

**- Function:** Bigebra:-VERSION - prints information about Bigebra and the version

### Calling Sequence:

VERSION()

### Parameters:

- none.

### Output:

- none.

### - Description:

- VERSION() displays information about the [Bigebra package](#)

### - Examples:

```
> restart:with(Bigebra) :
```

```
Increase verbosity by infolevel[`function`]=val -- use online help > ?Bigebra[h  
elp]
```

```
[ VERSION is a function hence parentheses are needed after the name!
```

```
> VERSION() ;
```

```
<=====>
```

```
°°Bi-Gebra Package VERSION 1.01 for Clifford version 11°°  
by Rafal Ablamowicz($) and Bertfried Fauser(*)  
(c) Dec-16-99 / Nov-16-2002 / Mar-3-1007 / Aug-14-2007 / Dec-20-2007  
Available from http://math.tntech.edu/rafal/
```

```
($) Department of Mathematics, Box 5054  
Tennessee Technological University  
Cookeville, TN 38505, U.S.A.  
Email: rablamowicz@tntech.edu  
URL: http://math.tntech.edu/rafal/
```

```
(*) Universit"at Konstanz  
Fachbereich Physik, Fach M678  
78457 Konstanz, Germany  
Email: Bertfried.Fauser@uni-konstanz.de  
URL: http://clifford.physik.uni-konstanz.de/~fauser/
```

```
Online help available with:
```

```
> ?Bigebra  
or use 'help' menu and search for topics
```

```
Copyright (c) Rafal Ablamowicz, Bertfried Fauser 1999-2008.  
All rights reserved. See also > ?Bigebra[help]
```

```
BUG-REPORTS to Bertfried Fauser
```

```
<=====>
```

[ >  
[ >

**- See Also:** [Bigebra helppage](#)

(c) Copyright 1999-2008, by Rafal Ablamowicz & Bertfried Fauser, all rights reserved.  
Last modified: December 20, 2007 /BF/RA.

## - Function: `&cco` - Clifford co-product

### Calling Sequence:

```
t1 = &cco(p1,i)
```

```
t1 = &cco(c1)
```

### Parameters:

- `p1` : a tensorpolynom (element of ``type/tensorpolynom``) of rank not less than `i` in each factor  
`i` : the slot number (first slot from the left is 1) on which the co-product acts
- `c1` : a Clifford polynom (element of one of these types: ``type/clipolynom``, ``type/climon``, ``type/clibasmon``)

### Output:

- `t1` : a tensor polynom

### Global variables:

- `BI_Id` - set by `make\_BI\_Id`
- `dim_V` - the dimension of the one-vector space `V`

### WARNING:

The Clifford co-product takes only one 'factor' (and one parameter), the **infix form** makes no sense with this function and yields *unpredictable nonsense*.

## - Description:

- Like the Clifford product, Clifford co-product needs a bilinear form defined on the base space of the Grassmann algebra. In the case of the co-product, this form is tied to co-one-vectors, so it is called co-scalar product. Since we deal with finite dimensional spaces, the dimension of the covector space is `dim_V`, the same as for the vector space `V` used by `CLIFFORD`. Hence we use the *global variable* `dim_V`, which has to be assigned. The matrix of the Clifford co-product w.r.t. the co-one-vector basis (in abuse of language also denoted by `e1`, see remarks in `EV`) is stored in `BI`. The elements of `BI` can be assigned freely, without any restrictions or relations to the matrix of the Clifford scalar product `B`. `BI` can be singular or non symmetric or even zero, in which case the Clifford co-product reduces to the `Graßmann co-product`.
- The Clifford co-product is based on Rota-Stein co-cliffordization `\[2,3,7,8\]`. This is the categorical dual of the Rota-Stein cliffordization of the Grassmann algebra which leads to the Clifford co-product. In Sweedler notation, the formula for Clifford co-product is:

$$\Delta_{\&c}(x) = (\text{wedge } \&t \text{ wedge})(\text{Id } \&t \text{ BI}_{(1)} \&t \text{ BI}_{(2)} \&t \text{ Id})\Delta(x)$$

where  $\Delta$  [c] is the Clifford co-product,  $\Delta$  is the [Grassmann co-product](#). The factor BI\_Id = BI\_(1) &t BI\_(2) (in fact internally represented by a list of type [ [sign,BI\_(1),BI\_(2)] , ... ]) has to be precomputed using [make BI\\_Id](#).

