

# Contents

<b>1</b>	<b>Resolutions of the ground ring</b>	<b>3</b>
<b>2</b>	<b>Resolutions of modules</b>	<b>4</b>
<b>3</b>	<b>Induced equivariant chain maps</b>	<b>5</b>
<b>4</b>	<b>Functors</b>	<b>6</b>
<b>5</b>	<b>Chain complexes</b>	<b>7</b>
<b>6</b>	<b>Homology and cohomology groups</b>	<b>8</b>
<b>7</b>	<b>Poincare series</b>	<b>9</b>
<b>8</b>	<b>Cohomology ring structure</b>	<b>10</b>
<b>9</b>	<b>Commutator and nonabelian tensor computations</b>	<b>11</b>
<b>10</b>	<b>Generators and relators of groups</b>	<b>12</b>
<b>11</b>	<b>Orbit polytopes and fundamental domains</b>	<b>13</b>
<b>12</b>	<b>Cocycles</b>	<b>15</b>
<b>13</b>	<b>Words in free <math>ZG</math>-modules</b>	<b>16</b>
<b>14</b>	<b><math>FpG</math>-modules</b>	<b>17</b>
<b>15</b>	<b>Meataxe modules</b>	<b>18</b>
<b>16</b>	<b>Coxeter diagrams and graphs of groups</b>	<b>19</b>
<b>17</b>	<b>Some functions for accessing basic data</b>	<b>20</b>
<b>18</b>	<b>Parallel Computation - Core Functions</b>	<b>21</b>
<b>19</b>	<b>Parallel Computation - Extra Functions</b>	<b>22</b>
<b>20</b>	<b>Topological Data Analysis</b>	<b>23</b>

2

**21 Pseudo lists**

**24**

**22 Miscellaneous**

**25**

# Chapter 1

## Resolutions of the ground ring

`ResolutionAbelianGroup(L, n)` `ResolutionAbelianGroup(G, n)` Inputs a list  $L := [m_1, m_2, \dots, m_d]$  of nonnegative integers and a positive integer  $n$ . It returns  $n$  terms of a free  $ZG$ -resolution of  $Z/L$ .

`ResolutionAlmostCrystalGroup(G, n)` Inputs a positive integer  $n$  and an almost crystallographic pcg group  $G$ . It returns  $n$  terms of a free  $ZG$ -resolution of  $Z$ .

`ResolutionAlmostCrystalQuotient(G, n, c)` `ResolutionAlmostCrystalQuotient(G, n, c, false)` An almost crystallographic pcg group  $G$  and a positive integer  $n$ . It returns  $n$  terms of a free  $ZG$ -resolution of  $Z$ .

`ResolutionArtinGroup(D, n)` Inputs a Coxeter diagram  $D$  and an integer  $n > 1$ . It returns  $n$  terms of a free  $ZG$ -resolution of  $Z$ .

`ResolutionAsphericalPresentation(F, R, n)` Inputs a free group  $F$ , a set  $R$  of words in  $F$  which constitute an aspherical presentation of a group  $G$ , and a positive integer  $n$ . It returns  $n$  terms of a free  $ZG$ -resolution of  $Z$ .

`ResolutionBieberbachGroup(G)` `ResolutionBieberbachGroup(G, v)` Inputs a Bieberbach group  $G$  (represented using `AffineCrystGroupOnRight`) and a positive integer  $v$ . It returns  $v$  terms of a free  $ZG$ -resolution of  $Z$ .

`ResolutionDirectProduct(R, S)` `ResolutionDirectProduct(R, S, "internal")` Inputs a  $ZG$ -resolution  $R$  and a  $ZH$ -resolution  $S$ . It returns  $R \otimes S$ .

`ResolutionExtension(g, R, S)` `ResolutionExtension(g, R, S, "TestFiniteness")` `ResolutionExtension(g, R, S, true)` Inputs a  $ZG$ -resolution  $R$ , a  $ZH$ -resolution  $S$ , and a group  $g$ . It returns  $R \otimes S$ .

`ResolutionFiniteDirectProduct(R, S)` `ResolutionFiniteDirectProduct(R, S, "internal")` Inputs a  $ZG$ -resolution  $R$  and a  $ZH$ -resolution  $S$ . It returns  $R \otimes S$ .

`ResolutionFiniteExtension(gensE, gensG, R, n)` `ResolutionFiniteExtension(gensE, gensG, R, n, true)` Inputs a  $ZG$ -resolution  $R$ , a list  $gensE$  of generators of a group  $E$ , a list  $gensG$  of generators of a group  $G$ , and a positive integer  $n$ . It returns  $n$  terms of a free  $ZG$ -resolution of  $Z$ .

`ResolutionFiniteGroup(gens, n)` `ResolutionFiniteGroup(gens, n, true)` `ResolutionFiniteGroup(gens, n, false)` Inputs a list  $gens$  of generators of a group  $G$  and a positive integer  $n$ . It returns  $n$  terms of a free  $ZG$ -resolution of  $Z$ .

`ResolutionFiniteSubgroup(R, K)` `ResolutionFiniteSubgroup(R, gensG, gensK)` Inputs a  $ZG$ -resolution  $R$  for a group  $G$  and a list  $gensK$  of generators of a subgroup  $K$  of finite index  $|G : K|$ . It returns  $n$  terms of a free  $ZG$ -resolution of  $Z$ .

