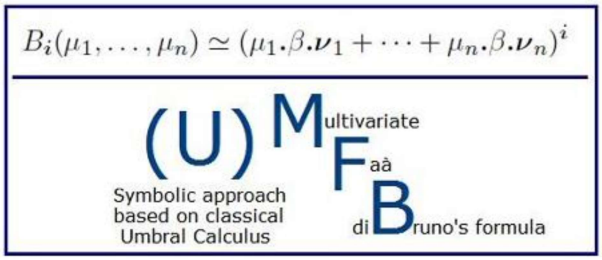
This Maple worksheet accompanies the paper [1]: Di Nardo E., G. Guarino, D. Senato (2011), *A new algorithm for computing the multivariate Faà di Bruno's formula*, Applied Mathematics and Computation. doi:10.1016/j.amc.2011.01.001

A new algorithm for computing the multivariate Faà di Bruno's formula



E. Di Nardo*
elvira.dinardo@unibas.it
http://www.unibas.it/utenti/dinardo/home.html;
Tel: +39 0971205890, Fax: +39 0971205896

G. Guarino**
giuseppe.guarino@aspbasilicata.it

D. Senato*
domenico.senato@unibas.it

*Department of Mathematics, Computer Science and Economics, University of Basilicata, viale dell'Ateneo Lucano n.10, 85100 Potenza, Italy

**Medical School, Università del Sacro Cuore (Rome branch), Largo Agostino Gemelli n.8, 00168 Roma, Italy

Introduction

Abstract: We provide a new algorithm for computing the multivariate Faà di Bruno's formula. We follow a symbolic approach based on the classical umbral calculus that leads back the computation of the multivariate Faà di Bruno's formula to a suitable multinomial expansion. The resulting computational times are faster compared with procedures existing in the literature.

Application Areas/Subject: Combinatorics & algebraic methods

Keyword: multivariate composite function, Faà di Bruno's formula, multivariate cumulants, multivariate Hermite polynomials, classical umbral calculus.

See Also: background on the classical umbral calculus in [2], background on multiset subdivisions in [3], for different applications of multiset subdivisions see [5]

Initialization

```
> restart;
> with(combinat, partition, multinomial, permute);
  with(ListTools, Transpose, Flatten);
```

[*partition*, *multinomial*, *permute*]

[*Transpose*, *Flatten*] **(2.1)**

Subdivisions of a multiset

This topic has been extensively discussed in [3].
See http://www.maplesoft.com/applications/view.aspx?SID=33039 for details.

```
> nRep := proc(u) mul(x[2]!, x = convert(u, multiset)); end:
> URv := proc(u, v)
    local U, ou, i, ptr, vI;
    ou := NULL; U := [ ]; vI := indets(v);
    for ptr from nops(u) by -1 to 2 do
      if has(u[ptr], v) then break; fi; od;
    for i from ptr to nops(u) do
      if not (u[i] = ou or has(u[i], vI)) then
        ou := u[i];
        U := [op(U), [ op(u[1..i-1]), u[i]·v, op(u[i+1..-1]) ]];
      fi;
      od; op(U), [op(u), v];
  end:
> URV := proc( )
    local U, V, i;
    U := [args[1, 1]]; V := args[2, 1];
    for i from 1 to nops(V) do
```

```
    U := [ seq( URv(u, V_i), u = U ) ];
    od; seq( [x, args_{1,2}·args_{2,2}], x = U ) ;
  end:
```

```
> URmV := proc( )
    local U, i;
    if nargs = 1 then
      args;
    else U := URV( args_1, args_2 );
        for i from 3 to nargs do
          U := seq( URV( u, args_i ), u = [U] );
        od; seq( [ x_1, x_2 / nRep(x_1) ], x = U );
    fi; end:
```

```
> comb := proc(V, ptr, Y)
     if ptr = nops(V) + 1 then return(Y); fi;
     seq( comb(V, ptr + 1, [ op(Y), L ] ), L = V_ptr );
   end:
```

```
> makeTab := proc( )
    local U;
    U := [ seq( `if`( args_i = 0, NULL,
          [ seq( [ [ seq( α_i^z, z = y ) ], multinomial( args_i, seq(r, r = y) ) ],
        y = partition( args_i ) ) ] ),
      i = 1 ..nargs ) ];
    if nops(U) = 1
      then [ seq( [ x_1, x_2 / nRep(x_1) ], x = op(U) ) ];
      else [ seq( URmV( op(x) ), x = [ comb( U, 1, [ ] ) ] ) ]; fi;
  end:
```

