalb}(\text{bas1}=\text{bas2}); \]
\[ \text{GTbas} := \text{map}(\text{bas2GTbas}, \text{bas1}); \]
\[ \text{bas2} := \text{map}(\text{GTbas2bas}, \text{GTbas}); \]
\[ \text{evalb}(\text{bas1}=\text{bas2}); \]

**Definition of the graded tensor Clifford product**

The (binary) graded tensor Clifford product cmulGTensor is defined as follows. Let \( A, B \) be Clifford algebras, with Clifford multiplications \( m_A \) and \( m_B \), then the graded tensor product algebra \( A \times B \) has the product:
m_{A (x) B} = (m_A (x) m_B)^\circ (1 (x) \text{gswitch} (x) 1)

m_{A (x) B}[X1 (x) X2, Y1 (x) Y2] = \&t(m_A(X1,Y1), m_B(X2,Y2))

Using the facilities of Bigebra we can literally use this definition:

\text{gswitch}(\&t(X1,X2,\ldots,Xl), i) = (-1)^{\lfloor |X(i+1)| \rfloor |Xi|} \times \&t(X1,\ldots,X(i-1),X(i+1),Xi,X(i+2),\ldots,Xl)

Remember that \text{cmul}[B](x,y) takes an argument `B` holding a bilinear form, so that m_A=\text{cmul}[B1] and m_B=\text{cmul}[B2] and we hand \text{cmulGTensor} two bilinear forms defining the tensor product factor Clifford algebras A=\text{CL}_p,q and B=\text{CL}_r,s.

> #
> # \text{cmulGTensor}: the (binary) graded tensor product Clifford algebra product
> # for arbitrary generators (symmetric bilinear forms)
> #
> \text{cmulGTensor}:=\text{proc}(x,y,B1,B2)
> \text{local } f4;
> f4:=(a,b,x,y)\rightarrow \&t(\text{cmul}[B1](a,b),\text{cmul}[B2](x,y));
> \text{eval(subs(`&t`=f4, \text{gswitch}(&t(x,y),2)));
> \text{end proc;}

\text{Note:} It would be faster to code the action of \text{gswitch} directly into the internal function \text{f4}. Here we do not do this only to show the facilities of Bigebra and how to deal with graded tensors in general.

\textbf{Proof:} \text{bas2GTbas} (and its inverse \text{GTbas2bas}) is a graded Clifford algebra isomorphism

Let A=\text{CL}_p,q, B=\text{CL}_r,s. To show this we need to show commutativity of a diagram that has these two edges:

i) first multiply in A (x) B, then apply the isomorphism \text{GTbas2bas}

ii) first apply to both arguments of the multiplication the isomorphism \text{GTbas2bas}, and then multiply in \text{CL}_p+r,q+s

To show this, we use a multiplication table, computing for all basis elements. This gives two matrices \text{mat1} and \text{mat2}, and their equality (one needs to collect terms first) shows that the commutative diagram described above commutes. That is, the difference \text{mat1}-\text{mat2} is the zero matrix. Note that we compute with \text{general bilinear} froms \text{B1} and \text{B2}.

We do not show output, but only the logical test at the end showing the matrix identity to be true.

The actual test that \text{cmulGTensor} is a graded algebra isomorphism is done by computing the respective multiplication tables and show (using the isomorphisms \text{bas2GTbas} and \text{GTbas2bas} defined above) that they are equal:
-- produce the Grassmann basis for CL(V,B), stored in bas
-- producte the basis of the graded tensor space, stored in GTbas
-- compute the Clifford product (multiplication table) in the ambient Clifford algebra CL(V,B), store in mat1
-- compute the Clifford product in the tensor product Clifford algebra CL(V1,B1) (x) CL(V2,B2) and map the result back to the Grassmann basis of CL(V,B) using GTbas2bas, store in mat2
-- check that the two multiplication tables are equal, QED.