`ResolutionGraphOfGroups(D, n)` `ResolutionGraphOfGroups(D, n, L)` Inputs a graph of groups  $D$  and a positive integer  $n$ . It returns  $n$  terms of a free  $ZG$ -resolution of  $Z$ .

`ResolutionNilpotentGroup(G, n)` `ResolutionNilpotentGroup(G, n, "TestFiniteness")` Inputs a nilpotent group  $G$  and a positive integer  $n$ . It returns  $n$  terms of a free  $ZG$ -resolution of  $Z$ .

`ResolutionNormalSeries(L, n)` `ResolutionNormalSeries(L, n, true)` `ResolutionNormalSeries(L, n, false)` Inputs a list  $L$  of subgroups of a finite group  $G$  and a positive integer  $n$ . It returns  $n$  terms of a free  $ZG$ -resolution of  $Z$ .

`ResolutionPrimePowerGroup(P, n)` `ResolutionPrimePowerGroup(G, n, p)` Inputs a  $p$ -group  $P$  and integer  $n > 0$ . It returns  $n$  terms of a free  $ZG$ -resolution of  $Z$ .

`ResolutionSmallFpGroup(G, n)` `ResolutionSmallFpGroup(G, n, p)` Inputs a small finitely presented group  $G$  and a positive integer  $n$ . It returns  $n$  terms of a free  $ZG$ -resolution of  $Z$ .

`ResolutionSubgroup(R, K)` Inputs a  $ZG$ -resolution for an (infinite) group  $G$  and a subgroup  $K$  of finite index  $|G : K|$ . It returns  $n$  terms of a free  $ZG$ -resolution of  $Z$ .

`ResolutionSubnormalSeries(L, n)` Inputs a positive integer  $n$  and a list  $L = [L_1, \dots, L_k]$  of subgroups  $L_i$  of a finite group  $G$ . It returns  $n$  terms of a free  $ZG$ -resolution of  $Z$ .

`TwistedTensorProduct(R, S, EhomG, GmapE, NhomE, NEhomN, EltsE, Mult, InvE)` Inputs a  $ZG$ -resolution  $R$ , a  $ZN$ -resolution  $S$ , and a group  $G$ . It returns  $R \otimes S$ .

## Chapter 2

# Resolutions of modules

| `ResolutionFpGModule(M, n)` Inputs an  $FpG$ -module  $M$  and a positive integer  $n$ . It returns  $n$  terms of a minimal free  $F$

## Chapter 3

# Induced equivariant chain maps

| `EquivariantChainMap(R, S, f)` Inputs a  $ZG$ -resolution  $R$ , a  $ZG'$ -resolution  $S$ , and a group homomorphism  $f : G \rightarrow G'$

# Chapter 4

## Functors

•

`HomToIntegers(X)` Inputs either a  $ZG$ -resolution  $X = R$ , or an equivariant chain map  $X = (F : R \rightarrow S)$ . It returns the

`HomToIntegersModP(R)` Inputs a  $ZG$ -resolution  $R$  and returns the cochain complex obtained by applying  $Hom_{ZG}(Z, Z)$

`HomToIntegralModule(R, f)` Inputs a  $ZG$ -resolution  $R$  and a group homomorphism  $f : G \rightarrow GL_n(Z)$  to the group of

`LowerCentralSeriesLieAlgebra(G)` `LowerCentralSeriesLieAlgebra(f)` Inputs a pcp group  $G$ . If each quo

`TensorWithIntegers(X)` Inputs either a  $ZG$ -resolution  $X = R$ , or an equivariant chain map  $X = (F : R \rightarrow S)$ . It retu

`TensorWithIntegersModP(X, p)` Inputs either a  $ZG$ -resolution  $X = R$ , or an equivariant chain map  $X = (F : R \rightarrow S)$

`TensorWithRationals(R)` Inputs a  $ZG$ -resolution  $R$  and returns the chain complex obtained by tensoring with the tri

## Chapter 5

# Chain complexes

`ChevalleyEilenbergComplex(X, n)` Inputs either a Lie algebra  $X = A$  (over the ring of integers  $Z$  or over a field  $K$ )  
`LeibnizComplex(X, n)` Inputs either a Lie or Leibniz algebra  $X = A$  (over the ring of integers  $Z$  or over a field  $K$ ) or

## Chapter 6

# Homology and cohomology groups

`Cohomology(X)` Inputs either a cochain complex  $X = C$  or a cochain map  $X = (C \rightarrow D)$  over the integers  $Z$ . If  $X = C$

`CohomologyPrimePart(C, n, p)` Inputs a cochain complex  $C$  in characteristic 0, a positive integer  $n$ , and a prime  $p$ . It re

`GroupCohomology(X, n)` `GroupCohomology(X, n, p)` Inputs a positive integer  $n$  and either a finite group  $X = G$  or a t

`GroupHomology(X, n)`

`GroupHomology(X, n, p)` Inputs a positive integer  $n$  and either a finite group  $X = G$  or a Coxeter diagram  $X = D$  represe

`Homology(X, n)` Inputs either a chain complex  $X = C$  or a chain map  $X = (C \rightarrow D)$ . If  $X = C$  then the torsion coefficient

`HomologyPb(C, n)` This is a back-up function which might work in some instances where `Homology(C, n)` fails. It is mo

