

The Crime Package

Version 0.2

Marcus Bishop

Copyright

© 2006 Marcus Bishop

We adopt the copyright regulations of GAP as detailed in the copyright notice in the GAP manual.

Acknowledgements

This project wouldn't have been possible without Jon Carlson. Jon devised the algorithms used by `ProjectiveResolution`, `CohomologyGenerators`, and `CohomologyRelators`, having already implemented them in Magma and sharing these programs with me.

Thank you also to Laurent Bartholdi for his helpful suggestions regarding the GAP implementation and the user interface. Laurent also tested the program extensively and uncovered many bugs.

Contents

1	Installation and Loading	4
2	Usage	5
2.1	Cohomology Objects	5
2.1.1	CohomologyObject	5
2.2	Minimal Projective Resolutions	5
2.2.1	ProjectiveResolution	6
2.3	Cohomology Generators and Relators	6
2.3.1	CohomologyGenerators	6
2.3.2	CohomologyRelators	6
2.4	Cohomology Rings	7
2.4.1	CohomologyRing	7
2.4.2	IsHomogeneous	7
2.4.3	Degree	7
2.4.4	LocateGeneratorsInCohomologyRing	8
2.5	Induced Maps	8
2.5.1	InducedHomomorphismOnCohomology	8
2.6	Massey Products	9
2.6.1	MasseyProduct	9
3	Leisure and Recreation: Cohomology Rings of all Groups of Size 16	10

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/Crime/crime-0.2.tar.gz'
% tar xzvzf crime-0.2.tar.gz
```

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

Example

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

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/Crime/crime-0.2.tar.gz'
$ tar xzvzf crime-0.2.tar.gz
$ gap -l '/tmp'
gap> LoadPackage("crime");
```

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

Example

```
gap> ReadPackage("crime", "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` except `CohomologyRing` do whatever the manual says they do until some stage `n`, where `n` is normally the homological degree. 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 p -group G , the field k of characteristic p , and the MeatAxe kG -module M . The second invocation creates a cohomology object, initially having components the p -group G , the field \mathbb{F}_p , and the trivial MeatAxe kG -module.

The cohomology object is used to store, in addition to the group, field, and module, the boundary maps, the Betti numbers, the multiplication table, etc.

2.2 Minimal Projective Resolutions

Given a p -group G , the field k of characteristic p and a kG -module M , the function below computes a minimal projective resolution

$$P_n \rightarrow \cdots \rightarrow P_2 \rightarrow P_1 \rightarrow P_0 \rightarrow k \rightarrow 0$$

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

2.2.1 ProjectiveResolution

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

Returns: a list containing the Betti numbers b_0, b_1, \dots, 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 Generators and Relators

2.3.1 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 $H^*(G, k)$ 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 $P_n \rightarrow k$ and are stored in C as column vectors. Only their degrees are returned.

2.3.2 CohomologyRelators

◇ `CohomologyRelators(C, n)` (operation)

Returns: a list of generators and a list of 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 $H^*(G, k)$ having multidegree less than or equal to n .

The function returns two lists, the first containing the variables z, y, x, \dots corresponding to the generators of $H^*(G, k)$ if there are fewer than 12 generators, and containing the variables x_1, x_2, x_3, \dots otherwise. The second is a list of polynomials in the variables from the first list.

These two lists should be interpreted as follows. The degree n truncation of the cohomology ring $H^*(G, k)$ is the polynomial ring in the non-commuting variables from the first list, having the 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 $H^*(G, k)$ is the graded-commutative polynomial ring in the variables z, y , and x of degrees 1, 1, and 2, subject to the relation $zy + y^2$. But since $H^*(G, k)$ is commutative, k being of characteristic 2, we have $H^*(G, k) = k[z, y, x] / (zy + y^2)$. This result can be further improved by taking $z = z + y$, giving $H^*(G, k) = k[z, y, x] / (zy)$.

2.4 Cohomology Rings

See [CTVEZ03] for the details of the calculation of cohomology products using composition of chain maps. See also the file `doc/explanation.tex` for an explanation of the implementation.

2.4.1 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 and possibly having a 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 of the group G immediately, throwing away all intermediate calculations.

Observe that the object returned is a degree n truncation of the infinite-dimensional cohomology ring. A consequence of this is that multiplying two elements whose product has degree greater than n results in zero.

Observe also that calling `CohomologyRing` a second time with a bigger n does *not* extend the previous ring, but rather, recalculates the entire ring from the beginning. Admittedly, this is rather inconvenient, and I promise to think about extending the previous ring the next time I find myself drinking a martini on the beach with my laptop.

2.4.2 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 .

2.4.3 Degree

◇ `Degree(e)` (method)

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, works for any element of any graded `SCAlgebra`. Specifically, if $A = A_0 \oplus A_1 \oplus A_2 \oplus \cdots$ with A_i 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
```

