Homological Algebra Programming

HAP

Version 1.79

01 September 2026

Graham Ellis

Graham Ellis Email: Graham.Ellis@nuigalway.ie

Address: School of Mathematics
National University of Irelnd, Galway
Galway
(Ireland)

Abstract

HAP is a homological algebra library for use with the GAP computer algebra system, and is still under development. The current version 1.79 was released on 01 September 2026 .
The initial focus of the library was on computations related to the cohomology of finite and infinite groups, with particular emphasis on integral coefficients. The focus has since broadened to include Steenrod algebras of finite groups, Bredon homology, cohomology of simplicial groups, and general computations in algebraic topology relating to finite CW-complexes, covering spaces, knots, knotted surfaces, and topics such as persitent homology arising in topological data analysis.
This document describes the functions available in HAP. Examples illustrating these functions are available in the HAP tutorial.

Copyright

© 2005-2019 by Graham Ellis

Acknowledgements

The HAP project has been supported by the School of Maths at NUI Galway, various PhD students and postdoctoral researchers at NUI Galway, the Irish Research Council, Science Foundation Ireland, and the EU Marie Curie programme.

Contents

1 Basic functionality for cellular complexes, fundamental groups and homology **7**
- 1.1 Data ⟶ Cellular Complexes . . . 7
- 1.2 Metric Spaces . . . 12
- 1.3 Cellular Complexes ⟶ Cellular Complexes . . . 13
- 1.4 Cellular Complexes ⟶ Cellular Complexes (Preserving Data Types) . . . 16
- 1.5 Cellular Complexes ⟶ Homotopy Invariants . . . 19
- 1.6 Data ⟶ Homotopy Invariants . . . 22
- 1.7 Cellular Complexes ⟶ Non Homotopy Invariants . . . 22
- 1.8 (Co)chain Complexes ⟶ (Co)chain Complexes . . . 24
- 1.9 (Co)chain Complexes ⟶ Homotopy Invariants . . . 25
- 1.10 Visualization . . . 27

2 Basic functionality for $\mathbb{Z}G$-resolutions and group cohomology **30**
- 2.1 Resolutions . . . 30
- 2.2 Algebras ⟶ (Co)chain Complexes . . . 32
- 2.3 Resolutions ⟶ (Co)chain Complexes . . . 32
- 2.4 Cohomology rings . . . 34
- 2.5 Group Invariants . . . 36
- 2.6 $\mathbb{F}_p$-modules . . . 38

3 Basic functionality for homological group theory **39**
- 3.1 Cocycles . . . 39
- 3.2 G-Outer Groups . . . 40
- 3.3 G-cocomplexes . . . 41

4 Basic functionality for parallel computation **42**
- 4.1 Six Core Functions . . . 42

5 Resolutions of the ground ring **44**
- 5.1 . . . 44

6 Resolutions of modules **53**
- 6.1 . . . 53

7 Induced equivariant chain maps **54**
- 7.1 . . . 54

8 Functors **55**
8.1 55

9 Chain complexes **59**
9.1 59

10 Sparse Chain complexes **62**
10.1 62

11 Homology and cohomology groups **65**
11.1 65

12 Poincare series **73**
12.1 73

13 Cohomology ring structure **75**
13.1 75

14 Cohomology rings of p-groups (mainly $p = 2$) **78**
14.1 78

15 Commutator and nonabelian tensor computations **80**
15.1 80

16 Lie commutators and nonabelian Lie tensors **85**
16.1 85

17 Generators and relators of groups **87**
17.1 87

18 Orbit polytopes and fundamental domains **89**
18.1 89

19 Cocycles **93**
19.1 93

20 Words in free ZG-modules **95**
20.1 95

21 FpG-modules **97**
21.1 97

22 Meataxe modules **102**
22.1 102

23 G-Outer Groups **103**
23.1 103

24 Cat-1-groups **105**
24.1 105

25 Simplicial groups **107**
25.1 107

26 Coxeter diagrams and graphs of groups **112**
26.1 112

27 Torsion Subcomplexes **116**
27.1 116

28 Simplicial Complexes **120**
28.1 120

29 Cubical Complexes **125**
29.1 125

30 Regular CW-Complexes **135**
30.1 135

31 Knots and Links **137**
31.1 137

32 Knots and Quandles **139**
32.1 139

33 Finite metric spaces and their filtered complexes **144**
33.1 144

34 Commutative diagrams and abstract categories **147**
34.1 147
34.2 148

35 Arrays and Pseudo lists **151**
35.1 151

36 Parallel Computation - Core Functions **155**
36.1 155

37 Parallel Computation - Extra Functions **158**
37.1 158

38 Some functions for accessing basic data **159**
38.1 159

39 Miscellaneous **161**
39.1 161

40 HAP variables that are not yet documented **164**
40.1 164

Index **247**

Chapter 1

Basic functionality for cellular complexes, fundamental groups and homology

This page covers the functions used in chapters 1 and 2 of the book An Invitation to Computational Homotopy.

1.1 Data ⟶ Cellular Complexes

1.1.1 RegularCWPolytope

▷ `RegularCWPolytope(L)` (function)
▷ `RegularCWPolytope(G, v)` (function)

Inputs a list L of vectors in $\mathbb{R}^n$ and outputs their convex hull as a regular CW-complex.

Inputs a permutation group G of degree d and vector $v \in \mathbb{R}^d$, and outputs the convex hull of the orbit $\{v^g : g \in G\}$ as a regular CW-complex.

EXAMPLES:

1.1.2 CubicalComplex

▷ `CubicalComplex(A)` (function)

Inputs a binary array A and returns the cubical complex represented by A. The array A must of course be such that it represents a cubical complex.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12

1.1.3 PureCubicalComplex

▷ `PureCubicalComplex(A)` (function)

Inputs a binary array A and returns the pure cubical complex represented by A.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12

1.1.4 PureCubicalKnot

▷ `PureCubicalKnot(n, k)` (function)
▷ `PureCubicalKnot(L)` (function)

Inputs integers n,k and returns the k-th prime knot on n crossings as a pure cubical complex (if this prime knot exists).

Inputs a list L describing an arc presentation for a knot or link and returns the knot or link as a pure cubical complex.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11

1.1.5 PurePermutahedralKnot

▷ `PurePermutahedralKnot(n, k)` (function)
▷ `PurePermutahedralKnot(L)` (function)

Inputs integers n,k and returns the k-th prime knot on n crossings as a pure permutahedral complex (if this prime knot exists).

Inputs a list L describing an arc presentation for a knot or link and returns the knot or link as a pure permutahedral complex.

EXAMPLES: 1 , 2

1.1.6 PurePermutahedralComplex

▷ `PurePermutahedralComplex(A)` (function)

Inputs a binary array A and returns the pure permutahedral complex represented by A.

EXAMPLES: 1 , 2 , 3 , 4

1.1.7 CayleyGraphOfGroup

▷ `CayleyGraphOfGroup(G, L)` (function)

Inputs a finite group G and a list L of elements in G.It returns the Cayley graph of the group generated by L.

EXAMPLES:

1.1.8 EquivariantEuclideanSpace

▷ `EquivariantEuclideanSpace(G, v)` (function)

Inputs a crystallographic group G with left action on $\mathbb{R}^n$ together with a row vector $v \in \mathbb{R}^n$. It returns an equivariant regular CW-space corresponding to the Dirichlet-Voronoi tessellation of $\mathbb{R}^n$ produced from the orbit of v under the action.

EXAMPLES: 1

1.1.9 EquivariantOrbitPolytope

▷ `EquivariantOrbitPolytope(G, v)` (function)

Inputs a permutation group G of degree n together with a row vector $v \in \mathbb{R}^n$. It returns, as an equivariant regular CW-space, the convex hull of the orbit of v under the canonical left action of G on $\mathbb{R}^n$.

EXAMPLES:

1.1.10 EquivariantTwoComplex

▷ `EquivariantTwoComplex(G)` (function)

Inputs a suitable group G and returns, as an equivariant regular CW-space, the 2-complex associated to some presentation of G.

EXAMPLES: 1

1.1.11 QuillenComplex

▷ `QuillenComplex(G, p)` (function)

Inputs a finite group G and prime p, and returns the simplicial complex arising as the order complex of the poset of elementary abelian p-subgroups of G.

EXAMPLES: 1 , 2 , 3 , 4

1.1.12 RestrictedEquivariantCWComplex

▷ `RestrictedEquivariantCWComplex(Y, H)` (function)

Inputs a G-equivariant regular CW-space Y and a subgroup $H \leq G$ for which GAP can find a transversal. It returns the equivariant regular CW-complex obtained by retricting the action to H.

EXAMPLES:

1.1.13 RandomSimplicialGraph

▷ `RandomSimplicialGraph(n, p)` (function)

Inputs an integer $n \geq 1$ and positive prime p, and returns an Erdős–Rényi random graph as a 1-dimensional simplicial complex. The graph has n vertices. Each pair of vertices is, with probability p, directly connected by an edge.

EXAMPLES: 1

1.1.14 RandomSimplicialTwoComplex

▷ `RandomSimplicialTwoComplex(n, p)` (function)

Inputs an integer $n \geq 1$ and positive prime p, and returns a Linial-Meshulam random simplicial 2-complex. The 1-skeleton of this simplicial complex is the complete graph on n vertices. Each triple of vertices lies, with probability p, in a common 2-simplex of the complex.

EXAMPLES: 1 , 2

1.1.15 ReadCSVfileAsPureCubicalKnot

▷ `ReadCSVfileAsPureCubicalKnot(str)` (function)
▷ `ReadCSVfileAsPureCubicalKnot(str, r)` (function)
▷ `ReadCSVfileAsPureCubicalKnot(L)` (function)
▷ `ReadCSVfileAsPureCubicalKnot(L, R)` (function)

Reads a CSV file identified by a string str such as "file.pdb" or "path/file.pdb" and returns a 3-dimensional pure cubical complex K. Each line of the file should contain the coordinates of a point in $\mathbb{R}^3$ and the complex K should represent a knot determined by the sequence of points, though the latter is not guaranteed. A useful check in this direction is to test that K has the homotopy type of a circle.

If the test fails then try the function again with an integer $r \geq 2$ entered as the optional second argument. The integer determines the resolution with which the knot is constructed.

The function can also read in a list L of strings identifying CSV files for several knots. In this case a list R of integer resolutions can also be entered. The lists L and R must be of equal length.

EXAMPLES: 1

1.1.16 ReadImageAsPureCubicalComplex

▷ `ReadImageAsPureCubicalComplex(str, t)` (function)

Reads an image file identified by a string str such as "file.bmp", "file.eps", "file.jpg", "path/file.png" etc., together with an integer t between 0 and 765. It returns a 2-dimensional pure cubical complex corresponding to a black/white version of the image determined by the threshold t. The 2-cells of the pure cubical complex correspond to pixels with RGB value $R+G+B \leq t$.

EXAMPLES: 1 , 2 , 3 , 4 , 5

1.1.17 ReadImageAsFilteredPureCubicalComplex

▷ `ReadImageAsFilteredPureCubicalComplex(str, n)` (function)

Reads an image file identified by a string str such as "file.bmp", "file.eps", "file.jpg", "path/file.png" etc., together with a positive integer n. It returns a 2-dimensional filtered pure cubical complex of filtration length n. The kth term in the filtration is a pure cubical complex corresponding to a black/white version of the image determined by the threshold $t_k = k \times 765/n$. The 2-cells of the kth term correspond to pixels with RGB value $R+G+B \leq t_k$.

EXAMPLES: 1

1.1.18 ReadImageAsWeightFunction

▷ `ReadImageAsWeightFunction(str, t)` (function)

Reads an image file identified by a string str such as "file.bmp", "file.eps", "file.jpg", "path/file.png" etc., together with an integer t. It constructs a 2-dimensional regular CW-complex Y from the image, together with a weight function $w: Y \to \mathbb{Z}$ corresponding to a filtration on Y of filtration length t. The pair $[Y, w]$ is returned.

EXAMPLES:

1.1.19 ReadPDBfileAsPureCubicalComplex

▷ `ReadPDBfileAsPureCubicalComplex(str)` (function)
▷ `ReadPDBfileAsPureCubicalComplex(str, r)` (function)

Reads a PDB (Protein Database) file identified by a string str such as "file.pdb" or "path/file.pdb" and returns a 3-dimensional pure cubical complex K. The complex K should represent a (protein backbone) knot but this is not guaranteed. A useful check in this direction is to test that K has the homotopy type of a circle.

If the test fails then try the function again with an integer $r \geq 2$ entered as the optional second argument. The integer determines the resolution with which the knot is constructed.

EXAMPLES: 1 , 2 , 3

1.1.20 ReadPDBfileAsPurepermutahedralComplex

▷ `ReadPDBfileAsPurepermutahedralComplex` (global variable)
▷ `ReadPDBfileAsPurePermutahedralComplex(str, r)` (function)

Reads a PDB (Protein Database) file identified by a string str such as "file.pdb" or "path/file.pdb" and returns a 3-dimensional pure permutahedral complex K. The complex K should represent a (protein backbone) knot but this is not guaranteed. A useful check in this direction is to test that K has the homotopy type of a circle.

If the test fails then try the function again with an integer $r \geq 2$ entered as the optional second argument. The integer determines the resolution with which the knot is constructed.

EXAMPLES:

1.1.21 RegularCWPolytope

▷ `RegularCWPolytope(L)` (function)
▷ `RegularCWPolytope(G, v)` (function)

Inputs a list L of vectors in $\mathbb{R}^n$ and outputs their convex hull as a regular CW-complex.

Inputs a permutation group G of degree d and vector $v \in \mathbb{R}^d$, and outputs the convex hull of the orbit $\{v^g : g \in G\}$ as a regular CW-complex.

EXAMPLES:

1.1.22 SimplicialComplex

▷ `SimplicialComplex(L)` (function)

Inputs a list L whose entries are lists of vertices representing the maximal simplices of a simplicial complex, and returns the simplicial complex. Here a "vertex" is a GAP object such as an integer or a subgroup. The list L can also contain non-maximal simplices.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12

1.1.23 SymmetricMatrixToFilteredGraph

▷ `SymmetricMatrixToFilteredGraph(A, m, s)` (function)
▷ `SymmetricMatrixToFilteredGraph(A, m)` (function)

Inputs an $n \times n$ symmetric matrix A, a positive integer m and a positive rational s. The function returns a filtered graph of filtration length m. The t-th term of the filtration is a graph with n vertices and an edge between the i-th and j-th vertices if the (i, j) entry of A is less than or equal to $t \times s/m$.

If the optional input s is omitted then it is set equal to the largest entry in the matrix A.

EXAMPLES: 1 , 2 , 3

1.1.24 SymmetricMatrixToGraph

▷ `SymmetricMatrixToGraph(A, t)` (function)

Inputs an $n \times n$ symmetric matrix A over the rationals and a rational number $t \geq 0$, and returns the graph on the vertices $1, 2, \ldots, n$ with an edge between distinct vertices i and j precisely when the (i, j) entry of A is $\leq t$.

EXAMPLES: 1 , 2

1.2 Metric Spaces

1.2.1 CayleyMetric

▷ `CayleyMetric(g, h)` (function)

Inputs two permutations g, h and optionally the degree N of a symmetric group containing them. It returns the minimum number of transpositions needed to express $g * h^{-1}$ as a product of transpositions.

EXAMPLES: 1

1.2.2 EuclideanMetric

▷ `EuclideanMetric` (global variable)

Inputs two vectors $v, w \in \mathbb{R}^n$ and returns a rational number approximating the Euclidean distance between them.

EXAMPLES:

1.2.3 EuclideanSquaredMetric

▷ `EuclideanSquaredMetric(g, h)` (function)

Inputs two vectors $v, w \in \mathbb{R}^n$ and returns the square of the Euclidean distance between them.
EXAMPLES:

1.2.4 HammingMetric

▷ `HammingMetric(g, h)` (function)

Inputs two permutations g, h and optionally the degree N of a symmetric group containing them. It returns the minimum number of integers moved by the permutation $g * h^{-1}$.
EXAMPLES:

1.2.5 KendallMetric

▷ `KendallMetric(g, h)` (function)

Inputs two permutations g, h and optionally the degree N of a symmetric group containing them. It returns the minimum number of adjacent transpositions needed to express $g * h^{-1}$ as a product of adjacent transpositions. An *adjacent* transposition is of the form $(i, i+1)$.
EXAMPLES:

1.2.6 ManhattanMetric

▷ `ManhattanMetric(g, h)` (function)

Inputs two vectors $v, w \in \mathbb{R}^n$ and returns the Manhattan distance between them.
EXAMPLES: 1

1.2.7 VectorsToSymmetricMatrix

▷ `VectorsToSymmetricMatrix(V)` (function)
▷ `VectorsToSymmetricMatrix(V, d)` (function)

Inputs a list $V = \{v_1, \ldots, v_k\} \in \mathbb{R}^n$ and returns the $k \times k$ symmetric matrix of Euclidean distances $d(v_i, v_j)$. When these distances are irrational they are approximated by a rational number.
As an optional second argument any rational valued function $d(x, y)$ can be entered.
EXAMPLES: 1 , 2 , 3

1.3 Cellular Complexes ⟶ Cellular Complexes

1.3.1 BoundaryMap

▷ `BoundaryMap(K)` (function)

Inputs a pure regular CW-complex K and returns the regular CW-inclusion map $\iota: \partial K \hookrightarrow K$ from the boundary ∂K into the complex K.
EXAMPLES: 1 , 2 , 3

1.3.2 CliqueComplex

▷ `CliqueComplex(G, n)` (function)
▷ `CliqueComplex(F, n)` (function)
▷ `CliqueComplex(K, n)` (function)

Inputs a graph G and integer $n \geq 1$. It returns the n-skeleton of a simplicial complex K with one k-simplex for each complete subgraph of G on $k+1$ vertices.

Inputs a fitered graph F and integer $n \geq 1$. It returns the n-skeleton of a filtered simplicial complex K whose t-term has one k-simplex for each complete subgraph of the t-th term of G on $k+1$ vertices.

Inputs a simplicial complex of dimension $d = 1$ or $d = 2$. If $d = 1$ then the clique complex of a graph returned. If $d = 2$ then the clique complex of a 2-complex is returned.

EXAMPLES: 1

1.3.3 ConcentricFiltration

▷ `ConcentricFiltration(K, n)` (function)

Inputs a pure cubical complex K and integer $n \geq 1$, and returns a filtered pure cubical complex of filtration length n. The t-th term of the filtration is the intersection of K with the ball of radius r_t centred on the centre of gravity of K, where $0 = r_1 \leq r_2 \leq r_3 \leq \cdots \leq r_n$ are equally spaced rational numbers. The complex K is contained in the ball of radius r_n. (At present, this is implemented only for 2- and 3-dimensional complexes.)

EXAMPLES:

1.3.4 DirectProduct

▷ `DirectProduct(M, N)` (function)
▷ `DirectProduct(M, N)` (function)

Inputs two or more regular CW-complexes or two or more pure cubical complexes and returns their direct product.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7

1.3.5 FiltrationTerm

▷ `FiltrationTerm(K, t)` (function)
▷ `FiltrationTerm(K, t)` (function)

Inputs a filtered regular CW-complex or a filtered pure cubical complex K together with an integer $t \geq 1$. The t-th term of the filtration is returned.