`HomologyPrimePart(C, n, p)` Inputs a chain complex  $C$  in characteristic 0, a positive integer  $n$ , and a prime  $p$ . It returns

`LeibnizAlgebraHomology(A, n)` Inputs a Lie or Leibniz algebra  $X = A$  (over the ring of integers  $Z$  or over a field  $K$ ), t

`LieAlgebraHomology(A, n)` Inputs a Lie algebra  $A$  (over the integers or a field) and a positive integer  $n$ . It returns the h

`PrimePartDerivedFunctor(G, R, F, n)` Inputs a finite group  $G$ , a positive integer  $n$ , at least  $n + 1$  terms of a  $ZP$ -resolut

`RankHomologyPGroup(G, n)` `RankHomologyPGroup(R, n)` `RankHomologyPGroup(G, n, "empirical")` Inputs a (sm

`RankPrimeHomology(G, n)` Inputs a (smallish)  $p$ -group  $G$  together with a positive integer  $n$ . It returns a function `dim(k)`

# Chapter 7

## Poincare series

EfficientNormalSubgroups( $G$ )

EfficientNormalSubgroups( $G, k$ ) Inputs a prime-power group  $G$  and, optionally, a positive integer  $k$ . The default is  $k=1$ .

ExpansionOfRationalFunction( $f, n$ ) Inputs a positive integer  $n$  and a rational function  $f(x) = p(x)/q(x)$  where the denominator  $q(x)$  is a product of linear factors.

PoincareSeries( $G, n$ ) PoincareSeries( $R, n$ )

PoincareSeries( $L, n$ )

PoincareSeries( $G$ ) Inputs a finite  $p$ -group  $G$  and a positive integer  $n$ . It returns a quotient of polynomials  $f(x) = P(x)/Q(x)$ .

PoincareSeriesPrimePart( $G, p, n$ ) Inputs a finite group  $G$ , a prime  $p$ , and a positive integer  $n$ . It returns a quotient of polynomials  $f(x) = P(x)/Q(x)$ .

Prank( $G$ ) Inputs a  $p$ -group  $G$  and returns the rank of the largest elementary abelian subgroup.

## Chapter 8

# Cohomology ring structure

`IntegralCupProduct (R, u, v, p, q)`

`IntegralCupProduct (R, u, v, p, q, P, Q, N)` (Various functions used to construct the cup product are also *CRfunctions*)

`IntegralRingGenerators (R, n)` Inputs at least  $n + 1$  terms of a  $ZG$ -resolution and integer  $n > 0$ . It returns a minimal

`ModPCohomologyGenerators (G, n)`

`ModPCohomologyGenerators (R)` Inputs either a  $p$ -group  $G$  and positive integer  $n$ , or else  $n$  terms of a minimal  $Z_p G$ -

`ModPCohomologyRing (G, n)`

`ModPCohomologyRing (G, n, level)`

`ModPCohomologyRing (R)`

`ModPCohomologyRing (R, level)` Inputs either a  $p$ -group  $G$  and positive integer  $n$ , or else  $n$  terms of a minimal  $Z_p G$ -

`ModPRingGenerators (A)` Inputs a mod  $p$  cohomology ring  $A$  (created using the preceding function). It returns a mi

## Chapter 9

# Commutator and nonabelian tensor computations

•  
BaerInvariant( $G, c$ ) Inputs a nilpotent group  $G$  and integer  $c > 0$ . It returns the Baer invariant  $M^{(c)}(G)$  defined as follows.  
Coclass( $G$ ) Inputs a group  $G$  of prime-power order  $p^n$  and nilpotency class  $c$  say. It returns the integer  $r = n - c$ .  
EpiCentre( $G, N$ )  
EpiCentre( $G$ ) Inputs a finite group  $G$  and normal subgroup  $N$  and returns a subgroup  $Z^*(G, N)$  of the centre of  $N$ . The  
NonabelianExteriorProduct( $G, N$ ) Inputs a finite group  $G$  and normal subgroup  $N$ . It returns a record  $E$  with the following  
NonabelianTensorProduct( $G, N$ ) Inputs a finite group  $G$  and normal subgroup  $N$ . It returns a record  $E$  with the following  
NonabelianTensorSquare( $G$ )  
NonabelianTensorSquare( $G, m$ ) Inputs a finite or nilpotent infinite group  $G$  and returns a record  $T$  with the following  
RelativeSchurMultiplier( $G, N$ ) Inputs a finite group  $G$  and normal subgroup  $N$ . It returns the homology group  $H_2$   
TensorCentre( $G$ ) Inputs a group  $G$  and returns the largest central subgroup  $N$  such that the induced homomorphism  
ThirdHomotopyGroupOfSuspensionB( $G$ )  
ThirdHomotopyGroupOfSuspensionB( $G, m$ ) Inputs a finite or nilpotent infinite group  $G$  and returns the abelian invariant  
UpperEpicentralSeries( $G, c$ ) Inputs a nilpotent group  $G$  and an integer  $c$ . It returns the  $c$ -th term of the upper epicentral series.

## Chapter 10

# Generators and relators of groups

•  
CayleyGraphDisplay( $G, X$ )

CayleyGraphDisplay( $G, X, \text{"mozilla"}$ ) Inputs a finite group  $G$  together with a subset  $X$  of  $G$ . It displays the corresponding Cayley graph.

