16.20 DFPART: Derivatives of generic functions

This package supports computations with total and partial derivatives of formal function objects. Such computations can be useful in the context of differential equations or power series expansions.

Author: Herbert Melenk.

The package `DFPART` supports computations with total and partial derivatives of formal function objects. Such computations can be useful in the context of differential equations or power series expansions.

16.20.1 Generic Functions

A generic function is a symbol which represents a mathematical function. The minimal information about a generic function function is the number of its arguments. In order to facilitate the programming and for a better readable output this package assumes that the arguments of a generic function have default names such as $f(x,y)$,$q(rho,phi)$. A generic function is declared by prototype form in a statement

`GENERIC_FUNCTION` ⟨*fname*⟩`(`⟨arg_1⟩,⟨arg_2⟩$\cdots$⟨arg_n⟩`);`

where $fname$ is the (new) name of a function and arg_i are symbols for its formal arguments. In the following $fname$ is referred to as “generic function", $arg_1, arg_2 \cdots arg_n$ as “generic arguments" and $fname(arg_1, arg_2 \cdots arg_n)$ as “generic form". Examples:

```
generic_function f(x,y);
generic_function g(z);
```

After this declaration REDUCE knows that

- there are formal partial derivatives $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$ $\frac{\partial g}{\partial z}$ and higher ones, while partial derivatives of f and g with respect to other variables are assumed as zero,
- expressions of the type $f()$, $g()$ are abbreviations for $f(x,y)$, $g(z)$,
- expressions of the type $f(u,v)$ are abbreviations for $sub(x=u, y=v, f(x,y))$
- a total derivative $\frac{df(u,v)}{dw}$ has to be computed as $\frac{\partial f}{\partial x}\frac{du}{dw} + \frac{\partial f}{\partial y}\frac{dv}{dw}$

16.20.2 Partial Derivatives

The operator `DFP` represents a partial derivative:

`DFP (`$\langle$*expr*$\rangle, \langle$*dfarg*$_1\rangle, \langle$*dfarg*$_2\rangle \cdots \langle$*dfarg*$_n\rangle$`) ;`

where $expr$ is a function expression and $dfarg_i$ are the differentiation variables. Examples:

```
dfp(f(),{x,y});
```

means $\frac{\partial^2 f}{\partial x \partial y}$ and

```
dfp(f(u,v),{x,y});
```

stands for $\frac{\partial^2 f}{\partial x \partial y}(u, v)$. For compatibility with the DF operator the differentiation variables need not be entered in list form; instead the syntax of `DF` can be used, where the function expression is followed by the differentiation variables, eventually with repetition numbers. Such forms are interenally converted to the above form with a list as second parameter.

The expression $expr$ can be a generic function with or without arguments, or an arithmetic expression built from generic functions and other algebraic parts. In the second case the standard differentiation rules are applied in order to reduce each derivative expressions to a minimal form.

When the switch `NAT` is on partial derivatives of generic functions are printed in standard index notation, that is f_{xy} for $\frac{\partial^2 f}{\partial x \partial y}$ and $f_{xy}(u, v)$ for $\frac{\partial^2 f}{\partial x \partial y}(u, v)$. Therefore single characters should be used for the arguments whenever possible. Examples:

```
generic_function f(x,y);
generic_function g(y);
dfp(f(),x,2);

F
 XX

dfp(f()*g(),x,2);

F  *G()
 XX

dfp(f()*g(),x,y);

F  *G() + F *G
 XY         X  Y
```

The difference between partial and total derivatives is illustrated by the following example:

```
generic_function h(x);
dfp(f(x,h(x))*g(h(x)),x);

F (X,H(X))*G(H(X))
 X

df(f(x,h(x))*g(h(x)),x);

F (X,H(X))*G(H(X)) + F (X,H(X))*H (X)*G(H(X))
 X                    Y          X

 + G (H(X))*H (X)*F(X,H(X))
    Y        X
```

Cooperation of partial derivatives and Taylor series under a differential side relation $\frac{dq}{dx} = f(x,q)$:

```
load_package taylor;
operator q;
let df(q(~x),x) => f(x,q(x));
taylor(q(x0+h),h,0,3);

                          F (X0,Q(X0)) + F (X0,Q(X0))*F(X0,Q(X0))
                           X              Y                          2
Q(X0) + F(X0,Q(X0))*H + ----------------------------------------*H
                                          2

+ (F  (X0,Q(X0)) + F  (X0,Q(X0))*F(X0,Q(X0))
    XX              XY

    + F (X0,Q(X0))*F (X0,Q(X0)) + F  (X0,Q(X0))*F(X0,Q(X0))
       X            Y              YX

                                  2               2                   3
    + F  (X0,Q(X0))*F(X0,Q(X0))  + F (X0,Q(X0)) *F(X0,Q(X0)))/6*H
       YY                          Y

      4
 + O(H )
```

Normally partial differentials are assumed as non-commutative

```
dfp(f(),x,y)-dfp(f(),y,x);

F    - F
 XY     YX
```

However, a generic function can be declared to have globally interchangeable partial derivatives using the declaration `DFP_COMMUTE` which takes the name of a generic function or a generic function form as argument. For such a function differentiation variables are rearranged corresponding to the sequence of the generic variables.

```
generic_function q(x,y);
dfp_commute q(x,y);
dfp(q(),{x,y,y}) + dfp(q(),{y,x,y}) + dfp(q(),{y,y,x});

3*Q
   XYY
```

If only a part of the derivatives commute, this has to be declared using the standard REDUCE rule mechanism. Please note that then the derivative variables must be written as list.

16.20.3 Substitutions

When a generic form or a `DFP` expression takes part in a substitution the following steps are performed:

1. The substitutions are performed for the arguments. If the argument list is empty the substitution is applied to the generic arguments of the function; if these change, the resulting forms are used as new actual arguments. If the generic function itself is not affected by the substitution, the process stops here.

2. If the function name or the generic function form occurs as a left hand side in the substitution list, it is replaced by the corresponding right hand side.

3. The new form is partially differentiated according to the list of partial derivative variables.

4. The (eventually modified) actual parameters are substituted into the form for their corresponding generic variables. This substitution is done by name.

Examples:

```
generic_function f(x,y);
sub(y=10,f());

F(X,10)

sub(y=10,dfp(f(),x,2));


F  (X,10)
 XX

sub(y=10,dfp(f(y,y),x,2));

F  (10,10)
 XX

sub(f=x**3*y**3,dfp(f(),x,2));

     3
6*X*Y

generic_function ff(y,z);
sub(f=ff,f(a,b));

FF(B,Z)
```

The dataset `dfpart.tst` contains more examples, including a complete application for computing the coefficient equations for Runge-Kutta ODE solvers.