```
   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> (declaim (sb-ext:muffle-conditions cl:warning))

   ((#<SB-KERNEL::CONDITION-CLASSOID WARNING> . MUFFLE-WARNING))
SBCL> (in-package :weyl)

   #<PACKAGE "WEYL">
SBCL> (reset-domains)

   NIL
SBCL> (ge-variable? 'x)

   NIL
SBCL> (setf ZZ (get-rational-integers))

   Z
SBCL> (setf R (get-polynomial-ring ZZ '(x1 x2 x3)))

   Z[x1, x2, x3]
SBCL> (setf x (coerce 'x1 R))
      (setf y (coerce 'x2 R))
      (setf z (coerce 'x3 R))

   x1
   x2
   x3
SBCL> (setf r1 (* (expt (- x) 3) (+ y (* 2 z))))

   (- x2 + -2 x3) x1^3
SBCL> (partial-deriv r1 x)

   (-3 x2 + -6 x3) x1^2
SBCL>
SBCL> (setf QQ (get-rational-numbers))

   Q
SBCL> (setf RR (get-real-numbers))

   R
SBCL> (setf CC (get-complex-numbers))

   C
```

```
SBCL> (setf Q4 (get-quaternion-domain QQ))
   Quat(Q)
SBCL> (setf a (coerce 1234567890 ZZ))
   1234567890
SBCL> (setf b (coerce 127654 ZZ))
   127654
SBCL> (setf c (/ a b))

   #<SIMPLE-ERROR "No applicable contagion method for ~S and ~S" {1003193BF3}>
SBCL> (setf q (coerce (/ 23 789) QQ))
   23/789
SBCL> (* a q)
   9465020490/263
SBCL> (prime? 77)
   NIL
SBCL> (factor 1234567890)
   ((2 . 1) (3 . 2) (5 . 1) (3607 . 1) (3803 . 1))
SBCL> (setq c 12345)
   12345
SBCL> (factor c)
   ((3 . 1) (5 . 1) (823 . 1))
SBCL> (factorial 32)
   263130836933693530167218012160000000
SBCL> (prime? 5678691)
   NIL
SBCL> (totient 123)
   80
SBCL> (pochhammer 123 6)
   3905000064000
SBCL> (combinations 23 12)
   1352078
SBCL> (newprime 2332)

   #<UNDEFINED-FUNCTION NEWPRIME {1003510CB3}>
SBCL> q
   23/789
SBCL> (floor q)
   0
   23
SBCL> (ceiling q)
   1
```

```
   -766
SBCL> (truncate q)
   0
   23
SBCL> (round q)
   0
   23
SBCL> (+ b (* a q))
   9498593492/263
SBCL> (setf q (coerce 'q *general*))
   q
SBCL> q
   q
SBCL> (setf p (coerce 'p *general*))
   p
SBCL> (cl-user::type-of q)
   WEYLI::GE-VARIABLE
SBCL> (ge-variable? q)
   T
SBCL> (setf ge1 (/ (* (+ 23 (* p q)) p) (- p q)))
   (-1 q + p)^-1 (23 + q p) p
SBCL> (setf ge2 (* (+ p q) (- p q)))
   (-1 q + p) (q + p)
SBCL> (simplify ge2)
   (-1 q + p) (q + p)
SBCL> (deriv ge1 'p 'q 'p)
   2 (-1 q + p)^-1 - (2 (-1 q + p)^-2 q) - (4 (-1 q + p)^-2 p) + 4 (-1 q + p)^-3
   (23 + q p) + 4 (-1 q + p)^-3 q p + 2 (-1 q + p)^-3 p^2 - (6 (-1 q + p)^-4 (23
   + q p) p)
SBCL> (expand ge2)
   -1 q^2 + p^2
SBCL> (setf ge3 (/ ge1 ge2))
   (-1 q + p)^-1 (23 + q p) p ((-1 q + p) (q + p))^-1
SBCL> (simplify ge3)
   (-1 q + p)^-1 (23 + q p) p ((-1 q + p) (q + p))^-1
SBCL> (expand ge3)
   0
SBCL> (simplify (deriv ge1 'p 'q 'p))
   2 (-1 q + p)^-1 - (2 (-1 q + p)^-2 q) - (4 (-1 q + p)^-2 p) + 4 (-1 q + p)^-3
   (23 + q p) + 4 (-1 q + p)^-3 q p + 2 (-1 q + p)^-3 p^2 - (6 (-1 q + p)^-4 (23
   + q p) p)
```

```
SBCL> (expand (deriv ge1 'p 'q 'p))
   0
SBCL> (simplify (/ p p))
   1
SBCL> (simplify (/ ge2 ge2))
   (-1 q + p) (q + p) ((-1 q + p) (q + p))^-1
SBCL> (expand (/ ge2 ge2))
   0
SBCL> (reset-domains)
      (setf R *general*)
   NIL
   #<Domain: GENERAL-EXPRESSIONS>
SBCL> (setf u (coerce 'u R))
      (setf v (coerce 'v R))
      (setf w (coerce 'w R))
   u
   v
   w
SBCL> (ge-variable? v)
   T
SBCL> (add-subscripts u '(1 2 'a))
   u((1 2 'A))
SBCL> (get-variable-property R u 'key)
   NIL
SBCL> (declare-dependencies u v w)
   (w v)
SBCL> (depends-on? u v)
   T
SBCL> (setf x (+ 2 (* u v)))
   2 + v u
SBCL> (different-kernels x '(v))
   (u v V)
SBCL> (different-kernels x (list v))
   (u v)
SBCL> (setf xx (simplify (- x x)))
   2 - (2 + v u) + v u
SBCL> (simplify xx)
   2 - (2 + v u) + v u
SBCL> (expand xx)
   0
SBCL> (deriv x 'v)
   u
```

```
SBCL> (deriv (* x x) 'v)

  2 (2 + v u) u

SBCL> (expand (deriv (* x x) 'v))

  2 v u^2 + 4 u

SBCL> (expand xx)

  0

SBCL> (let ((count 0)) (permute '(a b c d) (p) (print p) (incf count))
          (format t "~%~D permutations total. ~%" count))

  (D C B A)
  (C D B A)
  (D B C A)
  (B D C A)
  (C B D A)
  (B C D A)
  (D C A B)
  (C D A B)
  (D A C B)
  (A D C B)
  (C A D B)
  (A C D B)
  (D B A C)
  (B D A C)
  (D A B C)
  (A D B C)
  (B A D C)
  (A B D C)
  (C B A D)
  (B C A D)
  (C A B D)
  (A C B D)
  (B A C D)
  (A B C D)
  24 permutations total.