Examples

```
> makeTab(3);
```

$$\left[\left[\left[\alpha_1, \alpha_1, \alpha_1\right], 1\right], \left[\left[\alpha_1, \alpha_1^2\right], 3\right], \left[\left[\alpha_1^3\right], 1\right]\right] \qquad \textbf{(3.1)}$$

```
> makeTab(2, 1);
```

$$\left[\left[\left[\alpha_1\, \alpha_2, \alpha_1\right], 2\right], \left[\left[\alpha_1, \alpha_1, \alpha_2\right], 1\right], \left[\left[\alpha_1^2\, \alpha_2\right], 1\right], \left[\left[\alpha_1^2, \alpha_2\right], 1\right]\right] \qquad \textbf{(3.2)}$$

▼ Multivariate Faà di Bruno's formula

A MAPLE algorithm for the computation of the multivariate Faà di Bruno's formula by using the umbral equivalence (18) in [1].

Univariate and Multivariate Case

MFB Functions

Introduction

As we have proved in [3], the function *makeTab* generates a multiset subdivision (see the output of the function *makeTab* in the previous section).

Parameters: $MFB(n_1, n_2 \ldots)$, $n_i \in \mathbb{N}$

Univariate with Univariate

By using the subdivisions of a multiset with all equal elements, we are able to compute the univariate Faà di Bruno's formula, that is to expand $\frac{d^n}{dx^n} f(g(x))$.

For example in order to expand $\frac{d^2}{dx^2} f(g(x))$ we call *MFB*(2) which gives $f_2 g_1^2 + f_1 g_2$ as output . To see that $f_2 g_1^2 + f_1 g_2$ is the expansion of $\frac{d^2}{dx^2} f(g(x))$, replace $f_i = \left(D^{(i)}\right)(f)\,(g(x))$ for $i = 1, 2$ and $g_1 = \frac{d}{dx} g(x)$, $g_2 = \frac{d^2}{dx^2} g(x)$. Then we have $D^{(2)}(f)\,(g(x)) \left(\frac{d}{dx} g(x)\right)^2 + D(f)\,(g(x)) \left(\frac{d^2}{dx^2} g(x)\right)$.

Remark: the function *MFB* uses the function *makeTab* as we show in the following example. Calling *makeTab*(2) we have $\left[\left[\left[\alpha_1, \alpha_1\right], 1\right], \left[\left[\alpha_1^2\right], 1\right]\right]$. The first block $\left[\alpha_1, \alpha_1\right]$ has two elements: its cardinality gives the index of f_2. The second block $\left[\alpha_1^2\right]$ has one element: its cardinality gives the index of f_1. The first element α_1 in the first block $\left[\alpha_1, \alpha_1\right]$ has degree 1: its degree gives the index of g_1. Since in $\left[\alpha_1, \alpha_1\right]$ we have two equal elements of degree 1, we get g_1^2. This term must be multiplied with the integer (the multiplicity) that appears in each block (for example: 1 in $\left[\left[\alpha_1, \alpha_1\right], 1\right]$ or n in $\left[\left[\alpha_1, \ldots\ldots, \alpha_1\right], n\right]$). The second block $\left[\left[\alpha_1^2\right], 1\right]$ contains only one element with degree 2. This gives just g_2. At the end, the summation of the two terms returns $f_2 g_1^2 + f_1 g_2$.

Univariate with Multivariate

When in the multiset there are two or more different elements, the result of a call to *MFB* is different from the previous one. For esample *MFB*(1,1) generates the following output:

$f_1 g_{1,1} + f_2 g_{1,0} g_{0,1}$ that represents the expansion of

$$\frac{\partial^2}{\partial x_2\,\partial x_1} f\left(g\left(x_1, x_2\right)\right).$$

Indeed if in $f_1\, g_{1,1} + f_2\, g_{1,0}\, g_{0,1}$ we replace $f_i = \left(\mathrm{D}^{(i)}\right)(f)\left(g\left(x_1, x_2\right)\right)$ for $i = 1, 2$, $g_{1,1} = \frac{\partial^2}{\partial x_1\,\partial x_2} g\left(x_1, x_2\right)$ and $g_{1,0} = \frac{\partial}{\partial x_1} g\left(x_1, x_2\right)$, $g_{0,1} = \frac{\partial}{\partial x_2} g\left(x_1, x_2\right)$ we have $\mathrm{D}^{(2)}(f)\left(g\left(x_1, x_2\right)\right)\left(\frac{\partial}{\partial x_2} g\left(x_1, x_2\right)\right)\left(\frac{\partial}{\partial x_1} g\left(x_1, x_2\right)\right) + \mathrm{D}(f)\left(g\left(x_1, x_2\right)\right)\left(\frac{\partial^2}{\partial x_1\,\partial x_2} g\left(x_1, x_2\right)\right)$, which is the well known expansion of $\frac{\partial^2}{\partial x_1\,\partial x_2} f\left(g\left(x_1, x_2\right)\right)$

Remark: Similarly to what it happens for the univariate/univariate case, the function *MFB* calls the function *makeTab* as we show in the following example.