IsAspherical( $F, R$ ) Inputs a free group  $F$  and a set  $R$  of words in  $F$ . It performs a test on the 2-dimensional CW-space  $K(F/R)$ .

PresentationOfResolution( $R$ ) Inputs at least two terms of a reduced  $ZG$ -resolution  $R$  and returns a record  $P$  with the following fields:

TorsionGeneratorsAbelianGroup( $G$ ) Inputs an abelian group  $G$  and returns a generating set  $[x_1, \dots, x_n]$  where no proper subset of  $\{x_1, \dots, x_n\}$  generates  $G$ .

# Chapter 11

## Orbit polytopes and fundamental domains

`FundamentalDomainAffineCrystGroupOnRight(v, G)` Inputs a crystallographic group  $G$  (represented using AffineC

`OrbitPolytope(G, v, L)` Inputs a permutation group or matrix group  $G$  of degree  $n$  and a rational vector  $v$  of length  $n$ .  
The function uses Polymake software.

`PolytopalComplex(G, v)`

`PolytopalComplex(G, v, n)`

Inputs a permutation group or matrix group  $G$  of degree  $n$  and a rational vector  $v$  of length  $n$ . In both cases there is a natural action of  $G$  on  $v$ . Let  $P(G, v)$  be the convex polytope arising as the convex hull of the Euclidean points in the orbit of  $v$  under the action of  $G$ . The cellular chain complex  $C_* = C_*(P(G, v))$  is an exact sequence of (not necessarily free)  $ZG$ -modules. The function returns a component object  $R$  with components:

- $R!.dimension(k)$  is a function which returns the number of  $G$ -orbits of the  $k$ -dimensional faces in  $P(G, v)$ . If each  $k$ -face has trivial stabilizer subgroup in  $G$  then  $C_k$  is a free  $ZG$ -module of rank  $R!.dimension(k)$ .
- $R!.stabilizer(k, n)$  is a function which returns the stabilizer subgroup for a face in the  $n$ -th orbit of  $k$ -faces.
- If all faces of dimension  $< k + 1$  have trivial stabilizer group then the first  $k$  terms of  $C_*$  constitute part of a free  $ZG$ -resolution. The boundary map is described by the function  $boundary(k, n)$ . (If some faces have non-trivial stabilizer group then  $C_*$  is not free and no attempt is made to determine signs for the boundary map.)
- $R!.elements, R!.group, R!.properties$  are as in a  $ZG$ -resolution.

If an optional third input variable  $n$  is used, then only the first  $n$  terms of the resolution  $C_*$  will be computed.

The function uses Polymake software.

`PolytopalGenerators(G, v)`

Inputs a permutation group or matrix group  $G$  of degree  $n$  and a rational vector  $v$  of length  $n$ . In both cases there is a natural action of  $G$  on  $v$ , and the vector  $v$  must be chosen so that it has trivial stabilizer subgroup in  $G$ . Let  $P(G, v)$  be the convex polytope arising as the convex hull of the Euclidean points in the orbit of  $v$  under the action of  $G$ . The function returns a record  $P$  with components:

- *P.generators* is a list of all those elements  $g$  in  $G$  such that  $g \cdot v$  has an edge in common with  $v$ . The list is a generating set for  $G$ .
- *P.vector* is the vector  $v$ .
- *P.hasseDiagram* is the Hasse diagram of the cone at  $v$ .

The function uses Polymake software. The function is joint work with Seamus Kelly.

`VectorStabilizer(G, v)`

Inputs a permutation group or matrix group  $G$  of degree  $n$  and a rational vector of degree  $n$ . In both cases there is a natural action of  $G$  on  $v$  and the function returns the group of elements in  $G$  that fix  $v$ .

## Chapter 12

# Cocycles

`CocycleCondition(R, n)` Inputs a resolution  $R$  and an integer  $n > 0$ . It returns an integer matrix  $M$  with the following properties.

`StandardCocycle(R, f, n)`

`StandardCocycle(R, f, n, q)` Inputs a  $ZG$ -resolution  $R$  (with contracting homotopy), a positive integer  $n$  and an integer  $q$ .

`Syzygy(R, g)` Inputs a  $ZG$ -resolution  $R$  (with contracting homotopy) and a list  $g = [g[1], \dots, g[n]]$  of elements in  $G$ . It returns a list of syzygies.

## Chapter 13

# Words in free $ZG$ -modules

`AddFreeWords(v, w)` Inputs two words  $v, w$  in a free  $ZG$ -module and returns their sum  $v + w$ . If the characteristic of  $Z$  is  $p$ , the result is reduced modulo  $p$ .

`AddFreeWordsModP(v, w, p)` Inputs two words  $v, w$  in a free  $ZG$ -module and the characteristic  $p$  of  $Z$ . It returns the sum  $v + w$  reduced modulo  $p$ .

`AlgebraicReduction(w)` Inputs a word  $w$  in a free  $ZG$ -module and returns a reduced version of the word in which no two adjacent letters are inverses of each other.

`Multiply Word(n, w)` Inputs a word  $w$  and integer  $n$ . It returns the scalar multiple  $n \cdot w$ .

`Negate([i, j])` Inputs a pair  $[i, j]$  of integers and returns  $[-i, j]$ .

`NegateWord(w)` Inputs a word  $w$  in a free  $ZG$ -module and returns the negated word  $-w$ .