- The Clifford co-product is associative by construction, but it is not (graded) co-commutative.
- Clifford co-products lead to non-connected co-modules, see [Milnore and Moore](#). This is an important difference with Grassmann co-products [\[3,6\]](#).
- There is a sort of asymmetry in the BIGEBRA package, since the Clifford product is not (yet) computed using Rota-Stein cliffordization. The Clifford product is simply taken from CLIFFORD. This may cause problems if one tries to compute with q-deformed Clifford co-products. However, q-deformed Grassmann co-products are available by setting the global variable \_CLIENV[\_QDEF\_PREFACTOR] e.g. to -q.

### TO DO:

- The Clifford product has to be based on Rota-Stein cliffordization and a q-wedge product has to be created.

### Examples:

```
> restart:bench:=time():with(Clifford):with(Bigebra):
```

```
Increase verbosity by infolevel[function]=val -- use online help > ?Bigebr[h  
elp]
```

```
> dim_V:=2:
```

```
BI:=linalg[matrix](dim_V,dim_V,[a,b,c,d]): #co-scalarproduct  
m1:=make_BI_Id(): #remember this  
result in m1
```

```
Cliplus has been loaded. Definitions for type/climon and type/clipolynom now in  
clude &C and &C[K]. Type ?cliprod for help.
```

```
The Clifford co-product of Clifford polynomials needs no slot index:
```

```
> c1:= &cco(e1);  
c2:= &cco(&t(e1),1); # the same, &t(e1) = e1  
c3:= &cco(&t(e1),2); # the same, the slot is ignored here
```

```
c1 := (Id &t e1) - b (e1 &t e1we2) - d (e2 &t e1we2) + (e1 &t Id) + c (e1we2 &t e1)  
+ d (e1we2 &t e2)
```

```
c2 := (Id &t e1) - b (e1 &t e1we2) - d (e2 &t e1we2) + (e1 &t Id) + c (e1we2 &t e1)  
+ d (e1we2 &t e2)
```

```
c3 := (Id &t e1) - b (e1 &t e1we2) - d (e2 &t e1we2) + (e1 &t Id) + c (e1we2 &t e1)  
+ d (e1we2 &t e2)
```

```
Reduction to the Grassmann co-product is obtained by letting all parameters of the  
co-scalarproduct go to zero:
```

```
> subs(a=0,b=0,c=0,d=0,m1); ## &cco(Id) -> &gco(Id)
```

```

&gco (Id) ;
subs (a=0 ,b=0 ,c=0 ,d=0 ,c2) ; ## &cco (e1) -> &gco (e1)
&gco (e1) ;

```

$$\begin{aligned}
& Id \&t Id \\
& Id \&t Id \\
& (Id \&t e1) + (e1 \&t Id) \\
& (Id \&t e1) + (e1 \&t Id)
\end{aligned}$$

One can either change the co-product, or substitute the general parameters to get the result for another co-scalarproduct.

```

> BI:=linalg[matrix] (dim_V ,dim_V , [1 ,b ,b ,1]) : #new
co-scalarproduct
make_BI_Id() :

```

Compute once more &cco(e1) and compare it with the substituted result:

```

> c4:=&cco (e1) ;
c5:=subs (a=1 ,b=b ,c=b ,d=1 ,c1) ;

```

$$\begin{aligned}
c4 := & (Id \&t e1) - b (e1 \&t e1we2) - (e2 \&t e1we2) + (e1 \&t Id) + b (e1we2 \&t e1) \\
& + (e1we2 \&t e2)
\end{aligned}$$

$$\begin{aligned}
c5 := & (Id \&t e1) - b (e1 \&t e1we2) - (e2 \&t e1we2) + (e1 \&t Id) + b (e1we2 \&t e1) \\
& + (e1we2 \&t e2)
\end{aligned}$$

```

> &cco (Id) ;
subs (a=1 ,b=b ,c=b ,d=1 ,m1) ;