EXAMPLES: 1

1.3.6 Graph

▷ `Graph(K)` (function)
▷ `Graph(K)` (function)

Inputs a regular CW-complex or a simplicial complex K and returns its 1-skeleton as a graph.
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16

1.3.7 HomotopyGraph

▷ `HomotopyGraph(Y)` (function)

Inputs a regular CW-complex Y and returns a subgraph $M \subset Y^1$ of the 1-skeleton for which the induced homology homomorphisms $H_1(M,\mathbb{Z}) \to H_1(Y,\mathbb{Z})$ and $H_1(Y^1,\mathbb{Z}) \to H_1(Y,\mathbb{Z})$ have identical images. The construction tries to include as few edges in M as possible, though a minimum is not guaranteed.
EXAMPLES: 1

1.3.8 Nerve

▷ `Nerve(M)` (function)
▷ `Nerve(M)` (function)
▷ `Nerve(M, n)` (function)
▷ `Nerve(M, n)` (function)

Inputs a pure cubical complex or pure permutahedral complex M and returns the simplicial complex K obtained by taking the nerve of an open cover of $|M|$, the open sets in the cover being sufficiently small neighbourhoods of the top-dimensional cells of $|M|$. The spaces $|M|$ and $|K|$ are homotopy equivalent by the Nerve Theorem. If an integer $n \geq 0$ is supplied as the second argument then only the n-skeleton of K is returned.
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

1.3.9 RegularCWComplex

▷ `RegularCWComplex(K)` (function)
▷ `RegularCWComplex(K)` (function)
▷ `RegularCWComplex(K)` (function)
▷ `RegularCWComplex(K)` (function)
▷ `RegularCWComplex(L)` (function)
▷ `RegularCWComplex(L, M)` (function)

Inputs a simplicial, pure cubical, cubical or pure permutahedral complex K and returns the corresponding regular CW-complex. Inputs a list $L = Y!.boundaries$ of boundary incidences of a regular CW-complex Y and returns Y. Inputs a list $L = Y!.boundaries$ of boundary incidences of a regular CW-complex Y together with a list $M = Y!.orientation$ of incidence numbers and returns a regular CW-complex Y. The availability of precomputed incidence numbers saves recalculating them.
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15

1.3.10 RegularCWMap

▷ `RegularCWMap(M, A)` (function)

Inputs a pure cubical complex M and a subcomplex A and returns the inclusion map $A \to M$ as a map of regular CW complexes.

EXAMPLES: 1 , 2 , 3

1.3.11 ThickeningFiltration

▷ `ThickeningFiltration(K, n)` (function)
▷ `ThickeningFiltration(K, n, s)` (function)

Inputs a pure cubical complex K and integer $n \geq 1$, and returns a filtered pure cubical complex of filtration length n. The t-th term of the filtration is the t-fold thickening of K. If an integer $s \geq 1$ is entered as the optional third argument then the t-th term of the filtration is the ts-fold thickening of K.

EXAMPLES: 1 , 2

1.4 Cellular Complexes ⟶ Cellular Complexes (Preserving Data Types)

1.4.1 ContractedComplex

▷ `ContractedComplex(K)` (function)
▷ `ContractedComplex(K)` (function)
▷ `ContractedComplex(K)` (function)
▷ `ContractedComplex(K)` (function)
▷ `ContractedComplex(K, S)` (function)
▷ `ContractedComplex(K)` (function)
▷ `ContractedComplex(K)` (function)
▷ `ContractedComplex(K, S)` (function)
▷ `ContractedComplex(K)` (function)
▷ `ContractedComplex(G)` (function)

Inputs a complex (regular CW, Filtered regular CW, pure cubical etc.) and returns a homotopy equivalent subcomplex.

Inputs a pure cubical complex or pure permutahedral complex K and a subcomplex S. It returns a homotopy equivalent subcomplex of K that contains S.

Inputs a graph G and returns a subgraph S such that the clique complexes of G and S are homotopy equivalent.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13

1.4.2 ContractibleSubcomplex

▷ `ContractibleSubcomplex(K)` (function)
▷ `ContractibleSubcomplex(K)` (function)
▷ `ContractibleSubcomplex(K)` (function)

Inputs a non-empty pure cubical, pure permutahedral or simplicial complex K and returns a contractible subcomplex.

EXAMPLES: 1 , 2

1.4.3 KnotReflection

▷ `KnotReflection(K)` (function)

Inputs a pure cubical knot and returns the reflected knot.
EXAMPLES:

1.4.4 KnotSum

▷ `KnotSum(K, L)` (function)

Inputs two pure cubical knots and returns their sum.
EXAMPLES: 1 , 2 , 3 , 4 , 5

1.4.5 OrientRegularCWComplex

▷ `OrientRegularCWComplex(Y)` (function)

Inputs a regular CW-complex Y and computes and stores incidence numbers for Y. If Y already has incidence numbers then the function does nothing.
EXAMPLES:

1.4.6 PathComponent

▷ `PathComponent(K, n)` (function)
▷ `PathComponent(K, n)` (function)
▷ `PathComponent(K, n)` (function)

Inputs a simplicial, pure cubical or pure permutahedral complex K together with an integer $1 \leq n \leq \beta_0(K)$. The n-th path component of K is returned.
EXAMPLES: 1 , 2 , 3

1.4.7 PureComplexBoundary

▷ `PureComplexBoundary(M)` (function)
▷ `PureComplexBoundary(M)` (function)

Inputs a d-dimensional pure cubical or pure permutahedral complex M and returns a d-dimensional complex consisting of the closure of those d-cells whose boundaries contains some cell with coboundary of size less than the maximal possible size.
EXAMPLES: 1

1.4.8 PureComplexComplement

▷ `PureComplexComplement(`*M*`)` (function)
▷ `PureComplexComplement(`*M*`)` (function)

Inputs a pure cubical complex or a pure permutahedral complex and returns its complement.
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7

1.4.9 PureComplexDifference

▷ `PureComplexDifference(`*M*`, `*N*`)` (function)
▷ `PureComplexDifference(`*M*`, `*N*`)` (function)

Inputs two pure cubical complexes or two pure permutahedral complexes and returns the difference $M - N$.
EXAMPLES: 1

1.4.10 PureComplexInterstection

▷ `PureComplexInterstection` (global variable)
▷ `PureComplexIntersection(`*M*`, `*N*`)` (function)

Inputs two pure cubical complexes or two pure permutahedral complexes and returns their intersection.
EXAMPLES:

1.4.11 PureComplexThickened

▷ `PureComplexThickened(`*M*`)` (function)
▷ `PureComplexThickened(`*M*`)` (function)

Inputs a pure cubical complex or a pure permutahedral complex and returns the a thickened complex.
EXAMPLES: 1

1.4.12 PureComplexUnion

▷ `PureComplexUnion(`*M*`, `*N*`)` (function)
▷ `PureComplexUnion(`*M*`, `*N*`)` (function)

Inputs two pure cubical complexes or two pure permutahedral complexes and returns their union.
EXAMPLES: 1

1.4.13 SimplifiedComplex

▷ `SimplifiedComplex(`*K*`)` (function)
▷ `SimplifiedComplex(`*K*`)` (function)
▷ `SimplifiedComplex(`*R*`)` (function)

▷ SimplifiedComplex(C) (function)

Inputs a regular CW-complex or a pure permutahedral complex K and returns a homeomorphic complex with possibly fewer cells and certainly no more cells.

Inputs a free $\mathbb{Z}G$-resolution R of $\mathbb{Z}$ and returns a $\mathbb{Z}G$-resolution S with potentially fewer free generators.

Inputs a chain complex C of free abelian groups and returns a chain homotopic chain complex D with potentially fewer free generators.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8

1.4.14 ZigZagContractedComplex

▷ ZigZagContractedComplex(K) (function)
▷ ZigZagContractedComplex(K) (function)
▷ ZigZagContractedComplex(K) (function)

Inputs a pure cubical, filtered pure cubical or pure permutahedral complex and returns a homotopy equivalent complex. In the filtered case, the t-th term of the output is homotopy equivalent to the t-th term of the input for all t.

EXAMPLES: 1

1.5 Cellular Complexes ⟶ Homotopy Invariants

1.5.1 AlexanderPolynomial

▷ AlexanderPolynomial(K) (function)
▷ AlexanderPolynomial(K) (function)
▷ AlexanderPolynomial(G) (function)

Inputs a 3-dimensional pure cubical or pure permutahdral complex K representing a knot and returns the Alexander polynomial of the fundamental group $G = \pi_1(\mathbb{R}^3 \setminus K)$.

Inputs a finitely presented group G with infinite cyclic abelianization and returns its Alexander polynomial.

EXAMPLES: 1 , 2 , 3 , 4

1.5.2 BettiNumber

▷ BettiNumber(K, n) (function)
▷ BettiNumber(K, n) (function)
▷ BettiNumber(K, n) (function)
▷ BettiNumber(K, n) (function)
▷ BettiNumber(K, n) (function)
▷ BettiNumber(K, n) (function)
▷ BettiNumber(K, n) (function)
▷ BettiNumber(K, n, p) (function)
▷ BettiNumber(K, n, p) (function)
▷ BettiNumber(K, n, p) (function)

▷ BettiNumber(K, n, p) (function)
▷ BettiNumber(K, n, p) (function)

Inputs a simplicial, cubical, pure cubical, pure permutahedral, regular CW, chain or sparse chain complex K together with an integer $n \geq 0$ and returns the nth Betti number of K.

Inputs a simplicial, cubical, pure cubical, pure permutahedral or regular CW-complex K together with an integer $n \geq 0$ and a prime $p \geq 0$ or $p = 0$. In this case the nth Betti number of K over a field of characteristic p is returned.

EXAMPLES: 1

1.5.3 EulerCharacteristic

▷ EulerCharacteristic(C) (function)
▷ EulerCharacteristic(K) (function)
▷ EulerCharacteristic(K) (function)
▷ EulerCharacteristic(K) (function)
▷ EulerCharacteristic(K) (function)
▷ EulerCharacteristic(K) (function)

Inputs a chain complex C and returns its Euler characteristic.

Inputs a cubical, or pure cubical, or pure permutahedral or regular CW-, or simplicial complex K and returns its Euler characteristic.

EXAMPLES: 1

1.5.4 EulerIntegral

▷ EulerIntegral(Y, w) (function)

Inputs a regular CW-complex Y and a weight function $w: Y \to \mathbb{Z}$, and returns the Euler integral $\int_Y w\, d\chi$.

EXAMPLES:

1.5.5 FundamentalGroup

▷ FundamentalGroup(K) (function)
▷ FundamentalGroup(K, n) (function)
▷ FundamentalGroup(K) (function)
▷ FundamentalGroup(K) (function)
▷ FundamentalGroup(K) (function)
▷ FundamentalGroup(F) (function)
▷ FundamentalGroup(F, n) (function)

Inputs a regular CW, simplicial, pure cubical or pure permutahedral complex K and returns the fundamental group.

Inputs a regular CW complex K and the number n of some zero cell. It returns the fundamental group of K based at the n-th zero cell.

Inputs a regular CW map F and returns the induced homomorphism of fundamental groups. If the number of some zero cell in the domain of F is entered as an optional second variable then the fundamental group is based at this zero cell.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13

1.5.6 FundamentalGroupOfQuotient

▷ `FundamentalGroupOfQuotient(Y)` (function)

Inputs a G-equivariant regular CW complex Y and returns the group G.

EXAMPLES: 1

1.5.7 IsAspherical

▷ `IsAspherical(F, R)` (function)

Inputs a free group F and a list R of words in F. The function attempts to test if the quotient group $G = F/\langle R \rangle^F$ is aspherical. If it succeeds it returns $true$. Otherwise the test is inconclusive and $fail$ is returned.

EXAMPLES: 1 , 2 , 3 , 4

1.5.8 KnotGroup

▷ `KnotGroup(K)` (function)
▷ `KnotGroup(K)` (function)

Inputs a pure cubical or pure permutahedral complex K and returns the fundamental group of its complement. If the complement is path-connected then this fundamental group is unique up to isomorphism. Otherwise it will depend on the path-component in which the randomly chosen base-point lies.

EXAMPLES: 1

1.5.9 PiZero

▷ `PiZero(Y)` (function)
▷ `PiZero(Y)` (function)
▷ `PiZero(Y)` (function)

Inputs a regular CW-complex Y, or graph Y, or simplicial complex Y and returns a pair $[cells, r]$ where: $cells$ is a list of vertices of Y representing the distinct path-components; $r(v)$ is a function which, for each vertex v of Y returns the representative vertex $r(v) \in cells$.

EXAMPLES: 1

1.5.10 PersistentBettiNumbers

▷ `PersistentBettiNumbers(K, n)` (function)
▷ `PersistentBettiNumbers(K, n)` (function)
▷ `PersistentBettiNumbers(K, n)` (function)

▷ PersistentBettiNumbers(K, n) (function)
▷ PersistentBettiNumbers(K, n) (function)
▷ PersistentBettiNumbers(K, n, p) (function)
▷ PersistentBettiNumbers(K, n, p) (function)
▷ PersistentBettiNumbers(K, n, p) (function)
▷ PersistentBettiNumbers(K, n, p) (function)
▷ PersistentBettiNumbers(K, n, p) (function)

Inputs a filtered simplicial, filtered pure cubical, filtered regular CW, filtered chain or filtered sparse chain complex K together with an integer $n \geq 0$ and returns the nth PersistentBetti numbers of K as a list of lists of integers.

Inputs a filtered simplicial, filtered pure cubical, filtered regular CW, filtered chain or filtered sparse chain complex K together with an integer $n \geq 0$ and a prime $p \geq 0$ or $p = 0$. In this case the nth PersistentBetti numbers of K over a field of characteristic p are returned.

EXAMPLES: 1

1.6 Data ⟶ Homotopy Invariants

1.6.1 DendrogramMat

▷ DendrogramMat(A, t, s) (function)

Inputs an $n \times n$ symmetric matrix A over the rationals, a rational $t \geq 0$ and an integer $s \geq 1$. A list $[v_1, \ldots, v_{t+1}]$ is returned with each v_k a list of positive integers. Let $t_k = (k-1)s$. Let $G(A, t_k)$ denote the graph with vertices $1, \ldots, n$ and with distinct vertices i and j connected by an edge when the (i, j) entry of A is $\leq t_k$. The i-th path component of $G(A, t_k)$ is included in the $v_k[i]$-th path component of $G(A, t_{k+1})$. This defines the integer vector v_k. The vector v_k has length equal to the number of path components of $G(A, t_k)$.

EXAMPLES:

1.7 Cellular Complexes ⟶ Non Homotopy Invariants

1.7.1 ChainComplex

▷ ChainComplex(K) (function)
▷ ChainComplex(K) (function)
▷ ChainComplex(K) (function)
▷ ChainComplex(Y) (function)
▷ ChainComplex(K) (function)

Inputs a cubical, or pure cubical, or pure permutahedral or simplicial complex K and returns its chain complex of free abelian groups. In degree n this chain complex has one free generator for each n-dimensional cell of K.

Inputs a regular CW-complex Y and returns a chain complex C which is chain homotopy equivalent to the cellular chain complex of Y. In degree n the free abelian chain group C_n has one free generator for each critical n-dimensional cell of Y with respect to some discrete vector field on Y.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13

1.7.2 ChainComplexEquivalence

▷ `ChainComplexEquivalence` (global variable)

Inputs a regular CW-complex X and returns a pair $[f_*, g_*]$ of chain maps $f_*: C_*(X) \to D_*(X)$, $g_*: D_*(X) \to C_*(X)$. Here $C_*(X)$ is the standard cellular chain complex of X with one free generator for each cell in X. The chain complex $D_*(X)$ is a typically smaller chain complex arising from a discrete vector field on X. The chain maps f_*, g_* are chain homotopy equivalences.

EXAMPLES:

1.7.3 ChainComplexOfQuotient

▷ `ChainComplexOfQuotient(Y)` (function)

Inputs a G-equivariant regular CW-complex Y and returns the cellular chain complex of the quotient space Y/G.

EXAMPLES: 1

1.7.4 ChainMap

▷ `ChainMap(X, A, Y, B)` (function)
▷ `ChainMap(f)` (function)
▷ `ChainMap(f)` (function)

Inputs a pure cubical complex Y and pure cubical sucomplexes $X \subset Y$, $B \subset Y$,$A \subset B$. It returns the induced chain map $f_*: C_*(X/A) \to C_*(Y/B)$ of cellular chain complexes of pairs. (Typlically one takes A and B to be empty or contractible subspaces, in which case $C_*(X/A) \simeq C_*(X)$, $C_*(Y/B) \simeq C_*(Y)$.)

Inputs a map $f: X \to Y$ between two regular CW-complexes X, Y and returns an induced chain map $f_*: C_*(X) \to C_*(Y)$ where $C_*(X)$, $C_*(Y)$ are chain homotopic to (but usually smaller than) the cellular chain complexes of X, Y.

Inputs a map $f: X \to Y$ between two simplicial complexes X, Y and returns the induced chain map $f_*: C_*(X) \to C_*(Y)$ of cellular chain complexes.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8

1.7.5 CochainComplex

▷ `CochainComplex(K)` (function)
▷ `CochainComplex(K)` (function)
▷ `CochainComplex(K)` (function)
▷ `CochainComplex(Y)` (function)
▷ `CochainComplex(K)` (function)

Inputs a cubical, or pure cubical, or pure permutahedral or simplicial complex K and returns its cochain complex of free abelian groups. In degree n this cochain complex has one free generator for each n-dimensional cell of K.

Inputs a regular CW-complex Y and returns a cochain complex C which is chain homotopy equivalent to the cellular cochain complex of Y. In degree n the free abelian cochain group C_n has one free generator for each critical n-dimensional cell of Y with respect to some discrete vector field on Y.

EXAMPLES:

1.7.6 CriticalCells

▷ `CriticalCells(K)` (function)

Inputs a regular CW-complex K and returns its critical cells with respect to some discrete vector field on K. If no discrete vector field on K is available then one will be computed and stored.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

1.7.7 DiagonalApproximation

▷ `DiagonalApproximation(X)` (function)

Inputs a regular CW-complex X and outputs a pair $[p,\iota]$ of maps of CW-complexes. The map $p{:}X^{\Delta}\to X$ will often be a homotopy equivalence. This is always the case if X is the CW-space of any pure cubical complex. In general, one can test to see if the induced chain map $p_*{:}C_*(X^{\Delta})\to C_*(X)$ is an isomorphism on integral homology. The second map $\iota{:}X^{\Delta}\hookrightarrow X\times X$ is an inclusion into the direct product. If p_* induces an isomorphism on homology then the chain map $\iota_*{:}C_*(X^{\Delta})\to C_*(X\times X)$ can be used to compute the cup product.

EXAMPLES: 1

1.7.8 Size

▷ `Size(Y)` (function)
▷ `Size(Y)` (function)
▷ `Size(K)` (function)
▷ `Size(K)` (function)

Inputs a regular CW complex or a simplicial complex Y and returns the number of cells in the complex.