2.4.4 LocateGeneratorsInCohomologyRing

◇ `LocateGeneratorsInCohomologyRing(C)` (function)

Returns: a list containing the cohomology generators.

Having already called `CohomologyRing` (see 2.4.1), this function returns a list of elements of the cohomology ring which together with the identity element of the ring generate it as a ring.

This function is a wrapper for `CohomologyGenerators` (see 2.3.1), indicating which elements of the cohomology ring correspond with the generators found by `CohomologyGenerators`.

Example

```
gap> C:=CohomologyObject(SmallGroup(8,4));
<object>
gap> A:=CohomologyRing(C,10);
<algebra of dimension 17 over GF(2)>
gap> L:=LocateGeneratorsInCohomologyRing(C);
[ v.2, v.3, v.7 ]
gap> A:=Subalgebra(A,Concatenation(L,[One(A)]));
true
```

2.5 Induced Maps

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

2.5.1 InducedHomomorphismOnCohomology

◇ `InducedHomomorphismOnCohomology(C, D, f, n)` (function)

Returns: the induced homomorphism on cohomology rings.

This function returns the induced homomorphism on cohomology $H^*(H, k) \rightarrow H^*(G, k)$ where the groups G and H are the components of the cohomology objects C and D and $f : G \rightarrow H$ 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 2.4.1).

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 ]
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. See also the file `doc/explanation.tex` for an explanation of the implementation.

2.6.1 MasseyProduct

◇ **MasseyProduct**(*x1*, *x2*, ..., *xn*) (function)

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

Given elements x_1, x_2, \dots, x_n of a cohomology ring returned by `CohomologyRing` (see 2.4), this function computes the n -fold Massey product $\langle x_1, x_2, \dots, x_n \rangle$ provided that the lower-degree Massey products $\langle x_i, x_{i+1}, \dots, x_j \rangle$ vanish for all $1 \leq i < j \leq n$, 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 3.06 GHz Intel processor with 3.71 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 dealing with other users when running long-running batch processes.

Example

```
SmallGroup(16,1)
Betti Numbers: [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ]
Time: 0:00:04.209
Generators in degrees: [ 1, 2 ]
Time: 0:00:00.037
Relators: [ [ z, y ], [ z^2 ] ]
Time: 0:00:00.101

SmallGroup(16,2)
Betti Numbers: [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 ]
Time: 0:00:03.055
Generators in degrees: [ 1, 1, 2, 2 ]
Time: 0:00:09.322
Relators: [ [ z, y, x, w ], [ z^2, y^2 ] ]
Time: 0:00:23.386

SmallGroup(16,3)
Betti Numbers: [ 1, 2, 4, 6, 9, 12, 16, 20, 25, 30, 36 ]
Time: 0:00:54.653
Generators in degrees: [ 1, 1, 2, 2, 2 ]
Time: 0:03:29.691
Relators: [ [ z, y, x, w, v ], [ z^2, z*y, z*x, y^2*v+x^2 ] ]
Time: 0:06:33.189

SmallGroup(16,4)
```

