The Cohomology Package

Version 0.1

Marcus Bishop

Marcus Bishop — Email: marcus.bishop@epfl.ch

Copyright

Contents

Chapter 1

Installation and Loading

Like other GAP packages, you download and unpack this package into GAP's `pkg` directory. For example, if you were using some Unix derivative and GAP were installed in the directory `/usr/local/gap4r4`, you would do the following.

Example

```
$ cd /usr/local/gap4r4/pkg
$ su
% wget 'http://mad.epfl.ch/~bishop/Cohomology/cohomology-0.1.tar.gz'
% tar xvzvf cohomology-0.1.tar.gz
```

In this situation, users would then load the package with the `LoadPackage` command.

Example

```
$ gap
gap> LoadPackage("cohomology");
```

Users not having root access, using someone else's computer, or having bad relationships with their network administrators could install the package into their home directories or into some other writable directory such as `/tmp` as follows.

Example

```
$ mkdir /tmp/pkg
$ cd /tmp/pkg
$ wget 'http://mad.epfl.ch/~bishop/Cohomology/cohomology-0.1.tar.gz'
$ tar xvzvf cohomology-0.1.tar.gz
$ gap -l ';/tmp'
gap> LoadPackage("cohomology");
```

Finally, it would be a good idea to run the test file to confirm that all the functions work.

Example

```
gap> ReadPackage("cohomology","tst/test.g");
```

You can count yourself lucky if GAP doesn't complain about anything. There is also a longer running test file for those having ample free time described in Chapter 3.

Chapter 2

Usage

All the functions described below taking an argument `n` do whatever the manual says they do until some stage `n`. `n` is normally a homological degree of some kind. These functions are idempotent in the sense that called a second time with the same argument `n`, they do nothing, but called with a bigger `n`, they continue computing from where the previous calculations left off.

2.1 Cohomology Objects

The calculation of group cohomology involves several computations, the results of which are reused in later calculations, and are thus collected in an object of type `CObject`, created with the following operation.

2.1.1 CohomologyObject

◊ `CohomologyObject( G, k, M )` (operation)
◊ `CohomologyObject( G )` (operation)

Returns: a cohomology object

This function creates a cohomology object, initially having components the group G, the ring k, and the MeatAxe module M. The second invocation creates a cohomology object, initially having components the p-group G, the finite field $\mathbb{F}_p$, and the trivial MeatAxe module.

Notwithstanding,the reader shouldn't get too excited about the generality of the first invocation, as the package is not currently designed to work with groups which aren't p-groups or with rings which aren't $\mathbb{F}_p$. However, you can put whatever kG-module you like as the third argument if G is a p-group and $k = \mathbb{F}_p$. The cohomology object is used to store, in addition to the group, ring, and module mentioned above, the boundary maps, the Betti numbers, the multiplication table, etc.

2.2 Minimal Projective Resolutions

Given a p-group G, the field $k = \mathbb{F}_p$, and a kG-module M, the function below computes the minimal projective resolution

$$P_n \to \cdots \to P_2 \to P_1 \to P_0 \to 0$$

with $P_i = (kG)^{b_i}$ for certain numbers b_i, the *Betti numbers* of the resolution. Then the groups $\mathrm{Ext}^n_{kG}(M,N)$ are simply $\mathrm{Hom}_{kG}(P_n,N)$, and if $N = k$ is the trivial kG-module, then $H^n(G,k) = \mathrm{Ext}^n_{kG}(k,k) = k^{b_n}$.

2.2.1 ProjectiveResolution

◊ `ProjectiveResolution( C, n )` (operation)

Returns: a list containing the Betti numbers $b_0, b_1, \ldots, b_n$

Given a cohomology object `C` having components G, k, and M, this function computes the first `n`+1 terms of the minimal projective resolution P_* of M of the form $P_i = (kG)^{b_i}$ for $0 \leq i \leq n$, and returns the numbers b_i as a list.

2.3 Cohomology Rings

See [CTVEZ03] for the details of the calculation of cohomology products using composition of chain maps.

2.3.1 CohomologyRing

◊ `CohomologyRing( C, n )` (operation)

◊ `CohomologyRing( G, n )` (operation)

Returns: the cohomology ring of G

Given a cohomology object `C` having module component the trivial kG module, possibly with projective resolution already computed, this function returns the degree n truncation of the cohomology ring $H^*(G,k)$. The object returned is an structure constant algebra. Users interested only in working with the cohomology ring of a group as a GAP object, and not in calculating generators, relators, induced maps, etc, can use the second invocation of this function, which returns the cohomology ring immediately, throwing away all intermediate calculations.