N1:=2:N2:=2:
B1:=linalg[[-matrix]](N1,N1,(i,j)->if i<=j then b[i,j] else b[i,j] end if);
B2:=linalg[[-matrix]](N2,N2,(i,j)->if i=j then g[i,j] else g[i,j] end if); ##(-1)^(i+1) else 0 end if);
B1 :=
[ b1, 1 b1, 2 ]
[ b2, 1 b2, 2 ]
B2 :=
[ g1, 1 g1, 2 ]
[ g2, 1 g2, 2 ]

dim_V:=N1+N2:
B:=mergeBs(B1,B2):
bas:=cbasis(dim_V):
GTbas:=map(bas2GTbas,bas):
mat1:=linalg[[-matrix]](nops(bas),nops(bas),(i,j)->cmul[B](bas[i],bas[j]));
mat2:=linalg[[-matrix]](nops(GTbas),nops(GTbas),(i,j)->clicollect(GTbas2bas(cmulGTensor(GTbas[i],GTbas[j],B1,B2))));
evalb(convert(evalm(mat1-mat2),set)={0});
true

Benchmarks : cmul versus cmulGTensor

We compute on a set of randomly chosen (equally distributed over the basis) basis monomials, and compare the timings.
**Note1:** Clifford comes with several multiplication algorithms which perform differently in different cases. The user can switch between them (see `Clifford:-useproduct`) or even define own product routine (not necessarily computing a Clifford product). By default Clifford uses:
-- cmulRS, using the Rota Stein Hopf algebra based algorithm, optimised for symbolic bilinear forms with off diagonal entries
-- cmulNUM, using the Chevalley construction, optimised for bilinear forms with numeric entries and possibly many zeros in it.
-- cmulWalsh, experimental multiplication, optimized for diagonal bilinear forms (not yet part of Clifford, it needs to be loaded by the user).

Our benchmarks are for the symbolic case using cmulRS.

**Note2:** Clifford uses type checking and functions heavily use the remember mechanism of Maple to speed up (significantly) computations. For that reason recomputing a result is much faster than doing it for the first time. For that reason computing a loop over the same calculation would just benchmark largely the overhead of type checking and function calls done internally in Clifford. For that reason we use randomly chosen elements.

**Note3:** We would expect that cmulGTensor is slower than cmul, as it operates on more complex data types. However, the symbolic forms involved are of dimension dim B1=N1, dim B2=N2 and dim B=N1+N2, so cmul[B] computes on all the (zero) off diagonal elements in the block structure
\[
B = \begin{bmatrix}
  B1 & 0 \\
  0 & B2
\end{bmatrix}
\]
Now cmulGTensor does not compute these off diagonal terms, and hence gains some advantage. This effect becomes larger for larger N1,N2.

As this computation is randomized, it should be run several times and average times should be taken.

```plaintext
> ######################################
> N1:=4:N2:=3:
> B1:=linalg[matrix](N1,N1,(i,j)->if i<=j then b[i,j] else b[i,j] end if);;
> B2:=linalg[matrix](N2,N2,(i,j)->if i=j then g[i,j] else g[i,j] end if);;  ##(-1)^(i+1) else 0 end if);
> #
> dim_V:=N1+N2:
> B:=mergeBs(B1,B2):;
> ######################################
> # test the graded algebra isomorphism
> # CL_p+r,q+s ~= CL_p,q (x) CL_r,s
> #
> bas:=cbasis(dim_V):
> GTbas:=map(bas2GTbas,bas):
> #
> # benchmarks:
> #
```
 rnd:=rand(1..nops(GTbas)):
MaxIter:=50:
 run1:=time():
 for i from 1 to MaxIter do
 cmul[B](GTbas2bas(GTbas[rnd()]),GTbas2bas(GTbas[rnd()]));
 end do:
 printf("===============================\n");
 printf(" cmul[B] needed %a sec\n",time()−run1);
 #
 run2:=time():
 for i from 1 to MaxIter do
 GTbas2bas(cmulGTensor(GTbas[rnd()],GTbas[rnd()],B1,B2));
 end do:
 printf("cmulGTensor needed %a sec\n",time()−run2);
 printf("===============================\n");

