```
   This is SBCL 1.3.4.15614.texmacs.1-0729f5c41-WIP, an implementation of ANSI
   Common Lisp.
   More information about SBCL is available at <http://www.sbcl.org/>.

   SBCL is free software, provided as is, with absolutely no warranty.
   It is mostly in the public domain; some portions are provided under
   BSD-style licenses.  See the CREDITS and COPYING files in the
   distribution for more information.

SBCL> (ql:quickload :weyl)

   To load "weyl":
     Load 1 ASDF system:
       weyl
   ; Loading "weyl"
   ..................................
   (:WEYL)

SBCL> (in-package :weyl)

   #<PACKAGE "WEYL">

SBCL> (defvar x (coerce 'x *general*))
      (defvar y (coerce 'y *general*))
      (defvar z (coerce 'z *general*))
      (defvar p (coerce 'p *general*))
      (defvar q (coerce 'q *general*))
      (defvar r (coerce 'r *general*))


   X
   Y

SBCL> (weyli::ge-variables *general*)

   (r q p z y x v.1 x)

SBCL> (defvar ge1 (deriv (expt p q) q))

   GE1

SBCL> (defvar ge2 (* x (expt y 2) (expt z 3) (sin x)))

   GE2

SBCL> ge1

   (log(p)) p^q

SBCL> ge2

   z^3 y^2 x (sin(x))

SBCL> (deriv ge2 x)

   z^3 y^2 (sin(x)) + (cos(x)) z^3 y^2 x

SBCL> (deriv ge2 x x)

   2 (cos(x)) z^3 y^2 - ((sin(x)) z^3 y^2 x)

SBCL> (deriv ge2 x y z)

   6 z^2 y (sin(x)) + 6 z^2 y (cos(x)) x

SBCL> (defun wtype (obj) (cl::type-of obj))


   WTYPE
```

```
SBCL> (defun slot-names (cls)
        (mapcar #'sb-mop::slot-definition-name
          (sb-mop:class-slots (sb-mop::find-class cls ))))

   SLOT-NAMES

SBCL> (defun slot-iargs (cls)
        (mapcar #'sb-mop::slot-definition-initargs
          (sb-mop:class-slots (sb-mop::find-class cls))))

   SLOT-IARGS

SBCL> (defun slot-info (obj &key (prt t))
        (let* ((tobj (cl-user::type-of obj))
               (sn (slot-names tobj))
               (sv (map 'list (lambda (x) (slot-value obj x)) sn))
               (sa (slot-iargs tobj)))
                  (format prt "Obj:Type : ~a : ~a ~%" obj tobj)
                  (format prt "Names ...: ~{~a~^, ~} ~%" sn)
                  (format prt "Values ..: ~{~a~^, ~} ~%" sv)
                  (format prt "InitArgs : ~{~a~^, ~} ~%~%" sa)))
   SLOT-INFO

SBCL> (slot-info p)

   Obj:Type : p : GE-VARIABLE
   Names :::: PROPERTY-LIST, DOMAIN, SIMPLIFIED?, SYMBOL, STRING
   Values ..: NIL, #<Domain: GENERAL-EXPRESSIONS>, NIL, P, p
   InitArgs : NIL, (DOMAIN), NIL, (SYMBOL), (STRING)

   NIL

SBCL> (slot-info (* p q))

   Obj:Type : q p : GE-TIMES
   Names :::: DOMAIN, SIMPLIFIED?, TERMS
   Values ..: #<Domain: GENERAL-EXPRESSIONS>, NIL, (q p)
   InitArgs : (DOMAIN), NIL, (TERMS)

   NIL

SBCL> (slot-info (expt p q))

   Obj:Type : p^q : GE-EXPT
   Names :::: DOMAIN, SIMPLIFIED?, BASE, EXP
   Values ..: #<Domain: GENERAL-EXPRESSIONS>, NIL, p, q
   InitArgs : (DOMAIN), NIL, (BASE), (EXP)

   NIL

SBCL> (slot-info (sin p) )

   Obj:Type : sin(p) : GE-APPLICATION
   Names :::: DOMAIN, SIMPLIFIED?, FUNCT, ARGS
   Values ..: #<Domain: GENERAL-EXPRESSIONS>, NIL, sin, (p)
   InitArgs : (DOMAIN), NIL, (FUNCT), (ARGS)

   NIL

SBCL> (slot-value (sin p) 'funct)

   sin
```

```
SBCL> (slot-value (sin p) 'weyli::domain)
   #<Domain: GENERAL-EXPRESSIONS>
SBCL> (slot-value (sin p) 'weyli::args)
   (p)
SBCL> (make-ge-variable *general* 'g)
   g
SBCL> (weyli::ge-variables *general*)
   (g r q p z y x v.1 x)
SBCL> (substitute p q (* p q))
   p^2
SBCL> (substitute p q (+ p q))
   2 p
SBCL> (substitute 4 q (+ p q))
   4 + p
SBCL> (substitute x  q (+ p (sin (cos q)) ))
   p + sin(cos(x))
SBCL> (ge-variable? p)
   T
SBCL> (defvar f1 (weyli::make-app-function '(x y) (+ (* 'x 'y) (* 'x 'y 'x))))
   F1
SBCL> f1
   (lambda (v.1 v.2) v.2 v.1^2 + v.2 v.1)
SBCL> (deriv f1 0)
   (lambda (v.1 v.2) 2 v.2 v.1 + v.2)
SBCL> (deriv f1 1)
   (lambda (v.1 v.2) v.1 + v.1^2)
SBCL> (cl-user::type-of f1)
   WEYLI::APPLICABLE-FUNCTION
SBCL> (apply f1 '(p q))
   q p^2 + q p
SBCL> (apply (deriv f1 0) '(p q))
   2 q p + q
SBCL> (documentation 'weyli::make-ge-variable 'function)
   "Create a variable in a domain."
SBCL> (documentation 'weyli::coerce 'function)
   "Coerce the element into the domain."
SBCL> (documentation 'weyli::expand 'function)
   "Replaces all products of sums in exp by sums of products."
SBCL> (documentation 'weyli::memoize 'function)
   "Performs the same functions as ``weyli::%memoize`` except that the domain
      used is ``*general*``."
SBCL>
```