2.3.2 IsHomogeneous

◊ `IsHomogeneous( e )` (operation)

Returns: `true` or `false`

Given an element `e` of some cohomology ring A, this operation determines whether or not `e` is homogeneous, that is, whether or not `e` is contained in some `hom_component` of A.

2.3.3 Degree

◊ `Degree( e )` (operation)

Returns: the degree of `e`

This function is intended to return the degree of the possibly non-homogeneous element `e` of some cohomology ring A, but in principle, it works for any element of any graded `SCAlgebra`. Specifically, if $A = A_0 \oplus A_1 \oplus A_2 \oplus \cdots$, that is, if the A_i are the `hom_components` of A, then this function returns the minimum n such that `e` is in $A_0 \oplus A_1 \oplus \cdots \oplus A_n$.

Example

```
gap> A:=CohomologyRing(DihedralGroup(8),10);
<algebra of dimension 66 over GF(2)>
gap> b:=Basis(A);
CanonicalBasis( <algebra of dimension 66 over GF(2)> )
gap> x:=b[2]+b[4];
v.2+v.4
```

```
gap> IsHomogeneous(x);
false
gap> Degree(x);
2
```

2.4 Cohomology Generators and Relators

2.4.1 CohomologyGenerators

◊ CohomologyGenerators(C, n) (operation)

Returns: a list containing the degrees of the generators of the cohomology ring

Given a cohomology object C having components G, k, and M, this function computes the generators of $\mathrm{H}^*(G,k)$ of degree less than or equal to n, and stores them in C. The function returns a list of the degrees of the generators.

The actual cohomology generators are represented by maps $P_n \to k$ and are stored in C as column vectors. Only their degrees are returned.

2.4.2 CohomologyRelators

◊ CohomologyRelators(C, n) (operation)

Returns: the generators and the relators

Given a cohomology object C having components G, k, and M, this function computes a set of generators of the ideal of relators of $\mathrm{H}^*(G,k)$ having multidegree less than or equal to n. The function returns two lists. The first list contains the variables z, y, x, etc. corresponding to the generators of $\mathrm{H}^*(G,k)$ if there are fewer than 12 generators, and contains the variables X_1, X_2, X_3 otherwise. The second list is a list of polynomials in the variables of the first list. The result of this function should be interpreted as follows. The degree n truncation of the cohomology ring $\mathrm{H}^*(G,k)$ is the polynomial ring in the noncommuting variables in the first list, having degrees returned by CohomologyGenerators above, and subject to the relators in the second list.

For example, the following commands

Example

```
gap> C:=CohomologyObject(DihedralGroup(8));
<object>
gap> CohomologyGenerators(C,10);
[ 1, 1, 2 ]
gap> CohomologyRelators(C,10);
[ [ z, y, x ], [ z*y+y^2 ] ]
```

tell us that for $G = D_8$, the cohomology ring $\mathrm{H}^*(G,k)$ is the polynomial ring in non-commuting variables z, y, and x subject to the relations $zy+y^2$. In fact, since $\mathrm{H}^*(G,k)$ is graded-commutative, over $\mathrm{GF}(2)$, it is commutative so that $\mathrm{H}^*(G,k) = \mathbb{F}_2[z,y,x]/(zy+y^2)$.

2.5 Induced Maps

Let $f : G \to H$ be a group homomorphism for p-groups G and H. Then f induces a homomorphism on cohomology $\mathrm{H}^*(H,k) \to \mathrm{H}^*(G,k)$ which is returned by the following function.

2.5.1 InducedHomomorphismOnCohomology

◊ `InducedHomomorphismOnCohomology( C, D, f, n )` (operation)

Returns: the induced homomorphism on cohomology rings

This function returns the induced homomorphism on cohomology $\mathrm{H}^*(H,k) \to \mathrm{H}^*(G,k)$ where the groups G and H are the components of the cohomology objects `C` and `D` and $f : G \to H$ is a group homomorphism. If the cohomology rings have not yet been calculated, they will be computed to degree n, and in this case, they can then be accessed by calling `CohomologyRing` (see 2.3.1).

The following example calculates the homomorphism on cohomology induced by the inclusion of the cyclic group of size 4 into the dihedral group of size 8.