Inputs a d-dimensional pure cubical or pure permutahedral complex K and returns the number of d-dimensional cells in the complex.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22

1.8 (Co)chain Complexes ⟶ (Co)chain Complexes

1.8.1 FilteredTensorWithIntegers

▷ `FilteredTensorWithIntegers(R)` (function)

Inputs a free $\mathbb{Z}G$-resolution R for which "$filteredDimension$" lies in NAMESOFCOMPONENTS(R). (Such a resolution can be produced using TWISTERTENSORPRODUCT(), RESOLUTION-

NORMALSUBGROUPS() or FREEGRESOLUTION().) It returns the filtered chain complex obtained by tensoring with the trivial module $\mathbb{Z}$.

EXAMPLES: 1 , 2

1.8.2 FilteredTensorWithIntegersModP

▷ `FilteredTensorWithIntegersModP(R, p)` (function)

Inputs a free $\mathbb{Z}G$-resolution R for which "*filteredDimension*" lies in NAMESOFCOMPONENTS(R), together with a prime p. (Such a resolution can be produced using TWISTERTENSORPRODUCT(), RESOLUTIONNORMALSUBGROUPS() or FREEGRESOLUTION().) It returns the filtered chain complex obtained by tensoring with the trivial module $\mathbb{F}$, the field of p elements.

EXAMPLES: 1 , 2

1.8.3 HomToIntegers

▷ `HomToIntegers(C)` (function)
▷ `HomToIntegers(R)` (function)
▷ `HomToIntegers(F)` (function)

Inputs a chain complex C of free abelian groups and returns the cochain complex $Hom_{\mathbb{Z}}(C,\mathbb{Z})$.

Inputs a free $\mathbb{Z}G$-resolution R in characteristic 0 and returns the cochain complex $Hom_{\mathbb{Z}G}(R,\mathbb{Z})$.

Inputs an equivariant chain map $F:R \to S$ of resolutions and returns the induced cochain map $Hom_{\mathbb{Z}G}(S,\mathbb{Z}) \longrightarrow Hom_{\mathbb{Z}G}(R,\mathbb{Z})$.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

1.8.4 TensorWithIntegersModP

▷ `TensorWithIntegersModP(C, p)` (function)
▷ `TensorWithIntegersModP(R, p)` (function)
▷ `TensorWithIntegersModP(F, p)` (function)

Inputs a chain complex C of characteristic 0 and a prime integer p. It returns the chain complex $C \otimes_{\mathbb{Z}} \mathbb{Z}_p$ of characteristic p.

Inputs a free $\mathbb{Z}G$-resolution R of characteristic 0 and a prime integer p. It returns the chain complex $R \otimes_{\mathbb{Z}G} \mathbb{Z}_p$ of characteristic p.

Inputs an equivariant chain map $F:R \to S$ in characteristic 0 a prime integer p. It returns the induced chain map $F \otimes_{\mathbb{Z}G} \mathbb{Z}_p : R \otimes_{\mathbb{Z}G} \mathbb{Z}_p \longrightarrow S \otimes_{\mathbb{Z}G} \mathbb{Z}_p$.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

1.9 (Co)chain Complexes $\longrightarrow$ Homotopy Invariants

1.9.1 Cohomology

▷ `Cohomology(C, n)` (function)
▷ `Cohomology(F, n)` (function)
▷ `Cohomology(K, n)` (function)

▷ `Cohomology(K, n)` (function)
▷ `Cohomology(K, n)` (function)
▷ `Cohomology(K, n)` (function)
▷ `Cohomology(K, n)` (function)

Inputs a cochain complex C and integer $n \geq 0$ and returns the n-th cohomology group of C as a list of its abelian invariants.

Inputs a chain map F and integer $n \geq 0$. It returns the induced cohomology homomorphism $H_n(F)$ as a homomorphism of finitely presented groups.

Inputs a cubical, or pure cubical, or pure permutahedral or regular CW or simplicial complex K together with an integer $n \geq 0$. It returns the n-th integral cohomology group of K as a list of its abelian invariants.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26

1.9.2 CupProduct

▷ `CupProduct(Y)` (function)
▷ `CupProduct(R, p, q, P, Q)` (function)

Inputs a regular CW-complex Y and returns a function $f(p,q,P,Q)$. This function f inputs two integers $p,q \geq 0$ and two integer lists $P = [p_1,\ldots,p_m]$, $Q = [q_1,\ldots,q_n]$ representing elements $P \in H^p(Y,\mathbb{Z})$ and $Q \in H^q(Y,\mathbb{Z})$. The function f returns a list $P \cup Q$ representing the cup product $P \cup Q \in H^{p+q}(Y,\mathbb{Z})$.

Inputs a free $\mathbb{Z}G$ resolution R of $\mathbb{Z}$ for some group G, together with integers $p,q \geq 0$ and integer lists P,Q representing cohomology classes $P \in H^p(G,\mathbb{Z})$, $Q \in H^q(G,\mathbb{Z})$. An integer list representing the cup product $P \cup Q \in H^{p+q}(G,\mathbb{Z})$ is returned.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6

1.9.3 Homology

▷ `Homology(C, n)` (function)
▷ `Homology(F, n)` (function)
▷ `Homology(K, n)` (function)
▷ `Homology(K, n)` (function)
▷ `Homology(K, n)` (function)
▷ `Homology(K, n)` (function)
▷ `Homology(K, n)` (function)

Inputs a chain complex C and integer $n \geq 0$ and returns the n-th homology group of C as a list of its abelian invariants.

Inputs a chain map F and integer $n \geq 0$. It returns the induced homology homomorphism $H_n(F)$ as a homomorphism of finitely presented groups.

Inputs a cubical, or pure cubical, or pure permutahedral or regular CW or simplicial complex K together with an integer $n \geq 0$. It returns the n-th integral homology group of K as a list of its abelian invariants.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45

1.10 Visualization

1.10.1 BarCodeDisplay

▷ `BarCodeDisplay(L)` (function)

Displays a barcode L=PERSITENTBETTINUMBERS(X,N).
EXAMPLES: 1 , 2

1.10.2 BarCodeCompactDisplay

▷ `BarCodeCompactDisplay(L)` (function)

Displays a barcode L=PERSITENTBETTINUMBERS(X,N) in compact form.
EXAMPLES: 1 , 2 , 3

1.10.3 CayleyGraphOfGroup

▷ `CayleyGraphOfGroup(G, L)` (function)

Inputs a finite group G and a list L of elements in G.It displays the Cayley graph of the group generated by L where edge colours correspond to generators.
EXAMPLES:

1.10.4 Display

▷ `Display(G)` (function)
▷ `Display(M)` (function)
▷ `Display(M)` (function)

Displays a graph G; a 2- or 3-dimensional pure cubical complex M; a 3-dimensional pure permutahedral complex M.
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24

1.10.5 DisplayArcPresentation

▷ `DisplayArcPresentation(K)` (function)

Displays a 3-dimensional pure cubical knot K=PURECUBICALKNOT(L) in the form of an arc presentation.
EXAMPLES:

1.10.6 DisplayCSVKnotFile

▷ `DisplayCSVKnotFile` (global variable)

Inputs a string *str* that identifies a csv file containing the points on a piecewise linear knot in $\mathbb{R}^3$. It displays the knot.

EXAMPLES:

1.10.7 DisplayDendrogram

▷ `DisplayDendrogram(L)` (function)

Displays the dendrogram L:=DENDROGRAMMAT(A,T,S).

EXAMPLES:

1.10.8 DisplayDendrogramMat

▷ `DisplayDendrogramMat(A, t, s)` (function)

Inputs an $n \times n$ symmetric matrix A over the rationals, a rational $t \geq 0$ and an integer $s \geq 1$. The dendrogram defined by DENDROGRAMMAT(A,T,S) is displayed.

EXAMPLES:

1.10.9 DisplayPDBfile

▷ `DisplayPDBfile(str)` (function)

Displays the protein backone described in a PDB (Protein Database) file identified by a string str such as "file.pdb" or "path/file.pdb".

EXAMPLES: 1

1.10.10 OrbitPolytope

▷ `OrbitPolytope(G, v, L)` (function)

Inputs a permutation group or finite matrix group G of degree d and a rational vector $v \in \mathbb{R}^d$. In both cases there is a natural action of G on $\mathbb{R}^d$. Let $P(G,v)$ be the convex hull of the orbit of v under the action of G. The function also inputs a sublist L of the following list of strings: ["dimension","vertex_degree", "visual_graph", "schlegel", "visual"]

Depending on L, the function displays the following information:
the dimension of the orbit polytope $P(G,v)$;
the degree of a vertex in the graph of $P(G,v)$;
a visualization of the graph of $P(G,v)$;
a visualization of the Schlegel diagram of $P(G,v)$;
a visualization of the polytope $P(G,v)$ if $d = 2,3$.

The function requires Polymake software.

EXAMPLES: 1 , 2

1.10.11 ScatterPlot

▷ `ScatterPlot(L)` (function)

Inputs a list $L = [[x_1, y_1], \ldots, [x_n, y_n]]$ of pairs of rational numbers and displays a scatter plot of the points in the x-y-plane.

EXAMPLES: 1 , 2

Chapter 2

Basic functionality for $\mathbb{Z}G$-resolutions and group cohomology

This page covers the functions used in chapter 3 of the book An Invitation to Computational Homotopy.

2.1 Resolutions

2.1.1 EquivariantChainMap

▷ `EquivariantChainMap(R, S, f)` (function)

Inputs a free $\mathbb{Z}G$-resolution R of $\mathbb{Z}$, a free $\mathbb{Z}Q$-resolution S of $\mathbb{Z}$, and a group homomorphism $f: G \to Q$. It returns the induced f-equivariant chain map $F: R \to S$.

EXAMPLES: 1 , 2 , 3 , 4

2.1.2 FreeGResolution

▷ `FreeGResolution(P, n)` (function)

Inputs a non-free $\mathbb{Z}G$-resolution P_* and a positive integer n. It attempts to return n terms of a free $\mathbb{Z}G$-resolution of $\mathbb{Z}$. However, the stabilizer groups in the non-free resolution must be such that HAP can construct free resolutions with contracting homotopies for them.

The contracting homotopy on the resolution was implemented by Bui Anh Tuan.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10

2.1.3 ResolutionBieberbachGroup

▷ `ResolutionBieberbachGroup(G)` (function)
▷ `ResolutionBieberbachGroup(G, v)` (function)

Inputs a torsion free crystallographic group G, also known as a Bieberbach group, represented using AFFINECRYSTGROUPONRIGHT as in the GAP package Cryst. It also optionally inputs a choice of vector v in the Euclidean space $\mathbb{R}^n$ on which G acts freely. The function returns $n+1$ terms of the free ZG-resolution of $\mathbb{Z}$ arising as the cellular chain complex of the tessellation of $\mathbb{R}^n$ by the

Dirichlet-Voronoi fundamental domain determined by v. No contracting homotopy is returned with the resolution.

This function is part of the HAPcryst package written by Marc Roeder and thus requires the HAPcryst package to be loaded.

The function requires the use of Polymake software.

EXAMPLES: 1

2.1.4 ResolutionCubicalCrystGroup

▷ `ResolutionCubicalCrystGroup(G, k)` (function)

Inputs a crystallographic group G represented using AFFINECRYSTGROUPONRIGHT as in the GAP package *Cryst* together with an integer $k \geq 1$. The function tries to find a cubical fundamental domain in the Euclidean space $\mathbb{R}^n$ on which G acts. If it succeeds it uses this domain to return $k+1$ terms of a free ZG-resolution of $\mathbb{Z}$.

This function was written by Bui Anh Tuan.

EXAMPLES: 1 , 2 , 3

2.1.5 ResolutionFiniteGroup

▷ `ResolutionFiniteGroup(G, k)` (function)

Inputs a finite group G and an integer $k \geq 1$. It returns $k+1$ terms of a free ZG-resolution of $\mathbb{Z}$.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20

2.1.6 ResolutionNilpotentGroup

▷ `ResolutionNilpotentGroup(G, k)` (function)

Inputs a nilpotent group G (which can be infinite) and an integer $k \geq 1$. It returns $k+1$ terms of a free $\mathbb{Z}G$-resolution of $\mathbb{Z}$.

EXAMPLES: 1 , 2 , 3 , 4 , 5

2.1.7 ResolutionNormalSeries

▷ `ResolutionNormalSeries(L, k)` (function)

Inputs a a list L consisting of a chain $\$1 = N_1 \leq N_2 \leq \cdots \leq N_n = G$ of normal subgroups of G, together with an integer $k \geq 1$. It returns $k+1$ terms of a free ZG-resolution of $\mathbb{Z}$.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8

2.1.8 ResolutionPrimePowerGroup

▷ `ResolutionPrimePowerGroup(G, k)` (function)

Inputs a finite p-group G and an integer $k \geq 1$. It returns $k+1$ terms of a minimal free $\mathbb{F}G$-resolution of the field $\mathbb{F}$ of p elements.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6

2.1.9 ResolutionSL2Z

▷ `ResolutionSL2Z(m, k)` (function)

Inputs positive integers m,n and returns n terms of a free $\mathbb{Z}G$-resolution of $\mathbb{Z}$ for the group $G = SL_2(\mathbb{Z}[1/m])$.

This function is joint work with Bui Anh Tuan.

EXAMPLES: 1 , 2 , 3 , 4

2.1.10 ResolutionSmallGroup

▷ `ResolutionSmallGroup(G, k)` (function)
▷ `ResolutionSmallGroup(G, k)` (function)

Inputs a small group G and an integer $k \geq 1$. It returns $k+1$ terms of a free ZG-resolution of $\mathbb{Z}$.

If G is a finitely presented group then up to degree 2 the resolution coincides with cellular chain complex of the universal cover of the 2 complex associated to the presentation of G. Thus the boundaries of the generators in degree 3 provide a generating set for the module of identities of the presentation.

This function was written by Irina Kholodna.

EXAMPLES: 1 , 2

2.1.11 ResolutionSubgroup

▷ `ResolutionSubgroup(R, H)` (function)

Inputs a free ZG-resolution of $\mathbb{Z}$ and a finite index subgroup $H \leq G$. It returns a free ZH-resolution of $\mathbb{Z}$.

EXAMPLES: 1 , 2 , 3

2.2 Algebras $\longrightarrow$ (Co)chain Complexes

2.2.1 LeibnizComplex

▷ `LeibnizComplex(g, n)` (function)

Inputs a Leibniz algebra, or Lie algebra, $\mathfrak{g}$ over a ring $\mathbb{K}$ together with an integer $n \geq 0$. It returns the first n terms of the Leibniz chain complex over $\mathbb{K}$. The complex was implemented by Pablo Fernandez Ascariz.

EXAMPLES:

2.3 Resolutions $\longrightarrow$ (Co)chain Complexes

2.3.1 HomToIntegers

▷ `HomToIntegers(C)` (function)
▷ `HomToIntegers(R)` (function)

▷ `HomToIntegers(F)` (function)

Inputs a chain complex C of free abelian groups and returns the cochain complex $Hom_{\mathbb{Z}}(C,\mathbb{Z})$.

Inputs a free $\mathbb{Z}G$-resolution R in characteristic 0 and returns the cochain complex $Hom_{\mathbb{Z}G}(R,\mathbb{Z})$.

Inputs an equivariant chain map $F:R \to S$ of resolutions and returns the induced cochain map $Hom_{\mathbb{Z}G}(S,\mathbb{Z}) \longrightarrow Hom_{\mathbb{Z}G}(R,\mathbb{Z})$.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

2.3.2 HomToIntegralModule

▷ `HomToIntegralModule(R, A)` (function)

Inputs a free $\mathbb{Z}G$-resolution R in characteristic 0 and a group homomorphism $A:G \to \mathrm{GL}_n(\mathbb{Z})$. The homomorphism A can be viewed as the $\mathbb{Z}G$-module with underlying abelian group $\mathbb{Z}^n$ on which G acts via the homomorphism A. It returns the cochain complex $Hom_{\mathbb{Z}G}(R,A)$.

EXAMPLES: 1 , 2 , 3 , 4

2.3.3 TensorWithIntegers

▷ `TensorWithIntegers(R)` (function)
▷ `TensorWithIntegers(F)` (function)

Inputs a free $\mathbb{Z}G$-resolution R of characteristic 0 and returns the chain complex $R \otimes_{\mathbb{Z}G} \mathbb{Z}$.

Inputs an equivariant chain map $F:R \to S$ in characteristic 0 and returns the induced chain map $F \otimes_{\mathbb{Z}G} \mathbb{Z}:R \otimes_{\mathbb{Z}G} \mathbb{Z} \longrightarrow S \otimes_{\mathbb{Z}G} \mathbb{Z}$.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30

2.3.4 TensorWithIntegersModP

▷ `TensorWithIntegersModP(C, p)` (function)
▷ `TensorWithIntegersModP(R, p)` (function)
▷ `TensorWithIntegersModP(F, p)` (function)

Inputs a chain complex C of characteristic 0 and a prime integer p. It returns the chain complex $C \otimes_{\mathbb{Z}} \mathbb{Z}_p$ of characteristic p.

Inputs a free $\mathbb{Z}G$-resolution R of characteristic 0 and a prime integer p. It returns the chain complex $R \otimes_{\mathbb{Z}G} \mathbb{Z}_p$ of characteristic p.

Inputs an equivariant chain map $F:R \to S$ in characteristic 0 a prime integer p. It returns the induced chain map $F \otimes_{\mathbb{Z}G} \mathbb{Z}_p:R \otimes_{\mathbb{Z}G} \mathbb{Z}_p \longrightarrow S \otimes_{\mathbb{Z}G} \mathbb{Z}_p$.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

2.4 Cohomology rings

2.4.1 AreIsomorphicGradedAlgebras

▷ `AreIsomorphicGradedAlgebras(A, B)` (function)

Inputs two freely presented graded algebras $A = \mathbb{F}[x_1,\ldots,x_m]/I$ and $B = \mathbb{F}[y_1,\ldots,y_n]/J$ and returns TRUE if they are isomorphic, and FALSE otherwise. This function was implemented by Paul Smith.

EXAMPLES:

2.4.2 HAPDerivation

▷ `HAPDerivation(R, I, L)` (function)

Inputs a polynomial ring $R = \mathbb{F}[x_1,\ldots,x_m]$ over a field $\mathbb{F}$ together with a list I of generators for an ideal in R and a list $L = [y_1,\ldots,y_m] \subset R$. It returns the derivation $d:E \to E$ for $E = R/I$ defined by $d(x_i) = y_i$. This function was written by Paul Smith. It uses the Singular commutative algebra package.

EXAMPLES:

2.4.3 HilbertPoincareSeries

▷ `HilbertPoincareSeries(E)` (function)

Inputs a presentation $E = \mathbb{F}[x_1,\ldots,x_m]/I$ of a graded algebra and returns its Hilbert–Poincaré series. This function was written by Paul Smith and uses the Singular commutative algebra package. It is essentially a wrapper for Singular's Hilbert–Poincaré series.

EXAMPLES: 1

2.4.4 HomologyOfDerivation

▷ `HomologyOfDerivation(d)` (function)

Inputs a derivation $d:E \to E$ on a quotient $E = R/I$ of a polynomial ring $R = \mathbb{F}[x_1,\ldots,x_m]$ over a field $\mathbb{F}$. It returns a list $[S,J,h]$ where S is a polynomial ring and J is a list of generators for an ideal in S such that there is an isomorphism $\alpha:S/J \to \ker d/\operatorname{im} d$. This isomorphism lifts to the ring homomorphism $h:S \to \ker d$. This function was written by Paul Smith. It uses the Singular commutative algebra package.

