# Package ‘rcdd’

November 18, 2021

**Version** 1.5

**Date** 2021-11-17

**Title** Computational Geometry

**Author** Charles J. Geyer <charlie@stat.umn.edu> and
Glen D. Meeden <glen@stat.umn.edu>, incorporates code from
cddlib (ver 0.94f) written by Komei Fukuda <fukuda@ifor.math.ethz.ch>

**Maintainer** Charles J. Geyer <charlie@stat.umn.edu>

**Depends** R (>= 3.0.2)

**Imports** methods

**SystemRequirements** GMP (GNU MP bignum library from
<https://gmplib.org/>)

**Description** R interface to (some of) cddlib
(<https://github.com/cddlib/cddlib>).
Converts back and forth between two representations of a convex polytope:
as solution of a set of linear equalities and inequalities and as
convex hull of set of points and rays.
Also does linear programming and redundant generator elimination
(for example, convex hull in n dimensions). All functions can use exact
infinite-precision rational arithmetic.

**License** GPL-2

**URL** https://www.stat.umn.edu/geyer/rcdd/,
https://github.com/cjgeyer/rcdd

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2021-11-18 06:20:02 UTC

## R topics documented:

---

`allfaces` *All Faces of a Convex Polyhedron*

---

## Description

List all the nonempty faces of a convex polyhedron, giving for each the dimension, the active set of constraints, and a relative interior point.

## Usage

```
allfaces(hrep)
```

## Arguments

`hrep` H-representation of convex polyhedron (see details).

## Details

See `cddlibman.pdf` in the `doc` directory of this package, especially Sections 1 and 2.

This function lists all nonempty faces of a convex polyhedron given by the H-representation given by the matrix `hrep`. Let

```
l <- hrep[ , 1]
b <- hrep[ , 2]
v <- hrep[ , - c(1, 2)]
a <- (- v)
```

Then the convex polyhedron in question is the set of points `x` satisfying

```
axb <- a %*% x - b
all(axb <= 0)
all(l * axb == 0)
```

A nonempty *face* of a convex polyhedron $P$ is the subset of $P$ that is the set of points over which some linear function achieves its maximum over $P$. Note that $P$ is a face of $P$ and appears in the list of faces. By definition the empty set is also a face, but is not listed. These two faces are said to be *improper*, the other faces are *proper*.

A face in the listing is characterized by the set of constraints that are *active*, i. e., satisfied with equality, on the face.

The *relative interior* of a convex set its its interior considered as a subset of its affine hull. The relative interior of a one-point set is that point. The relative interior of a multi-point convex set is the union of open line segments $(x, y)$ with endpoints $x$ and $y$ in the set.

## Value

a list containing the following components:

`dimension` list of integers giving the dimensions of the faces.

`active.set` list of integer vectors giving for each face the set of constraints that are active (satisfied with equality) on the face, the integers referring to row numbers of `hrep`.

`relative.interior.point`
list of double or character vectors (same type as `hrep`) giving a point in the relative interior of each face.

## Rational Arithmetic

The argument `hrep` may have type `"character"` in which case its elements are interpreted as unlimited precision rational numbers. They consist of an optional minus sign, a string of digits of any length (the numerator), a slash, and another string of digits of any length (the denominator). The denominator must be positive. If the denominator is one, the slash and the denominator may be omitted. This package provides several functions (see ConvertGMP and ArithmeticGMP) for conversion back and forth between R floating point numbers and rationals and for arithmetic on GMP rationals.

## Warning

If you want correct answers, use rational arithmetic. If you do not, this function may (1) produce approximately correct answers, (2) fail with an error, (3) give answers that are nowhere near correct with no error or warning, or (4) crash R losing all work done to that point. In large simulations (1) is most frequent, (2) occurs roughly one time in a thousand, (3) occurs roughly one time in ten thousand, and (4) has only occurred once and only with the `redundant` function. So the R floating point arithmetic version does mostly work, but you cannot trust its results unless you can check them independently.

## See Also

`scdd`, `ArithmeticGMP`, `ConvertGMP`

### Examples

```
hrep <- rbind(c(0, 1,  1,  1, -1),
              c(0, 1,  1, -1, -1),
              c(0, 1, -1, -1, -1),
              c(0, 1, -1,  1, -1),
              c(0, 0,  0,  0,  1))

allfaces(d2q(hrep))
```

---

ArithmeticGMP *GMP Rational Arithmetic*

---

### Description

Add, subtract, multiply, or divide one object to/from/by another using GMP (GNU multiple precision) rational arithmetic. Any size integers in the numerator and denominator are allowed.