Calling *makeTab*(1,1) we obtain $\left[\left[\left[\alpha_1\ \alpha_2\right], 1\right], \left[\left[\alpha_1, \alpha_2\right], 1\right]\right]$.

The first block $\left[\alpha_1\ \alpha_2\right]$ has one element: its cardinality gives the index of f_1. The second block $\left[\alpha_1, \alpha_2\right]$ has two elements: its cardinality gives the index of f_2. In the blocks, two variables α_1, α_2 are involved. This explains why we use two indexes in $g_{i,j}$ elements. In particular we have: $\alpha_1\alpha_2 \to g_{1,1}$, $\alpha_1 \to g_{1,0}$ and $\alpha_2 \to g_{0,1}$. Therefore from the block $\left[\alpha_1\ \alpha_2\right]$ we have $g_{1,1}$, and from the block $\left[\alpha_1, \alpha_2\right]$ we get $g_{1,0}\, g_{0,1}$. At the end, the summation of the two terms returns $f_1\, g_{1,1} + f_2\, g_{1,0}\, g_{0,1}$.

The Code

```
> MFB := proc( )
      option remember;
      local n, vIndets, E;
      n := add(args_i, i = 1 ..nargs);
      if n = 0 then return 1 end if;
      vIndets := [seq(α_i, i = 1 ..nargs)];
      E := add(f_nops(y_1)·y[2]·mul(g_seq(degree(x, vIndets_i), i = 1 .. nops(vIndets)), x = y_1), y
       = makeTab(args))
  end proc:
```

Examples

Univariate with Univariate $\frac{d^n}{dx^n} f(g(x))$

```
> MFB(2);
```

$$f_2\, g_1^2 + f_1\, g_2 \qquad \textbf{(4.1.3.1)}$$

```
> MFB(3);
```

$$f_3 g_1^3 + 3 f_2 g_1 g_2 + f_1 g_3 \qquad \textbf{(4.1.3.2)}$$

Univariate with Multivariate $\frac{\partial^n}{\partial x_1 .. \partial x_n} f\left(g\left(x_1,.., x_n\right)\right)$

> $MFB(1, 0);$

$$f_1 g_{1,\,0} \qquad \textbf{(4.1.3.3)}$$

> $MFB(1, 1);$

$$f_1 g_{1,\,1} + f_2 g_{1,\,0} g_{0,\,1} \qquad \textbf{(4.1.3.4)}$$

> $MFB(2, 1);$

$$2 f_2 g_{1,\,1} g_{1,\,0} + f_3 g_{1,\,0}^2 g_{0,\,1} + f_1 g_{2,\,1} + f_2 g_{2,\,0} g_{0,\,1} \qquad \textbf{(4.1.3.5)}$$

> $MFB(1, 1, 1);$

$$f_1 g_{1,\,1,\,1} + f_2 g_{1,\,1,\,0} g_{0,\,0,\,1} + f_2 g_{1,\,0,\,1} g_{0,\,1,\,0} + f_2 g_{1,\,0,\,0} g_{0,\,1,\,1} + f_3 g_{1,\,0,\,0} g_{0,\,1,\,0} g_{0,\,0,\,1} \qquad \textbf{(4.1.3.6)}$$

The function *joint*

Introduction

The procedure *joint* provides a way to multiply the factors of formula (22) in [1].

Parameters: $joint\left(\left[p_1, p_2,...\right], \left[q_1, q_2 ....\right],...\{n\ blocks\}\right)$ where $p_i,.., q_i \in \mathbb{N}$

Example: $joint([1, 0], [1, 1])$

If we need to multiply $MFB(1,0)$ with $MFB(1,1)$ we must follow the following steps:

1) Call $MFB(1,0)$ and $MFB(1, 1)$.
2) For each output, make the following evalutions: $g \rightarrow g[i]; f_j \rightarrow f^j[i]$ where $i=1..n$.
   So from $MFB(1,0) = f_1 g_{1,\,0}$, by using the previous evaluations, we have $f1g1_{1,\,0}$.
   From $MFB(1,1) = f_1 g_{1,\,1} + f_2 g_{1,\,0} g_{0,\,1}$, by using the previous evaluations we have $f1\ g1_{1,\,1} + f2\ g2_{1,\,0} g2_{0,\,1}$.
3) Compute the following cefficients:
   $r = \left(p_1 + q_1 + ...\right)! \cdot \left(p_2 + q_2 + ...\right)! \cdot ..... \cdot \left(p_n + q_n + ...\right)!$
   $s = \left(p_1! \cdot p_2! \cdot ... \cdot q_1! \cdot q_2! \cdot ...\right)$, in the example $r = 2! \cdot 1! = 2$ and $s = 1! \cdot 0! \cdot 1! \cdot 1! = 1$
4) Multiply the two previous terms as follows:
   $\frac{r}{s}\left(f1g1_{1,\,0}\right)\left(f1\ g1_{1,\,1} + f2\ g2_{1,\,0} g2_{0,\,1}\right) =$
   $= 2\, f1\ g1_{1,\,0} f2\ g2_{1,\,1} + 2\, f1\ g1_{1,\,0} f2^2\ g2_{1,\,0} g2_{0,\,1}$

The Code

```
> joint := proc( )
  local p1, p2, M, V;
```

```
V := `if`(nargs = 1, [[args]], [args]);
p1 := mul(add(y, y = x)!, x = Transpose(V));
p2 := mul(x!, x = Flatten(V));
M := max(seq(add(y, y = x), x = V));
expand(p1/p2 · mul(eval(MFB(op(V_i)), [g = g||i,
        seq(f_j = (f||i)^j, j = 1..M)]), i = 1..nargs));
```

end:

Examples

```
> joint([1, 0], [1, 1]);
```

$$2\,f1\,g1_{1,0}f2\,g2_{1,1} + 2\,f1\,g1_{1,0}f2^2\,g2_{1,0}\,g2_{0,1} \qquad \textbf{(4.2.3.1)}$$

```
> joint([1, 1], [2, 1]);
```

$$12\,f1\,g1_{1,1}f2^2\,g2_{1,1}\,g2_{1,0} + 6\,f1\,g1_{1,1}f2^3\,g2_{1,0}^2\,g2_{0,1} + 6\,f1\,g1_{1,1}f2\,g2_{2,1} \qquad \textbf{(4.2.3.2)}$$
$$+ 6\,f1\,g1_{1,1}f2^2\,g2_{2,0}\,g2_{0,1} + 12\,f1^2\,g1_{1,0}\,g1_{0,1}f2^2\,g2_{1,1}\,g2_{1,0}$$
$$+ 6\,f1^2\,g1_{1,0}\,g1_{0,1}f2^3\,g2_{1,0}^2\,g2_{0,1} + 6\,f1^2\,g1_{1,0}\,g1_{0,1}f2\,g2_{2,1}$$
$$+ 6\,f1^2\,g1_{1,0}\,g1_{0,1}f2^2\,g2_{2,0}\,g2_{0,1}$$

```
> joint([1, 0], [1, 1], [0, 1]);
```

$$4\,f1\,g1_{1,0}f3\,g3_{0,1}f2\,g2_{1,1} + 4\,f1\,g1_{1,0}f3\,g3_{0,1}f2^2\,g2_{1,0}\,g2_{0,1} \qquad \textbf{(4.2.3.3)}$$

The function *mkT*

Introduction

This procedure gives a list of all subvectors having the same lenght of a vector V and such that their summation returns V.

Parameters: *mkT*(V, n), $n \in \mathbb{N}$

Note: the output of $V = [v_1,..,v_r]$ is $[p_1,..,p_r],...,[q_1,..,q_r]$ such that

$p_1 + .. + q_1 = v_1$

. . .

$p_r + .. + q_r = v_r$

Example: Suppose to call *mkt*([1,1],2).

1) First the function *makeTab* is called: *makeTab*(1,1)=$[[[\alpha_1\ \alpha_2], 1], [[\alpha_1, \alpha_2], 1]]$.

2) From this list and for each block, we consider only the first subblock, that is $[\alpha_1\ \alpha_2]$, $[\alpha_1, \alpha_2]$.

3) Since the first block $[\alpha_1\ \alpha_2]$ has cardinality one, we set $[\alpha_1\ \alpha_2, null]$ as we have asked for two blocks (that is two subvectors, note that the second parameter is $n=2$). So we have

[1, 1], [0, 0]. The second block $[\alpha_1, \alpha_2]$ has two elements and so we have [1,0],[0,1]

4) The result is the list [[1, 1], [0, 0]], [[1, 0], [0, 1]].

5) At the end, in order to have all lists, we need to permute the subblocks for each block, that is:

[[1, 1], [0, 0]]
[[1, 0], [0, 1]]
[[0, 0], [1, 1]]
[[0, 1], [1, 0]]

This function is used in the master function.