```

$$\begin{aligned}
& (Id \&t Id) + (e1 \&t e1) + b (e2 \&t e1) + b (e1 \&t e2) + (e2 \&t e2) \\
& + (b^2 - 1) (e1we2 \&t e1we2)
\end{aligned}$$

$$\begin{aligned}
& (Id \&t Id) + (e1 \&t e1) + b (e2 \&t e1) + b (e1 \&t e2) + (e2 \&t e2) \\
& + (b^2 - 1) (e1we2 \&t e1we2)
\end{aligned}$$

Reduction to the Grassmann co-product as a test:

```

> subs (a=0 ,b=0 ,c=0 ,d=0 ,m1) ; ## &cco (Id) -> &gco (Id)
&gco (Id) ;
subs (a=0 ,b=0 ,c=0 ,d=0 ,c2) ; ## &cco (e1) -> &gco (e1)
&gco (e1) ;

```

$$\begin{aligned}
& Id \&t Id \\
& Id \&t Id \\
& (Id \&t e1) + (e1 \&t Id) \\
& (Id \&t e1) + (e1 \&t Id)
\end{aligned}$$

Co-associativity of the Clifford co-product:

```
> &cco (&cco (&t (e1) , 1) , 1) ;
&cco (&cco (&t (e1) , 1) , 2) ;
```

$$\begin{aligned}
& -\&t(e1, e1we2, e1we2) + \&t(Id, e1, Id) - \&t(e2, e1, e2) + \&t(e2, e2, e1) + \&t(e1, Id, Id) \\
& + \&t(Id, Id, e1) - b \&t(Id, e1, e1we2) - \&t(Id, e2, e1we2) + b \&t(e1we2, e1, Id) \\
& + \&t(e1, e1, e1) + \&t(e1we2, Id, e2) + \&t(e1we2, e2, Id) + \&t(e1we2, e1, e1we2) \\
& - b \&t(e1, Id, e1we2) - b^2 \&t(e1we2, e1, e1we2) - \&t(e2, Id, e1we2) - b \&t(e1, e1we2, Id) \\
& + (b^2 - 1) \&t(e1we2, e1we2, e1) + \&t(e1, e2, e2) - \&t(e2, e1we2, Id) + 2 b \&t(e1, e2, e1) \\
& + \&t(Id, e1we2, e2) + b^2 \&t(e1, e1we2, e1we2) + b \&t(Id, e1we2, e1) + b \&t(e1we2, Id, e1) \\
& \&t(Id, e1, Id) - \&t(e2, e1, e2) + \&t(e2, e2, e1) + \&t(e1, Id, Id) + \&t(Id, Id, e1) \\
& - b \&t(Id, e1, e1we2) - \&t(Id, e2, e1we2) + b \&t(e1we2, e1, Id) + b^2 \&t(e1we2, e1we2, e1) \\
& + \&t(e1, e1, e1) + (b^2 - 1) \&t(e1, e1we2, e1we2) + \&t(e1we2, Id, e2) + \&t(e1we2, e2, Id) \\
& + \&t(e1we2, e1, e1we2) - b \&t(e1, Id, e1we2) - b^2 \&t(e1we2, e1, e1we2) \\
& - \&t(e2, Id, e1we2) - b \&t(e1, e1we2, Id) - \&t(e1we2, e1we2, e1) + \&t(e1, e2, e2) \\
& - \&t(e2, e1we2, Id) + 2 b \&t(e1, e2, e1) + \&t(Id, e1we2, e2) + b \&t(Id, e1we2, e1) \\
& + b \&t(e1we2, Id, e1)
\end{aligned}$$

```
> simplify (tcollect(%-%)) ;
```

0

Note however that acting on different slots of the same tensor gives different answers:

```
> res1 := &cco (&t (e1, e2) , 1) ;
res2 := &cco (&t (e1, e2) , 2) ;
print(`res1 - res2 = 0 is `, evalb(tcollect(res1-res2)=0)) ;
```

$$\begin{aligned}
res1 := & \&t(Id, e1, e2) - b \&t(e1, e1we2, e2) - \&t(e2, e1we2, e2) + \&t(e1, Id, e2) \\
& + b \&t(e1we2, e1, e2) + \&t(e1we2, e2, e2)
\end{aligned}$$

$$\begin{aligned}
res2 := & \&t(e1, Id, e2) + \&t(e1, e1, e1we2) + b \&t(e1, e2, e1we2) + \&t(e1, e2, Id) \\
& - \&t(e1, e1we2, e1) - b \&t(e1, e1we2, e2)
\end{aligned}$$