`PrintZGword(w, elts)` Inputs a word  $w$  in a free  $ZG$ -module and a (possibly partial but sufficient) listing `elts` of the elements of  $G$ . It prints the word  $w$  in terms of the elements of  $G$ .

`TietzeReduction(S, w)` Inputs a set  $S$  of words in a free  $ZG$ -module, and a word  $w$  in the module. The function returns a reduced version of  $w$  in which no two adjacent letters are inverses of each other, and no two adjacent letters are in  $S$ .

# Chapter 14

## *FpG*-modules

`DirectSumOfFpGModules (M, N)`  
`DirectSumOfFpGModules ([ M[1], M[2], ..., M[k] ])` Inputs two *FpG*-modules  $M$  and  $N$  with common group  $G$ . It returns a list of  $FpG$ -modules.  
`FpGModule (A, P)`  
`FpGModule (A, G, p)` Inputs a  $p$ -group  $P$  and a matrix  $A$  whose rows have length a multiple of the order of  $G$ . It returns an *FpG*-module.  
`FpGModuleDualBasis (M)` Inputs an *FpG*-module  $M$ . It returns a record  $R$  with two components:  $R.freeModule$  is the free module and  $R.dualBasis$  is a dual basis.  
`FpGModuleHomomorphism (M, N, A)`  
`FpGModuleHomomorphismNC (M, N, A)` Inputs *FpG*-modules  $M$  and  $N$  over a common  $p$ -group  $G$ . Also inputs a list  $A$  of matrices representing the homomorphism.  
`DesuspensionFpGModule (M, n)`  
`DesuspensionFpGModule (R, n)` Inputs a positive integer  $n$  and an *FpG*-module  $M$ . It returns an *FpG*-module  $D^n M$ .  
`RadicalOfFpGModule (M)` Inputs an *FpG*-module  $M$  with  $G$  a  $p$ -group, and returns the Radical of  $M$  as an *FpG*-module.  
`GeneratorsOfFpGModule (M)` Inputs an *FpG*-module  $M$  and returns a matrix whose rows correspond to a minimal generating set.  
`ImageOfFpGModuleHomomorphism (f)` Inputs an *FpG*-module homomorphism  $f : M \rightarrow N$  and returns its image  $f(M)$ .  
`IntersectionOfFpGModules (M, N)` Inputs two *FpG*-modules  $M, N$  arising as submodules in a common free module and returns their intersection.  
`IsFpGModuleHomomorphismData (M, N, A)` Inputs *FpG*-modules  $M$  and  $N$  over a common  $p$ -group  $G$ . Also inputs a list  $A$  of matrices representing the homomorphism.  
`MultipleOfFpGModule (w, M)` Inputs an *FpG*-module  $M$  and a list  $w := [g_1, \dots, g_t]$  of elements in the group  $G = M!.g$ . It returns a list of  $M$ .  
`ProjectedFpGModule (M, k)` Inputs an *FpG*-module  $M$  of ambient dimension  $n|G|$ , and an integer  $k$  between 1 and  $n$ . It returns a list of  $M$ .  
`RandomHomomorphismOfFpGModules (M, N)` Inputs two *FpG*-modules  $M$  and  $N$  over a common group  $G$ . It returns a random homomorphism.  
`Rank (f)` Inputs an *FpG*-module homomorphism  $f : M \rightarrow N$  and returns the dimension of the image of  $f$  as a vector space.  
`SumOfFpGModules (M, N)` Inputs two *FpG*-modules  $M, N$  arising as submodules in a common free module  $(FG)^n$  and returns their sum.  
`SumOp (f, g)` Inputs two *FpG*-module homomorphisms  $f, g : M \rightarrow N$  with common source and common target. It returns their sum.  
`VectorsToFpGModuleWords (M, L)` Inputs an *FpG*-module  $M$  and a list  $L = [v_1, \dots, v_k]$  of vectors in  $M$ . It returns a list of words.

## Chapter 15

# Meataxe modules

•  
DesuspensionMtxModule( $M$ ) Inputs a meataxe module  $M$  over the field of  $p$  elements and returns an FpG-module  $DM$ .

FpG\_to\_MtxModule( $M$ ) Inputs an FpG-module  $M$  and returns an isomorphic meataxe module.

GeneratorsOfMtxModule( $M$ ) Inputs a meataxe module  $M$  acting on, say, the vector space  $V$ . The function returns a m

## Chapter 16

# Coxeter diagrams and graphs of groups

`CoxeterDiagramComponents(D)` Inputs a Coxeter diagram  $D$  and returns a list  $[D_1, \dots, D_d]$  of the maximal connected components of  $D$ .

`CoxeterDiagramDegree(D, v)` Inputs a Coxeter diagram  $D$  and vertex  $v$ . It returns the degree of  $v$  (i.e. the number of neighbors of  $v$  in  $D$ ).

`CoxeterDiagramDisplay(D)` Inputs a Coxeter diagram  $D$  and displays it as a .gif file. It uses the `GraphOfGroupsDisplay` function.

`CoxeterDiagramDisplay(D, "web browser")` Inputs a Coxeter diagram  $D$  and displays it as a .gif file. It uses the `GraphOfGroupsDisplay` function.

`CoxeterDiagramFpArtinGroup(D)` Inputs a Coxeter diagram  $D$  and returns the corresponding finitely presented Artin group.