### Usage

```
qpq(x, y)
qmq(x, y)
qxq(x, y)
qdq(x, y)
qmatmult(x, y)
qsum(x)
qprod(x)
qmax(x)
qmin(x)
qsign(x)
qneg(x)
qabs(x)
qinv(x)
```

### Arguments

| | |
|---|---|
| `x,y` | objects of type `"numeric"` or `"character"`. If `"numeric"` are converted to rational using `d2q`. Objects must have the same length. |

### Details

`qpq` is “plus”, `qmq` is “minus”, `qxq` is “times”, `qdq` is “divide”. Divide by zero is an error. There are no rational NA, NaN, Inf. `qsum` is vectorizing summation like `sum` for ordinary numeric. `qprod` is vectorizing product like `prod` for ordinary numeric. `qmax` is like `max` for ordinary numeric. `qmin` is like `min` for ordinary numeric. `qsign` is vectorizing sign like `sign` for ordinary numeric. `qmatmult` is matrix multiplication like `%*%` for ordinary numeric; both arguments must be matrices. `qneg` is vectorizing negation like unary minus for ordinary numeric. `qabs` is vectorizing absolute value like `abs` for ordinary numeric. `qinv` is vectorizing inversion like `1 / x` for ordinary numeric.

### Value

an object of the same form as x that is the sum, difference, product, quotient, or sign or (for qsum and qprod) a scalar that is the sum or product.

### See Also

ConvertGMP

### Examples

```
qmq("1/3", "1/2")
# note inexactness of floating point representations
qmq("1/5", 1/5)
qdq("1/5", 1/5)
qsum(c("1", "1/2", "1/4", "1/8"))
qprod(c("1", "1/2", "1/4", "1/8"))
qmax(c("-1", "1/2", "1/-4", "1/8"))
qmin(c("-1", "1/2", "1/-4", "1/8"))
qsign(c("-1", "1/2", "1/-4", "1/8"))
qmatmult(matrix(c("1", "2", "3", "4"), 2, 2),
    matrix(c("1/1", "1/2", "1/3", "1/4"), 2, 2))
qneg(seq(-3, 3))
```

---

## ConvertGMP *Convert Between Real, Integer, and GMP Rational*

---

### Description

Converts to and from GMP (GNU multiple precision) rational numbers. Any size integers in the numerator and denominator are allowed.

### Usage

```
d2q(x)
q2d(x)
q2q(x)
z2q(numer, denom, canonicalize = TRUE)
```

### Arguments

| | |
|---|---|
| x,numer,denom | objects of type "numeric" or "character". |
| canonicalize | if TRUE (the default) canonicalize (see below). |

### Value

d2q converts from real to rational, q2d converts from rational to real, q2q canonicalizes (no common factors in numerator and denominator) rationals, z2q converts integer numerator and denominator to rational canonicalizing if canonicalize = TRUE (the default).

### See Also

ArithmeticGMP

### Examples

```
d2q(runif(1))
q2d("-1234567891234567899876543210/33")
z2q(44, 11)
```

## linearity — *Find implicit linearities in H-representation and V-representation of convex polyhedron*

### Description

Given V-representation (convex hull of points and directions) or H-representation (intersection of half spaces) of convex polyhedron find non-linearity generators that can be made linearity without changing polyhedron

### Usage

```
linearity(input, representation = c("H", "V"))
```

### Arguments

| | |
|---|---|
| `input` | either H-representation or V-representation of convex polyhedron (see details). |
| `representation` | if `"H"`, then `input` is an H-representation, otherwise a V-representation. May also be obtained from a `"representation"` attribute of `input`, if present. |

### Details

Interface to the function `dd_ImpliedLinearityRows` of the `cddlib` library, see `cddlibman.pdf` in the `doc` directory of this package, especially Sections 1 and 2 and page 9. See also `scdd` for a description of the way this package codes H-representations and V-representations as R matrices.

A row of a matrix that is an H-representation or V-representation is a linearity row if the first element of that row is 1. The row is an implied linearity row if the first element of that row is 0 but if it were 1 the convex polyhedron described would be unchanged.