Grade involution and reversion for the graded tensor algebra

We extend the grade and reversion involutions to the graded tensor case as follows:

GTgradeinv = gradeinv (x) gradeinv


GTreversion(X (x) Y, B1, B2) = (-1)^( |X| |Y| ) reversion(X,B1) (x) reversion(Y,B2)

on an element `X (x) Y`.

The grade involution is straightforward. The reversion needs both, the switch and the graded switch, to
account for the sign factor (and keep the order of the algebra A (x) B. We encode the switch by a switch
in the local function `f2' swapping (without sign) the factors back.

> #
# GTgradeinv : grade involution on CL_p,q (x) CL_r,s
#
GTgradeinv:=proc(x)
 local f2;
 f2:=(a,b)->&t(gradeinv(a),gradeinv(b));
 eval(subs(`&t`=f2,x));
end proc:
#
# GTreversion : reversion involution on CL_p,q (x) CL_r,s
Proof that GTgradeinv and GTreversion are the correct involutions on $C = A (x) B$

To proof that GTgradeinv and GTreversion are the correct involutions on $A (x) B$, we need to verify that they coincide with the respective involutions on the ambient Clifford algebra. Doing so means that they need to commute with the graded algebra isomorphisms bas2GTbas and GTbas2bas. That is we need to show that

(i) $\text{bas2GTbas} \circ \text{gradeinv} = \text{GTgradeinv} \circ \text{bas2GTbas}$

(ii) $\text{bas2GTbas} \circ \text{reversion}[B] = \text{GTreversion}[B1,B2] \circ \text{bas2GTbas}$

where we have written the dependency of the reversion involutions on the respective bilinear forms as indices. In actual code they are passed as a second argument, hence GTreversion(X, B1, B2) will take the two bilinear forms B1,B2, making up the bilinear form $B = B1 + B2$ (direct sum).

Note: If we choose a basis for the Clifford algebra $C = CL_{p+r,q+s}$ such that the bilinear form $B$ does not decompose as a direct sum $B = B1 + B2$, then the reversion needs to be modified, as the switch $\circ$ gswitch part needs to be modified to encode this fact. This needs a smash product structure on the tensor (R-matrix).


```plaintext
G{T}bas:=map(bas2G{T}bas,bas):
basl:=map(bas2G{T}bas,map(gradeinv, bas));;
bas2:=map(G{T}gradeinv,G{T}bas,B1,B2):
printf("=========================================================
";
printf(" For general B=B1+B2 we have:\n");
printf(" grade involution is a graded algebra isomorphism :
%a\n",evalb(bas1=bas2));
#
#
bas1:=map(bas2G{T}bas,map(reversion[B], bas));;
bas2:=map(G{T}reversion,G{T}bas,B1,B2):
printf("reversion involution is a graded algebra isomorphism :
%a\n",evalb(bas1=bas2));
printf("=========================================================
";

B1 :=
[ [ b_{1,1} b_{1,2} b_{1,3} b_{1,4} ]
  [ b_{2,1} b_{2,2} b_{2,3} b_{2,4} ]
  [ b_{3,1} b_{3,2} b_{3,3} b_{3,4} ]
  [ b_{4,1} b_{4,2} b_{4,3} b_{4,4} ] ]

B2 :=
[ [ g_{1,1} g_{1,2} ]
  [ g_{2,1} g_{2,2} ] ]

===============================================================
For general B=B1+B2 we have:
 grade involution is a graded algebra isomorphism : true
reversion involution is a graded algebra isomorphism : true
===============================================================
```

June 28, 2012