

Contents

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

21 Topological Data Analysis	24
22 Pseudo lists	25
23 Miscellaneous	26

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 a resolution of the abelian group $\mathbb{Z}^d / \langle L \rangle$.

`ResolutionAlmostCrystalGroup(G,n)` Inputs a positive integer n and an almost crystallographic pcg group G . It returns a resolution of the group G .

`ResolutionAlmostCrystalQuotient(G,n,c)` `ResolutionAlmostCrystalQuotient(G,n,c,false)` An almost crystallographic pcg group G and a positive integer n . It returns a resolution of the group G .

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

`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 a resolution of the group G .

`ResolutionBieberbachGroup(G)` `ResolutionBieberbachGroup(G,v)` Inputs a Bieberbach group G (represented using `AffineCrystGroupOnRight`) and a positive integer v . It returns a resolution of the group G .

`ResolutionDirectProduct(R,S)` `ResolutionDirectProduct(R,S,"internal")` Inputs a ZG -resolution R and a ZG -resolution S . It returns a resolution of the direct product of the groups represented by R and S .

`ResolutionExtension(g,R,S)` `ResolutionExtension(g,R,S,"TestFiniteness")` `ResolutionExtension(g,R,S,true)` Inputs a group g , a ZG -resolution R , and a ZG -resolution S . It returns a resolution of the extension of g by S .

`ResolutionFiniteDirectProduct(R,S)` `ResolutionFiniteDirectProduct(R,S,"internal")` Inputs a ZG -resolution R and a ZG -resolution S . It returns a resolution of the direct product of the groups represented by R and S .

`ResolutionFiniteExtension(gensE,gensG,R,n)` `ResolutionFiniteExtension(gensE,gensG,R,n,true)` Inputs a list of generators $gensE$, a list of generators $gensG$, a ZG -resolution R , and a positive integer n . It returns a resolution of the extension of g by S .

`ResolutionFiniteGroup(gens,n)` `ResolutionFiniteGroup(gens,n,true)` `ResolutionFiniteGroup(gens,n,true)` Inputs a list of generators $gens$ and a positive integer n . It returns a resolution of the group G .

`ResolutionFiniteSubgroup(R,K)` `ResolutionFiniteSubgroup(R,gensG,gensK)` Inputs a ZG -resolution R for a group G and a list of generators $gensK$ for a subgroup K of G . It returns a resolution of the subgroup K .

`ResolutionGraphOfGroups(D,n)` `ResolutionGraphOfGroups(D,n,L)` Inputs a graph of groups D and a positive integer n . It returns a resolution of the group G .

`ResolutionNilpotentGroup(G,n)` `ResolutionNilpotentGroup(G,n,"TestFiniteness")` Inputs a nilpotent group G and a positive integer n . It returns a resolution of the group G .

`ResolutionNormalSeries(L,n)` `ResolutionNormalSeries(L,n,true)` `ResolutionNormalSeries(L,n,false)` Inputs a list of subgroups L and a positive integer n . It returns a resolution of the group G .

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

`ResolutionSmallFpGroup(G,n)` `ResolutionSmallFpGroup(G,n,p)` Inputs a small finitely presented group G and integer $n > 0$. It returns a resolution of the group G .

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

`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 a resolution of the group G .

`TwistedTensorProduct(R,S,EhomG,GmapE,NhomE,NEhomN,EltSE,Mult,InvE)` Inputs a ZG -resolution R , a ZN -resolution S , and a list of maps E . It returns a resolution of the tensor product of R and 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 $f : G \longrightarrow G'$

Chapter 4

Functors

•

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

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

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

`LowerCentralSeriesLieAlgebra(G)` `LowerCentralSeriesLieAlgebra(f)` Inputs a pcg group G . If each quotient G_i/G_{i+1} is

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

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

`TensorWithRationals(R)` Inputs a ZG -resolution R and returns the chain complex obtained by tensoring with the trivial ZG -module \mathbb{Q} .