Example
```
gap> G:=CyclicGroup(4);H:=DihedralGroup(8);
<pc group of size 4 with 2 generators>
<pc group of size 8 with 3 generators>
gap> C:=CohomologyObject(G);D:=CohomologyObject(H);
<object>
<object>
gap> f:=GroupHomomorphismByImages(G,H,[G.1],[H.2]);
[ f1 ] -> [ f2 ]
gap> F:=InducedHomomorphismOnCohomology(C,D,f,10);
CanonicalBasis( <algebra of dimension 66 over GF(2)> ) ->
[ v.1, 0*v.1, v.2, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1,
  0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1,
  0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1,
  0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1,
  0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1,
  0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1,
  0*v.1, 0*v.1, 0*v.1, 0*v.1, 0*v.1 ]
gap> B:=CohomologyRing(D,10);
<algebra of dimension 66 over GF(2)>
gap> B.1^F;B.2^F;
v.1
0*v.1
```

2.6 Massey Products

See [Kra66] for the definitions and [Bor01] for the details of the calculation using the Yoneda cocomplex.

2.6.1 MasseyProduct

◊ `MasseyProduct( x1, x2, ..., xn )` (function)

Returns: the Massey product $\langle x_1, x_2, \ldots, x_n \rangle$

Given $= x_1, x_2, \ldots, x_n$ elements of some cohomology ring returned by `CohomologyRing` (see 2.3), this function computes the n-fold Massey product $\langle x_1, x_2, \ldots, x_n \rangle$ provided that the lower-degree Massey products $\left\langle x_i, x_{i+1}, \ldots, x_j \right\rangle$ vanish for all $1 \le i < j \le n$, and returns `fail` otherwise.

As an example, recall that the cohomology rings of the cyclic groups C_3 and C_9 of size 3 and 9 over $k = \mathbb{F}_3$ are both given by $k\langle z,y\rangle / (z^2)$, that is, they are isomorphic as rings. However, the following example shows that $\langle z,z,z \rangle$ is non-zero in $H^*(C_3,k)$ but is zero in $H^*(C_9,k)$.

Example

```
gap> A:=CohomologyRing(CyclicGroup(3),10);
<algebra of dimension 11 over GF(3)>
gap> z:=Basis(A)[2];
v.2
gap> MasseyProduct(z,z);
0*v.1
gap> MasseyProduct(z,z,z);
v.3
gap> A:=CohomologyRing(CyclicGroup(9),10);
<algebra of dimension 11 over GF(3)>
gap> z:=Basis(A)[2];
v.2
gap> MasseyProduct(z,z);
0*v.1
gap> MasseyProduct(z,z,z);
0*v.1
gap> MasseyProduct(z,z,z,z,z,z,z,z,z);
v.3
```

Chapter 3

Leisure and Recreation: Cohomology Rings of All Groups of Size 16

Below is the output of the test file `tst/batch.g`. The file runs through all groups of size *n*, which is initially set to 16, and runs `ProjectiveResolution`, `CohomologyGenerators` and `CohomologyRelators` for each group, and prints the results as well as the timings for each operation to a file. The output below was computed on a 1.8 GHz Intel processor with 2 GB of RAM. The projective resolutions are calculated initially to degree 10 and the generators and relators to degree 6, due to the fact that I already knew all the generators and relators to be of degree less than 6, see http://www.math.uga.edu/~lvalero/cohointro.html. See also the file `tst/README` for suggestions on running long-running batch processes.