EXAMPLES:

2.4.5 IntegralCohomologyGenerators

▷ `IntegralCohomologyGenerators(R, n)` (function)

Inputs at least $n+1$ terms of a free $\mathbb{Z}G$-resolution of $\mathbb{Z}$ and the integer $n \geq 1$. It returns a minimal list of cohomology classes in $H^n(G,\mathbb{Z})$ which, together with all cup products of lower degree classes, generate the group $H^n(G,\mathbb{Z})$. (Let a_i be the i-th canonical generator of the d-generator abelian group

$H^n(G,Z)$. The cohomology class $n_1a_1+...+n_da_d$ is represented by the integer vector $u=(n_1,...,n_d)$.
)

EXAMPLES:

2.4.6 LHSSpectralSequence

▷ `LHSSpectralSequence(G, N, r)` (function)

Inputs a finite 2-group G, and normal subgroup N and an integer r. It returns a list of length r whose i-th term is a presentation for the i-th page of the Lyndon-Hochschild-Serre spectral sequence. This function was written by Paul Smith. It uses the Singular commutative algebra package.

EXAMPLES:

2.4.7 LHSSpectralSequenceLastSheet

▷ `LHSSpectralSequenceLastSheet(G, N)` (function)

Inputs a finite 2-group G and normal subgroup N. It returns presentation for the E_∞ page of the Lyndon-Hochschild-Serre spectral sequence. This function was written by Paul Smith. It uses the Singular commutative algebra package.

EXAMPLES:

2.4.8 ModPCohomologyGenerators

▷ `ModPCohomologyGenerators(G, n)` (function)
▷ `ModPCohomologyGenerators(R)` (function)

Inputs either a p-group G and positive integer n, or else $n+1$ terms of a minimal $\mathbb{F}G$-resolution R of the field $\mathbb{F}$ of p elements. It returns a pair whose first entry is a minimal list of homogeneous generators for the cohomology ring $A=H^*(G,\mathbb{F})$ modulo all elements in degree greater than n. The second entry of the pair is a function DEG which, when applied to a minimal generator, yields its degree. WARNING: the following rule must be applied when multiplying generators x_i together. Only products of the form $x_1*(x_2*(x_3*(x_4*...)))$ with $deg(x_i)\leq deg(x_{i+1})$ should be computed (since the x_i belong to a structure constant algebra with only a partially defined structure constants table).

EXAMPLES: 1

2.4.9 ModPCohomologyRing ( )

▷ `ModPCohomologyRing(G, n)` (function)
▷ `ModPCohomologyRing(G, n, level)` (function)
▷ `ModPCohomologyRing(R)` (function)
▷ `ModPCohomologyRing(R, level)` (function)

Inputs either a p-group G and positive integer n, or else n terms of a minimal $\mathbb{F}G$-resolution R of the field $\mathbb{F}$ of p elements. It returns the cohomology ring $A=H^*(G,\mathbb{F})$ modulo all elements in degree greater than n. The ring is returned as a structure constant algebra A. The ring A is graded. It has a component A!.DEGREE(X) which is a function returning the degree of each (homogeneous) element x

in GENERATORSOFALGEBRA(A). An optional input variable "*level*" can be set to one of the strings "*medium*" or "*high*". These settings determine parameters in the algorithm. The default setting is "*medium*". When "*level*" is set to "*high*" the ring A is returned with a component A!.NICEBASIS. This component is a pair $[Coeff, Bas]$. Here Bas is a list of integer lists; a "nice" basis for the vector space A can be constructed using the command LIST(BAS,X->PRODUCT(LIST(X,I->BASIS(A)[I])). The coefficients of the canonical basis element BASIS(A)[I] are stored as COEFF[I]. If the ring A is computed using the setting "*level*" = "*medium*" then the component A!.NICEBASIS can be added to A using the command A:=MODPCOHOMOLOGYRING_PART_2(A).

EXAMPLES: 1 , 2

2.4.10 Mod2CohomologyRingPresentation

▷ `Mod2CohomologyRingPresentation(G)` (function)
▷ `Mod2CohomologyRingPresentation(G, n)` (function)
▷ `Mod2CohomologyRingPresentation(A)` (function)
▷ `Mod2CohomologyRingPresentation(R)` (function)

When applied to a finite 2-group G this function returns a presentation for the mod-2 cohomology ring $H^*(G,\mathbb{F})$. The Lyndon-Hochschild-Serre spectral sequence is used to prove that the presentation is complete. When the function is applied to a 2-group G and positive integer n the function first constructs $n+1$ terms of a free $\mathbb{F}G$-resolution R, then constructs the finite-dimensional graded algebra $A = H^{(*\leq n)}(G,\mathbb{F})$, and finally uses A to approximate a presentation for $H^*(G,\mathbb{F})$. For "sufficiently large" n the approximation will be a correct presentation for $H^*(G,\mathbb{F})$. Alternatively, the function can be applied directly to either the resolution R or graded algebra A. This function was written by Paul Smith. It uses the Singular commutative algebra package to handle the Lyndon-Hochschild-Serre spectral sequence.

EXAMPLES: 1 , 2

2.5 Group Invariants

2.5.1 GroupCohomology

▷ `GroupCohomology(G, k)` (function)
▷ `GroupCohomology(G, k, p)` (function)

Inputs a group G and integer $k \geq 0$. The group G should either be finite or else lie in one of a range of classes of infinite groups (such as nilpotent, crystallographic, Artin etc.). The function returns the list of abelian invariants of $H^k(G,\mathbb{Z})$.

If a prime p is given as an optional third input variable then the function returns the list of abelian invariants of $H^k(G,\mathbb{Z}_p)$. In this case each abelian invariant will be equal to p and the length of the list will be the dimension of the vector space $H^k(G,\mathbb{Z}_p)$.

EXAMPLES: 1 , 2

2.5.2 GroupHomology

▷ `GroupHomology(G, k)` (function)
▷ `GroupHomology(G, k, p)` (function)

Inputs a group G and integer $k \geq 0$. The group G should either be finite or else lie in one of a range of classes of infinite groups (such as nilpotent, crystallographic, Artin etc.). The function returns the list of abelian invariants of $H_k(G,\mathbb{Z})$.

If a prime p is given as an optional third input variable then the function returns the list of abelian invariants of $H_k(G,\mathbb{Z}_p)$. In this case each abelian invariant will be equal to p and the length of the list will be the dimension of the vector space $H_k(G,\mathbb{Z}_p)$.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

2.5.3 PrimePartDerivedFunctor

▷ `PrimePartDerivedFunctor(G, R, A, k)` (function)

Inputs a group G, an integer $k \geq 0$, at least $k+1$ terms of a free $\mathbb{Z}P$-resolution of $\mathbb{Z}$ for P a Sylow p-subgroup of G. A function such as A=TENSORWITHINTEGERS is also entered. The abelian invariants of the p-primary part $H_k(G,A)_{(p)}$ of the homology with coefficients in A is returned.

EXAMPLES: 1 , 2 , 3 , 4

2.5.4 PoincareSeries

▷ `PoincareSeries(G, n)` (function)
▷ `PoincareSeries(G)` (function)
▷ `PoincareSeries(R, n)` (function)
▷ `PoincareSeries(L, n)` (function)

Inputs a finite p-group G and a positive integer n. It returns a quotient of polynomials $f(x) = P(x)/Q(x)$ whose expansion has coefficient of x^k equal to the rank of the vector space $H_k(G,\mathbb{F}_p)$ for all k in the range $1 \leq k \leq n$. (The second input variable can be omitted, in which case the function tries to choose a 'reasonable' value for n. For 2-groups the function POINCARESERIESLHS(G) can be used to produce an $f(x)$ that is correct in all degrees.) In place of the group G the function can also input (at least n terms of) a minimal mod-p resolution R for G. Alternatively, the first input variable can be a list L of integers. In this case the coefficient of x^k in $f(x)$ is equal to the $(k+1)$st term in the list.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

2.5.5 PoincareSeries

▷ `PoincareSeries(G, n)` (function)
▷ `PoincareSeries(G)` (function)
▷ `PoincareSeries(R, n)` (function)
▷ `PoincareSeries(L, n)` (function)

Inputs a finite p-group G and a positive integer n. It returns a quotient of polynomials $f(x) = P(x)/Q(x)$ whose expansion has coefficient of x^k equal to the rank of the vector space $H_k(G,\mathbb{F}_p)$ for all k in the range $1 \leq k \leq n$. (The second input variable can be omitted, in which case the function tries to choose a 'reasonable' value for n. For 2-groups the function POINCARESERIESLHS(G) can be used to produce an $f(x)$ that is correct in all degrees.) In place of the group G the function can also

input (at least n terms of) a minimal mod-p resolution R for G. Alternatively, the first input variable can be a list L of integers. In this case the coefficient of x^k in $f(x)$ is equal to the $(k+1)$st term in the list.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

2.5.6 RankHomologyPGroup

▷ `RankHomologyPGroup(G, P, n)` (function)

Inputs a p-group G, a rational function P representing the Poincaré series of the mod-p cohomology of G and a positive integer n. It returns the minimum number of generators for the finite abelian p-group $H_n(G,\mathbb{Z})$.

EXAMPLES: 1

2.6 $\mathbb{F}_p$-modules

2.6.1 GroupAlgebraAsFpGModule

▷ `GroupAlgebraAsFpGModule(G)` (function)

Inputs a finite p-group G and returns the modular group algebra $\mathbb{F}_pG$ in the form of an $\mathbb{F}_pG$-module.

EXAMPLES:

2.6.2 Radical

▷ `Radical(M)` (function)

Inputs an $\mathbb{F}_pG$-module and returns its radical.

EXAMPLES:

2.6.3 RadicalSeries

▷ `RadicalSeries(M)` (function)
▷ `RadicalSeries(R)` (function)

Inputs an $\mathbb{F}_pG$-module M and returns its radical series as a list of $\mathbb{F}_pG$-modules.

Inputs a free $\mathbb{F}_pG$-resolution R and returns the filtered chain complex $\cdots Rad_2(\mathbb{F}_pG)R \leq Rad_1(\mathbb{F}_pG)R \leq R$.

EXAMPLES:

Chapter 3

Basic functionality for homological group theory

This page covers the functions used in chapter 4 of the book An Invitation to Computational Homotopy.

3.1 Cocycles

3.1.1 CcGroup

▷ `CcGroup(N, f)` (function)

Inputs a G-outer group N with nonabelian cocycle describing some extension $N \rightarrowtail E \twoheadrightarrow G$ together with standard 2-cocycle $f: G \times G \to A$ where $A = Z(N)$. It returns the extension group determined by the cocycle f. The group is returned as a cocyclic group.

This function is part of the HAPcocyclic package of functions implemented by Robert F. Morse.

EXAMPLES: 1 , 2

3.1.2 CocycleCondition

▷ `CocycleCondition(R, n)` (function)

Inputs a free $\mathbb{Z}G$-resolution R of $\mathbb{Z}$ and an integer $n \geq 1$. It returns an integer matrix M with the following property. Let d be the $\mathbb{Z}G$-rank of R_n. An integer vector $f = [f_1, ..., f_d]$ then represents a $\mathbb{Z}G$-homomorphism $R_n \to \mathbb{Z}_q$ which sends the ith generator of R_n to the integer f_i in the trivial $\mathbb{Z}G$-module $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ (where possibly $q = 0$). The homomorphism f is a cocycle if and only if $M^t f = 0 \bmod q$.

EXAMPLES: 1 , 2

3.1.3 StandardCocycle

▷ `StandardCocycle(R, f, n)` (function)

▷ `StandardCocycle(R, f, n, q)` (function)

Inputs a free $\mathbb{Z}G$-resolution R (with contracting homotopy), a positive integer n and an integer vector f representing an n-cocycle $R_n \to \mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ where G acts trivially on $\mathbb{Z}_q$. It is assumed $q = 0$ unless a value for q is entered. The command returns a function $F(g_1,...,g_n)$ which is the standard cocycle $G^n \to \mathbb{Z}_q$ corresponding to f. At present the command is implemented only for $n = 2$ or 3.

EXAMPLES: 1 , 2

3.2 G-Outer Groups

3.2.1 ActedGroup

▷ `ActedGroup(M)` (function)

Inputs a G-outer group M corresponding to a homomorphism $\alpha: G \to \mathrm{Out}(N)$ and returns the group N.

EXAMPLES: 1 , 2 , 3 , 4

3.2.2 ActingGroup

▷ `ActingGroup(M)` (function)

Inputs a G-outer group M corresponding to a homomorphism $\alpha: G \to \mathrm{Out}(N)$ and returns the group G.

EXAMPLES: 1 , 2

3.2.3 Centre

▷ `Centre(M)` (function)

Inputs a G-outer group M and returns its group-theoretic centre as a G-outer group.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6

3.2.4 GOuterGroup

▷ `GOuterGroup(E, N)` (function)
▷ `GOuterGroup()` (function)

Inputs a group E and normal subgroup N. It returns N as a G-outer group where $G = E/N$. A nonabelian cocycle $f: G \times G \to N$ is attached as a component of the G-Outer group.

The function can be used without an argument. In this case an empty outer group C is returned. The components must be set using SETACTINGGROUP(C,G), SETACTEDGROUP(C,N) and SETOUTERACTION(C,ALPHA).

EXAMPLES: 1 , 2 , 3 , 4

3.3 G-cocomplexes

3.3.1 CohomologyModule

▷ `CohomologyModule(C, n)` (function)

Inputs a G-cocomplex C together with a non-negative integer n. It returns the cohomology $H^n(C)$ as a G-outer group. If C was constructed from a $\mathbb{Z}G$-resolution R by homing to an abelian G-outer group A then, for each x in $H := CohomologyModule(C,n)$, there is a function $f := H!.representativeCocycle(x)$ which is a standard n-cocycle corresponding to the cohomology class x. (At present this is implemented only for $n = 1,2,3$.)

EXAMPLES: 1 , 2 , 3

3.3.2 HomToGModule

▷ `HomToGModule(R, A)` (function)

Inputs a $\mathbb{Z}G$-resolution R and an abelian G-outer group A. It returns the G-cocomplex obtained by applying $HomZG(_,A)$. (At present this function does not handle equivariant chain maps.)

EXAMPLES: 1 , 2 , 3 , 4

Chapter 4

Basic functionality for parallel computation

This page covers the functions used for parallel computation in the book An Invitation to Computational Homotopy.

4.1 Six Core Functions

4.1.1 ChildCreate

▷ `ChildCreate()` (function)
▷ `ChildProcess(`*str*`)` (function)
▷ `ChildProcess(`*str*`)` (function)
▷ `ChildProcess(`*str*`)` (function)

Starts a GAP session as a child process and returns a stream to the child process. If no argument is given then the child process is created on the local machine; otherwise the argument should be: (1) *str*="computer.address.ie" the address of a remote computer for which ssh has been configured to require no password from the user; (2) *str*=["-m", "100000M", "-T"] a list of GAP command line options; (3) *str*="computer.ac.wales", ["-m", "100000M", "-T"] the address of a computer followed by a list of command line options.

EXAMPLES:

4.1.2 ChildCreate

▷ `ChildCreate()` (function)
▷ `ChildProcess(`*str*`)` (function)
▷ `ChildProcess(`*str*`)` (function)
▷ `ChildProcess(`*str*`)` (function)

Starts a GAP session as a child process and returns a stream to the child process. If no argument is given then the child process is created on the local machine; otherwise the argument should be: (1) *str*="computer.address.ie" the address of a remote computer for which ssh has been configured to require no password from the user; (2) *str*=["-m", "100000M", "-T"] a list of GAP command line

options; (3) *str*="computer.ac.wales", ["-m", "100000M", "-T"] the address of a computer followed by a list of command line options.

EXAMPLES:

Chapter 5

Resolutions of the ground ring

5.1

5.1.1 TietzeReducedResolution

▷ `TietzeReducedResolution(R)` (function)

Inputs a $\mathbb{Z}G$-resolution R and returns a $\mathbb{Z}G$-resolution S which is obtained from R by applying "Tietze like operations" in each dimension. The hope is that S has fewer free generators than R.

EXAMPLES: 1

5.1.2 ResolutionArithmeticGroup

▷ `ResolutionArithmeticGroup(P, n)` (function)

Inputs a positive integer n and a string P equal to one of the following:

"SL(2,Z)" , "SL(3,Z)" , "PGL(3,Z[i])" , "PGL(3,Eisenstein_Integers)" , "PSL(4,Z)" , "PSL(4,Z)_b" , "PSL(4,Z)_c" , "PSL(4,Z)_d" , "Sp(4,Z)"

or the string

"GL(2,O(-d))"

for d=1, 2, 3, 5, 6, 7, 10, 11, 13, 14, 15, 17, 19, 21, 22, 23, 26, 43

or the string

"SL(2,O(-d))"

for d=2, 3, 5, 7, 10, 11, 13, 14, 15, 17, 19, 21, 22, 23, 26, 43, 67, 163

or the string

"SL(2,O(-d))_a"

for d=2, 7, 11, 19.

It returns n terms of a free ZG-resolution for the group G described by the string. Here O(-d) denotes the ring of integers of Q(sqrt(-d)) and subscripts _a, _b , _c , _d denote alternative non-free ZG-resolutions for a given group G.

Data for the first list of resolutions was provided provided by MATHIEU DUTOUR. Data for GL(2,O(-d)) was provided by SEBASTIAN SCHOENENNBECK. Data for SL(2,O(-d)) was provided bySEBASTIAN SCHOENNENBECK for d <= 26 and by ALEXANDER RAHM for d>26 and for the alternative complexes.

EXAMPLES: 1

5.1.3 FreeGResolution

▷ `FreeGResolution(P, n)` (function)
▷ `FreeGResolution(P, n, p)` (function)

Inputs a non-free ZG-resolution P with finite stabilizer groups, and a positive integer n. It returns a free ZG-resolution of length equal to the minimum of n and the length of P. If one requires only a mod p resolution then the prime p can be entered as an optional third argument.

The free resolution is returned without a contracting homotopy.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10

5.1.4 ResolutionGTree

▷ `ResolutionGTree(P, n)` (function)

Inputs a non-free ZG-resolution P of dimension 1 (i.e. a G-tree) with finite stabilizer groups, and a positive integer n. It returns a free ZG-resolution of length equal to n.

If P has a contracting homotopy then the free resolution is returned with a contracting homotopy.

This function was written by BUI ANH TUAN.

EXAMPLES:

5.1.5 ResolutionSL2Z

▷ `ResolutionSL2Z(p, n)` (function)

Inputs positive integers m,n and returns n terms of a ZG-resolution for the group $G = SL(2,Z[1/m])$.

This function is joint work with BUI ANH TUAN.

EXAMPLES: 1 , 2 , 3 , 4

5.1.6 ResolutionAbelianGroup

▷ `ResolutionAbelianGroup(L, n)` (function)
▷ `ResolutionAbelianGroup(G, n)` (function)

Inputs a list $L := [m_1, m_2, ..., m_d]$ of nonnegative integers, and a positive integer n. It returns n terms of a $\mathbb{Z}G$-resolution for the abelian group $G = Z_{L[1]} + Z_{L[2]} + + Z_L[d]$.