The interpretation is as follows. For an H-representation, the linearity (given plus implied) determines the affine hull of the polyhedron (the smallest translate of a subspace containing it). For a V-representation, the linearity (given plus implied) determines the smallest affine set (translate of a subspace) contained in the polyhedron.

### Value

a numeric vector, the indices of the implied linearity rows. (Note: rows that are linearity rows in the input matrix are not contained in this vector.)

## Rational Arithmetic

The input representation may have type "character" in which case its elements are interpreted as unlimited precision rational numbers. They consist of an optional minus sign, a string of digits of any length (the numerator), a slash, and another string of digits of any length (the denominator). The denominator must be positive. If the denominator is one, the slash and the denominator may be omitted. This package provides several functions (see ConvertGMP and ArithmeticGMP) for conversion back and forth between R floating point numbers and rationals and for arithmetic on GMP rationals.

## Warning

If you want correct answers, use rational arithmetic. If you do not, this function may (1) produce approximately correct answers, (2) fail with an error, (3) give answers that are nowhere near correct with no error or warning, or (4) crash R losing all work done to that point. In large simulations (1) is most frequent, (2) occurs roughly one time in a thousand, (3) occurs roughly one time in ten thousand, and (4) has only occurred once and only with the `redundant` function. So the R floating point arithmetic version does mostly work, but you cannot trust its results unless you can check them independently.

## See Also

`ArithmeticGMP`, `ConvertGMP`, `validcdd`, `makeH`

## Examples

```
### calculate affine hull
### determined by given + implied linearity rows
qux <- rbind(c(0, 2, 0, 0, 1),
             c(0, 3, 1, 0, 0),
             c(0, 4, 0, 1, 0),
             c(0, -7, -1, -1, 0))
out <- linearity(qux, representation = "H")
print(out)
qux[out, 1] <- 1
redundant(qux, representation = "H")$output

### calculate minimal nonempty face of polyhedral convex cone
### determined by given + implied linearity rows
qux <- rbind(c(0, 0, 0, 0, 1),
             c(0, 0, 1, 0, 0),
             c(0, 0, 0, 1, 0),
             c(0, 0, -1, -1, 0))
out <- linearity(qux, representation = "V")
print(out)
redundant(qux, representation = "V")$output
```

lpcdd *linear programming with exact arithmetic*

### Description

Solve linear program or explain why it has no solution.

### Usage

```
lpcdd(hrep, objgrd, objcon, minimize = TRUE,
    solver = c("DualSimplex", "CrissCross"))
```

### Arguments

| | |
|---|---|
| hrep | H-representation of convex polyhedron (see details) over which an affine function is maximized or minimized. |
| objgrd | gradient vector of affine function. |
| objcon | constant term of affine function. May be missing, in which case, taken to be zero. |
| minimize | minimize if TRUE, otherwise maximize. |
| solver | type of solver. Use the default unless you know better. |

### Details

See cddlibman.pdf in the doc directory of this package, especially Sections 1 and 2 and the documentation of the function dd_LPSolve in Section 4.2.

This function minimizes or maximizes an affine function x maps to sum(objgrd * x) + objcon over a convex polyhedron given by the H-representation given by the matrix hrep. Let

```
l <- hrep[ , 1]
b <- hrep[ , 2]
v <- hrep[ , - c(1, 2)]
a <- (- v)
```

Then the convex polyhedron in question is the set of points x satisfying

```
axb <- a %*% x - b
all(axb <= 0)
all(l * axb == 0)
```

### Value

a list containing some of the following components:

solution.type
: character string describing the solution type. "Optimal" indicates the optimum is achieved. "Inconsistent" indicates the feasible region is empty (no points satisfy the constraints, the polyhedron specified by hrep is empty). "DualInconsistent" or "StrucDualInconsistent" indicates the feasible region is unbounded and the objective function is unbounded below when minimize = TRUE or above when minimize = FALSE.

primal.solution
: Returned only when solution.type = "Optimal", the solution to the stated (primal) problem.

dual.solution
: Returned only when solution.type = "Optimal", the solution to the dual problem, Lagrange multipliers for the primal problem.

dual.direction
: Returned only when solution.type = "Inconsistent", coefficients of a linear combination of original inequalities and equalities that proves the inconsistency. Coefficients for original inequalities are nonnegative.

primal.direction
: Returned only when solution.type = "DualInconsistent" or solution.type = "StrucDualInconsistent", coefficients of the linear combination of columns that proves the dual inconsistency, also an unbounded direction for the primal LP.