The Code

```
> mkT := proc(V, n)
    local vE, L, nV;
    nV := nops(V);
    vE := [seq(α[i], i=1..nV)];
    L := seq(`if`(nops(x[1]) <= n, x[1], NULL), x=makeTab(op(V)));
    L := seq([seq([seq(degree(y, z), z=vE)], y=x), [0$nV]$(n-nops(x))], x=[L]);
    L := seq(op(permute(x)), x=[L]);
  end:
```

Examples

```
> mkT([1, 1], 1);
```

[[1, 1]] **(4.3.3.1)**

```
> mkT([1, 1], 2);
```

[[1, 1], [0, 0]], [[0, 0], [1, 1]], [[1, 0], [0, 1]], [[0, 1], [1, 0]] **(4.3.3.2)**

```
> for L in mkT([2, 1], 2) do print(L); od;
```

[[1, 1], [1, 0]]
[[1, 0], [1, 1]]
[[2, 1], [0, 0]]
[[0, 0], [2, 1]]
[[2, 0], [0, 1]]
[[0, 1], [2, 0]] **(4.3.3.3)**

The Master Function *UMFB*

Introduction

(Umbral Multivariate Fa`a di Bruno's Formula)

All previous steps are combined in this procedure. The *UMFB* function recalls the *MFB* function for the univariate and the multivariate case (parameter *n=1*) and allows us to construct the multivariate-multivariate case (parameter *n>1*).

Parameters: *UMFB*(V, n), $V=\{v_1, v_2 \ldots\}$, *n = #different g functions*

the output is the expansion of

$$\left(\frac{\partial^{h_1+..+h_r}}{\partial x_1^{h_1} ... \partial x_r^{h_r}} \right) f\left(g[1]\left(x_1, (\,)..(\,), x_r\right), (\,)..(\,), g[n]\left(x_1, (\,)..(\,), x_r\right) \right)$$

Esample: $UMFB([1,3],2) = \frac{\partial^3}{\partial y^2 \, \partial x} f(g[1](x,y), g[2](x,y))$

Pseudocode:

```
UMFB(V, n)
  if n = 1 then return( MFB(V) );
  S := 0;
  for v in mkT( V, n ) do
      S := S + joint(v) {with f1^i f2^j .. fn^k → f_{i, j,.., k}}
  next for
  return(S);
end:
```

The Code

```
> UMFB := proc(V, n)
    local S, vE;
    if n = 1 then return(expand(eval(MFB(op(V)), [g = g1]))); fi;
    vE := [seq(f||i = 1, i = 1..n)];
    S := add(joint(op(x)), x = [mkT(V, n)]);
    add(f_{seq(degree(x, f||i), i = 1..n)} · eval(x, vE), x = S);
  end:
```

Examples

Univariate with Univariate $\frac{d^r}{dx^r} f(g(x))$ we have the same result of *MFB* function.

```
> UMFB([2], 1);
```

$$f_2 g1_1^2 + f_1 g1_2 \qquad \textbf{(4.4.3.1)}$$

```
> UMFB([3], 1);
```

$$f_3 g1_1^3 + 3 f_2 g1_1 g1_2 + f_1 g1_3 \qquad \textbf{(4.4.3.2)}$$

Univariate with Multivariate

$\frac{\partial^r}{\partial x_1 .. \partial x_r} f\left(g\left(x_1,.., x_r\right)\right)$ we have the same result of MFB function.

```
> UMFB([1, 1], 1);
```

$$f_1 g1_{1,1} + f_2 g1_{1,0} g1_{0,1} \qquad \textbf{(4.4.3.3)}$$

```
> UMFB([2, 1], 1);
```

$$2 f_2 g1_{1,1} g1_{1,0} + f_3 g1_{1,0}^2 g1_{0,1} + f_1 g1_{2,1} + f_2 g1_{2,0} g1_{0,1} \qquad \textbf{(4.4.3.4)}$$

Multivariate with Multivariate $\frac{\partial^r}{\partial x_1 .. \partial x_r} f\left(g_1(x_1,..,x_r),..,g_n(x_1,..,x_r)\right)$.

```
> UMFB([1],2);
```

$$f_{1,0}\,g1_1 + f_{0,1}\,g2_1 \qquad \textbf{(4.4.3.5)}$$

```
> UMFB([2],2);
```

$$2f_{1,1}\,g1_1 g2_1 + f_{2,0}\,g1_1^2 + f_{1,0}\,g1_2 + f_{0,2}\,g2_1^2 + f_{0,1}\,g2_2 \qquad \textbf{(4.4.3.6)}$$

```
> UMFB([1],3);
```

$$f_{1,0,0}\,g1_1 + f_{0,1,0}\,g2_1 + f_{0,0,1}\,g3_1 \qquad \textbf{(4.4.3.7)}$$

```
> UMFB([1,1],2);
```

$$f_{1,0}\,g1_{1,1} + f_{2,0}\,g1_{1,0}g1_{0,1} + f_{0,1}\,g2_{1,1} + f_{0,2}\,g2_{1,0}g2_{0,1} + f_{1,1}\,g1_{1,0}g2_{0,1} + f_{1,1}\,g1_{0,1}g2_{1,0} \qquad \textbf{(4.4.3.8)}$$