$res1 - res2 = 0$  is ,false

If the index is not in the range of the tensor slots, an error occurs so the user has to account for that.

```
> &cco (&t (e1, e2) , 3) ; ####<<<<-- Intended error
Error, (in Bigebra:-&cco) invalid subscript selector
```

```
> printf("Worksheet took %f seconds to compute on AMD Athlon
2700+ 1GB RAM machine\n", time() - bench) ;
```

Worksheet took 2.438000 seconds to compute on AMD Athlon 2700+ 1GB RAM machine

```
>
```

See Also: [Bigebra:-`&gco`](#), [Bigebra:-`&t`](#), [Bigebra:-drop t](#)

(c) Copyright 1999-2008, by Rafal Ablamowicz & Bertfried Fauser, all rights reserved.  
Last modified: December 20, 2007 /BF/RA.

**- Function:** `&gco_d` - dotted Grassmann co-product for a different filtration

### Calling Sequence:

```
t2 := &gco_d(t1,i)
```

```
t2 := &gco_d(c1)
```

### Parameters:

- `t1` : tensor polynoms
- `i` : tensor slot to be acted on
- `c1` : Clifford polynom

### Output:

- `t1` : a tensor polynom

### Global:

- for the transition to the regular wedge basis and back to the dotted basis (both represented as `eawebw...`) the two global matrices `F` and `FT` are used.

### - Description:

- The dotted Grassmann co-product is isomorphic to the regular Grassmann co-product on the dotted wedge basis. The function `&gco_d(t1,i)` computes this product using the original Grassmann co-product but w.r.t. the **undotted** basis. It is hence the counter part to the function [Cliplus:-dwedge](#) which computes the dotted wedge product in the **undotted** basis. The dotted and undotted bases arise from different filtrations of the underlying Grassmann algebra. **As Grassmann algebras they are isomorphic, but they are not isomorphic as Hopf algebras!**
- The function `&gco_d` needs the [Cliplus](#) and it will load it automatically if it was not done previously.
- This functionality is simply gained by wrapping the original Grassmann co-product and using internally the two functions ``convert/wedge to dwedge`` and ``convert/dwedge to wedge`` from the [Cliplus](#) package.
- If `F` and `FT` are antisymmetric arrays which are mutually transposed to each other (negative of each other) this mapping is an isomorphism (`F` and `FT` need not be non-singular !). Hence the dotted Grassmann co-product can be computed in the undotted basis by transforming back and forth.
- From a physical point of view, it is possible to consider the undotted basis as a fermionic time-ordered product and the dotted basis as a fermionic normal-ordered product. Hence, the

dotted Grassmann co-product employs the co-product of normal-ordered fields in the time-ordered basis, see [1].

- It should be noted that the loop tangle 'product \circ coproduct' works out completely differently if one mixes products and co-products for the same algebra with different filtration (ordering).

- **References:**

[1] B. Fauser, On the Hopf algebraic origin of Wick normal ordering, J. Math. Phys.: Gen. Math. 34, 2001:105-115

## Examples:

```
> restart:bench:=time():with(Clifford):with(Bigebra):
```

```
Increase verbosity by infolevel[function]=val -- use online help > ?Bigebra[help]
```

Define the dimension of the vector space V under consideration to be 3, and define F and FT

```
> dim_V:=3:
F:=array(antisymmetric,1..dim_V,1..dim_V):
F:=evalm(F);
FT:=evalm(-1*F);
w_bas:=cbasis(dim_V); ## the wedge basis
```

$$F = \begin{bmatrix} 0 & F_{1,2} & F_{1,3} \\ -F_{1,2} & 0 & F_{2,3} \\ -F_{1,3} & -F_{2,3} & 0 \end{bmatrix}$$

$$FT := \begin{bmatrix} 0 & -F_{1,2} & -F_{1,3} \\ F_{1,2} & 0 & -F_{2,3} \\ F_{1,3} & F_{2,3} & 0 \end{bmatrix}$$

```
w_bas := [Id, e1, e2, e3, e1we2, e1we3, e2we3, e1we2we3]
```