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) or

`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 $X = (C \longrightarrow D)$ over the integers Z . If $X = C$ then it returns the cohomology groups $H^n(C, Z)$.
`CohomologyPrimePart(C, n, p)` Inputs a cochain complex C in characteristic 0, a positive integer n , and a prime p . It returns the p -part of the cohomology group $H^n(C, Z)$.
`GroupCohomology(X, n)` `GroupCohomology(X, n, p)` Inputs a positive integer n and either a finite group $X = G$ or a Coxeter diagram $X = D$ representing a finite group. It returns the cohomology group $H^n(G, Z)$ or $H^n(G, F_p)$ respectively.
`GroupHomology(X, n)` `GroupHomology(X, n, p)` Inputs a positive integer n and either a finite group $X = G$ or a Coxeter diagram $X = D$ representing a finite group. It returns the homology group $H_n(G, Z)$ or $H_n(G, F_p)$ respectively.
`Homology(X, n)` Inputs either a chain complex $X = C$ or a chain map $X = (C \longrightarrow D)$. If $X = C$ then the torsion coefficients of the homology groups $H_n(C, Z)$ are returned.
`HomologyPb(C, n)` This is a back-up function which might work in some instances where `Homology(C, n)` fails. It is more computationally intensive.
`HomologyPrimePart(C, n, p)` Inputs a chain complex C in characteristic 0, a positive integer n , and a prime p . It returns the p -part of the homology group $H_n(C, Z)$.
`LeibnizAlgebraHomology(A, n)` Inputs a Lie or Leibniz algebra $X = A$ (over the ring of integers Z or over a field K), and a positive integer n . It returns the homology group $H_n(A, Z)$ or $H_n(A, K)$ respectively.
`LieAlgebraHomology(A, n)` Inputs a Lie algebra A (over the integers or a field) and a positive integer n . It returns the homology group $H_n(A, Z)$ or $H_n(A, K)$ respectively.
`PrimePartDerivedFunctor(G, R, F, n)` Inputs a finite group G , a positive integer n , at least $n + 1$ terms of a ZP -resolution of Z over R , and a field F . It returns the p -part of the homology group $H_n(G, F)$.
`RankHomologyPGroup(G, n)` `RankHomologyPGroup(R, n)` `RankHomologyPGroup(G, n, "empirical")` Inputs a (smallish) p -group G , a positive integer n , and a flag "empirical". It returns the rank of the homology group $H_n(G, F_p)$.
`RankPrimeHomology(G, n)` Inputs a (smallish) p -group G together with a positive integer n . It returns a function $dim(k)$ which gives the rank of the homology group $H_n(G, F_p)$ for any prime p .

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 not zero.

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 $f(x) = P_n(x)/Q_n(x)$.

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

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 resolution.
`ModPCohomologyGenerators (G, n)`
`ModPCohomologyGenerators (R)` Inputs either a p -group G and positive integer n , or else n terms of a minimal $Z_p G$ -resolution.
`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 $Z_p G$ -resolution.
`ModPRingGenerators (A)` Inputs a mod p cohomology ring A (created using the preceding function). It returns a minimal resolution.

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 $H_2(N, Z)$.
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

Lie commutators and nonabelian Lie tensors

•
All functions on this page were implemented by Hamid Mohammadzadeh.

`LieCoveringHomomorphism(L)` Inputs a finite dimensional Lie algebra L over a field, and returns a surjective Lie homomorphism π such that $\pi(L)$ is a free Lie algebra.

`LieEpiCentre(L)` Inputs a finite dimensional Lie algebra L over a field, and returns an ideal $Z^*(L)$ of the centre of L .

`LieExteriorSquare(L)` Inputs a finite dimensional Lie algebra L over a field. It returns a record E with the following fields:

`LieTensorSquare(L)` Inputs a finite dimensional Lie algebra L over a field and returns a record T with the following fields:

`TensorCentre(L)` Inputs a finite dimensional Lie algebra L over a field and returns the largest ideal N such that the quotient L/N is a free Lie algebra.

Chapter 11

Generators and relators of groups

•

`CayleyGraphDisplay(G,X)`

`CayleyGraphDisplay(G,X,"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 with presentation $\langle F, R \rangle$.

`PresentationOfResolution(R)` Inputs at least two terms of a reduced ZG -resolution R and returns a record P with the presentation of the resolution.

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

Chapter 12

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 13

Cocycles

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

Chapter 14

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 , then $v + w$ 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 subword is a power of a generator.