If G is finite then the first argument can also be the abelian group G itself.

EXAMPLES: 1 , 2 , 3

5.1.7 ResolutionAlmostCrystalGroup

▷ `ResolutionAlmostCrystalGroup(G, n)` (function)

Inputs a positive integer n and an almost crystallographic pcp group G. It returns n terms of a free ZG-resolution. (A group is almost crystallographic if it is nilpotent-by-finite and has no non-trivial finite normal subgroup. Such groups can be constructed using the ACLIB package.)

EXAMPLES: 1 , 2

5.1.8 ResolutionAlmostCrystalQuotient

▷ `ResolutionAlmostCrystalQuotient(G, n, c)` (function)
▷ `ResolutionAlmostCrystalQuotient(G, n, c, false)` (function)

An almost crystallographic group G is an extension of a finite group P by a nilpotent group T, and has no non-trivial finite normal subgroup. We define the relative lower central series by setting $T_1 = T$ and $T_{i+1} = [T_i, G]$.

This function inputs an almost crystallographic group G together with positive integers n and c. It returns n terms of a free ZQ-resolution R for the group $Q = G/T_c$.

In addition to the usual components, the resolution R has the component $R.quotientHomomorphism$ which gives the quotient homomorphism $G \longrightarrow Q$.

If a fourth optional variable is set equal to "false" then the function omits to test whether Q is finite and a "more canonical" resolution is constructed.

EXAMPLES: 1

5.1.9 ResolutionArtinGroup

▷ `ResolutionArtinGroup(D, n)` (function)

Inputs a Coxeter diagram D and an integer $n > 1$. It returns n terms of a free ZG-resolution R where G is the Artin monoid associated to D. It is conjectured that R is also a free resolution for the Artin group G. The conjecture is known to hold in certain cases.

$G = R.group$ is infinite and returned as a finitely presented group. The list $R.elts$ is a partial listing of the elements of G which grows as R is used. Initially $R.elts$ is empty and then, any time the boundary of a resolution generator is called, $R.elts$ is updated to include elements of G involved in the boundary.

The contracting homotopy on R has not yet been implemented! Furthermore, the group G is currently returned only as a finitely presented group (without any method for solving the word problem).

EXAMPLES: 1 , 2 , 3 , 4 , 5

5.1.10 ResolutionAsphericalPresentation

▷ `ResolutionAsphericalPresentation(F, R, n)` (function)

Inputs a free group F, a set R of words in F which constitute an aspherical presentation for a group G, and a positive integer n. (Asphericity can be a difficult property to verify. The function $IsAspherical(F,R)$ could be of help.)

The function returns n terms of a free ZG-resolution R which has generators in dimensions < 3 only. No contracting homotopy on R will be returned.

EXAMPLES: 1 , 2 , 3

5.1.11 ResolutionBieberbachGroup

▷ `ResolutionBieberbachGroup(G)` (function)
▷ `ResolutionBieberbachGroup(G, v)` (function)

Inputs a torsion free crystallographic group G, also known as a Bieberbach group, represented using AffineCrystGroupOnRight as in the GAP package Cryst. It also optionally inputs a choice of vector v in the euclidean space R^n on which G acts freely. The function returns $n+1$ terms of the free ZG-resolution of Z arising as the cellular chain complex of the tesselation of R^n by the Dirichlet-Voronoi fundamental domain determined by v.

This function is part of the HAPcryst package written by MARC ROEDER and thus requires the HAPcryst package to be loaded.

The function requires the use of Polymake software.

EXAMPLES: 1

5.1.12 ResolutionCoxeterGroup

▷ `ResolutionCoxeterGroup(D, n)` (function)

Inputs a Coxeter diagram D and an integer $n > 1$. It returns k terms of a free ZG-resolution R where G is the Coxeter group associated to D. Here k is the maximum of n and the number of vertices in the Coxeter diagram. At present the implementation is only for finite Coxeter groups and the group G is returned as a permutation group. The contracting homotopy on R has not yet been implemented!

EXAMPLES: 1 , 2 , 3

5.1.13 ResolutionDirectProduct

▷ `ResolutionDirectProduct(R, S)` (function)
▷ `ResolutionDirectProduct(R, S, str)` (function)

Inputs a ZG-resolution R and ZH-resolution S. It outputs a ZD-resolution for the direct product $D = GxH$.

If G and H lie in a common group K, and if they commute and have trivial intersection, then an optional third variable str="internal" can be used. This will force D to be the subgroup GH in K.

EXAMPLES: 1

5.1.14 ResolutionExtension

▷ `ResolutionExtension(g, R, S)` (function)
▷ `ResolutionExtension(g, R, S, str)` (function)
▷ `ResolutionExtension(g, R, S, str, GmapE)` (function)

Inputs a surjective group homomorphism $g : E \longrightarrow G$ with kernel N. It also inputs a ZN-resolution R and a ZG-resolution S. It returns a ZE-resolution. The groups E and G can be infinite.

If an optional fourth argument $str$is set equal to "TestFiniteness" then the groups N and G will be tested to see if they are finite. If they are finite then some speed saving routines will be invoked. One can also set str="NoTest".

If the homomorphism g is such that the GAP function $PreImagesElement(g,x)$ doesn't work, then a function $GmapE()$ should be included as a fifth input. For any x in G this function should return an element $GmapE(x)$ in E which gets mapped onto x by g.

The contracting homotopy on the ZE-resolution has not yet been fully implemented for infinite groups!

EXAMPLES: 1 , 2

5.1.15 ResolutionFiniteDirectProduct

▷ `ResolutionFiniteDirectProduct(R, S)` (function)
▷ `ResolutionFiniteDirectProduct(R, S, str)` (function)

Inputs a ZG-resolution R and ZH-resolution S where G and H are finite groups. It outputs a ZD-resolution for the direct product $D = G \times H$.

If G and H lie in a common group K, and if they commute and have trivial intersection, then an optional third variable str="internal" can be used. This will force D to be the subgroup GH in K.

EXAMPLES:

5.1.16 ResolutionFiniteExtension

▷ `ResolutionFiniteExtension(gensE, gensG, R, n)` (function)
▷ `ResolutionFiniteExtension(gensE, gensG, R, n, true)` (function)
▷ `ResolutionFiniteExtension(gensE, gensG, R, n, false, S)` (function)

Inputs: a set $gensE$ of generators for a finite group E; a set $gensG$ equal to the image of $gensE$ in a quotient group G of E; a ZG-resolution R up to dimension at least n; a positive integer n. It uses the $TwistedTensorProduct()$ construction to return n terms of a ZE-resolution.

The function has an optional fourth argument which, when set equal to "true", invokes tietze reductions in the construction of a resolution for the kernel of $E \longrightarrow G$.

If a ZN-resolution S is available, where N is the kernel of the quotient $E \longrightarrow G$, then this can be incorporated into the computations using an optional fifth argument.

EXAMPLES: 1 , 2

5.1.17 ResolutionFiniteGroup

▷ `ResolutionFiniteGroup(gens, n)` (function)
▷ `ResolutionFiniteGroup(gens, n, true)` (function)

▷ ResolutionFiniteGroup(*gens*, *n*, *false*, *p*) (function)
▷ ResolutionFiniteGroup(*gens*, *n*, *false*, *0*, *str*) (function)

Inputs a set *gens* of generators for a finite group G and a positive integer n. It outputs n terms of a ZG-resolution.

The function has an optional third argument which, when set equal to *true*, invokes tietze reductions in the construction of the resolution.

The function has an optional fourth argument which, when set equal to a prime p, records the fact that the resolution will only be used for mod p calculations. This could speed up subsequent constructions.

The function has an optional fifth argument *str* which, when set equal to "extendible", returns a resolution whose length can be increased using the command R!.extend() .

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20

5.1.18 ResolutionFiniteSubgroup

▷ ResolutionFiniteSubgroup(*R*, *K*) (function)
▷ ResolutionFiniteSubgroup(*R*, *gensG*, *gensK*) (function)

Inputs a ZG-resolution for a finite group G and a subgroup K of index $|G:K|$. It returns a free ZK-resolution whose ZK-rank is $|G:K|$ times the ZG-rank in each dimension.

Generating sets *gensG*, *gensK* for G and K can also be input to the function (though the method does not depend on a choice of generators).

This ZK-resolution is not reduced. ie. it has more than one generator in dimension 0.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7

5.1.19 ResolutionGraphOfGroups

▷ ResolutionGraphOfGroups(*D*, *n*) (function)
▷ ResolutionGraphOfGroups(*D*, *n*, *L*) (function)

Inputs a graph of groups D and a positive integer n. It returns n terms of a free ZG-resolution for the fundamental group G of D.

An optional third argument $L = [R_1, \ldots, R_t]$ can be used to list (in any order) free resolutions for some/all of the vertex and edge groups in D. If for some vertex or edge group no resolution is listed in L then the function *ResolutionFiniteGroup*() will be used to try to construct the resolution.

The ZG-resolution is usually not reduced. i.e. it has more than one generator in dimension 0.

The contracting homotopy on the ZG-resolution has not yet been implemented! Furthermore, the group G is currently returned only as a finitely presented group (without any method for solving the word problem).

EXAMPLES: 1 , 2 , 3

5.1.20 ResolutionNilpotentGroup

▷ ResolutionNilpotentGroup(*G*, *n*) (function)
▷ ResolutionNilpotentGroup(*G*, *n*, *str*) (function)

Inputs a nilpotent group G and positive integer n. It returns n terms of a free ZG-resolution. The resolution is computed using a divide-and-conquer technique involving the lower central series.

This function can be applied to infinite groups G. For finite groups the function *ResolutionNormalSeries*() probably gives better results.

If an optional third argument *str* is set equal to "TestFiniteness" then the groups N and G will be tested to see if they are finite. If they are finite then some speed saving routines will be invoked.

The contracting homotopy on the ZE-resolution has not yet been fully implemented for infinite groups.

EXAMPLES: 1 , 2 , 3 , 4 , 5

5.1.21 ResolutionNormalSeries

▷ `ResolutionNormalSeries(L, n)` (function)
▷ `ResolutionNormalSeries(L, n, true)` (function)
▷ `ResolutionNormalSeries(L, n, false, p)` (function)

Inputs a positive integer n and a list $L = [L_1, ..., L_k]$ of normal subgroups L_i of a finite group G satisfying $G = L_1 > L2 > \ldots > L_k$. Alternatively, $L = [gensL_1, ...gensL_k]$ can be a list of generating sets for the L_i (and these particular generators will be used in the construction of resolutions). It returns a ZG-resolution by repeatedly using the function *ResolutionFiniteExtension*().

The function has an optional third argument which, if set equal to true, invokes tietze reductions in the construction of resolutions.

The function has an optional fourth argument which, if set equal to p > 0, produces a resolution which is only valid for mod p calculations.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8

5.1.22 ResolutionPrimePowerGroup

▷ `ResolutionPrimePowerGroup(P, n)` (function)
▷ `ResolutionPrimePowerGroup(G, n, p)` (function)

Inputs a p-group P and integer n>0. It uses GAP's standard linear algebra functions over the field F of p elements to construct a free FP-resolution for mod p calculations only. The resolution is minimal - meaning that the number of generators of R_n equals the rank of $H_n(P,F)$.

The function can also be used to obtain a free non-minimal FG-resolution of a small nilpotent group G of non-prime-power order. In this case the prime p must be entered as the third input variable. (In the non-prime-power nilpotent case the algorithm is naive and not very good.)

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6

5.1.23 ResolutionSmallFpGroup

▷ `ResolutionSmallFpGroup(G, n)` (function)
▷ `ResolutionSmallFpGroup(G, n, p)` (function)

Inputs a small finitely presented group G and an integer n>0. It returns n terms of a ZG-resolution which, in dimensions 1 and 2, corresponds to the given presentation for G. The method returns no contracting homotopy for the resolution.

The function has an optional fourth argument which, when set equal to a prime p, records the fact that the resolution will only be used for mod p calculations. This could speed up subsequent constructions.

This function was written by Irina Kholodna.

EXAMPLES: 1 , 2

5.1.24 ResolutionSubgroup

▷ `ResolutionSubgroup(R, K)` (function)

Inputs a ZG-resolution for an (infinite) group G and a subgroup K of finite index $|G:K|$. It returns a free ZK-resolution whose ZK-rank is $|G:K|$ times the ZG-rank in each dimension.

If G is finite then the function $ResolutionFiniteSubgroup(R,G,K)$ will probably work better. In particular, resolutions from this function probably won't work with the function $EquivariantChainMap()$. This ZK-resolution is not reduced. i.e. it has more than one generator in dimension 0.

EXAMPLES: 1 , 2 , 3

5.1.25 ResolutionSubnormalSeries

▷ `ResolutionSubnormalSeries(L, n)` (function)

Inputs a positive integer n and a list $L = [L_1,\ldots,L_k]$ of subgroups L_i of a finite group $G = L_1$ such that $L_1 > L2 \ldots > L_k$ is a subnormal series in G (meaning that each L_{i+1} must be normal in L_i). It returns a ZG-resolution by repeatedly using the function $ResolutionFiniteExtension()$.

If L is a series of normal subgroups in G then the function $ResolutionNormalSeries(L,n)$ will possibly work more efficiently.

EXAMPLES: 1 , 2 , 3

5.1.26 TwistedTensorProduct

▷ `TwistedTensorProduct(R, S, EhomG, GmapE, NhomE, NEhomN, EltsE, Mult, InvE)` (function)

Inputs a ZG-resolution R, a ZN-resolution S, and other data relating to a short exact sequence $1 \longrightarrow N \longrightarrow E \longrightarrow G \longrightarrow 1$. It uses a perturbation technique of CTC Wall to construct a ZE-resolution F. Both G and N could be infinite. The "length" of F is equal to the minimum of the "length"s of R and S. The resolution R needs no contracting homotopy if no such homotopy is requied for F.

EXAMPLES: 1

5.1.27 ConjugatedResolution

▷ `ConjugatedResolution(R, x)` (function)

Inputs a ZG-resoluton R and an element x from some group containing G. It returns a ZG^x-resolution S where the group G^x is the conjugate of G by x. (The component $S!.elts$ will be a pseudolist rather than a list.)

EXAMPLES:

5.1.28 RecalculateIncidenceNumbers

▷ `RecalculateIncidenceNumbers(`*R*`)` (function)

Inputs a ZG-resoluton *R* which arises as the cellular chain complex of a regular CW-complex. (Thus the boundary of any cell is a list of distinct cells.) It recalculates the incidence numbers for *R*. If it is applied to a resolution that is not regular then a wrong answer may be returned.

EXAMPLES:

Chapter 6

Resolutions of modules

6.1

6.1.1 ResolutionFpGModule

▷ `ResolutionFpGModule(M, n)` (function)

Inputs an FpG-module M and a positive integer n where G is a finite p-group and F the field of p elements. It returns n terms of a minimal free FG-resolution of the module M.

EXAMPLES: 1 , 2 , 3

Chapter 7

Induced equivariant chain maps

7.1

7.1.1 EquivariantChainMap

▷ `EquivariantChainMap(R, S, f)` (function)

Inputs a ZG-resolution R, a ZG'-resolution S, and a group homomorphism $f : G \longrightarrow G'$. It outputs a component object M with the following components.

- M!.*source* is the resolution R.
- M!.*target* is the resolution S.
- M!.*mapping*(w,n) is a function which gives the image in S_n, under a chain map induced by f, of a word w in R_n. (Here R_n and S_n are the n-th modules in the resolutions R and S.)
- F!.*properties* is a list of pairs such as ["type", "equivariantChainMap"].

The resolution S must have a contracting homotopy.
EXAMPLES: 1 , 2 , 3 , 4

Chapter 8

Functors

8.1

8.1.1 ExtendScalars

▷ `ExtendScalars(R, G, EltsG)` (function)

Inputs a ZH-resolution R, a group G containing H as a subgroup, and a list $EltsG$ of elements of G. It returns the free ZG-resolution $(R \otimes_{ZH} ZG)$. The returned resolution S has S!.elts:=EltsG. This is a resolution of the ZG-module $(Z \otimes_{ZH} ZG)$. (Here $\otimes_{ZH}$ means tensor over ZH.)

EXAMPLES:

8.1.2 HomToIntegers

▷ `HomToIntegers(X)` (function)

Inputs either a ZG-resolution $X = R$, or an equivariant chain map $X = (F : R \longrightarrow S)$. It returns the cochain complex or cochain map obtained by applying $HomZG(_, Z)$ where Z is the trivial module of integers (characteristic 0).

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

8.1.3 HomToIntegersModP

▷ `HomToIntegersModP(R)` (function)

Inputs a ZG-resolution R and returns the cochain complex obtained by applying $HomZG(_, Z_p)$ where Z_p is the trivial module of integers mod p. (At present this functor does not handle equivariant chain maps.)

EXAMPLES: 1 , 2 , 3 , 4

8.1.4 HomToIntegralModule

▷ `HomToIntegralModule(R, f)` (function)

Inputs a ZG-resolution R and a group homomorphism $f : G \longrightarrow GL_n(Z)$ to the group of $n \times n$ invertible integer matrices. Here Z must have characteristic 0. It returns the cochain complex obtained by applying $HomZG(,A)$ where A is the ZG-module Z^n with G action via f. (At present this function does not handle equivariant chain maps.)

EXAMPLES: 1 , 2 , 3 , 4

8.1.5 TensorWithIntegralModule

▷ `TensorWithIntegralModule(R, f)` (function)

Inputs a ZG-resolution R and a group homomorphism $f : G \longrightarrow GL_n(Z)$ to the group of $n \times n$ invertible integer matrices. Here Z must have characteristic 0. It returns the chain complex obtained by tensoring over ZG with the ZG-module $A = Z^n$ with G action via f. (At present this function does not handle equivariant chain maps.)

EXAMPLES: 1

8.1.6 HomToGModule

▷ `HomToGModule(R, A)` (function)

Inputs a ZG-resolution R and an abelian G-outer group A. It returns the G-cocomplex obtained by applying $HomZG(,A)$. (At present this function does not handle equivariant chain maps.)

EXAMPLES: 1 , 2 , 3 , 4

8.1.7 InduceScalars

▷ `InduceScalars(R, hom)` (function)

Inputs a ZQ-resolution R and a surjective group homomorphism $hom : G \to Q$. It returns the unduced non-free ZG-resolution.

EXAMPLES:

8.1.8 LowerCentralSeriesLieAlgebra

▷ `LowerCentralSeriesLieAlgebra(G)` (function)
▷ `LowerCentralSeriesLieAlgebra(f)` (function)

Inputs a pcp group G. If each quotient G_c/G_{c+1} of the lower central series is free abelian or p-elementary abelian (for fixed prime p) then a Lie algebra $L(G)$ is returned. The abelian group underlying $L(G)$ is the direct sum of the quotients G_c/G_{c+1} . The Lie bracket on $L(G)$ is induced by the commutator in G. (Here $G_1 = G$, $G_{c+1} = [G_c, G]$.)

The function can also be applied to a group homomorphism $f : G \longrightarrow G'$. In this case the induced homomorphism of Lie algebras $L(f) : L(G) \longrightarrow L(G')$ is returned.

If the quotients of the lower central series are not all free or p-elementary abelian then the function returns fail.