Now we invoke for the first time the dotted Grassmann co-product which **needs** also load the Cliplus package:

```
> with(Cliplus):
&gco_d(e1we2);
```

```
Cliplus has been loaded. Definitions for type/climon and type/clipolynom now include &C and &C[K]. Type ?cliprod for help.
```

$$(Id \&t e1we2) + F_{1,2} (Id \&t Id) + (e1 \&t e2) - (e2 \&t e1) + (e1we2 \&t Id)$$

Note that Cliplus was loaded. As the Grassmann co-product, &gco\_d can act on tensors in the i-th slot. For example, let's act on the second tensor slot of the tensor &t(e1,e2we3,e3) occupied by e2we3:

```
> &gco_d(&t(e1,e2we3,e3),2);
```

$$\&t(e1, Id, e2we3, e3) + F_{2,3} \&t(e1, Id, Id, e3) + \&t(e1, e2, e3, e3) - \&t(e1, e3, e2, e3)$$

+ &t(e1, e2we3, Id, e3)

Now let us show how the co-product acts on dotted and undotted elements

```
> w_p1:=elwe2;           #selection of an element in undotted
basis
w_c1:=&gco_d(w_p1);      #action of &gco_d on undotted element
elwe2
d_p1:=dwedge[F](e1,e2);  #transformation of elwe2 to dotted
basis
d_p2:=convert(w_p1,wedge_to_dwedge,F); #another way to
accomplish the same transformation
d_c1:=&gco_d(d_p1);      #action of &gco_d on the image of
elwe2 in dotted basis
```

$$w_{p1} := elwe2$$
$$w_{c1} := (Id \&t elwe2) + F_{1,2} (Id \&t Id) + (e1 \&t e2) - (e2 \&t e1) + (elwe2 \&t Id)$$
$$d_{p1} := elwe2 + F_{1,2} Id$$
$$d_{p2} := elwe2 + F_{1,2} Id$$
$$d_{c1} := (Id \&t elwe2) + 2 F_{1,2} (Id \&t Id) + (e1 \&t e2) - (e2 \&t e1) + (elwe2 \&t Id)$$

The following examples compose the dotted co-product with dotted and undotted wedge (acting on a wedge basis!!). First, let's show a dotted basis:

```
> Grassmann_basis:=cbasis(3);
#Grassmann un-dotted basis
dotted_basis:=map(convert,Grassmann_basis,wedge_to_dwedge,F);
#dotted basis
```

$$Grassmann\_basis := [Id, e1, e2, e3, elwe2, elwe3, e2we3, elwe2we3]$$
$$dotted\_basis := [Id, e1, e2, e3, elwe2 + F_{1,2} Id, elwe3 + F_{1,3} Id, e2we3 + F_{2,3} Id,$$
$$elwe2we3 + F_{2,3} e1 - F_{1,3} e2 + F_{1,2} e3]$$

We will use the following notation for the dotted basis, e.g., e1We2 = e1we2+F[1,2]\*Id, etc.:

```
> S:={e1we2+F[1,2]*Id=e1We2, e1we3+F[1,3]*Id=e1We3, e2we3+F[2,3]*Id
=e2We3,
elwe2we3+F[2,3]*e1-F[1,3]*e2+F[1,2]*e3=e1We2We3}:
subs(S,dotted_basis); #dotted basis in shorter (dotted wedge)
notation
```

$$[Id, e1, e2, e3, e1We2, e1We3, e2We3, e1We2We3]$$

Then, we compose dotted co-product with undotted and dotted wedge:

```
> for i in dotted_basis do
d_p1:=&gco_d(i):
drop_t(&map(d_p1,1,dwedge[F]));
```

```

`action_dwedge_o_&gco_d` =
2^maxgrade(%)*subs(S,%/2^maxgrade(%));
d_p2:=convert(%%,dwedge_to_wedge,-F);
print(`*****`);
od;

