blade_operator_syntax.wxm

GNU General Public License 2019 Stephen Athel Abbott.

A test document for Geometric Algebra with wxMaxima
contains...
Initialization
Loading of functions (intrinsic and GA specific)
Pseudoscalar definition (specifies the space dimension) and
calculation of the inverse pseudoscalar used to generate the dual of a multivector
Enumeration of the standard basis for the specified dimension

Examples of syntax and precedence of the infix blade operators

Initialization

```
(%i1)  reset()$
       kill(all)$
       stardisp:true$
       stringdisp:true$
       noundisp:true$
       simp:true$
       dotdistrib:true$
       derivabbrev:true$
       lispdisp:true$
```

load intrinsic (maxima or lisp) function files

```
(%i8)  load("basic")$
       load("facexp")$
       load("functs")$
       load("scifac")$
```

batchload GA specific (maxima) function files;

```
(%i12)  ext:["wxm"]$
        file_type_maxima:append(ext,file_type_maxima)$

(%i14)  batchload("gafns0")$
        batchload("gafns1")$
        batchload("gafns2")$
        batchload("gafns3")$
        batchload("gafns4")$
        batchload("gafns5")$
        batchload("gafns6")$
```

batchload GC specific (maxima) function files;

```
(%i21)  batchload("gcfns1")$
        batchload("gcfns2")$
        batchload("gcfns3")$
```

the pseudoscalar and its inverse
the lowest useable dimension pseudoscalar should be {e1,e2} i.e. Plen = 2

```
(%i24)  Pseudos: {e1,e2,e3}$
        Pvar:listofvars(Pseudos)$
        Plen:length(Pvar)$
        I:Pseudos$
        ni:(Plen-1)*Plen/2$
        Ii:(-1)^ni*I$
        kill(ni)$
        ldisplay(Pvar)$

 (%t31) Pvar=[e1,e2,e3]
```

initialize this list with the only list we have so far; Pvar

```
(%i32)  lstbases:makelist(Pvar[n],n,1,Plen)$
```

the integer array, nbases (n:0,...3) is used for grader(M) in gafns4.wxm;
in order to use the same indices for the lists, (n:1,...3) define it using function array()

```
(%i33)  array(nbases,Plen)$
        eset:setify(Pvar)$
        for ng:1 thru Plen do
        block(nbases[ng]:combination(Plen,ng),
        lstbases[ng]:full_listify(powerset(eset,ng)))$
        maxnbases:(combination(Plen,floor(Plen/2)))$
        nbases[0]:1$ /*an array index zero cannot be used to index any of the lists*/
        ldisplay(lstbases)$
        kill(eset,ng)$

 (%t38) lstbases=[[[e1],[e2],[e3]],[[e1,e2],[e1,e3],[e2,e3]],[[e1,e2,e3]]]
```

initialize this list again with Pvar

```
(%i40)  lstblds:makelist(Pvar[n],n,1,Plen)$
```

the list named lstblds is used for grader(M) in gafns4.wxm;
lstblds is an list of lists of blades and allblds is a list of all blades

```
(%i41)  for ng:1 thru Plen do
        block(lstb:lstbases[ng],
        lstblds[ng]:makelist(list2blade(lst),lst,lstb))$
        allblds:[]$
        for ng:1 thru Plen do
        block(allblds:append(allblds,lstblds[ng]))$
        ldisplay(lstblds)$
        ldisplay(allblds)$
        kill(lstb,ng)$

 (%t44) lstblds=[[{e1},{e2},{e3}],[{e1,e2},{e1,e3},{e2,e3}],[{e1,e2,e3}]]
 (%t45) allblds=[{e1},{e2},{e3},{e1,e2},{e1,e3},{e2,e3},{e1,e2,e3}]
```

end of Initialization

definition and precedence of the infix operators is found in gafns0.wxm so that document
should be opened and used as a reference when studying the values of the expressions below

the binary blade product operators and their functions are as taken from gafns0.wxm...
"~*"(A,B):=bladep(A,B);
"~^"(A,B):=bladepout(A,B);
"~."(A,B):=bladepinn(A,B);

the blade product operators all have a precedence less than the set, {,}, so that the base
vectors remain bound to each other as a bivector set in the evaluation of the expressions

```
(%i47)  {e1,e2}~*{e3};
        {e1,e2}~^{e3};
        {e1}~*{e2,e3};
        {e1}~^{e2,e3};
        {e1}~.{e1,e2};

(%o47) {e1,e2,e3}
(%o48) {e1,e2,e3}
(%o49) {e1,e2,e3}
(%o50) {e1,e2,e3}
(%o51) {e2}
```

blade geometric and outer product syntax and precedence using the scalars, a and b

the blade product operators have a higher precedence than scalar multiplication, so the
blade operators can handle two inputs of (scalar*vector)

```
(%i52)  (a*{e1})~*(b*{e2});
        (a*{e1})~^(b*{e2});

(%o52) a*b*{e1,e2}
(%o53) a*b*{e1,e2}
```

when there is equal precedence, the order of execution is left to right

```
(%i54)  a~*{e1}~*b~*{e2};
        a~^{e1}~^b~^{e2};

(%o54) a*b*{e1,e2}
(%o55) a*b*{e1,e2}
```

however, invalid products will produce invalid expressions

```
(%i56)  a*{e1}~*b*{e2};
        a*{e1}~^b*{e2};

(%o56) a*b*{e1}*{e2}
(%o57) a*b*{e1}*{e2}
```

blade inner product syntax and precedence using the scalar, a

```
(%i58)  {e1}~.{e1,e2};
        a*{e1}~.{e1,e2};
        {e1}~.a*{e1,e2};

(%o58) {e2}
(%o59) a*{e2}
(%o60) 0
```

the blade power operators are rarely used; they have the same precedence
as the blade product operators

```
(%i61)  {e1}@*3;
        {e1}@*3~*{e2};
        {e1,e2}@*3;
        {e2}~*{e1}@*3;

(%o61) {e1}
(%o62) {e1,e2}
(%o63) -{e1,e2}
(%o64) {e1,e2}
```