Test Results

Introduction

In this section we analyze the accuracy of the results comparing the output obtained by using the Maple *diff* function (routine by which the multivariate Faà di Bruno's formula is computable in Maple) .To this aim, we set:

$$f_{\#1,\#2,...,\#n} = \left(D_{1,\,1,\,1\,...2,\,2,\,2\,...n,\,n,\,n}\right)(f)\left(g[1](x_1,..,x_r),..,g[n](x_1,..,x_r)\right)$$

$$g[k]_{i,..j.} = \frac{\partial^{i+..+j}}{\partial x_1^i .. \partial x_r^j} g[k](x_1,..,x_r)$$

In order to compare the results we have set up three functions: *decoSub, deco* and *testUMFB*. The *decoSub* function is an auxiliary routine useful for the *deco* function. The *deco* function takes the result of the function *UMFB* as input and gives an output equal to the Maple function *diff*. The *testUMFB* function checks the accuracy of the result.

Parameter: *testUMFB*(*V, n*, tDebug) where *V*, *n* are the same as the *UMFB* function and *tDebug* $\in$ *{true, false}* [set *tDebug* = *true* if you need to display also the output of the function *diff*]

Example: $\frac{\partial^2}{\partial x_1 \partial x_2} f\left(g(x_1,x_2), g(x_1,x_2)\right) =$

$$\left(\frac{\partial}{\partial x_1} g1(x_1,x_2)\right)\left(\frac{\partial}{\partial x_2} g1(x_1,x_2)\right) D_{1,1}(f)\left(g1(x_1,x_2), g2(x_1,x_2)\right) +$$
$$\left(\frac{\partial}{\partial x_1} g1(x_1,x_2)\right)\left(\frac{\partial}{\partial x_2} g2(x_1,x_2)\right) D_{1,2}(f)\left(g1(x_1,x_2), g2(x_1,x_2)\right) +$$
$$\left(\frac{\partial^2}{\partial x_1 \partial x_2} g1(x_1,x_2)\right) D_1(f)\left(g1(x_1,x_2), g2(x_1,x_2)\right) +$$
$$\left(\frac{\partial}{\partial x_2} g1(x_1,x_2)\right)\left(\frac{\partial}{\partial x_1} g2(x_1,x_2)\right) D_{1,2}(f)\left(g1(x_1,x_2), g2(x_1,x_2)\right) +$$

$$\left(\frac{\partial}{\partial x_1}\, g2(x_1, x_2)\right)\left(\frac{\partial}{\partial x_2}\, g2(x_1, x_2)\right) \mathrm{D}_{2,2}(f)\,(g1(x_1, x_2), g2(x_1, x_2)) +$$
$$\left(\frac{\partial^2}{\partial x_1\, \partial x_2}\, g2(x_1, x_2)\right) \mathrm{D}_2(f)\,(g1(x_1, x_2), g2(x_1, x_2))$$

We get the same result from

$$UMFB([1, 1], 2) = f_{1,0}\, g1_{1,1} + f_{2,0}\, g1_{1,0}\, g1_{0,1} + f_{0,1}\, g2_{1,1} + f_{0,2}\, g2_{1,0}\, g2_{0,1} + f_{1,1}\, g1_{1,0}\, g2_{0,1} + f_{1,1}\, g1_{0,1}\, g2_{1,0}$$

setting

$$f_{2,0} = \mathrm{D}_{1,1}(f)\,(g1(x_1, x_2), g2(x_1, x_2));$$
$$f_{0,2} = \mathrm{D}_{2,2}(f)\,(g1(x_1, x_2), g2(x_1, x_2));$$
$$f_{1,1} = \mathrm{D}_{1,2}(f)\,(g1(x_1, x_2), g2(x_1, x_2));$$
$$f_{1,0} = \mathrm{D}_1(f)\,(g1(x_1, x_2), g2(x_1, x_2));$$
$$f_{0,1} = \mathrm{D}_2(f)\,(g1(x_1, x_2), g2(x_1, x_2));$$

$$g1_{1,1} = \frac{\partial^2}{\partial x_1\, \partial x_2}\, g1(x_1, x_2); \quad \left[g2_{1,1} = \frac{\partial^2}{\partial x_1\, \partial x_2}\, g2(x_1, x_2)\right]$$
$$g1_{1,0} = \frac{\partial}{\partial x_1}\, g1(x_1, x_2); \quad \left[g2_{1,0} = \frac{\partial}{\partial x_2}\, g2(x_1, x_2)\right]$$
$$g1_{0,1} = \frac{\partial}{\partial x_2}\, g1(x_1, x_2); \quad \left[g2_{0,1} = \frac{\partial}{\partial x_2}\, g2(x_1, x_2)\right]$$