Betti Numbers: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

Time: 0:00:03.163

Generators in degrees: [1, 1, 2, 2]

Time: 0:00:09.873

Relators: [[z, y, x, w], [z², z*y+y², y³]]

Time: 0:00:25.149

SmallGroup(16,5)

Betti Numbers: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

Time: 0:00:03.080

Generators in degrees: [1, 1, 2]

Time: 0:00:07.356

Relators: [[z, y, x], [z²]]

Time: 0:00:22.859

SmallGroup(16,6)

Betti Numbers: [1, 2, 2, 2, 3, 4, 4, 4, 5, 6, 6]

Time: 0:00:00.674

Generators in degrees: [1, 1, 3, 4]

Time: 0:00:02.575

Relators: [[z, y, x, w], [z², z*y², z*x, x²]]

Time: 0:00:03.675

SmallGroup(16,7)

Betti Numbers: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

Time: 0:00:03.071

Generators in degrees: [1, 1, 2]

Time: 0:00:07.282

Relators: [[z, y, x], [z*y]]

Time: 0:00:22.786

SmallGroup(16,8)

Betti Numbers: [1, 2, 2, 2, 3, 4, 4, 4, 5, 6, 6]

Time: 0:00:00.676

Generators in degrees: [1, 1, 3, 4]

Time: 0:00:02.584

Relators: [[z, y, x, w], [z*y, z³, z*x, y²*w+x²]]

Time: 0:00:03.825

SmallGroup(16,9)

Betti Numbers: [1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2]

Time: 0:00:00.087

Generators in degrees: [1, 1, 4]

Time: 0:00:00.139

Relators: [[z, y, x], [z*y, z³+y³, y⁴]]

Time: 0:00:00.374

SmallGroup(16,10)

Betti Numbers: [1, 3, 6, 10, 15, 21, 28, 36, 45, 55, 66]

Time: 0:05:37.603

Generators in degrees: [1, 1, 1, 2]

Time: 0:16:52.067

Relators: [[z, y, x, w], [z²]]

Time: 0:52:54.579

SmallGroup(16,11)

Betti Numbers: [1, 3, 6, 10, 15, 21, 28, 36, 45, 55, 66]

Time: 0:05:30.506

Generators in degrees: [1, 1, 1, 2]

Time: 0:16:29.940

Relators: [[z, y, x, w], [z*y]]

Time: 0:52:04.624

SmallGroup(16,12)

Betti Numbers: [1, 3, 5, 6, 7, 9, 11, 12, 13, 15, 17]

Time: 0:00:10.051

Generators in degrees: [1, 1, 1, 4]

Time: 0:00:43.703

Relators: [[z, y, x, w], [z^2+z*y+y^2, y^3]]

Time: 0:02:02.128

SmallGroup(16,13)

Betti Numbers: [1, 3, 5, 6, 7, 9, 11, 12, 13, 15, 17]

Time: 0:00:09.991

Generators in degrees: [1, 1, 1, 4]

Time: 0:00:43.443

Relators: [[z, y, x, w], [z*y+x^2, z*x^2+y*x^2, y^2*x^2+x^4]]

Time: 0:01:59.953

SmallGroup(16,14)

Betti Numbers: [1, 4, 10, 20, 35, 56, 84, 120, 165, 220, 286]

Time: 5:03:44.290

Generators in degrees: [1, 1, 1, 1]

Time: 8:14:32.187

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. [9](#)
- [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. [7](#)
- [Kra66] David Kraines. Massey higher products. *Trans. Amer. Math. Soc.*, 124:431–449, 1966. [9](#)

Index

CohomologyGenerators, [6](#)

CohomologyObject, [5](#)

CohomologyRelators, [6](#)

CohomologyRing, [7](#)

Degree, [7](#)

InducedHomomorphismOnCohomology, [8](#)

IsHomogeneous, [7](#)

LocateGeneratorsInCohomologyRing, [8](#)

MasseyProduct, [9](#)

ProjectiveResolution, [6](#)