### Rational Arithmetic

The arguments hrep, objgrd, and objcon may have type "character" in which case their elements are interpreted as unlimited precision rational numbers. They consist of an optional minus sign, a string of digits of any length (the numerator), a slash, and another string of digits of any length (the denominator). The denominator must be positive. If the denominator is one, the slash and the denominator may be omitted. This package provides several functions (see ConvertGMP and ArithmeticGMP) for conversion back and forth between R floating point numbers and rationals and for arithmetic on GMP rationals.

### Warning

If you want correct answers, use rational arithmetic. If you do not, this function may (1) produce approximately correct answers, (2) fail with an error, (3) give answers that are nowhere near correct with no error or warning, or (4) crash R losing all work done to that point. In large simulations (1) is most frequent, (2) occurs roughly one time in a thousand, (3) occurs roughly one time in ten thousand, and (4) has only occurred once and only with the redundant function. So the R floating point arithmetic version does mostly work, but you cannot trust its results unless you can check them independently.

### See Also

scdd, ArithmeticGMP, ConvertGMP

## Examples

```
# first two rows are inequalities, second two equalities
hrep <- rbind(c("0", "0", "1", "1", "0", "0"),
              c("0", "0", "0", "2", "0", "0"),
              c("1", "3", "0", "-1", "0", "0"),
              c("1", "9/2", "0", "0", "-1", "-1"))
a <- c("2", "3/5", "0", "0")
lpcdd(hrep, a)

# primal inconsistent problem
hrep <- rbind(c("0", "0", "1", "0"),
              c("0", "0", "0", "1"),
              c("0", "-2", "-1", "-1"))
a <- c("1", "1")
lpcdd(hrep, a)

# dual inconsistent problem
hrep <- rbind(c("0", "0", "1", "0"),
              c("0", "0", "0", "1"))
a <- c("1", "1")
lpcdd(hrep, a, minimize = FALSE)
```

---

makeH *make H-representation of convex polyhedron*

---

## Description

Construct H-representation of convex polyhedron, set of points x satisfying

```
a1 %*% x <= b1
a2 %*% x == b2
```

see scdd for description of valid representations.

## Usage

```
makeH(a1, b1, a2, b2, x = NULL)
addHeq(a, b, x)
addHin(a, b, x)
```

## Arguments

| | |
|---|---|
| a1 | numerical or character matrix for inequality constraints. If vector, treated as matrix with one row. |
| b1 | numerical or character right hand side vector for inequality constraints. |
| a2 | numerical or character matrix for equality constraints. If vector, treated as matrix with one row. |

- `b2` — numerical or character right hand side vector for equality constraints.
- `x` — if not `NULL`, a valid H-representation.
- `a` — numerical or character matrix for constraints. If vector, treated as matrix with one row. Constraints are equality in `addHeq` and inequality in `addHin`.
- `b` — numerical or character right hand side vector for constraints.

  Arguments `a1`, `b1`, `a2`, and `b2` may be missing, but must be missing in pairs. Rows in `x`, if any, are added to new rows corresponding to the constraints given by the other arguments.

## Value

a valid H-representation that can be handed to `scdd`.

## Rational Arithmetic

The input representation may have type `"character"` in which case its elements are interpreted as unlimited precision rational numbers. They consist of an optional minus sign, a string of digits of any length (the numerator), a slash, and another string of digits of any length (the denominator). The denominator must be positive. If the denominator is one, the slash and the denominator may be omitted. This package provides several functions (see ConvertGMP and ArithmeticGMP) for conversion back and forth between R floating point numbers and rationals and for arithmetic on GMP rationals.

Arguments may be a mix of numeric and character in which case all are converted to GMP rationals (character) and the output is GMP rational.

## See Also

`scdd`, `validcdd`

## Examples

```
d <- 4
# unit simplex in H-representation
qux <- makeH(- diag(d), rep(0, d), rep(1, d), 1)
print(qux)
# add an inequality constraint
qux <- addHin(c(1, -1, 0, 0), 0, qux)
print(qux)
# drop a constraint
qux <- qux[- 3, ]
print(qux)
```

makeV *make V-representation of convex polyhedron*

## Description

Construct V-representation of convex polyhedron. See scdd for description of valid representations.