`AlgebraicReduction(w, p)` Inputs a word w in a free ZG -module and returns a reduced version of the word in which no subword is a power of a generator, and the result is reduced modulo p .

`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 subword is in S .

Chapter 15

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 the direct sum of M and N .
`FpGModule (A, 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)` Inputs *FpG*-modules M and N over a common p -group G . Also inputs a list A of coefficients. It returns a homomorphism.
`DesuspensionFpGModule (M, n)` Inputs an *FpG*-module M and a positive integer n . It returns an *FpG*-module $D^n M$.
`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. It 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 coefficients. It returns a boolean.
`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 elements.
`ProjectedFpGModule (M, k)` Inputs an *FpG*-module M of ambient dimension $n|G|$, and an integer k between 1 and n . It returns a projected module.
`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 $(FG)^n$ where n is the ambient dimension. It 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 16

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 17

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 edges incident to v).

`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 are elements of 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 18

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 $C!.dimension$. Alternatively, `Dimension(M)` returns the dimension of the resolution M .

`EvaluateProperty(X,"name")` Inputs a component object X (such as a ZG -resolution or chain map) and a string "name". It 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 mapping object).

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

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

Chapter 19

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 process.

- open a shell on thishost
- `cd .ssh`
- `ls`
- *if id_{dsa}, id_{rsa} exist, skip the next two steps!*
- `ssh-keygen -t rsa`
- `ssh-keygen -t dsa`
- `scp *.pub user@remotehost: /`
- `ssh remotehost -l user`
- `cat idrsa.pub >> .ssh/authorizedkeys`
- `cat iddsa.pub >> .ssh/authorizedkeys`
- `rm idrsa.pub iddsa.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 "cmd" is a GAP command.

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

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

`ChildPut(A, "B", s)` This copies a GAP object `A` on the parent process to an object `B` on the child process `s`. (The copy is made immediately.)

`ChildGet("A", s)` This function copies a GAP object `A` on the child process `s` and returns it on the parent process. (The copy is made immediately.)

Chapter 20

Parallel Computation - Extra Functions

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

`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 returns a list of the same length as I , where the i -th element is the result of applying fn to the i -th element of I , using the child process $L[i]$.

Chapter 21

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 of dimension n .

`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". It writes the space T to the file "filename.ext".

`ViewTopologicalSpace(T)` `ViewTopologicalSpace(T, "mozilla")` Inputs a topological space T , and optionally a browser name "mozilla". It displays the space T in a browser window.

`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 of thickness n .

`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 dimension.

Chapter 22

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 23

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 $f : M \longrightarrow N$ and $g : L \longrightarrow M$ 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 $f : A \longrightarrow B$ 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 $f : G \longrightarrow GL(n, F)$.

`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, [25](#)
- AddFreeWords, [17](#)
- AddFreeWordsModP, [17](#)
- AlgebraicReduction, [17](#)
- Append, [25](#)

- BaerInvariant, [11](#)
- BettiNumbers, [24](#)
- BigStepLCS, [26](#)
- BoundaryMap, [21](#)
- BoundaryMatrix, [21](#)
- BoundarySingularities, [24](#)
- BoundaryTopologicalSpace, [24](#)

- CayleyGraphDisplay, [13](#)
- ChevalleyEilenbergComplex, [7](#)
- ChildClose, [22](#)
- ChildCommand, [22](#)
- ChildFunction, [23](#)
- ChildGet, [22](#)
- ChildProcess, [22](#)
- ChildPut, [22](#)
- ChildRead, [23](#)
- ChildReadEval, [23](#)
- Coclass, [11](#)
- CocycleCondition, [16](#)
- Cohomology, [8](#)
- CohomologyPrimePart, [8](#)
- ComplementTopologicalSpace, [24](#)
- Compose(f,g), [26](#)
- ConcatenatedTopologicalSpace, [24](#)
- ContractTopologicalSpace, [24](#)
- CoxeterDiagramComponents, [20](#)
- CoxeterDiagramDegree, [20](#)
- CoxeterDiagramDisplay, [20](#)
- CoxeterDiagramFpArtinGroup, [20](#)
- CoxeterDiagramFpCoxeterGroup, [20](#)
- CoxeterDiagramIsSpherical, [20](#)
- CoxeterDiagramMatrix, [20](#)