`CoxeterDiagramFpCoxeterGroup(D)` Inputs a Coxeter diagram  $D$  and returns the corresponding finitely presented Coxeter group.

`CoxeterDiagramIsSpherical(D)` Inputs a Coxeter diagram  $D$  and returns "true" if the associated Coxeter groups is spherical.

`CoxeterDiagramMatrix(D)` Inputs a Coxeter diagram  $D$  and returns a matrix representation of it. The matrix is given by  $(m_{ij})$  where  $m_{ij}$  is the order of  $s_i s_j$ .

`CoxeterSubDiagram(D, V)` Inputs a Coxeter diagram  $D$  and a subset  $V$  of its vertices. It returns the full sub-diagram of  $D$  with vertices  $V$ .

`CoxeterDiagramVertices(D)` Inputs a Coxeter diagram  $D$  and returns its set of vertices.

`EvenSubgroup(G)` Inputs a group  $G$  and returns a subgroup  $G^+$ . The subgroup is that generated by all products  $xy$  where  $x, y \in G$ .

`GraphOfGroupsDisplay(D)` Inputs a graph of groups  $D$  and displays it as a .gif file. It uses the `GraphOfGroupsDisplay` function.

`GraphOfGroupsDisplay(D, "web browser")` Inputs a graph of groups  $D$  and displays it as a .gif file. It uses the `GraphOfGroupsDisplay` function.

`GraphOfGroupsTest(D)` Inputs an object  $D$  and tries to test whether it is a Graph of Groups. However, it DOES NOT work.

## Chapter 17

### Some functions for accessing basic data

`BoundaryMap(C)` Inputs a resolution, chain complex or cochain complex  $C$  and returns the function  $C!.boundary$ .

`BoundaryMatrix(C, n)` Inputs a chain or cochain complex  $C$  and integer  $n > 0$ . It returns the  $n$ -th boundary map of  $C$ .

`Dimension(C)`

`Dimension(M)` Inputs a resolution, chain complex or cochain complex  $C$  and returns the function  $C!.dimension$ . Also returns the dimension of the module  $M$ .

`EvaluateProperty(X, "name")` Inputs a component object  $X$  (such as a  $ZG$ -resolution or chain map) and a string "name". Returns the value of the property "name" of  $X$ .

`GroupOfResolution(R)` Inputs a  $ZG$ -resolution  $R$  and returns the group  $G$ .

`Length(R)` Inputs a resolution  $R$  and returns its length (i.e. the number of terms of  $R$  that HAP has computed).

`Map(f)` Inputs a chain map, or cochain map or equivariant chain map  $f$  and returns the mapping function (as opposed to the map itself).

`Source(f)` Inputs a chain map, or cochain map, or equivariant chain map, or  $FpG$ -module homomorphism  $f$  and returns the source module.

`Target(f)` Inputs a chain map, or cochain map, or equivariant chain map, or  $FpG$ -module homomorphism  $f$  and returns the target module.

## Chapter 18

# Parallel Computation - Core Functions

ChildProcess ()

ChildProcess ("computer.ac.wales") This starts a GAP session as a child process and returns a stream to the child p

- open a shell on thishost
- cd .ssh
- ls
- *! if id<sub>d</sub>sa, id<sub>r</sub>saet cexists, skipthenexttwosteps!*
- ssh-keygen -t rsa
- ssh-keygen -t dsa
- scp \*.pub user@remotehost: /
- ssh remotehost -l user
- cat id<sub>r</sub>sa.pub >> .ssh/authorized<sub>k</sub>keys
- cat id<sub>d</sub>sa.pub >> .ssh/authorized<sub>k</sub>keys
- rm id<sub>r</sub>sa.pub id<sub>d</sub>sa.pub
- exit

You should now be able to connect from "thishost" to "remotehost" without a password prompt.)

ChildClose(s) This closes the stream s to a child GAP process.

ChildCommand("cmd"; s) This runs a GAP command "cmd;" on the child process accessed by the stream s. Here "cm

NextAvailableChild(L) Inputs a list L of child processes and returns a child in L which is ready for computation (as

IsAvailableChild(s) Inputs a child process s and returns true if s is currently available for computations, and false o

ChildPut(A, "B", s) This copies a GAP object A on the parent process to an object B on the child process s. (The cop

ChildGet("A", s) This functions copies a GAP object A on the child process s and returns it on the parent process. (T

## Chapter 19

# Parallel Computation - Extra Functions

`ChildFunction("function(arg);", s)` This runs the GAP function "function(arg);" on a child process accessed by the

`ChildRead(s)` This returns, as a string, the output of the last application of `ChildFunction("function(arg);", s)`.

`ChildReadEval(s)` This returns, as an evaluated string, the output of the last application of `ChildFunction("function(arg);", s)`.

`ParallelList(I, fn, L)` Inputs a list  $I$ , a function  $fn$  such that  $fn(x)$  is defined for all  $x$  in  $I$ , and a list of children  $L$ . It

## Chapter 20

# Topological Data Analysis

`MatrixToTopologicalSpace(A, n)` Inputs an integer matrix  $A$  and an integer  $n$ . It returns a 2-dimensional topological space.

`ReadImageAsTopologicalSpace("file.png", n)` `ReadImageAsTopologicalSpace("file.png", [m, n])` Reads an image file ("file.png") and returns a topological space.