## Usage

```
makeV(points, rays, lines, x = NULL)
addVpoints(points, x)
addVrays(rays, x)
addVlines(lines, x)
```

## Arguments

| | |
|---|---|
| points | numerical or character matrix for points. If vector, treated as matrix with one row. Each row is one point. |
| rays | numerical or character matrix for points. If vector, treated as matrix with one row. Each row represents one ray consisting of all nonnegative multiples of the vector which is the row. |
| lines | numerical or character matrix for points. If vector, treated as matrix with one row. Each row represents one line consisting of all scalar multiples of the vector which is the row. |
| x | if not `NULL`, a valid V-representation. |

## Details

In `makeV` the arguments `points` and `rays` and `lines` may be missing.

## Value

a valid V-representation that can be handed to `scdd`.

## Rational Arithmetic

The input representation may have type `"character"` in which case its elements are interpreted as unlimited precision rational numbers. They consist of an optional minus sign, a string of digits of any length (the numerator), a slash, and another string of digits of any length (the denominator). The denominator must be positive. If the denominator is one, the slash and the denominator may be omitted. This package provides several functions (see ConvertGMP and ArithmeticGMP) for conversion back and forth between R floating point numbers and rationals and for arithmetic on GMP rationals.

Arguments may be a mix of numeric and character in which case all are converted to GMP rationals (character) and the output is GMP rational.

### See Also

scdd, validcdd

### Examples

```
d <- 4
n <- 7
qux <- makeV(points = matrix(rnorm(n * d), ncol = d))
out <- scdd(qux)
out$output
```

---

## qgram *GMP Rational Gram-Schmidt*

---

### Description

Find Orthogonal Basis

### Usage

```
qgram(x, remove.zero.vectors = TRUE)
```

### Arguments

| | |
|---|---|
| `x` | matrix of type `"numeric"` or `"character"`. If `"numeric"` are converted to rational using `d2q`. Columns are considered vectors in space of dimension `nrow(x)`. |
| `remove.zero.vectors` | logical. |

### Value

If `remove.zero.vectors == FALSE`, a matrix of the same dimensions as `x` whose columns are orthogonal and span the same vector subspace as the columns of `x`. Since making the columns unit vectors in the L2 sense could require irrational numbers, the columns are made unit vectors in the L1 sense unless they are zero vectors (which, of course, cannot be normalized).

If `remove.zero.vectors == TRUE`, then the result is the same except zero vectors are removed, so the columns of the result form a basis of the subspace.

### See Also

ConvertGMP

### Examples

```
foo <- cbind(c("1", "1", "0", "0", "0"),
             c("2", "1", "0", "0", "0"),
             c("3", "1", "0", "0", "0"),
             c("1", "2", "3", "4", "5"))
qgram(foo)
qgram(foo, remo = FALSE)
```

---

## `redundant` *Eliminate redundant rows of H-representation and V-representation*

---

### Description

Eliminate redundant rows from H-representation (intersection of half spaces) or V-representation (convex hull of points and directions) of convex polytope.

### Usage

```
redundant(input, representation = c("H", "V"))
```

### Arguments

| | |
|---|---|
| `input` | either H-representation or V-representation of convex polyhedron (see details). |
| `representation` | if `"H"`, then `input` is an H-representation, otherwise a V-representation. May also be obtained from a `"representation"` attribute of `input`, if present. |

### Details

See `cddlibman.pdf` in the `doc` directory of this package, especially Sections 1 and 2.

Both representations are (in R) matrices, the first two columns are special. Let `foo` be either an H-representation or a V-representation and

```
l <- foo[ , 1]
b <- foo[ , 2]
v <- foo[ , - c(1, 2)]
a <- (- v)
```

In the H-representation the convex polyhedron in question is the set of points `x` satisfying

```
axb <- a %*% x - b
all(axb <= 0)
all(l * axb == 0)
```

In the V-representation the convex polyhedron in question is the set of points `x` for which there exists a `lambda` such that

```
x <- t(lambda) %*% v
```

where `lambda` satisfies the constraints

```
all(lambda * (1 - l) >= 0)
sum(b * lambda) == max(b)
```

An H-representation or V-representation object can be checked for validity using the function `validcdd`.