The code

```
> decoSub := proc(vE, vVars, nTot_x)
    local v, nTot_g, vG, vX, cE;
    cE := coeffs(vE);
    v := vE/cE;
    nTot_g := nops(vVars) - 1;
    v := [seq( [seq(`if`(has(x, u), u, NULL), u = vVars), cE, [op(op(x)[1])], op(x)[2]/2 ], x
      = op(v^2) )];
    vX := seq(x||j, j = 1..nTot_x);
    vG := seq(g||i(vX), i = 1..nTot_g);
    cE * mul( `if`(x[1] = f, D[seq(i$(x[3, i]), i = 1..nTot_g)](f)(vG),
          diff(x[1](vX), seq(x||i$x[3, i], i = 1..nops(x[3])))^x[4]), x = v);
  end:
```

```
> deco :=proc( )
      local v, V, vG, nTot_g, nTot_x;
      V := `if`(nargs = 1, indets(args), indets(op(args[1])[1]));
      for v in V do
          if has(v, 'f') then
              nTot_g := nops([op(v)]);
              break
          end if
      end do;
      for v in V do
          if not has(v, 'f') then
              nTot_x := nops([op(v)]);
              break
          end if
      end do;
      vG :=[seq(g||i, i = 1 ..nTot_g), f];
      if nops(V) = 2 then
          decoSub(args, vG, nTot_x)
      else
          add(decoSub(x, vG, nTot_x), x = [op(args)])
      end if
  end proc:

> testUMFB := proc(vX, ng, tDebug)
    local R0, R1, R2, F, uX, st :
    F := f(seq(g||i(seq(x||j, j = 1 ..nops(vX))), i = 1 ..ng));
    uX := seq(x||j$vX[j], j = 1 ..nops(vX));
    R0 := UMFB(vX, ng) :
    R1 := deco(R0) :
    R2 := expand(diff(F, uX));
    if tDebug then print(R0); print(R2); fi;
    if expand(R1 − R2) = 0 then print(TEST_OK) else print(___ERROR___);fi;
  end:
```

Execution Samples

```
> UMFB([1, 1], 1);
```

$$f_1\, g1_{1,1} + f_2\, g1_{1,0}\, g1_{0,1} \tag{5.3.1}$$

```
> Result1 := deco(%);
```

$$Result1 := \mathrm{D}(f)\,(g1(x1,x2))\left(\frac{\partial^2}{\partial x2\,\partial x1}\,g1(x1,x2)\right) + \mathrm{D}^{(2)}(f)\,(g1(x1,x2))\left(\frac{\partial}{\partial x1}\,g1(x1,x2)\right)\left(\frac{\partial}{\partial x2}\,g1(x1,x2)\right) \tag{5.3.2}$$

```
> Result2 := diff(f(g1(x1, x2)), x1, x2);
```

$$Result2 := \mathrm{D}(f)\,(g1(x1,x2))\left(\frac{\partial^2}{\partial x2\,\partial x1}\,g1(x1,x2)\right) + \mathrm{D}^{(2)}(f)\,(g1(x1, \tag{5.3.3}$$

$x2))\left(\frac{\partial}{\partial x1}\,g1(x1,x2)\right)\left(\frac{\partial}{\partial x2}\,g1(x1,x2)\right)$

> $evalb(\text{Result1} = \text{Result2});$

$$true \tag{5.3.4}$$

> $testUMFB([1,1],1,true);$

$$f_1\,g1_{1,1}+f_2\,g1_{1,0}\,g1_{0,1}$$

$$\mathrm{D}(f)\,(g1(x1,x2))\left(\frac{\partial^2}{\partial x2\,\partial x1}\,g1(x1,x2)\right)+\mathrm{D}^{(2)}(f)\,(g1(x1,x2))\left(\frac{\partial}{\partial x1}\,g1(x1,x2)\right)\left(\frac{\partial}{\partial x2}\,g1(x1,x2)\right)$$

$$TEST\_OK \tag{5.3.5}$$

> $testUMFB([1,1],2,true);$

$$f_{1,0}\,g1_{1,1}+f_{2,0}\,g1_{1,0}\,g1_{0,1}+f_{0,1}\,g2_{1,1}+f_{0,2}\,g2_{1,0}\,g2_{0,1}+f_{1,1}\,g1_{1,0}\,g2_{0,1}+f_{1,1}\,g1_{0,1}\,g2_{1,0}$$