This function was written by Pablo Fernandez Ascariz

EXAMPLES: 1 , 2 , 3

8.1.9 TensorWithIntegers

▷ `TensorWithIntegers(X)` (function)

Inputs either a ZG-resolution $X = R$, or an equivariant chain map $X = (F : R \longrightarrow S)$. It returns the chain complex or chain map obtained by tensoring with the trivial module of integers (characteristic 0).

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30

8.1.10 FilteredTensorWithIntegers

▷ `FilteredTensorWithIntegers(R)` (function)

Inputs a ZG-resolution R for which "filteredDimension" lies in NamesOfComponents(R). (Such a resolution can be produced using TwisterTensorProduct(), ResolutionNormalSubgroups() or FreeGResolution().) It returns the filtered chain complex obtained by tensoring with the trivial module of integers (characteristic 0).

EXAMPLES: 1 , 2

8.1.11 TensorWithTwistedIntegers

▷ `TensorWithTwistedIntegers(X, rho)` (function)

Inputs either a ZG-resolution $X = R$, or an equivariant chain map $X = (F : R \longrightarrow S)$. It also inputs a function $rho: G \to \mathbb{Z}$ where the action of $g \in G$ on $\mathbb{Z}$ is such that $g.1 = rho(g)$. It returns the chain complex or chain map obtained by tensoring with the (twisted) module of integers (characteristic 0).

EXAMPLES: 1 , 2 , 3 , 4

8.1.12 TensorWithIntegersModP

▷ `TensorWithIntegersModP(X, p)` (function)

Inputs either a ZG-resolution $X = R$, or a characteristics 0 chain complex, or an equivariant chain map $X = (F : R \longrightarrow S)$, or a chain map between characteristic 0 chain complexes, together with a prime p. It returns the chain complex or chain map obtained by tensoring with the trivial module of integers modulo p.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

8.1.13 TensorWithTwistedIntegersModP

▷ `TensorWithTwistedIntegersModP(X, p, rho)` (function)

Inputs either a ZG-resolution $X = R$, or an equivariant chain map $X = (F : R \longrightarrow S)$, and a prime p. It also inputs a function $rho: G \to \mathbb{Z}$ where the action of $g \in G$ on $\mathbb{Z}$ is such that $g.1 = rho(g)$. It returns the chain complex or chain map obtained by tensoring with the trivial module of integers modulo p.

EXAMPLES: 1

8.1.14 TensorWithRationals

▷ `TensorWithRationals(`*R*`)` (function)

Inputs a ZG-resolution R and returns the chain complex obtained by tensoring with the trivial module of rational numbers.

EXAMPLES: 1 , 2 , 3 , 4

Chapter 9

Chain complexes

9.1

9.1.1 ChainComplex

▷ `ChainComplex(T)` (function)

Inputs a pure cubical complex, or cubical complex, or simplicial complex T and returns the (often very large) cellular chain complex of T.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13

9.1.2 ChainComplexOfPair

▷ `ChainComplexOfPair(T, S)` (function)

Inputs a pure cubical complex or cubical complex T and contractible subcomplex S. It returns the quotient $C(T)/C(S)$ of cellular chain complexes.

EXAMPLES: 1 , 2

9.1.3 ChevalleyEilenbergComplex

▷ `ChevalleyEilenbergComplex(X, n)` (function)

Inputs either a Lie algebra $X = A$ (over the ring of integers Z or over a field K) or a homomorphism of Lie algebras $X = (f : A \longrightarrow B)$, together with a positive integer n. It returns either the first n terms of the Chevalley-Eilenberg chain complex $C(A)$, or the induced map of Chevalley-Eilenberg complexes $C(f) : C(A) \longrightarrow C(B)$.

(The homology of the Chevalley-Eilenberg complex $C(A)$ is by definition the homology of the Lie algebra A with trivial coefficients in Z or K).

This function was written by PABLO FERNANDEZ ASCARIZ

EXAMPLES: 1

9.1.4 LeibnizComplex

▷ `LeibnizComplex(X, n)` (function)

Inputs either a Lie or Leibniz algebra $X = A$ (over the ring of integers Z or over a field K) or a homomorphism of Lie or Leibniz algebras $X = (f : A \longrightarrow B)$, together with a positive integer n. It returns either the first n terms of the Leibniz chain complex $C(A)$, or the induced map of Leibniz complexes $C(f) : C(A) \longrightarrow C(B)$.

(The Leibniz complex $C(A)$ was defined by J.-L.Loday. Its homology is by definition the Leibniz homology of the algebra A).

This function was written by PABLO FERNANDEZ ASCARIZ

EXAMPLES:

9.1.5 SuspendedChainComplex

▷ `SuspendedChainComplex(C)` (function)

Inputs a chain complex C and returns the chain complex S defined by applying the degree shift $S_n = C_{n-1}$ to chain groups and boundary homomorphisms.

EXAMPLES: 1

9.1.6 ReducedSuspendedChainComplex

▷ `ReducedSuspendedChainComplex(C)` (function)

Inputs a chain complex C and returns the chain complex S defined by applying the degree shift $S_n = C_{n-1}$ to chain groups and boundary homomorphisms for all $n > 0$. The chain complex S has trivial homology in degree 0 and $S_0 = \mathbb{Z}$.

EXAMPLES: 1

9.1.7 CoreducedChainComplex

▷ `CoreducedChainComplex(C)` (function)
▷ `CoreducedChainComplex(C, 2)` (function)

Inputs a chain complex C and returns a quasi-isomorphic chain complex D. In many cases the complex D should be smaller than C. If an optional second input argument is set equal to 2 then an alternative method is used for reducing the size of the chain complex.

EXAMPLES: 1

9.1.8 TensorProductOfChainComplexes

▷ `TensorProductOfChainComplexes(C, D)` (function)

Inputs two chain complexes C and D of the same characteristic and returns their tensor product as a chain complex.

This function was written by LE VAN LUYEN.

EXAMPLES: 1

9.1.9 LefschetzNumber

▷ `LefschetzNumber(`*F*`)` (function)

Inputs a chain map $F:C \to C$ with common source and target. It returns the Lefschetz number of the map (that is, the alternating sum of the traces of the homology maps in each degree).

EXAMPLES:

Chapter 10

Sparse Chain complexes

10.1

10.1.1 SparseMat

▷ `SparseMat(A)` (function)

Inputs a matrix A and returns the matrix in sparse format.
EXAMPLES: 1

10.1.2 TransposeOfSparseMat

▷ `TransposeOfSparseMat(A)` (function)

Inputs a sparse matrix A and returns its transpose sparse format.
EXAMPLES:

10.1.3 ReverseSparseMat

▷ `ReverseSparseMat(A)` (function)

Inputs a sparse matrix A and modifies it by reversing the order of the columns. This function modifies A and returns no value.
EXAMPLES:

10.1.4 SparseRowMult

▷ `SparseRowMult(A, i, k)` (function)

Multiplies the i-th row of a sparse matrix A by k. The sparse matrix A is modified but nothing is returned.
EXAMPLES:

10.1.5 SparseRowInterchange

▷ `SparseRowInterchange(A, i, k)` (function)

Interchanges the i-th and j-th rows of a sparse matrix A by k. The sparse matrix A is modified but nothing is returned.

EXAMPLES:

10.1.6 SparseRowAdd

▷ `SparseRowAdd(A, i, j, k)` (function)

Adds k times the j-th row to the i-th row of a sparse matrix A. The sparse matrix A is modified but nothing is returned.

EXAMPLES:

10.1.7 SparseSemiEchelon

▷ `SparseSemiEchelon(A)` (function)

Converts a sparse matrix A to semi-echelon form (which means echelon form up to a permutation of rows). The sparse matrix A is modified but nothing is returned.

EXAMPLES:

10.1.8 RankMatDestructive

▷ `RankMatDestructive(A)` (function)

Returns the rank of a sparse matrix A. The sparse matrix A is modified during the calculation.

EXAMPLES:

10.1.9 RankMat

▷ `RankMat(A)` (function)

Returns the rank of a sparse matrix A.

EXAMPLES:

10.1.10 SparseChainComplex

▷ `SparseChainComplex(Y)` (function)

Inputs a regular CW-complex Y and returns a sparse chain complex which is chain homotopy equivalent to the cellular chain complex of Y. The function uses discrete vector fields to calculate a smallish chain complex.

EXAMPLES: 1 , 2

10.1.11 SparseChainComplexOfRegularCWComplex

▷ `SparseChainComplexOfRegularCWComplex(Y)` (function)

Inputs a regular CW-complex Y and returns its cellular chain complex as a sparse chain complex. The function SparseChainComplex(Y) will usually return a smaller chain complex.

EXAMPLES:

10.1.12 SparseBoundaryMatrix

▷ `SparseBoundaryMatrix(C, n)` (function)

Inputs a sparse chain complex C and integer n. Returns the n-th boundary matrix of the chain complex in sparse format.

EXAMPLES:

10.1.13 Bettinumbers

▷ `Bettinumbers(C, n)` (function)

Inputs a sparse chain complex C and integer n. Returns the n-th Netti number of the chain complex.

EXAMPLES: 1 , 2

Chapter 11

Homology and cohomology groups

11.1

11.1.1 Cohomology

▷ `Cohomology(X, n)` (function)

Inputs either a cochain complex $X = C$ (or G-cocomplex C) or a cochain map $X = (C \longrightarrow D)$ in characteristic p together with a non-negative intereg n.

- If $X = C$ and $p = 0$ then the torsion coefficients of $H^n(C)$ are retuned. If $X = C$ and p is prime then the dimension of $H^n(C)$ are retuned.
- If $X = (C \longrightarrow D)$ then the induced homomorphism $H^n(C) \longrightarrow H^n(D)$ is returned as a homomorphism of finitely presented groups.

A G-cocomplex C can also be input. The cohomology groups of such a complex may not be abelian. WARNING: in this case Cohomology(C,n) returns the abelian invariants of the n-th cohomology group of C.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26

11.1.2 CohomologyModule

▷ `CohomologyModule(C, n)` (function)

Inputs a G-cocomplex C together with a non-negative integer n. It returns the cohomology $H^n(C)$ as a G-outer group. If C was constructed from a resolution R by homing to an abelian G-outer group A then, for each x in H:=CohomologyModule(C,n), there is a function f:=H!.representativeCocycle(x) which is a standard n-cocycle corresponding to the cohomology class x. (At present this works only for n=1,2,3.)

EXAMPLES: 1 , 2 , 3

11.1.3 CohomologyPrimePart

▷ `CohomologyPrimePart(C, n, p)` (function)

Inputs a cochain complex C in characteristic 0, a positive integer n, and a prime p. It returns a list of those torsion coefficients of $H^n(C)$ that are positive powers of p. The function uses the EDIM package by Frank Luebeck.

EXAMPLES:

11.1.4 GroupCohomology

▷ `GroupCohomology(X, n)` (function)
▷ `GroupCohomology(X, n, p)` (function)

Inputs a positive integer n and either

- a finite group $X = G$
- or a nilpotent Pcp-group $X = G$
- or a space group $X = G$
- or a list $X = D$ representing a graph of groups
- or a pair $X = [\text{"}Artin\text{"}, D]$ where D is a Coxeter diagram representing an infinite Artin group G.
- or a pair $X = [\text{"}Coxeter\text{"}, D]$ where D is a Coxeter diagram representing a finite Coxeter group G.

It returns the torsion coefficients of the integral cohomology $H^n(G,Z)$.

There is an optional third argument which, when set equal to a prime p, causes the function to return the the mod p cohomology $H^n(G,Z_p)$ as a list of length equal to its rank.

This function is a composite of more basic functions, and makes choices for a number of parameters. For a particular group you would almost certainly be better using the more basic functions and making the choices yourself!

EXAMPLES: 1 , 2

11.1.5 GroupHomology

▷ `GroupHomology(X, n)` (function)
▷ `GroupHomology(X, n, p)` (function)

Inputs a positive integer n and either

- a finite group $X = G$
- or a nilpotent Pcp-group $X = G$
- or a space group $X = G$
- or a list $X = D$ representing a graph of groups
- or a pair $X = [\text{"}Artin\text{"}, D]$ where D is a Coxeter diagram representing an infinite Artin group G.
- or a pair $X = [\text{"}Coxeter\text{"}, D]$ where D is a Coxeter diagram representing a finite Coxeter group G.

It returns the torsion coefficients of the integral homology $H_n(G,Z)$.

There is an optional third argument which, when set equal to a prime p, causes the function to return the mod p homology $H_n(G,Z_p)$ as a list of lenth equal to its rank.

This function is a composite of more basic functions, and makes choices for a number of parameters. For a particular group you would almost certainly be better using the more basic functions and making the choices yourself!

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

11.1.6 PersistentHomologyOfQuotientGroupSeries

▷ `PersistentHomologyOfQuotientGroupSeries(S, n)` (function)
▷ `PersistentHomologyOfQuotientGroupSeries(S, n, p, Resolution_Algorithm)` (function)

Inputs a positive integer n and a decreasing chain $S = [S_1, S_2, ..., S_k]$ of normal subgroups in a finite p-group $G = S_1$. It returns the bar code of the persistent mod p homology in degree n of the sequence of quotient homomorphisms $G \to G/S_k \to G/S_{k-1} \to ... \to G/S_2$. The bar code is returned as a matrix containing the dimensions of the images of the induced homology maps.

If one sets $p = 0$ then the integral persitent homology bar code is returned. This is a matrix whose entries are pairs of the lists: the list of abelian invariants of the images of the induced homology maps and the cokernels of the induced homology maps. (The matrix probably does not uniquely determine the induced homology maps.)

Non prime-power (and possibly infinite) groups G can also be handled; in this case the prime must be entered as a third argument, and the resolution algorithm (e.g. ResolutionNilpotentGroup) can be entered as a fourth argument. (The default algorithm is ResolutionFiniteGroup, so this must be changed for infinite groups.)

EXAMPLES:

11.1.7 PersistentCohomologyOfQuotientGroupSeries

▷ `PersistentCohomologyOfQuotientGroupSeries(S, n)` (function)
▷ `PersistentCohomologyOfQuotientGroupSeries(S, n, p, Resolution_Algorithm)` (function)

Inputs a positive integer n and a decreasing chain $S = [S_1, S_2, ..., S_k]$ of normal subgroups in a finite p-group $G = S_1$. It returns the bar code of the persistent mod p cohomology in degree n of the sequence of quotient homomorphisms $G \to G/S_k \to G/S_{k-1} \to ... \to G/S_2$. The bar code is returned as a matrix containing the dimensions of the images of the induced homology maps.

If one sets $p = 0$ then the integral persitent cohomology bar code is returned. This is a matrix whose entries are pairs of the lists: the list of abelian invariants of the images of the induced cohomology maps and the cokernels of the induced cohomology maps. (The matrix probably does not uniquely determine the induced homology maps.)

Non prime-power (and possibly infinite) groups G can also be handled; in this case the prime must be entered as a third argument, and the resolution algorithm (e.g. ResolutionNilpotentGroup) can be entered as a fourth argument. (The default algorithm is ResolutionFiniteGroup, so this must be changed for infinite groups.)

(The implementation is possibly a little less efficient than that of the corresponding persistent homology function.)

EXAMPLES:

11.1.8 UniversalBarCode

▷ `UniversalBarCode(str, n, d)` (function)
▷ `UniversalBarCode(str, n, d, k)` (function)

Inputs integers n,d that identify a prime power group G=SmallGroup(n,d), together with one of the strings *str*= "UpperCentralSeries", LowerCentralSeries", "DerivedSeries", "UpperPCentralSeries", "LowerPCentralSeries". The function returns a matrix of rational functions; the coefficients of x^k in their expansions yield the persistence matrix for the degree k homology with trivial mod p coefficients associated to the quotients of G by the terms of the given series.

If the additional integer argument k is supplied then the function returns the degree k homology persistence matrix.

EXAMPLES: 1

11.1.9 PersistentHomologyOfSubGroupSeries

▷ `PersistentHomologyOfSubGroupSeries(S, n)` (function)
▷ `PersistentHomologyOfSubGroupSeries(S, n, p, Resolution_Algorithm)` (function)

Inputs a positive integer n and a decreasing chain $S = [S_1, S_2, ..., S_k]$ of subgroups in a finite p-group $G = S_1$. It returns the bar code of the persistent mod p homology in degree n of the sequence of inclusion homomorphisms $S_k \to S_{k-1} \to ... \to S_1 = G$. The bar code is returned as a binary matrix.

Non prime-power (and possibly infinite) groups G can also be handled; in this case the prime must be entered as a third argument, and the resolution algorithm (e.g. ResolutionNilpotentGroup) must be entered as a fourth argument.

EXAMPLES:

11.1.10 PersistentHomologyOfFilteredChainComplex

▷ `PersistentHomologyOfFilteredChainComplex(C, n, p)` (function)

Inputs a filtered chain complex C (of characteristic 0 or p) together with a positive integer n and prime p. It returns the bar code of the persistent mod p homology in degree n of the filtered chain complex C. (This function needs a more efficient implementation. Its fine as it stands for investigation in group homology, but not sufficiently efficient for the homology of large complexes arising in applied topology.)

EXAMPLES: 1 , 2

11.1.11 PersistentHomologyOfCommutativeDiagramOfPGroups

▷ `PersistentHomologyOfCommutativeDiagramOfPGroups(D, n)` (function)

Inputs a commutative diagram D of finite p-groups and a positive integer n. It returns a list containing, for each homomorphism in the nerve of D, a triple $[k,l,m]$ where k is the dimension of the source of the induced mod p homology map in degree n, l is the dimension of the image, and m is the dimension of the cokernel.

EXAMPLES:

11.1.12 PersistentHomologyOfFilteredPureCubicalComplex

▷ `PersistentHomologyOfFilteredPureCubicalComplex(M, n)` (function)

Inputs a filtered pure cubical complex M and a non-negative integer n. It returns the degree n persistent homology of M with rational coefficients.

EXAMPLES:

11.1.13 PersistentHomologyOfPureCubicalComplex

▷ `PersistentHomologyOfPureCubicalComplex(L, n, p)` (function)

Inputs a positive integer n, a prime p and an increasing chain $L = [L_1, L_2, ..., L_k]$ of subcomplexes in a pure cubical complex L_k. It returns the bar code of the persistent mod p homology in degree n of the sequence of inclusion maps. The bar code is returned as a matrix. (This function is extremely inefficient and it is better to use PersistentHomologyOFilteredfPureCubicalComplex.

EXAMPLES:

11.1.14 ZZPersistentHomologyOfPureCubicalComplex

▷ `ZZPersistentHomologyOfPureCubicalComplex(L, n, p)` (function)

Inputs a positive integer n, a prime p and any sequence $L = [L_1, L_2, ..., L_k]$ of subcomplexes of some pure cubical complex. It returns the bar code of the zig-zag persistent mod p homology in degree n of the sequence of maps $L_1 \to L_1 \cup L_2 \leftarrow L_2 \to L_2 \cup L_3 \leftarrow L_4 \to ... \leftarrow L_k$. The bar code is returned as a matrix.

EXAMPLES:

11.1.15 RipsHomology

▷ `RipsHomology(G, n)` (function)
▷ `RipsHomology(G, n, p)` (function)

Inputs a graph G, a non-negative integer n (and optionally a prime number p). It returns the integral homology (or mod p homology) in degree n of the Rips complex of G.