### Value

a list containing some of the following components:

- `output` — The input matrix with redundant rows removed.
- `implied.linearity` — For an H-representation, row numbers of inequality constraint rows that together imply equality constraints. For a V-representation, row numbers of rays that together imply lines.
- `redundant` — Row numbers of redundant rows. Note: this is set `redset` output by the `dd_MatrixCanonicalize` function in `cddlib`. It apparently does not consider all rows it deletes “redundant”. Redundancy can also be determined from the following component.
- `new.position` — Integer vector of length `nrow(input)`. Says for each input row which output row it becomes or zero to indicate redundant.

### Rational Arithmetic

The input representation may have type `"character"` in which case its elements are interpreted as unlimited precision rational numbers. They consist of an optional minus sign, a string of digits of any length (the numerator), a slash, and another string of digits of any length (the denominator). The denominator must be positive. If the denominator is one, the slash and the denominator may be omitted. This package provides several functions (see ConvertGMP and ArithmeticGMP) for conversion back and forth between R floating point numbers and rationals and for arithmetic on GMP rationals.

### Warning

If you want correct answers, use rational arithmetic. If you do not, this function may (1) produce approximately correct answers, (2) fail with an error, (3) give answers that are nowhere near correct with no error or warning, or (4) crash R losing all work done to that point. In large simulations (1) is most frequent, (2) occurs roughly one time in a thousand, (3) occurs roughly one time in ten thousand, and (4) has only occurred once. So the R floating point arithmetic version does mostly work, but you cannot trust its results unless you can check them independently.

### See Also

`ArithmeticGMP`, `ConvertGMP`, `validcdd`, `makeH`

## Examples

```
hrep <- rbind(c(0, 0,  1,  1,  0),
              c(0, 0, -1,  0,  0),
              c(0, 0,  0, -1,  0),
              c(0, 0,  0,  0, -1),
              c(0, 0, -1, -1, -1))

redundant(d2q(hrep), representation = "H")

foo <- c(1, 0, -1)
hrep <- cbind(0, 1, rep(foo, each = 9), rep(foo, each = 3), foo)
print(hrep)
redundant(d2q(hrep), representation = "V")
```

---

`scdd` *Go between H-representation and V-representation of convex polyhedron*

---

## Description

Calculate V-representation (convex hull of points and directions) of convex polytope given H-representation (intersection of half spaces) or vice versa.

## Usage

```
scdd(input, adjacency = FALSE, inputadjacency = FALSE,
    incidence = FALSE, inputincidence = FALSE, roworder = c("lexmin",
    "maxindex", "minindex", "mincutoff", "maxcutoff", "mixcutoff", "lexmax",
    "randomrow"), keepinput = c("maybe", "TRUE", "FALSE"),
    representation = c("H", "V"))
```

## Arguments

| | |
|---|---|
| `input` | either H-representation or V-representation of convex polyhedron (see details). |
| `adjacency` | if `TRUE` produce adjacency list of output generators. |
| `inputadjacency` | if `TRUE` produce adjacency list of input generators. |
| `incidence` | if `TRUE` produce incidence list of output generators with respect to input generators. |
| `inputincidence` | if `TRUE` produce incidence list of input generators with respect to output generators. |
| `roworder` | during the computation, take input rows in the specified order. The default `"lexmin"` is usually o. k. The option `"maxcutoff"` might be efficient if the input contains many redundant inequalities or generators. |
| `keepinput` | if `"TRUE"` or `"maybe"` and an adjacency or incidence list involving the input is requested, save the input. |
| `representation` | if `"H"`, then `input` is an H-representation, otherwise a V-representation. May also be obtained from a `"representation"` attribute of `input`, if present. |

## Details

See `cddlibman.pdf` in the `doc` directory of this package, especially Sections 1 and 2.

Both representations are (in R) matrices, the first two columns are special. Let `foo` be either an H-representation or a V-representation and

```
l <- foo[ , 1]
b <- foo[ , 2]
v <- foo[ , - c(1, 2)]
a <- (- v)
```

In the H-representation the convex polyhedron in question is the set of points `x` satisfying

```
axb <- a %*% x - b
all(axb <= 0)
all(l * axb == 0)
```

In the V-representation the convex polyhedron in question is the set of points `x` for which there exists a `lambda` such that

```
x <- t(lambda) %*% v
```

where `lambda` satisfies the constraints

```
all(lambda * (1 - l) >= 0)
sum(b * lambda) == max(b)
```

An H-representation or V-representation object can be checked for validity using the function `validcdd`.

