Package 'freealg'

September 23, 2019

Type Package

Title The Free Algebra

Version 1.0-0

Maintainer Robin K. S. Hankin <hankin.robin@gmail.com>

Description The free algebra in R; multivariate polynomials with non-commuting indeterminates.

License GPL (>= 2)

Imports Rcpp (>= 0.12.3)

LinkingTo Rcpp

SystemRequirements C++11

Suggests knitr,testthat

VignetteBuilder knitr

URL https://github.com/RobinHankin/freealg.git

BugReports https://github.com/RobinHankin/freealg/issues

NeedsCompilation yes

Author Robin K. S. Hankin [aut, cre] (<https://orcid.org/0000-0001-5982-0415>)

Repository CRAN

Date/Publication 2019-09-23 14:50:02 UTC

R topics documented:

freealg-package *The Free Algebra*

Description

The free algebra in R; multivariate polynomials with non-commuting indeterminates.

Details

The DESCRIPTION file:

Package:	freealg
Type:	Package
Title:	The Free Algebra
Version:	1.0-0
Authors@R:	person(given=c("Robin", "K. S."), family="Hankin", role = c("aut","cre"), email="hankin.robin@gma
Maintainer:	Robin K. S. Hankin <hankin.robin@gmail.com>
Description:	The free algebra in R; multivariate polynomials with non-commuting indeterminates.
License:	GPL (>= 2)
Imports:	Rcpp (>= 0.12.3)
LinkingTo:	Rcpp
SystemRequirements:	C++11
Suggests:	knitr,testthat
VignetteBuilder:	knitr
URL:	https://github.com/RobinHankin/freealg.git
BugReports:	https://github.com/RobinHankin/freealg/issues
Author:	Robin K. S. Hankin [aut, cre] (<https://orcid.org/0000-0001-5982-0415>)

Index of help topics:

Author(s)

NA

Maintainer: Robin K. S. Hankin <hankin.robin@gmail.com>

Examples

```
a <- as.freealg("x+xyx")
b <- as.freealg("4x +XyX")  # upper-case interpreted as inverse

a+b
stopifnot(a+b==b+a)   # should be TRUE

a*b ==b*a # FALSE; noncommutative algebra

as.freealg("1+X+xy")^3

rfalg()
rfalg()^2
```

---

accessor *Accessor methods for freealg objects*

Description

Accessor methods for free algebra objects

Usage

```
words(x)
coeffs(x)
coeffs(x) <- value
```

Arguments

x	Object of class freealg
value	Numeric vector of length 1

Details

Access or set the different parts of an freealg object. The constant term is technically a coefficient but is documented under constant.Rd.

Note

Threre is an extended discussion of this issue in the mvp object at accessor.Rd.

Author(s)

Robin K. S. Hankin

See Also

constant

Examples

```
a <- rfalg()

coeffs(a)
coeffs(a) <- 7
```

---

constant *The constant term*

Description

Get and set the constant term of a freealg object

Usage

```
## S3 method for class 'freealg'
constant(x)
## S3 method for class 'numeric'
constant(x)
## S3 replacement method for class 'freealg'
constant(x) <- value
is.constant(x)
```

Arguments

x	Object of class freealg
value	Scalar value for the constant

Details

The constant term in a free algebra object is the coefficient of the empty term. In a freealg object, the map including {} -> v implies that v is the constant.

If x is a freealg object, constant(x) returns the value of the constant in the multivariate polynomial; if x is numeric, it returns a constant freealg object with value x.

Function is.constant() returns TRUE if its argument has no variables and FALSE otherwise.

Author(s)

Robin K. S. Hankin

Examples

```
p <- as.freealg("1+X+Y+xy")

constant(p)
constant(p^5)

constant(p) <- 1000
p
```

`freealg` *The free algebra*

Description

Create, test for, an coerce to, `freealg` objects

Usage

```
freealg(words, coeffs)
is_ok_free(words,coeffs)
is.freealg(x)
as.freealg(x,...)
char_to_freealg(ch)
natural_char_to_freealg(string)
string_to_freealg(string)
vector_to_free(v,coeffs)
```

Arguments

`words`	Terms of the algebra object, eg `[1,2,-1,3,2]` corresponds to `abACB` (uppercase, or negative number, means inverse)
`coeffs`	Numeric vector corresponding to the coefficients to each element of the `word` list
`string`	Character string
`ch`	Character vector
`v`	Vector of integers
`x`	Object possibly of class `freealg`
`...`	Further arguments, passed to the methods

Details

Function `freealg()` is the formal creation mechanism for `freealg` objects. However, it is not very user-friendly; it is better to use `as.freealg()` in day-to-day use.

Function `is_ok_freealg()` checks for consistency of its arguments.