```

$$d_{p1} := Id \&t Id$$

$$Id$$

$$action\_dwedge\_o\_&gco\_d = Id$$

$$d_{p2} := Id$$

\*\*\*\*\*

$$d_{p1} := (Id \&t e1) + (e1 \&t Id)$$

$$2 e1$$

$$action\_dwedge\_o\_&gco\_d = 2 e1$$

$$d_{p2} := 2 e1$$

\*\*\*\*\*

$$d_{p1} := (Id \&t e2) + (e2 \&t Id)$$

$$2 e2$$

$$action\_dwedge\_o\_&gco\_d = 2 e2$$

$$d_{p2} := 2 e2$$

\*\*\*\*\*

$$d_{p1} := (Id \&t e3) + (e3 \&t Id)$$

$$2 e3$$

$$action\_dwedge\_o\_&gco\_d = 2 e3$$

$$d_{p2} := 2 e3$$

\*\*\*\*\*

$$d_{p1} := (Id \&t e1we2) + 2 F_{1,2} (Id \&t Id) + (e1 \&t e2) - (e2 \&t e1) + (e1we2 \&t Id)$$

$$4 e1we2 + 4 F_{1,2} Id$$

$$action\_dwedge\_o\_&gco\_d = 4 e1We2$$

$$d_{p2} := 4 e1we2$$

\*\*\*\*\*

$$d_{p1} := (Id \&t e1we3) + 2 F_{1,3} (Id \&t Id) + (e1 \&t e3) - (e3 \&t e1) + (e1we3 \&t Id)$$

$$4 e1we3 + 4 F_{1,3} Id$$

$$action\_dwedge\_o\_&gco\_d = 4 e1We3$$

$$d_{p2} := 4 e1we3$$

\*\*\*\*\*

$$d_{p1} := (Id \&t e2we3) + 2 F_{2,3} (Id \&t Id) + (e2 \&t e3) - (e3 \&t e2) + (e2we3 \&t Id)$$

$$4 e_2 w e_3 + 4 F_{2,3} Id$$

$$action\_dwedge\_o\_&gco\_d = 4 e_2 w e_3$$

$$d\_p2 := 4 e_2 w e_3$$

\*\*\*\*\*

$$d\_p1 := (Id \&t e_1 w e_2 w e_3) + 2 F_{2,3} (Id \&t e_1) - 2 F_{1,3} (Id \&t e_2) + 2 F_{1,2} (Id \&t e_3)$$

$$+ (e_1 \&t e_2 w e_3) + 2 F_{2,3} (e_1 \&t Id) - (e_2 \&t e_1 w e_3) - 2 F_{1,3} (e_2 \&t Id) + (e_1 w e_2 \&t e_3)$$

$$+ (e_3 \&t e_1 w e_2) + 2 F_{1,2} (e_3 \&t Id) - (e_1 w e_3 \&t e_2) + (e_2 w e_3 \&t e_1) + (e_1 w e_2 w e_3 \&t Id)$$

$$8 F_{1,2} e_3 + 8 e_1 w e_2 w e_3 + 8 F_{2,3} e_1 - 8 F_{1,3} e_2$$

$$action\_dwedge\_o\_&gco\_d = 8 e_1 w e_2 w e_3$$

$$d\_p2 := 8 e_1 w e_2 w e_3$$

\*\*\*\*\*

Thus, the above shows that **(wedge o Grassmann co-product)(x) = 2^(grade of x) \* x** for any x in the Grassmann basis (Grassmann\_basis) shown above and that the same is true, namely, **(dotted wedge o dotted Grassmann co-product)(y) = 2^(grade of y) \* y** for any y in the dotted wedge basis (dotted\_basis) shown above.

```
> printf("Worksheet took %f seconds to compute on Intel Pentium M
2.13 GHz 2GB RAM\n", time()-bench);
Worksheet took 3.814000 seconds to compute on Intel Pentium M 2.13 GHz 2GB RAM
>
```



**See Also:** [Bigebra:-`&gco`](#), [Bigebra:-`&cco`](#), [Bigebra:-`&t`](#), [Bigebra:-drop\\_t](#), [Bigebra:-`&map`](#), [Cliplus:-dwedge](#)

## - Function: `&gco` - Grassmann co-product

### Calling Sequence:

`t1 := &gco(t2,i)`

`t1 := &gco(c1)`

### Parameters:

- `t2` : a tensorpolynom (an element of ``type/tensorpolynom``) of rank not less than `i` in each factor  
`i` : the slot number (first slot is from the left is 1) on which the co-product acts
- `c1` is a Clifford polynom (an element of one of these types: ``type/clibasmon``, ``type/climon``, ``type/clipolynom``)

### Output:

- `t1` : is a tensorpolynom.