`ReadImageAsMatrix("file.png")` Reads an image file ("file.png", "file.eps", "file.bmp" etc) and returns an integer matrix.

`WriteTopologicalSpaceAsImage(T, "filename", "ext")` Inputs a 2-dimensional topological space  $T$ , and a filename "filename" and an extension "ext".

`ViewTopologicalSpace(T)` `ViewTopologicalSpace(T, "mozilla")` Inputs a topological space  $T$ , and optionally a browser name "mozilla".

`BettiNumbers(T, n)` `BettiNumbers(T)` Inputs a topological space  $T$  and a non-negative integer  $n$ . It returns the  $n$ -th Betti number of  $T$ .

`PathComponent(T, n)` Inputs a topological space  $T$  and an integer  $n$  in the range  $0, \dots, \text{BettiNumbers}(T, 0)$ . It returns the  $n$ -th path component of  $T$ .

`SingularChainComplex(A)` Inputs a topological space  $T$  and returns a (usually very large) integral chain complex that computes the homology of  $T$ .

`ContractTopologicalSpace(T)` Inputs a topological space  $T$  of dimension  $d$  and removes  $d$ -dimensional cells from  $T$ .

`BoundaryTopologicalSpace(T)` Inputs a topological space  $T$  and returns its boundary as a topological space.

`BoundarySingularities(T)` Inputs a topological space  $T$  and returns the subspace of points in the boundary where the boundary is singular.

`ThickenedTopologicalSpace(T)` `ThickenedTopologicalSpace(T, n)` Inputs a topological space  $T$  and returns a thickened version of  $T$ .

`ComplementTopologicalSpace(T)` Inputs a topological space  $T$  and returns a topological space  $S$ . A euclidean point  $x$  is in  $S$  if and only if  $x$  is not in  $T$ .

`ConcatenatedTopologicalSpace(L)` Inputs a list  $L$  of topological spaces whose underlying arrays of numbers all have the same length.

## Chapter 21

### Pseudo lists

`Add(L, x)` Let  $L$  be a pseudo list of length  $n$ , and  $x$  an object compatible with the entries in  $L$ . If  $x$  is not in  $L$  then this operation adds  $x$  to the end of  $L$ .

`Append(L, K)` Let  $L$  be a pseudo list and  $K$  a list whose objects are compatible with those in  $L$ . This operation appends the elements of  $K$  to the end of  $L$ .

`ListToPseudoList(L)` Inputs a list  $L$  and returns the pseudo list representation of  $L$ .

## Chapter 22

### Miscellaneous

`BigStepLCS(G, n)` Inputs a group  $G$  and a positive integer  $n$ . It returns a subseries  $G = L_1 > L_2 > \dots > L_k = 1$  of the lower central series of  $G$ .

`Compose(f, g)` Inputs two  $FpG$ -module homomorphisms  $f : M \rightarrow N$  and  $g : L \rightarrow M$  with  $Source(f) = Target(g)$ .

`HAPcopyright()` This function provides details of HAP'S GNU public copyright licence.

`IsLieAlgebraHomomorphism(f)` Inputs an object  $f$  and returns true if  $f$  is a homomorphism  $f : A \rightarrow B$  of Lie algebras.

`IsSuperperfect(G)` Inputs a group  $G$  and returns "true" if both the first and second integral homology of  $G$  is trivial.

`MakeHAPManual()` This function creates the manual for HAP from an XML file.

`PermToMatrixGroup(G, n)` Inputs a permutation group  $G$  and its degree  $n$ . Returns a bijective homomorphism  $f : G \rightarrow GL(n, F)$ .

`SolutionsMatDestructive(M, B)` Inputs an  $m \times n$  matrix  $M$  and a  $k \times n$  matrix  $B$  over a field. It returns a  $k \times m$  matrix  $C$  such that  $CM = B$ .

`TestHap()` This runs a representative sample of HAP functions and checks to see that they produce the correct output.

# Index

- Add, 24
- AddFreeWords, 16
- AddFreeWordsModP, 16
- AlgebraicReduction, 16
- Append, 24
  
- BaerInvariant, 11
- BettiNumbers, 23
- BigStepLCS, 25
- BoundaryMap, 20
- BoundaryMatrix, 20
- BoundarySingularities, 23
- BoundaryTopologicalSpace, 23
  
- CayleyGraphDisplay, 12
- ChevalleyEilenbergComplex, 7
- ChildClose, 21
- ChildCommand, 21
- ChildFunction, 22
- ChildGet, 21
- ChildProcess, 21
- ChildPut, 21
- ChildRead, 22
- ChildReadEval, 22
- Coclass, 11
- CocycleCondition, 15
- Cohomology, 8
- CohomologyPrimePart, 8
- ComplementTopologicalSpace, 23
- Compose(f,g), 25
- ConcatenatedTopologicalSpace, 23
- ContractTopologicalSpace, 23
- CoxeterDiagramComponents, 19
- CoxeterDiagramDegree, 19
- CoxeterDiagramDisplay, 19
- CoxeterDiagramFpArtinGroup, 19
- CoxeterDiagramFpCoxeterGroup, 19
- CoxeterDiagramIsSpherical, 19
- CoxeterDiagramMatrix, 19
  