$$\mathrm{D}_1(f)\,(g1(x1,x2),g2(x1,x2))\left(\frac{\partial^2}{\partial x2\,\partial x1}\,g1(x1,x2)\right)+\mathrm{D}_{1,1}(f)\,(g1(x1,x2),g2(x1,x2))\left(\frac{\partial}{\partial x1}\,g1(x1,x2)\right)\left(\frac{\partial}{\partial x2}\,g1(x1,x2)\right)+\mathrm{D}_2(f)\,(g1(x1,x2),g2(x1,x2))\left(\frac{\partial^2}{\partial x2\,\partial x1}\,g2(x1,x2)\right)+\mathrm{D}_{2,2}(f)\,(g1(x1,x2),g2(x1,x2))\left(\frac{\partial}{\partial x1}\,g2(x1,x2)\right)\left(\frac{\partial}{\partial x2}\,g2(x1,x2)\right)+\mathrm{D}_{1,2}(f)\,(g1(x1,x2),g2(x1,x2))\left(\frac{\partial}{\partial x1}\,g1(x1,x2)\right)\left(\frac{\partial}{\partial x2}\,g2(x1,x2)\right)+\mathrm{D}_{1,2}(f)\,(g1(x1,x2),g2(x1,x2))\left(\frac{\partial}{\partial x2}\,g1(x1,x2)\right)\left(\frac{\partial}{\partial x1}\,g2(x1,x2)\right)$$

$$TEST\_OK \tag{5.3.6}$$

> $testUMFB([4,3],2,false);$

$$TEST\_OK \tag{5.3.7}$$

Execution Time

In this section we analize the difference of the execution time between the *UMFB* function and the Maple *diff* function.
Notice: to perform an appropriate test, make a restart (using "!!!" for recall the entire worksheet) to clear any cash areas.

> $testTime := \mathbf{proc}(V,n)$
$\quad \mathbf{local}\ r1, r2, h, lx, st;$
$\quad h := f\left(seq\left(g\,||\,i\left(seq\left(x\,||\,j, j=1..nops(V)\right)\right), i=1..n\right)\right);$

```
    lx := seq( `$`(x||j, V[j]), j = 1 ..nops(V) );
    st := time( );
    r1 := UMFB(V, n);
    print(UMFB_Time = time( ) - st);
    print(UMFB_Tot_Elements = nops( [op(r1)] ) );

    st := time( );
    r2 := expand(diff(h, lx) );
    print(DIFF_Time = time( ) - st); st := time( );
    print(UMFB_Tot_Elements = nops( [op(r2)] ) );
end proc:

> testTime( [7, 6], 2 );
```

$$UMFB\_Time = 3.406$$

$$UMFB\_Tot\_Elements = 60190$$

$$DIFF\_Time = 30.188$$

$$UMFB\_Tot\_Elements = 60190 \qquad \textbf{(6.1)}$$

Conclusions

By using the classical umbral calculus, we provide a new Maple algorithm for computing the multivariate Faà di Bruno's formula for the following compositions: univariate with univariate, univariate with multivariate, multivariate with univariate, multivariate with the same multivariate, multivariate with different multivariates in any arbitrary number of components. The resulting algorithm is speeder of the routine MAPLE *diff* performing the same computations.

References

[1] Di Nardo E., G. Guarino, D. Senato (2011), *A new algorithm for computing the multivariate Faà di Bruno's formula*, **Appl. Math. Comp.** doi:10.1016/j.amc.2011.01.001

[2] Di Nardo, E., Senato, D. (2006) *An umbral setting for cumulants and factorial moments*. **European J. Combin.** 27, No. 3, 394–413. (http://www.arxiv.org/abs/math/0412052)

[3] Di Nardo E., G. Guarino, D. Senato, *Multiset Subdivision*, source Maple algorithm located in www.maplesoft.com (http://www.maplesoft.com/applications/view.aspx?SID=33039)

[4] Di Nardo E., G. Guarino, D. Senato (2008) *An unifying framework for k-statistics, polykays and their generalizations*. **Bernoulli**. Vol. 14(2), 440-468. (http://www.unibas.it/utenti/dinardo/BEJ6163290408.pdf)

[5] Di Nardo E., G. Guarino, D. Senato*, A Maple algorithm for k-statistics, polykays and their multivariate generalization,* source Maple algorithm located in www.maplesoft.com (http://www.maplesoft.com/applications/view.aspx?SID=33040)

Contact the author for permission if you wish to use this application in for-profit activities