If you find it awkward to evaluate Lisp forms in Pure, you can also achieve the same with the `declare` function which covers most of the common Reduce declarations that might be needed:

```
> declare operator myop;
[]
> declare odd myop;
[]
> simplify (myop (-x));
-myop x
```

This example shows how you can do a simple plot using Reduce's gnuplot module:

```
> simplify $ plot [sin x/x, x==’(-20..20), terminal=="wxt"];
0
```

This pops up a wxWidgets window (`terminal=="wxt"`) with a plot of the given function in it. The `x==’(-20..20)` argument specifies the desired range of the `x` variable (note that the range needs to be quoted so that it gets through to Reduce rather than being evaluated on the Pure side). The same plot can be written to a PostScript file sinc.ps as follows:

```
> simplify $ plot [sin x/x, x==’(-20..20), terminal=="postscript", output=="sinc.ps"];
0
```

Many more examples can be found in the reduce_examp.pure and tests.pure scripts included in the distribution.

19.5 Examples By Topic

19.5.1 Differentiation

The operator `df` is used to represent partial differentiation with respect to one or more variables.

```
syntax:  df exprn [var <num>]+.
```

Differentiation of the function $x^2y^3z^4$ with respect to x, y, z, two, three and four times respectively, i.e $\frac{\partial^9 x^2y^3z^4}{\partial x^2 \partial y^3 \partial z^4}$:

```
> simplify $ df (x^2*y^3*z^4) x 2 y 3 z 4 ;
288
```

The derivative of $\log \sin(x)^2$:

```
> simplify $ df (log(sin x)^2) x;
2*cos x*log (sin x)/sin x
```

Note the parentheses.

Suppose $z(\cos(x), y)$: Let's calculate $\frac{\partial \sin(z)}{\partial \cos(x)}$ and $\frac{\partial z^2}{\partial x}$:

```
> declare depend [z,cos x,y];
[]
> simplify (df (sin z) (cos x));
cos z*df z (cos x)
> simplify (df (z^2) x);
2*df z x*z
```

Note how to declare dependencies.

The results are $\cos(z)\frac{\partial z}{\partial \cos(x)}$ and $2z\frac{\partial z}{\partial x}$) respectively, as expected.

19.5.2 Integration

`int` is an operator in REDUCE for indefinite integration using a combination of the Risch-Norman algorithm and pattern matching.

```
syntax:  intg exprn var.
```

Note that in `Pure` the operator is called `intg` in order not to clash with the integer type `int`.

Example 1:

$$\int \frac{1}{ax+b} dx$$

```
> simplify $ intg (1/(a*x+b)) x;
log (a*x+b)/a
```

Example 2:

$$I(a,b,n) = \int x^2 (ax+b)^n dx$$

```
> I a b n = simplify $ intg (x^2*(a*x+b)^n) x;
> I a b n;
((a*x+b)^n*a^3*n^2*x^3+3*(a*x+b)^n*a^3*n*x^3+2*(a*x+b)^n*a^3*x^3+
 (a*x+b)^n*a^2*b*n^2*x^2+(a*x+b)^n*a^2*b*n*x^2-2*(a*x+b)^n*a*b^2*
  n*x+2*(a*x+b)^n*b^3)/(a^3*n^3+6*a^3*n^2+11*a^3*n+6*a^3)
> I a b 0 ;
x^3/3
> I 0 b n;
b^n*x^3/3
> I a 0 k;
x^k*a^k*x^3/(k+3)
```

Example 3:

$$\int \frac{\sqrt{x+\sqrt{x^2+1}}}{x}$$

```
> simplify $ intg (sqrt(x+sqrt(x^2+1))/x) x ;
intg (sqrt (sqrt (x^2+1)+x)/x) x
```

Apparently no solution was found. There is a package `ALGINT` in REDUCE, that is specialized to deal with algebraic functions. The [UserGuide] says

> *... will analytically integrate a wide range of expressions involving square roots where the answer exists in that class of functions. It is an implementation of the work described in J.H. Davenport* [LNCS102]

```
> reduce::load "algint" ;
0
> simplify $ intg (sqrt(x+sqrt(x^2+1))/x) x ;
atan ((sqrt (sqrt (x^2+1)+x)*sqrt (x^2+1)-sqrt (sqrt (x^2+1)+x)*x-sqrt
(sqrt (x^2+1)+x))/2)+2*sqrt (sqrt (x^2+1)+x)+log (sqrt (sqrt
(x^2+1)+x)-1)-log (sqrt (sqrt (x^2+1)+x)+1)
```

Note how to load packages.