- CoxeterDiagramVertices, 19
- CoxeterSubDiagram, 19
  
- DesuspensionFpGModule, 17
- DesuspensionMtxModule, 18
- Dimension, 20
- DirectSumOfFpGModules, 17
  
- EpiCentre, 11
- EquivariantChainMap, 5
- EvaluateProperty, 20
- EvenSubgroup, 19
- ExpansionOfRationalFunction, 9
  
- FpGModule, 17
- FpGModuleDualBasis, 17
- FpGModuleHomomorphism, 17
- FpG\_to\_MtxModule, 18
- Fundamental domains (HAPcryst), 13
  
- GeneratorsOfFpGModule, 17
- GeneratorsOfMtxModule, 18
- GraphOfGroupsDisplay, 19
- GraphOfGroupsTest, 19
- GroupCohomology, 8
- GroupHomology, 8
- GroupOfResolution, 20
  
- HAPcopyright, 25
- Homology, 8
- HomologyPb, 8
- HomologyPrimePart, 8
- HomToIntegers, 6
- HomToIntegersModP, 6
- HomToIntegralModule, 6
  
- ImageOfFpGModuleHomomorphism, 17
- IntegralCupProduct, 10
- IntegralRingGenerators, 10
- IntersectionOfFpGModules, 17
- IsAspherical, 12

[IsAvailableChild](#), 21  
[IsFpGModuleHomomorphismData](#), 17  
[IsLieAlgebraHomomorphism](#), 25  
[IsSuperperfect](#), 25  
  
[LeibnizAlgebraHomology](#), 8  
[LeibnizComplex](#), 7  
[Length](#), 20  
[LieAlgebraHomology](#), 8  
[ListToPseudoList](#), 24  
[LowerCentralSeriesLieAlgebra](#), 6  
  
[MakeHAPManual](#), 25  
[Map](#), 20  
[MatrixToTopologicalSpace](#), 23  
[ModPCohomologyGenerators](#), 10  
[ModPCohomologyRing](#), 10  
[ModP RingGenerators](#), 10  
[MultipleOfFpGModule](#), 17  
[MultiplyWord](#), 16  
  
[Negate](#), 16  
[NegateWord](#), 16  
[NextAvailableChild](#), 21  
[NonabelianExteriorProduct](#), 11  
[NonabelianTensorProduct](#), 11  
[NonabelianTensorSquare](#), 11  
  
[OrbitPolytope](#), 13  
  
[ParallelList](#), 22  
[PathComponent](#), 23  
[PermToMatrixGroup](#), 25  
[PoincareSeries](#), 9  
[PoincareSeriesPrimePart](#), 9  
[PolytopalComplex](#), 13  
[PolytopalGenerators](#), 13  
[Prank](#), 9  
[PresentationOfResolution](#), 12  
[PrimePartDerivedFunctor](#), 8  
[PrintZGword](#), 16  
[ProjectedFpGModule](#), 17  
  
[RadicalOfFpGModule](#), 17  
[RandomHomomorphismOfFpGModules](#), 17  
[Rank](#), 17  
[RankHomologyPGroup](#), 8  
[RankPrimeHomology](#), 8  
  
[ReadImageAsMatrix](#), 23  
[ReadImageAsTopologicalSpace](#), 23  
[RelativeSchurMultiplier](#), 11  
[ResolutionAbelianGroup](#), 3  
[ResolutionAlmostCrystalGroup](#), 3  
[ResolutionAlmostCrystalQuotient](#), 3  
[ResolutionArtinGroup](#), 3  
[ResolutionAsphericalPresentation](#), 3  
[ResolutionBieberbachGroup \(HAPcryst\)](#), 3  
[ResolutionDirectProduct](#), 3  
[ResolutionExtension](#), 3  
[ResolutionFiniteDirectProduct](#), 3  
[ResolutionFiniteExtension](#), 3  
[ResolutionFiniteGroup](#), 3  
[ResolutionFiniteSubgroup](#), 3  
[ResolutionFpGModule](#), 4  
[ResolutionGraphOfGroups](#), 3  
[ResolutionNilpotentGroup](#), 3  
[ResolutionNormalSeries](#), 3  
[ResolutionPrimePowerGroup](#), 3  
[ResolutionSmallFpGroup](#), 3  
[ResolutionSubgroup](#), 3  
[ResolutionSubnormalSeries](#), 3  
  
[SingularChainComplex](#), 23  
[SolutionsMatDestructive](#), 25  
[Source](#), 20  
[StandardCocycle](#), 15  
[SumOfFpGModules](#), 17  
[SumOp](#), 17  
[Syzygy](#), 15  
  
[Target](#), 20  
[TensorCentre](#), 11  
[TensorWithIntegers](#), 6  
[TensorWithIntegersModP](#), 6  
[TensorWithRationals](#), 6  
[TestHap](#), 25  
[ThickenedTopologicalSpace](#), 23  
[ThirdHomotopyGroupOfSuspensionB](#), 11  
[TietzeReduction](#), 16  
[TorsionGeneratorsAbelianGroup](#), 12  
[TwistedTensorProduct](#), 3  
  
[UpperEpicentralSeries](#), 11  
  
[VectorStabilizer](#), 14  
[VectorsToFpGModuleWords](#), 17

`ViewTopologicalSpace`, [23](#)

`WriteTopologicalSpaceAsImage`, [23](#)