EXAMPLES:

11.1.16 BarCode

▷ `BarCode(P)` (function)

Inputs an integer persistence matrix P and returns the same information in the form of a binary matrix (corresponding to the usual bar code).

EXAMPLES: 1 , 2 , 3

11.1.17 BarCodeDisplay

▷ `BarCodeDisplay(P)` (function)
▷ `BarCodeDisplay(P, str)` (function)
▷ `BarCodeCompactDisplay(P)` (function)
▷ `BarCodeCompactDisplay(P, str)` (function)

Inputs an integer persistence matrix P, and an optional string, such as *str*="mozilla" specifying a viewer/browser. It displays a picture of the bar code (using GraphViz software). The compact display is better for large bar codes.

EXAMPLES: 1 , 2

11.1.18 Homology

▷ `Homology(X, n)` (function)

Inputs either a chain complex $X = C$ or a chain map $X = (C \longrightarrow D)$.

- If $X = C$ then the torsion coefficients of $H_n(C)$ are retuned.
- If $X = (C \longrightarrow D)$ then the induced homomorphism $H_n(C) \longrightarrow H_n(D)$ is returned as a homomorphism of finitely presented groups.

A G-complex C can also be input. The homology groups of such a complex may not be abelian. WARNING: in this case Homology(C,n) returns the abelian invariants of the n-th homology group of C.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45

11.1.19 HomologyPb

▷ `HomologyPb(C, n)` (function)

This is a back-up function which might work in some instances where $Homology(C,n)$ fails. It is most useful for chain complexes whose boundary homomorphisms are sparse.

It inputs a chain complex C in characteristic 0 and returns the torsion coefficients of $H_n(C)$. There is a small probability that an incorrect answer could be returned. The computation relies on probabilistic Smith Normal Form algorithms implemented in the Simplicial Homology GAP package. This package therefore needs to be loaded. The computation is stored as a component of C so, when called a second time for a given C and n, the calculation is recalled without rerunning the algorithm.

The choice of probabalistic algorithm can be changed using the command

SetHomologyAlgorithm(HomologyAlgorithm[i]);

where i = 1,2,3 or 4. The upper limit for the probability of an incorrect answer can be set to any rational number 0<e<= 1 using the following command.

SetUncertaintyTolerence(e);
See the Simplicial Homology package manual for further details.
EXAMPLES:

11.1.20 HomologyVectorSpace

▷ `HomologyVectorSpace(X, n)` (function)

Inputs either a chain complex $X = C$ or a chain map $X = (C \longrightarrow D)$ in prime characteristic.

- If $X = C$ then $H_n(C)$ is retuned as a vector space.
- If $X = (C \longrightarrow D)$ then the induced homomorphism $H_n(C) \longrightarrow H_n(D)$ is returned as a homomorphism of vector spaces.

EXAMPLES:

11.1.21 HomologyPrimePart

▷ `HomologyPrimePart(C, n, p)` (function)

Inputs a chain complex C in characteristic 0, a positive integer n, and a prime p. It returns a list of those torsion coefficients of $H_n(C)$ that are positive powers of p. The function uses the EDIM GAP package by Frank Luebeck.
EXAMPLES:

11.1.22 LeibnizAlgebraHomology

▷ `LeibnizAlgebraHomology(A, n)` (function)

Inputs a Lie or Leibniz algebra $X = A$ (over the ring of integers Z or over a field K), together with a positive integer n. It returns the n-dimensional Leibniz homology of A.
EXAMPLES: 1 , 2

11.1.23 LieAlgebraHomology

▷ `LieAlgebraHomology(A, n)` (function)

Inputs a Lie algebra A (over the integers or a field) and a positive integer n. It returns the homology $H_n(A,k)$ where k denotes the ground ring.
EXAMPLES: 1 , 2 , 3 , 4

11.1.24 PrimePartDerivedFunctor

▷ `PrimePartDerivedFunctor(G, R, F, n)` (function)

Inputs a finite group G, a positive integer n, at least $n+1$ terms of a ZP-resolution for a Sylow subgroup $P<G$ and a "mathematically suitable" covariant additive functor F such as TensorWithIntegers . It returns the abelian invariants of the p-component of the homology $H_n(F(R))$.

Warning: All calculations are assumed to be in characteristic 0. The function should not be used if the coefficient module is over the field of p elements.

"Mathematically suitable" means that the Cartan-Eilenberg double coset formula must hold.

EXAMPLES: 1 , 2 , 3 , 4

11.1.25 RankHomologyPGroup

▷ `RankHomologyPGroup(G, n)` (function)
▷ `RankHomologyPGroup(R, n)` (function)
▷ `RankHomologyPGroup(G, n, str)` (function)

Inputs a (smallish) p-group G, or n terms of a minimal Z_pG-resolution R of Z_p , together with a positive integer n. It returns the minimal number of generators of the integral homology group $H_n(G,Z)$.

If an option third string argument str="empirical" is included then an empirical algorithm will be used. This is one which always seems to yield the right answer but which we can't prove yields the correct answer.

EXAMPLES: 1

11.1.26 RankPrimeHomology

▷ `RankPrimeHomology(G, n)` (function)

Inputs a (smallish) p-group G together with a positive integer n. It returns a function $dim(k)$ which gives the rank of the vector space $H_k(G,Z_p)$ for all 0 <= k <= n.

EXAMPLES:

Chapter 12

Poincare series

12.1

12.1.1 EfficientNormalSubgroups

▷ `EfficientNormalSubgroups(G)` (function)
▷ `EfficientNormalSubgroups(G, k)` (function)

Inputs a prime-power group G and, optionally, a positive integer k. The default is $k = 4$. The function returns a list of normal subgroups N in G such that the Poincare series for G equals the Poincare series for the direct product $(N \times (G/N))$ up to degree k.

EXAMPLES: 1

12.1.2 ExpansionOfRationalFunction

▷ `ExpansionOfRationalFunction(f, n)` (function)

Inputs a positive integer n and a rational function $f(x) = p(x)/q(x)$ where the degree of the polynomial $p(x)$ is less than that of $q(x)$. It returns a list $[a_0, a_1, a_2, a_3, \ldots, a_n]$ of the first $n+1$ coefficients of the infinite expansion

$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \ldots$.

EXAMPLES: 1 , 2

12.1.3 PoincareSeries

▷ `PoincareSeries(G, n)` (function)
▷ `PoincareSeries(R, n)` (function)
▷ `PoincareSeries(L, n)` (function)
▷ `PoincareSeries(G)` (function)

Inputs a finite p-group G and a positive integer n. It returns a quotient of polynomials $f(x) = P(x)/Q(x)$ whose coefficient of x^k equals the rank of the vector space $H_k(G, Z_p)$ for all k in the range $k = 1$ to $k = n$. (The second input variable can be omitted, in which case the function tries to choose a "reasonable" value for n. For 2-groups the function PoincareSeriesLHS(G) can be used to produce an $f(x)$ that is correct in all degrees.)

In place of the group G the function can also input (at least n terms of) a minimal mod p resolution R for G.

Alternatively, the first input variable can be a list L of integers. In this case the coefficient of x^k in $f(x)$ is equal to the $(k+1)$st term in the list.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

12.1.4 PoincareSeriesPrimePart

▷ `PoincareSeriesPrimePart(G, p, n)` (function)

Inputs a finite group G, a prime p, and a positive integer n. It returns a quotient of polynomials $f(x) = P(x)/Q(x)$ whose coefficient of x^k equals the rank of the vector space $H_k(G, Z_p)$ for all k in the range $k = 1$ to $k = n$.

The efficiency of this function needs to be improved.

EXAMPLES: 1 , 2 , 3

12.1.5 PoincareSeriesLHS

▷ `PoincareSeriesLHS` (global variable)

Inputs a finite 2-group G and returns a quotient of polynomials $f(x) = P(x)/Q(x)$ whose coefficient of x^k equals the rank of the vector space $H_k(G, Z_2)$ for all k.

This function was written by PAUL SMITH. It use the Singular system for commutative algebra.

EXAMPLES:

12.1.6 Prank

▷ `Prank(G)` (function)

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

EXAMPLES:

Chapter 13

Cohomology ring structure

13.1

13.1.1 IntegralCupProduct

▷ `IntegralCupProduct(R, u, v, p, q)` (function)
▷ `IntegralCupProduct(R, u, v, p, q, P, Q, N)` (function)

(Various functions used to construct the cup product are also available.)

Inputs a ZG-resolution R, a vector u representing an element in $H^p(G,Z)$, a vector v representing an element in $H^q(G,Z)$ and the two integers p,q >0. It returns a vector w representing the cup product $u \cdot v$ in $H^{p+q}(G,Z)$. This product is associative and $u \cdot v = (-1)pqv \cdot u$. It provides $H^*(G,Z)$ with the structure of an anti-commutative graded ring. This function implements the cup product for characteristic 0 only.

The resolution R needs a contracting homotopy.

To save the function from having to calculate the abelian groups $H^n(G,Z)$ additional input variables can be used in the form $IntegralCupProduct(R,u,v,p,q,P,Q,N)$, where

- P is the output of the command $CR_{C}ocyclesAndCoboundaries(R,p,true)$
- Q is the output of the command $CR_{C}ocyclesAndCoboundaries(R,q,true)$
- N is the output of the command $CR_{C}ocyclesAndCoboundaries(R,p+q,true)$.

EXAMPLES: 1 , 2

13.1.2 IntegralRingGenerators

▷ `IntegralRingGenerators(R, n)` (function)

Inputs at least $n+1$ terms of a ZG-resolution and integer n> 0. It returns a minimal list of cohomology classes in $H^n(G,Z)$ which, together with all cup products of lower degree classes, generate the group $H^n(G,Z)$.

(Let a_i be the i-th canonical generator of the d-generator abelian group $H^n(G,Z)$. The cohomology class $n_1a_1 + ... + n_da_d$ is represented by the integer vector $u = (n_1,...,n_d)$.)

EXAMPLES: 1 , 2

13.1.3 ModPCohomologyGenerators

▷ `ModPCohomologyGenerators(G, n)` (function)
▷ `ModPCohomologyGenerators(R)` (function)

Inputs either a p-group G and positive integer n, or else n terms of a minimal Z_pG-resolution R of Z_p. It returns a pair whose first entry is a minimal set of homogeneous generators for the cohomology ring $A = H^*(G,Z_p)$ modulo all elements in degree greater than n. The second entry of the pair is a function *deg* which, when applied to a minimal generator, yields its degree.

WARNING: the following rule must be applied when multiplying generators x_i together. Only products of the form $x_1 * (x_2 * (x_3 * (x_4 * ...)))$ with $deg(x_i) \leq deg(x_{i+1})$ should be computed (since the x_i belong to a structure constant algebra with only a partially defined structure constants table).

EXAMPLES: 1

13.1.4 ModPCohomologyRing ( )

▷ `ModPCohomologyRing(G, n)` (function)
▷ `ModPCohomologyRing(G, n, level)` (function)
▷ `ModPCohomologyRing(R)` (function)
▷ `ModPCohomologyRing(R, level)` (function)

Inputs either a p-group G and positive integer n, or else n terms of a minimal Z_pG-resolution R of Z_p. It returns the cohomology ring $A = H^*(G,Z_p)$ modulo all elements in degree greater than n.

The ring is returned as a structure constant algebra A.

The ring A is graded. It has a component $A!.degree(x)$ which is a function returning the degree of each (homogeneous) element x in $GeneratorsOfAlgebra(A)$.

An optional input variable "level" can be set to one of the strings "medium" or "high". These settings determine parameters in the algorithm. The default setting is "medium".

When "level" is set to "high" the ring A is returned with a component $A!.niceBasis$. This component is a pair $[Coeff, Bas]$. Here Bas is a list of integer lists; a "nice" basis for the vector space A can be constructed using the command $List(Bas, x- > Product(List(x, i- > Basis(A)[i]))$. The coefficients of the canonical basis element $Basis(A)[i]$ are stored as $Coeff[i]$.

If the ring A is computed using the setting "level"="medium" then the component $A!.niceBasis$ can be added to A using the command $A := ModPCohomologyRing_part_2(A)$.

EXAMPLES: 1 , 2

13.1.5 ModPRingGenerators

▷ `ModPRingGenerators(A)` (function)

Inputs a mod p cohomology ring A (created using the preceeding function). It returns a minimal generating set for the ring A. Each generator is homogeneous.

EXAMPLES: 1 , 2

13.1.6 Mod2CohomologyRingPresentation

▷ `Mod2CohomologyRingPresentation(G)` (function)
▷ `Mod2CohomologyRingPresentation(G, n)` (function)

▷ `Mod2CohomologyRingPresentation(A)` (function)
▷ `Mod2CohomologyRingPresentation(R)` (function)

When applied to a finite 2-group G this function returns a presentation for the mod 2 cohomology ring $H^*(G,Z_2)$. The Lyndon-Hochschild-Serre spectral sequence is used to prove that the presentation is correct.

When the function is applied to a 2-group G and positive integer n the function first constructs n terms of a free Z_2G-resolution R, then constructs the finite-dimensional graded algebra $A = H^(* \leq n)(G,Z_2)$, and finally uses A to approximate a presentation for $H^*(G,Z_2)$. For "sufficiently large" the approximation will be a correct presentation for $H^*(G,Z_2)$.

Alternatively, the function can be applied directly to either the resolution R or graded algebra A.

This function was written by PAUL SMITH. It uses the Singular commutative algebra package to handle the Lyndon-Hochschild-Serre spectral sequence.

EXAMPLES: 1 , 2

Chapter 14

Cohomology rings of p-groups (mainly $p = 2$)

The functions on this page were written by PAUL SMITH. (They are included in HAP but they are also independently included in Paul Smiths HAPprime package.)

14.1

14.1.1 Mod2CohomologyRingPresentation

▷ `Mod2CohomologyRingPresentation(G)` (function)
▷ `Mod2CohomologyRingPresentation(G, n)` (function)
▷ `Mod2CohomologyRingPresentation(A)` (function)
▷ `Mod2CohomologyRingPresentation(R)` (function)

When applied to a finite 2-group G this function returns a presentation for the mod 2 cohomology ring $H^*(G,Z_2)$. The Lyndon-Hochschild-Serre spectral sequence is used to prove that the presentation is correct.

When the function is applied to a 2-group G and positive integer n the function first constructs n terms of a free Z_2G-resolution R, then constructs the finite-dimensional graded algebra $A = H^(* \leq n)(G,Z_2)$, and finally uses A to approximate a presentation for $H^*(G,Z_2)$. For "sufficiently large" the approximation will be a correct presentation for $H^*(G,Z_2)$.

Alternatively, the function can be applied directly to either the resolution R or graded algebra A.

This function was written by PAUL SMITH. It uses the Singular commutative algebra package to handle the Lyndon-Hochschild-Serre spectral sequence.

EXAMPLES: 1 , 2

14.1.2 PoincareSeriesLHS

▷ `PoincareSeriesLHS` (global variable)

Inputs a finite 2-group G and returns a quotient of polynomials $f(x) = P(x)/Q(x)$ whose coefficient of x^k equals the rank of the vector space $H_k(G,Z_2)$ for all k.

This function was written by PAUL SMITH. It use the Singular system for commutative algebra.

EXAMPLES:

Chapter 15

Commutator and nonabelian tensor computations

15.1

15.1.1 BaerInvariant

▷ `BaerInvariant(G, c)` (function)

Inputs a nilpotent group G and integer c>0. It returns the Baer invariant $M^(c)(G)$ defined as follows. For an arbitrary group G let $L^*_{c+1}(G)$ be the $(c+1)$-st term of the upper central series of the group $U = F/[[[R,F],F]...]$ (with c copies of F in the denominator) where F/R is any free presentation of G. This is an invariant of G and we define $M^{(c)}(G)$ to be the kernel of the canonical homomorphism $M^{(c)}(G) \longrightarrow G$. For $c = 1$ the Baer invariant $M^(1)(G)$ is isomorphic to the second integral homology $H_2(G,Z)$.

This function requires the NQ package.

EXAMPLES: 1

15.1.2 BogomolovMultiplier

▷ `BogomolovMultiplier(G)` (function)
▷ `BogomolovMultiplier(G, str)` (function)

Inputs a finite group G and an optional string str="standard" or str="homology" or str="tensor". It returns the quotient $H_2(G,Z)/K(G)$ of the second integral homology of G where $K(G)$ is the subgroup of $H_2(G,Z)$ generated by the images of all homomorphisms $H_2(A,Z) \to H_2(G,Z)$ induced from abelian subgroups of G.

Three slight variants of the implementation are available. The default "standard" implementation seems to work best on average. But for some groups the "homology" implementation or the "tensor" implementation will be faster. The variants are called by including the appropriate string as the second argument.

EXAMPLES: 1 , 2

15.1.3 Bogomology

▷ `Bogomology(G, n)` (function)

Inputs a finite group G and positive integer n, and returns the quotient $H_n(G,Z)/K(G)$ of the degree n integral homology of G where $K(G)$ is the subgroup of $H_n(G,Z)$ generated by the images of all homomorphisms $H_n(A,Z) \to H_n(G,Z)$ induced from abelian subgroups of G.

EXAMPLES: 1

15.1.4 Coclass

▷ `Coclass` (global variable)

Inputs a group G of prime-power order p^n and nilpotency class c say. It returns the integer $r = n - c$.

EXAMPLES:

15.1.5 EpiCentre

▷ `EpiCentre(G, N)` (function)
▷ `EpiCentre(G)` (function)

Inputs a finite group G and normal subgroup N and returns a subgroup $Z^*(G,N)$ of the centre of N. The group $Z^*(G,N)$ is trivial if and only if there is a crossed module $d : E \longrightarrow G$ with $N = Image(d)$ and with $Ker(d)$ equal to the subgroup of E consisting of those elements on which G acts trivially.

If no value for N is entered then it is assumed that $N = G$. In this case the group $Z^*(G,G)$ is trivial if and only if G is isomorphic to a quotient $G = E/Z(E)$ of some group E by the centre of E. (See also the command $UpperEpicentralSeries(G,c)$.)

EXAMPLES: 1 , 2 , 3

15.1.6 NonabelianExteriorProduct

▷ `NonabelianExteriorProduct(G, N)` (function)

Inputs a finite group G and normal subgroup N. It returns a record E with the following components.

- $E.homomorphism$ is a group homomorphism $\mu : (G \wedge N) \longrightarrow G$ from the nonabelian exterior product $(G \wedge N)$ to G. The kernel of μ is the relative Schur multiplier.
- $E.pairing(x,y)$ is a function which inputs an element x in G and an element y in N and returns $(x \wedge y)$ in the exterior product $(G \wedge N)$.

This function should work for reasonably small nilpotent groups or extremely small non-nilpotent groups.

EXAMPLES: 1

15.1.7 NonabelianSymmetricKernel

▷ `NonabelianSymmetricKernel(G)` (function)
▷ `NonabelianSymmetricKernel(G, m)` (function)

Inputs a finite or nilpotent infinite group G and returns the abelian invariants of the Fourth homotopy group SG of the double suspension $SSK(G,1)$ of the Eilenberg-Mac Lane space $K(G,1)$.