### WARNING:

The Clifford co-product takes only one 'factor' (and one parameter), the **infix form** makes no sense with this function and yields *unpredictable nonsense*.

## - Description:

- A Grassmann algebra leads naturally to a multi-vector space  $\wedge V$ . This space has a dual which we call  $(\wedge V)^* = \wedge V^*$ . There is a natural pairing between one-vectors and co-one-vectors which can be extended to a graded scalar valued (target/domain is the ring  $k$ ) action of co-multi-vectors on multivectors called pairing:  $\langle A, B \rangle : \wedge V^* \times \wedge V \rightarrow k$ . Let us denote the Grassmann product of co-multi-vectors by  $\wedge V^*$ , i.e using a `&v` (vee)-product. One obtains by categorical duality (i.e. by reversing arrows in commutative diagrams) the coproduct  $\Delta$ . For two-vectors this reads:

$$\begin{aligned}\langle a1 \vee a2 \mid b \rangle &= \langle a1 \ \&t \ a2 \mid \Delta(b) \rangle \\ &= \langle a1 \ \&t \ a2 \mid \sum_i (b)_{(1i)} \ \&t \ (b)_{(2i)} \rangle \\ &= \sum_i \langle a1 \mid (b)_{(1i)} \rangle \langle a2 \mid (b)_{(2i)} \rangle\end{aligned}$$

Since the co-vectors `a1` and `a2` are arbitrary co-multi-vectors, this defines the coproduct on an arbitrary multi-vector Grassmann element `b` in  $\wedge V$ . If `a1` is a co-one-multivector, this turns out to be the Laplace row expansion of the pairing, see [\[8.3\]](#). The same consideration can be done for columns, i.e. moving the wedge  $\wedge$  from right to left in the pairing to generate a Grassmann co-product on co-multi-vectors. Since we denote currently vectors and co-vectors by the *same* vector symbol ``e``, the user has to take care of the fact in which slot of a tensor a vector or co-vector resides, see [EV](#).

- Expanded in our basis, the above formalism leads to a combinatorial formula using split-sums and shuffles which are internally computed in BIGEBRA, [4]. From this construction, one concludes that the Grassmann co-product enjoys the following properties:
- The Grassmann co-product is associative,  $(\Delta \&t Id) \Delta = (Id \&t \Delta) \Delta$ .
- The Grassmann co-product is graded co-commutative  $\Delta = \tau \Delta$ , where  $\tau$  is the [graded switch](#).
- The Grassmann co-product is linear.
- Together with the Grassmann wedge product one proves this structure to be a Grassmann Hopf algebra, which possesses an [antipode](#).
- The Grassmann Hopf algebra is a bi-augmented bi-connected Hopf algebra, which is also called a non interacting Hopf algebra [3].

## Examples:

```
> restart:bench:=time():with(Clifford):with(Bigebra):
```

```
Increase verbosity by infolevel[function]=val -- use online help > ?Bigebra[h  
elp]
```

Some examples of Grassmann co-products of Clifford polynomials:

```
> &gco(e1);  
&gco(&t(e1),1); # the same, since  
e1 = drop_t(&t(e1));
```

$$(Id \&t e1) + (e1 \&t Id)$$

Cliplus has been loaded. Definitions for type/climon and type/clipolynom now in  
clude &C and &C[K]. Type ?cliprod for help.

$$(Id \&t e1) + (e1 \&t Id)$$

$$e1 = e1$$

```
> &gco(e1we2);  
&gco(a*e3);  
&gco(e1we2+a*e3);
```

$$(Id \&t e1we2) + (e1 \&t e2) - (e2 \&t e1) + (e1we2 \&t Id)$$

$$a (Id \&t e3) + a (e3 \&t Id)$$

$$(Id \&t e1we2) + (e1 \&t e2) - (e2 \&t e1) + (e1we2 \&t Id) + a (Id \&t e3) + a (e3 \&t Id)$$