A `freealg` object is a two-element list. The first element is a list of integer vectors representing the indices and the second is a numeric vector of coefficients. Thus, for example:

```
> as.freealg("a+4bd+3abbbbc")
free algebra element algebraically equal to
 + 1*a + 3*abbbbc + 4*bd
> dput(as.freealg("a+4bd+3abbbbc"))
structure(list(indices = list(1L, c(1L, 2L, 2L, 2L, 2L, 3L),
    c(2L, 4L)), coeffs = c(1, 3, 4)), class = "freealg")
```

Observe that the order of the terms is not preserved and indeed is undefined (implementation-specific). Zero entries are stripped out.

Character strings may be coerced to `freealg` objects; `as.freealg()` calls `natural_char_to_freealg()`, which is user-friendly. Functions `char_to_freealg()` and `string_to_freealg()` are low-level helper functions. These functions assume that upper-case letters are the multiplicative inverses of the lower-case equivalents; so for example `as.freealg("aA")` and `as.freealg(aBcCbA)` evaluate to one. This can be confusing with the default print method.

Author(s)

Robin K. S. Hankin

Examples

```
freealg(sapply(1:5,seq_len),1:5)

freealg(replicate(5,sample(-5:5,rgeom(1,1/5),replace=TRUE)),1:5)


as.freealg("1+xaX")^5
```

`Ops.freealg` *Arithmetic Ops methods for the the free algebra*

Description

Arithmetic operators for manipulation of freealg objects such as addition, multiplication, powers, etc

Usage

```
## S3 method for class 'freealg'
Ops(e1, e2)
free_negative(S)
free_power_scalar(S,n)
free_eq_free(e1,e2)
free_plus_numeric(S,x)
free_plus_free(e1,e2)
lowlevel_simplify(words,coeffs)
lowlevel_free_prod(words1,coeffs1,words2,coeffs2)
lowlevel_free_sum(words1,coeffs1,words2,coeffs2)
lowlevel_free_power(words,coeffs,n)
```

Arguments

`S,e1,e2`	Objects of class `freealg`
`n`	An integer, possibly non-positive
`x`	Scalar value
`words,words1,words2`	A list of words, that is, a list of integer vectors representing the variables in each term
`coeffs,coeffs1,coeffs2`	Numeric vector representing the coefficients of each word

Details

The function `Ops.freealg()` passes binary arithmetic operators ("+", "-", "*", "^", and "==") to the appropriate specialist function.

The caret, as in `a^n`, denotes arithmetic exponentiation, as in `x^3==x*x*x`.

Functions `lowlevel_foo()` are low-level functions that interface directly with the `C` routines in the `src/` directory and are not intended for the end-user.

Author(s)

Robin K. S. Hankin

Examples

```
rfalg()
as.freealg("1+x+xy+yx")  # variables are non-commutative
as.freealg("x") * as.freealg("X") # upper-case letters are lower-case inverses



constant(as.freealg("x+y+X+Y")^6)  # OEIS sequence A035610
```

print *Print freealg objects*

Description

Print methods for free algebra objects

Usage

```
## S3 method for class 'freealg'
print(x,...)
```

Arguments

x	Object of class freealg in the print method
...	Further arguments, currently ignored

Note

The print method does not change the internal representation of a freealg object, which is a two-element list, the first of which is a list of integer vectors representing words, and the second is a numeric vector of coefficients.

The print method has special dispensation for length-zero freealg objects but these are not handled entirely consistently.

The print method is sensitive to the value of getOption("usecaret"), defaulting to "no". The default is to use uppercase letters to represent multiplicative inverses, but if TRUE, use a^-1.

Author(s)

Robin K. S. Hankin

See Also

freealg

Examples

```
rfalg()

x <- rfalg(inc=TRUE)
x                           # default
options("usecaret" = TRUE)  # use caret
x
options("usecaret" = FALSE) # back to the default
x
```

`rfalg` *Random free algebra objects*

Description

Random elements of the free algebra, intended as quick "get you going" examples of `freealg` objects

Usage

```
rfalg(n=7, distinct=3, maxsize=4, include.negative=FALSE)
```

Arguments

`n`	Number of terms to generate
`distinct`	Number of distinct symbols to use
`maxsize`	Maximum number of symbols in any word
`include.negative`	Boolean, with default `FALSE` meaning to use only positive symbols (lower-case letters) and `TRUE` meaning to use upper-case letters as well, corresponding to the inverse of the lower-case symbols

Details

What you see is what you get, basically. A term such as `aabaAbBB` will be simplified to `aabbBB`.

Author(s)

Robin K. S. Hankin

Examples

```
rfalg()
rfalg()^3

constant(rfalg())
```

zero *The zero algebraic object*

Description

Test for a freealg object's being zero

Usage

```
is.zero(x)
```

Arguments

x	Object of class freealg

Details

Function is.zero() returns TRUE if x is indeed the zero free algebra object. It is defined as length(coeffs(x))==0 for reasons of efficiency, but conceptually it returns x==constant(0).

(Use constant(0) to create the zero object).

Author(s)

Robin K. S. Hankin

See Also

constant

Examples

```
stopifnot(is.zero(constant(0)))
```

Index