- CoxeterDiagramVertices, [20](#)
- CoxeterSubDiagram, [20](#)

- DesuspensionFpGModule, [18](#)
- DesuspensionMtxModule, [19](#)
- Dimension, [21](#)
- DirectSumOfFpGModules, [18](#)

- EpiCentre, [11](#)
- EquivariantChainMap, [5](#)
- EvaluateProperty, [21](#)
- EvenSubgroup, [20](#)
- ExpansionOfRationalFunction, [9](#)

- FpGModule, [18](#)
- FpGModuleDualBasis, [18](#)
- FpGModuleHomomorphism, [18](#)
- FpG_to_MtxModule, [19](#)
- Fundamental domains (HAPcryst), [14](#)

- GeneratorsOfFpGModule, [18](#)
- GeneratorsOfMtxModule, [19](#)
- GraphOfGroupsDisplay, [20](#)
- GraphOfGroupsTest, [20](#)
- GroupCohomology, [8](#)
- GroupHomology, [8](#)
- GroupOfResolution, [21](#)

- HAPcopyright, [26](#)
- Homology, [8](#)
- HomologyPb, [8](#)
- HomologyPrimePart, [8](#)
- HomToIntegers, [6](#)
- HomToIntegersModP, [6](#)
- HomToIntegralModule, [6](#)

- ImageOfFpGModuleHomomorphism, [18](#)
- IntegralCupProduct, [10](#)
- IntegralRingGenerators, [10](#)
- IntersectionOfFpGModules, [18](#)
- IsAspherical, [13](#)

IsAvailableChild, 22
 IsFpGModuleHomomorphismData, 18
 IsLieAlgebraHomomorphism, 26
 IsSuperperfect, 26

 LeibnizAlgebraHomology, 8
 LeibnizComplex, 7
 Length, 21
 LieAlgebraHomology, 8
 LieCoveringHomomorphism, 12
 LieEpiCentre, 12
 LieExteriorSquare, 12
 LieTensorSquare, 12
 ListToPseudoList, 25
 LowerCentralSeriesLieAlgebra, 6

 MakeHAPManual, 26
 Map, 21
 MatrixToTopologicalSpace, 24
 ModPCohomologyGenerators, 10
 ModPCohomologyRing, 10
 ModP RingGenerators, 10
 MultipleOfFpGModule, 18
 MultiplyWord, 17

 Negate, 17
 NegateWord, 17
 NextAvailableChild, 22
 NonabelianExteriorProduct, 11
 NonabelianTensorProduct, 11
 NonabelianTensorSquare, 11

 OrbitPolytope, 14

 ParallelList, 23
 PathComponent, 24
 PermToMatrixGroup, 26
 PoincareSeries, 9
 PoincareSeriesPrimePart, 9
 PolytopalComplex, 14
 PolytopalGenerators, 14
 Prank, 9
 PresentationOfResolution, 13
 PrimePartDerivedFunctor, 8
 PrintZGword, 17
 ProjectedFpGModule, 18

 RadicalOfFpGModule, 18

 RandomHomomorphismOfFpGModules, 18
 Rank, 18
 RankHomologyPGroup, 8
 RankPrimeHomology, 8
 ReadImageAsMatrix, 24
 ReadImageAsTopologicalSpace, 24
 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, 24
 SolutionsMatDestructive, 26
 Source, 21
 StandardCocycle, 16
 SumOfFpGModules, 18
 SumOp, 18
 Syzygy, 16

 Target, 21
 TensorCentre, 11, 12
 TensorWithIntegers, 6
 TensorWithIntegersModP, 6
 TensorWithRationals, 6
 TestHap, 26
 ThickenedTopologicalSpace, 24
 ThirdHomotopyGroupOfSuspensionB, 11
 TietzeReduction, 17
 TorsionGeneratorsAbelianGroup, 13
 TwistedTensorProduct, 3

UpperEpicentralSeries, [11](#)

VectorStabilizer, [15](#)

VectorsToFpGModuleWords, [18](#)

ViewTopologicalSpace, [24](#)

WriteTopologicalSpaceAsImage, [24](#)