Example

```
SmallGroup(16,1)
Betti Numbers: [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ]
Time:  0:00:03.290
Generators in degrees: [ 1, 2 ]
Time:  0:00:00.030
Relators: [ [ z, y ], [ z^2 ] ]
Time:  0:00:00.000

SmallGroup(16,2)
Betti Numbers: [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 ]
Time:  0:00:04.750
Generators in degrees: [ 1, 1, 2, 2 ]
Time:  0:00:01.050
Relators: [ [ z, y, x, w ], [ y^2, z^2 ] ]
Time:  0:00:00.000

SmallGroup(16,3)
Betti Numbers: [ 1, 2, 4, 6, 9, 12, 16, 20, 25, 30, 36 ]
Time:  0:01:18.190
Generators in degrees: [ 1, 1, 2, 2, 2 ]
Time:  0:00:06.780
Relators: [ [ z, y, x, w, v ], [ z*y, z^2, z*v, y^2*x+v^2 ] ]
Time:  0:00:00.000

SmallGroup(16,4)
```

```
Betti Numbers: [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 ]
Time:  0:00:14.020
Generators in degrees: [ 1, 1, 2, 2 ]
Time:  0:00:01.080
Relators: [ [ z, y, x, w ], [ z^2, z*y+y^2, y^3 ] ]
Time:  0:00:00.000

SmallGroup(16,5)
Betti Numbers: [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 ]
Time:  0:00:06.640
Generators in degrees: [ 1, 1, 2 ]
Time:  0:00:00.800
Relators: [ [ z, y, x ], [ z^2 ] ]
Time:  0:00:00.000

SmallGroup(16,6)
Betti Numbers: [ 1, 2, 2, 2, 3, 4, 4, 4, 5, 6, 6 ]
Time:  0:00:02.990
Generators in degrees: [ 1, 1, 3, 4 ]
Time:  0:00:00.310
Relators: [ [ z, y, x, w ], [ z^2, z*y^2, z*x, x^2 ] ]
Time:  0:00:00.000

SmallGroup(16,7)
Betti Numbers: [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 ]
Time:  0:00:05.330
Generators in degrees: [ 1, 1, 2 ]
Time:  0:00:00.790
Relators: [ [ z, y, x ], [ z*y ] ]
Time:  0:00:00.000

SmallGroup(16,8)
Betti Numbers: [ 1, 2, 2, 2, 3, 4, 4, 4, 5, 6, 6 ]
Time:  0:00:02.970
Generators in degrees: [ 1, 1, 3, 4 ]
Time:  0:00:00.310
Relators: [ [ z, y, x, w ], [ z*y, z^3, z*x, y^2*w+x^2 ] ]
Time:  0:00:00.000

SmallGroup(16,9)
Betti Numbers: [ 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2 ]
Time:  0:00:00.670
Generators in degrees: [ 1, 1, 4 ]
Time:  0:00:00.070
Relators: [ [ z, y, x ], [ z*y, z^3+y^3, y^4 ] ]
Time:  0:00:00.000

SmallGroup(16,10)
Betti Numbers: [ 1, 3, 6, 10, 15, 21, 28, 36, 45, 55, 66 ]
Time:  0:07:29.440
Generators in degrees: [ 1, 1, 1, 2 ]
Time:  0:00:22.680
Relators: [ [ z, y, x, w ], [ z^2 ] ]
```

```
Time:  0:00:00.000

SmallGroup(16,11)
Betti Numbers: [ 1, 3, 6, 10, 15, 21, 28, 36, 45, 55, 66 ]
Time:  0:08:22.950
Generators in degrees: [ 1, 1, 1, 2 ]
Time:  0:00:22.640
Relators: [ [ z, y, x, w ], [ z*y ] ]
Time:  0:00:00.000

SmallGroup(16,12)
Betti Numbers: [ 1, 3, 5, 6, 7, 9, 11, 12, 13, 15, 17 ]
Time:  0:01:13.200
Generators in degrees: [ 1, 1, 1, 4 ]
Time:  0:00:03.720
Relators: [ [ z, y, x, w ], [ z^2+z*y+y^2, y^3 ] ]
Time:  0:00:00.000

SmallGroup(16,13)
Betti Numbers: [ 1, 3, 5, 6, 7, 9, 11, 12, 13, 15, 17 ]
Time:  0:00:23.330
Generators in degrees: [ 1, 1, 1, 4 ]
Time:  0:00:03.430
Relators: [ [ z, y, x, w ], [ z*y+x^2, z*x^2+y*x^2, y^2*x^2+x^4 ] ]
Time:  0:00:00.000

SmallGroup(16,14)
Betti Numbers: [ 1, 4, 10, 20, 35, 56, 84, 120, 165, 220, 286 ]
Time:  6:37:11.670
Generators in degrees: [ 1, 1, 1, 1 ]
Time:  0:04:17.050
Relators: [ [ z, y, x, w ], [  ] ]
Time:  0:00:00.000
```

References

[Bor01] Inger Christin Borge. *A cohomological approach to the classification of p-groups*. PhD thesis, Oxford, http://www.maths.abdn.ac.uk/∼bensondj/html/archive/borge.html, 2001. 8

[CTVEZ03] Jon F. Carlson, Lisa Townsley, Luis Valeri-Elizondo, and Mucheng Zhang. *Cohomology rings of finite groups*, volume 3 of *Algebras and Applications*. Kluwer Academic Publishers, Dordrecht, 2003. 6

[Kra66] David Kraines. Massey higher products. *Trans. Amer. Math. Soc.*, 124:431–449, 1966. 8

Index