Acting on tensor slots:

```
> &gco(&t(e1,e2),1);  
&gco(&t(e1,e2),2);
```

$$\&t(Id, e1, e2) + \&t(e1, Id, e2)$$

$$\&t(e1, Id, e2) + \&t(e1, e2, Id)$$

```
> &gco(Id);
```

```
&gco(%,1);
```

$Id \&t Id$   
 $\&t(Id, Id, Id)$

```
> &gco(a*&t(e1,e2)+b*&t(e3,e4),1);
```

$a \&t(Id, e1, e2) + a \&t(e1, Id, e2) + b \&t(Id, e3, e4) + b \&t(e3, Id, e4)$

Checking co-associativity:

```
> &gco(&gco(&t(e1we2),1),1);  
&gco(&gco(&t(e1we2),1),2);  
evalb(%-%%=0);
```

$\&t(Id, Id, e1we2) + \&t(Id, e1, e2) + \&t(e1, Id, e2) - \&t(Id, e2, e1) - \&t(e2, Id, e1)$   
 $+ \&t(Id, e1we2, Id) + \&t(e1, e2, Id) - \&t(e2, e1, Id) + \&t(e1we2, Id, Id)$   
 $\&t(Id, Id, e1we2) + \&t(Id, e1, e2) + \&t(e1, Id, e2) - \&t(Id, e2, e1) - \&t(e2, Id, e1)$   
 $+ \&t(Id, e1we2, Id) + \&t(e1, e2, Id) - \&t(e2, e1, Id) + \&t(e1we2, Id, Id)$   
 $true$

Checking graded co-commutativity:

```
> g1:=&gco(e1we2+e1we2we3);  
g2:=gswitch(g1,1);
```

$g1 := (Id \&t e1we2) + (e1 \&t e2) - (e2 \&t e1) + (e1we2 \&t Id) + (Id \&t e1we2we3)$   
 $+ (e1 \&t e2we3) - (e2 \&t e1we3) + (e1we2 \&t e3) + (e3 \&t e1we2) - (e1we3 \&t e2)$   
 $+ (e2we3 \&t e1) + (e1we2we3 \&t Id)$   
 $g2 := (Id \&t e1we2) + (e1 \&t e2) - (e2 \&t e1) + (e1we2 \&t Id) + (Id \&t e1we2we3)$   
 $+ (e1 \&t e2we3) - (e2 \&t e1we3) + (e1we2 \&t e3) + (e3 \&t e1we2) - (e1we3 \&t e2)$   
 $+ (e2we3 \&t e1) + (e1we2we3 \&t Id)$

```
> evalb(%-%%=0);
```

$true$

Note however that acting on different slots of the same tensor gives different answers:

```
> res1:=&gco(&t(e1,e2),1);  
res2:=&gco(&t(e1,e2),2);  
printf("res1 - res2 =0 is %s !",evalb(tcollect(res1-res2)=0));
```

$res1 := \&t(Id, e1, e2) + \&t(e1, Id, e2)$   
 $res2 := \&t(e1, Id, e2) + \&t(e1, e2, Id)$

```
res1 - res2 =0 is false !
```

If the index is not in the range of the tensor slots, an error occurs, so the user has to account for that.

```
> &gco (&t(e1,e2),3);
```

```
Error, (in Bigebra:-&gco) invalid subscript selector
```

```
> printf("Worksheet took %f seconds to compute on Intel Pentium M  
2.13 GHz 2GB RAM\n",time()-bench);
```

```
Worksheet took 0.064000 seconds to compute on Intel Pentium M 2.13 GHz 2GB RAM
```

```
>
```

**- See Also:** [Bigebra:-`&cco`](#), [Bigebra:-`&t`](#), [Bigebra:-drop\\_t](#), [Bigebra:-`&map`](#)

(c) Copyright 1999-2008, by Rafal Ablamowicz & Bertfried Fauser, all rights reserved.

Last modified: December 20, 2007 /BF/RA.