

The adjoint operator in the freealg package

Robin K. S. Hankin
University of Stirling

Abstract

In this very short document I discuss the adjoint operator `ad()` and illustrate some of its properties.

Keywords: Adjoint operator, free algebra.



```
> ad

function (x)
{
  function(y) {
    new("dot")[as.freealg(x), as.freealg(y)]
  }
}
<bytecode: 0x556d55b519b0>
<environment: namespace:freealg>
```

The adjoint operator: definition

Given an associative algebra \mathcal{A} and $X, Y \in \mathcal{A}$, we define the *Lie Bracket* $[X, Y]$ as $XY - YX$. In the `freealg` package this is implemented with the `.[]` construction:

```
> X <- as.freealg("X")
> Y <- as.freealg("Y")
> .[X, Y]
```

```
free algebra element algebraically equal to
- YX + XY
```

The Jacobi identity

The Lie bracket is bilinear and satisfies the Jacobi condition:

```
> X <- rfa1g(3)
> Y <- rfa1g(3)
> Z <- rfa1g(3)
> X # Y and Z are similar objects
```

```
free algebra element algebraically equal to
+ aba + 2ca + 3cb
```

```
> .[X,Y] # quite complicated
```

```
free algebra element algebraically equal to
- 3aaababa - 6aaabca - 9aaabcb - aaba + abaa + 3abaaaab + 2abab - 2aca - 3acb -
2baba - 4bca - 6bcb + 2caa + 6caaaab + 4cab + 3cba + 9cbaaab + 6cbb
```

```
> .[X,.[Y,Z]] + .[Y,.[Z,X]] + .[Z,.[X,Y]] # Zero by Jacobi
```

```
free algebra element algebraically equal to
0
```

The adjoint map: definition

Now we define the adjoint as follows. Given a Lie algebra \mathfrak{g} , and $X \in \mathcal{A}$, we define a linear map $\text{ad}_X: \mathfrak{g} \rightarrow \mathfrak{g}$ with

$$\text{ad}_X(Y) = [X, Y]$$

In the `freealg` package, this is implemented using the `ad()` function:

```
> ad(X)

function (y)
{
  new("dot")[as.freealg(x), as.freealg(y)]
}
<bytecode: 0x556d55b51518>
<environment: 0x556d532f9cc8>
```

See how function `ad()` returns a *function*. We can play with this:

```
> f <- ad(X)
> f(Y)
```

```
free algebra element algebraically equal to
- 3aaababa - 6aaabca - 9aaabcb - aaba + abaa + 3abaaaab + 2abab - 2aca - 3acb -
2baba - 4bca - 6bcb + 2caa + 6caaaab + 4cab + 3cba + 9cbaaab + 6cbb
```

```
> f(Y) == X*Y-Y*X
```

```
[1] TRUE
```

The first thing to note is that ad_X is NOT a Lie homomorphism, for any particular (non-constant) value of X . If ϕ is a Lie homomorphism then $\phi([x, y]) = [\phi(x), \phi(y)]$. There is no reason to expect the adjoint to be a Lie homomorphism, but it does not hurt to check:

```
> phi <- ad(Z)
> phi(. [X, Y]) == . [phi(X), phi(Y)]
```

```
[1] FALSE
```

With this definition, it is easy to calculate, say, $[Z, [Z, [Z, [Z, [Z, X]]]]$:

```
> f <- ad("x")
> f(f(f(f(f("y")))))
```

```
free algebra element algebraically equal to
+ xxxxyx - 5xxxxyx + 10xxxxyx - 10xxyxxx + 5xyxxxx - yxxxxx
```

Above, we see that $\text{ad}()$ coerces its argument to a `freealg` object.

The adjoint operator is a derivation

A *derivation* of a Lie bracket is a function $\phi: \mathfrak{g} \rightarrow \mathfrak{g}$ that satisfies

$$\phi([Y, Z]) = [\phi(Y), Z] + [Y, \phi(Z)].$$

We will verify that ad_X is indeed a derivation:

```
> phi <- ad(X)
> phi(. [Y, Z]) == . [phi(Y), Z] + . [Y, phi(Z)]
```

```
[1] TRUE
```

The adjoint operator $\text{ad}: \mathfrak{g} \rightarrow \text{End}(\mathfrak{g})$ is a Lie homomorphism

Even though ad_X is not a Lie homomorphism, we can view the adjoint operator as a map from a Lie algebra to its endomorphism group, and this *is* a Lie homomorphism. We are asserting that

$$\text{ad}_{[X, Y]} = [\text{ad}_X, \text{ad}_Y]$$

In package idiom we would have:

```
> ad(. [X,Y]) (Z) == . [ad(X),ad(Y)] (Z)
```

```
[1] TRUE
```

Observe that “. [ad(X),ad(Y)]” is a function:

```
> . [ad(X),ad(Y)]
```

```
function (z)
{
  i(j(z)) - j(i(z))
}
<environment: 0x556d5161fda8>
```

which we evaluate (on the right hand side) at Z.

Adjoint in other contexts

Function `ad()` works in a more general context than the free algebra. For example, we might use it for matrices:

```
> f <- ad(matrix(c(4,6,2,3),2,2))
> M <- matrix(1:4,2,2)
> f(M)
```

```
free algebra element algebraically equal to
- ab - ac - ad - af + ba - bf + ca - cf + da - df + fa + fb + fc + fd
```

Note on the definition of `ad()`

It would seem that one could define `ad()` as follows:

```
`ad` <- function(x){
  function(y){
    . [as.freealg(x),as.freealg(y)]
  }
}
```

which would be a lot clearer. However, “.” is an object, loaded via the `lazydata` system. *Writing R extensions* says, in a footnote:

Note that lazy-loaded datasets are *not* in the package’s namespace so need to be accessed via `::`, e.g. `survival::survexp.us`.

This would make it “`freealg::. [x,y]`”, which is not really any better IMO.

Affiliation:

Robin K. S. Hankin
University of Stirling

E-mail: hankin.rob@gmail.com