## Value

a list containing some of the following components:

| | |
|---|---|
| `output` | An H-representation if input was V-representation and vice versa. |
| `input` | The argument `input`, if requested. |
| `adjacency` | The adjacency list, if requested. |
| `inputadjacency` | The input adjacency list, if requested. |
| `incidence` | The incidence list, if requested. |
| `inputincidence` | The input incidence list, if requested. |

### Rational Arithmetic

The input representation may have type "character" in which case its elements are interpreted as unlimited precision rational numbers. They consist of an optional minus sign, a string of digits of any length (the numerator), a slash, and another string of digits of any length (the denominator). The denominator must be positive. If the denominator is one, the slash and the denominator may be omitted. This package provides several functions (see ConvertGMP and ArithmeticGMP) for conversion back and forth between R floating point numbers and rationals and for arithmetic on GMP rationals.

### Warning

If you want correct answers, use rational arithmetic. If you do not, this function may (1) produce approximately correct answers, (2) fail with an error, (3) give answers that are nowhere near correct with no error or warning, or (4) crash R losing all work done to that point. In large simulations (1) is most frequent, (2) occurs roughly one time in a thousand, (3) occurs roughly one time in ten thousand, and (4) has only occurred once and only with the `redundant` function. So the R floating point arithmetic version does mostly work, but you cannot trust its results unless you can check them independently.

### See Also

`ArithmeticGMP`, `ConvertGMP`, `validcdd`, `makeH`

### Examples

```
d <- 4
# unit simplex in H-representation
qux <- makeH(- diag(d), rep(0, d), rep(1, d), 1)
print(qux)
# unit simplex in V-representation
out <- scdd(qux)
print(out)
# unit simplex in H-representation
# note: different from original, but equivalent
out <- scdd(out$output)
print(out)

# add equality constraint
quux <- addHeq(1:d, (d + 1) / 2, qux)
print(quux)
out <- scdd(quux)
print(out)

# use some options
out <- scdd(quux, roworder = "maxcutoff", adjacency = TRUE)
print(out)

# convex hull
# not the efficient way to do convex hull
# see ?redundant and sections 5.4 and 6.2 of the package vignette
np <- 50
```

```
x <- matrix(rnorm(d * np), ncol = d)
foo <- cbind(0, cbind(1, x))
out <- scdd(d2q(foo), inputincidence = TRUE, representation = "V")
boundies <- sapply(out$inputincidence, length) > 0
sum(boundies) ## number of points on boundary of convex hull
```

---

Subset *subsets and maximal sets*

---

## Description

Given a list of positive integer vectors representing sets, return a vector of all pairwise intersections (`allIntersect`), return a vector of all pairwise unions (`allUnion`), or a vector indicating the sets that are maximal in the sense of not being a subset of any other set in the list (`maximal`). If the list contains duplicate sets, at most one of each class of duplicates is declared maximal.

## Usage

```
allIntersect(sets, pow2)
allUnion(sets, pow2)
maximal(sets, pow2)
```

## Arguments

| | |
|---|---|
| `sets` | a list of vectors of `storage.mode "integer"`. (Unlike most R functions we do not coerce real to integer.) |
| `pow2` | use hash table of size `2^pow2`. May be missing. |

## Value

For `allIntersect` or `allUnion` a list of length `choose(length(sets),2)` giving all pairwise intersections (resp. unions) of elements of `sets`. For `maximal` a logical vector of the same length as `sets` indicating the maximal elements.

Note: `allIntersect` and `allUnion` run over the pairs in the same order so they can be matched up.

## Note

The functions `allIntersect` and `allUnion` were called `all.intersect` and `all.union` in previous versions of this package. The names were changed because the `all` function was made generic and these function are not methods of that one. These functions were originally intended to be used to find the faces of a convex set using the output of `scdd` but now the `allfaces` function does a better job and does it much more efficiently. Hence these functions have no known use, but have not been deleted for reasons of backwards compatibility.

`validcdd` *validate an H-representation or V-representation of convex polyhedron*

## Description

Validate an H-representation or V-representation of convex polyhedron, see `scdd` for description of valid representations.

## Usage

```
validcdd(x, representation = c("H", "V"))
```

## Arguments

| | |
|---|---|
| `x` | an H-representation or V-representation to be validated. |
| `representation` | if `"H"`, validate `x` as an H-representation, otherwise as a V-representation. May also be obtained from a `"representation"` attribute of `x`, if present. |

## Value

always `TRUE`. Fails with error message if not a valid object.

## See Also

`scdd`

# Index