  NIL

SBCL> (weyli:partition (l 10) (print l))


  (1 1 1 1 1 1 1 1 1 1)
  (2 1 1 1 1 1 1 1 1)
  (3 1 1 1 1 1 1 1)
  (2 2 1 1 1 1 1 1)
  (4 1 1 1 1 1 1)
  (3 2 1 1 1 1 1)
  (5 1 1 1 1 1)
  (2 2 2 1 1 1 1)
  (4 2 1 1 1 1)
  (3 3 1 1 1 1)
  (6 1 1 1 1)
  (3 2 2 1 1 1)
  (5 2 1 1 1)
  (4 3 1 1 1)
  (7 1 1 1)
  (2 2 2 2 1 1)
```

```
   (4 2 2 1 1)
   (3 3 2 1 1)
   (6 2 1 1)
   (5 3 1 1)
   (4 4 1 1)
   (8 1 1)
   (3 2 2 2 1)
   (5 2 2 1)
   (4 3 2 1)
   (7 2 1)
   (3 3 3 1)
   (6 3 1)
   (5 4 1)
   (9 1)
   (2 2 2 2 2)
   (4 2 2 2)
   (3 3 2 2)
   (6 2 2)
   (5 3 2)
   (4 4 2)
   (8 2)
   (4 3 3)
   (7 3)
   (6 4)
   (5 5)
   (10)
SBCL> (require :sb-introspect)
   ("SB-INTROSPECT")
SBCL> (sb-introspect::who-calls  'weyli:ge-variable? )
   ((WEYLI::STANDARD-DERIVATION
               . #S(SB-INTROSPECT:DEFINITION-SOURCE
                    :PATHNAME #P"/home/kfp/quicklisp/local-projects/weyl/
   differential-domains.lisp"
                    :FORM-PATH (12)
                    :FORM-NUMBER 20
                    :CHARACTER-OFFSET 5097
                    :FILE-WRITE-DATE 3938251414
                    :PLIST NIL
                    :DESCRIPTION NIL))
              (WEYLI::STANDARD-DERIVATION
               . #S(SB-INTROSPECT:DEFINITION-SOURCE
                    :PATHNAME #P"/home/kfp/quicklisp/local-projects/weyl/
   differential-domains.lisp"
                    :FORM-PATH (12)
                    :FORM-NUMBER 33
                    :CHARACTER-OFFSET 5097
                    :FILE-WRITE-DATE 3938251414
                    :PLIST NIL
                    :DESCRIPTION NIL))
              (WEYLI::SAFE-DISPLAY
               . #S(SB-INTROSPECT:DEFINITION-SOURCE
                    :PATHNAME #P"/home/kfp/quicklisp/local-projects/weyl/
   general.lisp"
                    :FORM-PATH (100)
```

```
                    :FORM-NUMBER 14
                    :CHARACTER-OFFSET 20112
                    :FILE-WRITE-DATE 3938251414
                    :PLIST NIL
                    :DESCRIPTION NIL)))
SBCL>  (sb-introspect::function-lambda-list  'weyli:deriv)
   (WEYLI::EXPRESSION &REST WEYLI::VARIABLES)
SBCL>  (sb-introspect::function-type   'weyli:deriv)
   (FUNCTION (T &REST T) COMMON-LISP:*)
SBCL>  (sb-introspect::function-type   'weyli:expand)
   (FUNCTION (T) COMMON-LISP:*)
SBCL>  (sb-introspect::find-function-callers  'weyli:expand)
   (#<FUNCTION MAKE-APP-FUNCTION>
               #<FUNCTION (SB-PCL::FAST-METHOD EXPAND (WEYLI::GE-EXPT))>
               #<FUNCTION (SB-PCL::FAST-METHOD EXPAND (WEYLI::GE-PLUS))>
               #<FUNCTION WEYLI::EXPAND-PRODUCT1>)
SBCL> (cl-user::quit)
   Busy...
SBCL>
```