For non-nilpotent groups the implementation of the function $NonabelianSymmetricKernel(G)$ is far from optimal and will soon be improved. As a temporary solution to this problem, an optional second variable m can be set equal to 0, and then the function efficiently returns the abelian invariants of groups A and B such that there is an exact sequence $0 \longrightarrow B \longrightarrow SG \longrightarrow A \longrightarrow 0$.

Alternatively, the optional second varible m can be set equal to a positive multiple of the order of the symmetric square $(G\tilde{\otimes}G)$. In this case the function returns the abelian invariants of SG. This might help when G is solvable but not nilpotent (especially if the estimated upper bound m is reasonable accurate).

EXAMPLES: 1

15.1.8 NonabelianSymmetricSquare

▷ `NonabelianSymmetricSquare(G)` (function)
▷ `NonabelianSymmetricSquare(G, m)` (function)

Inputs a finite or nilpotent infinite group G and returns a record T with the following components.

- $T.homomorphism$ is a group homomorphism $\mu : (G\tilde{\otimes}G) \longrightarrow G$ from the nonabelian symmetric square of G to G. The kernel of μ is isomorphic to the fourth homotopy group of the double suspension $SSK(G,1)$ of an Eilenberg-Mac Lane space.
- $T.pairing(x,y)$ is a function which inputs two elements x,y in G and returns the tensor $(x \otimes y)$ in the symmetric square $(G \otimes G)$.

An optional second varible m can be set equal to a multiple of the order of the symmetric square $(G\tilde{\otimes}G)$. This might help when G is solvable but not nilpotent (especially if the estimated upper bound m is reasonable accurate) as the bound is used in the solvable quotient algorithm.

The optional second variable m can also be set equal to 0. In this case the Todd-Coxeter procedure will be used to enumerate the symmetric square even when G is solvable.

This function should work for reasonably small solvable groups or extremely small non-solvable groups.

EXAMPLES:

15.1.9 NonabelianTensorProduct

▷ `NonabelianTensorProduct(G, N)` (function)

Inputs a finite group G and normal subgroup N. It returns a record E with the following components.

- $E.homomorphism$ is a group homomorphism $\mu : (G \otimes N) \longrightarrow G$ from the nonabelian exterior product $(G \otimes N)$ to G.

- $E.pairing(x,y)$ is a function which inputs an element x in G and an element y in N and returns $(x \otimes y)$ in the tensor product $(G \otimes N)$.

This function should work for reasonably small nilpotent groups or extremely small non-nilpotent groups.

EXAMPLES: 1

15.1.10 NonabelianTensorSquare

▷ `NonabelianTensorSquare(G)` (function)
▷ `NonabelianTensorSquare(G, m)` (function)

Inputs a finite or nilpotent infinite group G and returns a record T with the following components.

- $T.homomorphism$ is a group homomorphism $\mu : (G \otimes G) \longrightarrow G$ from the nonabelian tensor square of G to G. The kernel of μ is isomorphic to the third homotopy group of the suspension $SK(G,1)$ of an Eilenberg-Mac Lane space.
- $T.pairing(x,y)$ is a function which inputs two elements x,y in G and returns the tensor $(x \otimes y)$ in the tensor square $(G \otimes G)$.

An optional second varible m can be set equal to a multiple of the order of the tensor square $(G \otimes G)$. This might help when G is solvable but not nilpotent (especially if the estimated upper bound m is reasonable accurate) as the bound is used in the solvable quotient algorithm.

The optional second variable m can also be set equal to 0. In this case the Todd-Coxeter procedure will be used to enumerate the tensor square even when G is solvable.

This function should work for reasonably small solvable groups or extremely small non-solvable groups.

EXAMPLES: 1 , 2

15.1.11 RelativeSchurMultiplier

▷ `RelativeSchurMultiplier(G, N)` (function)

Inputs a finite group G and normal subgroup N. It returns the homology group $H_2(G,N,Z)$ that fits into the exact sequence

$\ldots \longrightarrow H_3(G,Z) \longrightarrow H_3(G/N,Z) \longrightarrow H_2(G,N,Z) \longrightarrow H_3(G,Z) \longrightarrow H_3(G/N,Z) \longrightarrow \ldots$.

This function should work for reasonably small nilpotent groups G or extremely small non-nilpotent groups.

EXAMPLES: 1

15.1.12 TensorCentre

▷ `TensorCentre(G)` (function)

Inputs a group G and returns the largest central subgroup N such that the induced homomorphism of nonabelian tensor squares $(G \otimes G) \longrightarrow (G/N \otimes G/N)$ is an isomorphism. Equivalently, N is the largest central subgroup such that $\pi_3(SK(G,1)) \longrightarrow \pi_3(SK(G/N,1))$ is injective.

EXAMPLES:

15.1.13 ThirdHomotopyGroupOfSuspensionB

▷ `ThirdHomotopyGroupOfSuspensionB(G)` (function)
▷ `ThirdHomotopyGroupOfSuspensionB(G, m)` (function)

Inputs a finite or nilpotent infinite group G and returns the abelian invariants of the third homotopy group JG of the suspension $SK(G,1)$ of the Eilenberg-Mac Lane space $K(G,1)$.

For non-nilpotent groups the implementation of the function $ThirdHomotopyGroupOfSuspensionB(G)$ is far from optimal and will soon be improved. As a temporary solution to this problem, an optional second variable m can be set equal to 0, and then the function efficiently returns the abelian invariants of groups A and B such that there is an exact sequence $0 \longrightarrow B \longrightarrow JG \longrightarrow A \longrightarrow 0$.

Alternatively, the optional second varible m can be set equal to a positive multiple of the order of the tensor square $(G \otimes G)$. In this case the function returns the abelian invariants of JG. This might help when G is solvable but not nilpotent (especially if the estimated upper bound m is reasonable accurate).

EXAMPLES: 1 , 2 , 3 , 4

15.1.14 UpperEpicentralSeries

▷ `UpperEpicentralSeries(G, c)` (function)

Inputs a nilpotent group G and an integer c. It returns the c-th term of the upper epicentral series $1 < Z_1^*(G) < Z_2^*(G) < \ldots$.

The upper epicentral series is defined for an arbitrary group G. The group $Z_c^*(G)$ is the image in G of the c-th term $Z_c(U)$ of the upper central series of the group $U = F/[[[R,F],F]\ldots]$ (with c copies of F in the denominator) where F/R is any free presentation of G.

This functions requires the NQ package.

EXAMPLES: 1

Chapter 16

Lie commutators and nonabelian Lie tensors

Functions on this page are joint work with HAMID MOHAMMADZADEH, and implemented by him.

16.1

16.1.1 LieCoveringHomomorphism

▷ `LieCoveringHomomorphism(L)` (function)

Inputs a finite dimensional Lie algebra L over a field, and returns a surjective Lie homomorphism $phi : C \to L$ where:

- the kernel of *phi* lies in both the centre of C and the derived subalgebra of C,
- the kernel of *phi* is a vector space of rank equal to the rank of the second Chevalley-Eilenberg homology of L.

EXAMPLES: 1 , 2

16.1.2 LeibnizQuasiCoveringHomomorphism

▷ `LeibnizQuasiCoveringHomomorphism(L)` (function)

Inputs a finite dimensional Lie algebra L over a field, and returns a surjective homomorphism $phi : C \to L$ of Leibniz algebras where:

- the kernel of *phi* lies in both the centre of C and the derived subalgebra of C,
- the kernel of *phi* is a vector space of rank equal to the rank of the kernel J of the homomorphism $L \otimes L \to L$ from the tensor square to L. (We note that, in general, J is NOT equal to the second Leibniz homology of L.)

EXAMPLES:

16.1.3 LieEpiCentre

▷ `LieEpiCentre(L)` (function)

Inputs a finite dimensional Lie algebra L over a field, and returns an ideal $Z^*(L)$ of the centre of L. The ideal $Z^*(L)$ is trivial if and only if L is isomorphic to a quotient $L = E/Z(E)$ of some Lie algebra E by the centre of E.

EXAMPLES: 1 , 2

16.1.4 LieExteriorSquare

▷ `LieExteriorSquare(L)` (function)

Inputs a finite dimensional Lie algebra L over a field. It returns a record E with the following components.

- $E.homomorphism$ is a Lie homomorphism $\mu : (L \wedge L) \longrightarrow L$ from the nonabelian exterior square $(L \wedge L)$ to L. The kernel of μ is the Lie multiplier.
- $E.pairing(x,y)$ is a function which inputs elements x, y in L and returns $(x \wedge y)$ in the exterior square $(L \wedge L)$.

EXAMPLES:

16.1.5 LieTensorSquare

▷ `LieTensorSquare(L)` (function)

Inputs a finite dimensional Lie algebra L over a field and returns a record T with the following components.

- $T.homomorphism$ is a Lie homomorphism $\mu : (L \otimes L) \longrightarrow L$ from the nonabelian tensor square of L to L.
- $T.pairing(x,y)$ is a function which inputs two elements x, y in L and returns the tensor $(x \otimes y)$ in the tensor square $(L \otimes L)$.

EXAMPLES:

16.1.6 LieTensorCentre

▷ `LieTensorCentre(L)` (function)

Inputs a finite dimensional Lie algebra L over a field and returns the largest ideal N such that the induced homomorphism of nonabelian tensor squares $(L \otimes L) \longrightarrow (L/N \otimes L/N)$ is an isomorphism.

EXAMPLES:

Chapter 17

Generators and relators of groups

17.1

17.1.1 CayleyGraphOfGroupDisplay

▷ `CayleyGraphOfGroupDisplay(G, X)` (function)
▷ `CayleyGraphOfGroupDisplay(G, X, str)` (function)

Inputs a finite group G together with a subset X of G. It displays the corresponding Cayley graph as a .gif file. It uses the Mozilla web browser as a default to view the diagram. An alternative browser can be set using a second argument *str*="mozilla".

The argument G can also be a finite set of elements in a (possibly infinite) group containing X. The edges of the graph are coloured according to which element of X they are labelled by. The list X corresponds to the list of colours [blue, red, green, yellow, brown, black] in that order.

This function requires Graphviz software.

EXAMPLES:

17.1.2 IdentityAmongRelatorsDisplay

▷ `IdentityAmongRelatorsDisplay(R, n)` (function)
▷ `IdentityAmongRelatorsDisplay(R, n, str)` (function)

Inputs a free ZG-resolution R and an integer n. It displays the boundary R!.boundary(3,n) as a tessellation of a sphere. It displays the tessellation as a .gif file and uses the Mozilla web browser as a default display mechanism. An alternative browser can be set using the second argument *str*="mozilla". (The resolution R should be reduced and, preferably, in dimension 1 it should correspond to a Cayley graph for G.)

This function uses GraphViz software.

EXAMPLES: 1 , 2 , 3

17.1.3 IsAspherical

▷ `IsAspherical(F, R)` (function)

Inputs a free group F and a set R of words in F. It performs a test on the 2-dimensional CW-space K associated to this presentation for the group $G = F/\langle R \rangle^F$.

The function returns "true" if K has trivial second homotopy group. In this case it prints: Presentation is aspherical.

Otherwise it returns "fail" and prints: Presentation is NOT piece-wise Euclidean non-positively curved. (In this case K may or may not have trivial second homotopy group. But it is NOT possible to impose a metric on K which restricts to a Euclidean metric on each 2-cell.)

The function uses Polymake software.

EXAMPLES: 1 , 2 , 3 , 4

17.1.4 PresentationOfResolution

▷ `PresentationOfResolution(R)` (function)

Inputs at least two terms of a reduced ZG-resolution R and returns a record P with components

- $P.freeGroup$ is a free group F,
- $P.relators$ is a list S of words in F,
- $P.gens$ is a list of positive integers such that the i-th generator of the presentation corresponds to the group element R!.elts[P[i]] .

where G is isomorphic to F modulo the normal closure of S. This presentation for G corresponds to the 2-skeleton of the classifying CW-space from which R was constructed. The resolution R requires no contracting homotopy.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6

17.1.5 TorsionGeneratorsAbelianGroup

▷ `TorsionGeneratorsAbelianGroup(G)` (function)

Inputs an abelian group G and returns a generating set $[x_1, \ldots, x_n]$ where no pair of generators have coprime orders.

EXAMPLES:

Chapter 18

Orbit polytopes and fundamental domains

18.1

18.1.1 CoxeterComplex

▷ `CoxeterComplex(`*D*`)` (function)
▷ `CoxeterComplex(`*D*`, `*n*`)` (function)

Inputs a Coxeter diagram *D* of finite type. It returns a non-free ZW-resolution for the associated Coxeter group *W*. The non-free resolution is obtained from the permutahedron of type *W*. A positive integer *n* can be entered as an optional second variable; just the first *n* terms of the non-free resolution are then returned.

EXAMPLES: 1

18.1.2 ContractibleGcomplex

▷ `ContractibleGcomplex(`*str*`)` (function)

Inputs one of the following strings *str*=:

"SL(2,Z)" , "SL(3,Z)" , "PGL(3,Z[i])" , "PGL(3,Eisenstein_Integers)" , "PSL(4,Z)" , "PSL(4,Z)_b" , "PSL(4,Z)_c" , "PSL(4,Z)_d" , "Sp(4,Z)"

or one of the following strings

"SL(2,O-2)" , "SL(2,O-7)" , "SL(2,O-11)" , "SL(2,O-19)" , "SL(2,O-43)" , "SL(2,O-67)" , "SL(2,O-163)"

It returns a non-free ZG-resolution for the group *G* described by the string. The stabilizer groups of cells are finite. (Subscripts _b , _c , _d denote alternative non-free ZG-resolutions for a given group G.)

Data for the first list of non-free resolutions was provided provided by MATHIEU DUTOUR. Data for the second list was provided by ALEXANDER RAHM.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10

18.1.3 QuotientOfContractibleGcomplex

▷ `QuotientOfContractibleGcomplex(C, D)` (function)

Inputs a non-free ZG-resolution C and a finite subgroup D of G which is a subgroup of each cell stabilizer group for C. Each element of D must preserves the orientation of any cell stabilized by it. It returns the corresponding non-free $Z(G/D)$-resolution. (So, for instance, from the $SL(2,O)$ complex $C = ContractibleGcomplex("SL(2,O-2)")$; we can construct a $PSL(2,O)$-complex using this function.)

EXAMPLES: 1 , 2

18.1.4 TruncatedGComplex

▷ `TruncatedGComplex(R, m, n)` (function)

Inputs a non-free ZG-resolution R and two positive integers m and n. It returns the non-free ZG-resolution consisting of those modules in R of degree at least m and at most n.

EXAMPLES:

18.1.5 FundamentalDomainStandardSpaceGroup

▷ `FundamentalDomainStandardSpaceGroup(v, G)` (function)

Inputs a crystallographic group G (represented using AffineCrystGroupOnRight as in the GAP package Cryst). It also inputs a choice of vector v in the euclidean space R^n on which G acts. It returns the Dirichlet-Voronoi fundamental cell for the action of G on euclidean space corresponding to the vector v. The fundamental cell is a fundamental domain if G is Bieberbach. The fundamental cell/domain is returned as a "Polymake object". Currently the function only applies to certain crystallographic groups. See the manuals to HAPcryst and HAPpolymake for full details.

This function is part of the HAPcryst package written by MARC ROEDER and is thus only available if HAPcryst is loaded.

The function requires the use of Polymake software.

EXAMPLES: 1 , 2

18.1.6 OrbitPolytope

▷ `OrbitPolytope(G, v, L)` (function)

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 function also inputs a sublist L of the following list of strings:

["dimension","vertex_degree", "visual_graph", "schlegel","visual"]

Depending on the sublist, the function:

- prints the dimension of the orbit polytope $P(G,v)$;
- prints the degree of a vertex in the graph of $P(G,v)$;
- visualizes the graph of $P(G,v)$;
- visualizes the Schlegel diagram of $P(G,v)$;
- visualizes $P(G,v)$ if the polytope is of dimension 2 or 3.

The function uses Polymake software.
EXAMPLES: 1 , 2

18.1.7 PolytopalComplex

▷ `PolytopalComplex(G, v)` (function)
▷ `PolytopalComplex(G, v, n)` (function)

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.
EXAMPLES: 1 , 2

18.1.8 PolytopalGenerators

▷ `PolytopalGenerators(G, v)` (function)

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.

EXAMPLES:

18.1.9 VectorStabilizer

▷ `VectorStabilizer(G, v)` (function)

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.

EXAMPLES:

Chapter 19

Cocycles

19.1

19.1.1 CcGroup

▷ `CcGroup(A, f)` (function)

Inputs a G-module A (i.e. an abelian G-outer group) and a standard 2-cocycle f $GxG - - - > A$. It returns the extension group determined by the cocycle. The group is returned as a CcGroup.

This is a HAPcocyclic function and thus only works when HAPcocyclic is loaded.

EXAMPLES: 1 , 2

19.1.2 CocycleCondition

▷ `CocycleCondition(R, n)` (function)

Inputs a resolution R and an integer n>0. It returns an integer matrix M with the following property. Suppose $d = R.dimension(n)$. An integer vector $f = [f_1, \ldots, f_d]$ then represents a ZG-homomorphism $R_n \longrightarrow Z_q$ which sends the ith generator of R_n to the integer f_i in the trivial ZG-module Z_q (where possibly $q = 0$). The homomorphism f is a cocycle if and only if $M^t f = 0 \bmod q$.

EXAMPLES: 1 , 2

19.1.3 StandardCocycle

▷ `StandardCocycle(R, f, n)` (function)

▷ `StandardCocycle(R, f, n, q)` (function)

Inputs a ZG-resolution R (with contracting homotopy), a positive integer n and an integer vector f representing an n-cocycle $R_n \longrightarrow Z_q$ where G acts trivially on Z_q. It is assumed $q = 0$ unless a value for q is entered. The command returns a function $F(g_1, ..., g_n)$ which is the standard cocycle $G_n \longrightarrow Z_q$ corresponding to f. At present the command is implemented only for $n = 2$ or 3.

EXAMPLES: 1 , 2

19.1.4 Syzygy

▷ `Syzygy(R, g)` (function)

Inputs a ZG-resolution R (with contracting homotopy) and a list $g = [g[1], ..., g[n]]$ of elements in G. It returns a word w in R_n. The word w is the image of the n-simplex in the standard bar resolution corresponding to the n-tuple g. This function can be used to construct explicit standard n-cocycles. (Currently implemented only for n<4.)

EXAMPLES:

Chapter 20

Words in free ZG-modules

20.1

20.1.1 AddFreeWords

▷ `AddFreeWords(v, w)` (function)

Inputs two words v, w in a free ZG-module and returns their sum $v+w$. If the characteristic of Z is greater than 0 then the next function might be more efficient.

EXAMPLES:

20.1.2 AddFreeWordsModP

▷ `AddFreeWordsModP(v, w, p)` (function)

Inputs two words v, w in a free ZG-module and the characteristic p of Z. It returns the sum $v+w$. If $p = 0$ the previous function might be fractionally quicker.

EXAMPLES:

20.1.3 AlgebraicReduction

▷ `AlgebraicReduction(w)` (function)
▷ `AlgebraicReduction(w, p)` (function)

Inputs a word w in a free ZG-module and returns a reduced version of the word in which all pairs of mutually inverse letters have been cancelled. The reduction is performed in a free abelian group unless the characteristic p of Z is entered.

EXAMPLES:

20.1.4 MultiplyWord

▷ `MultiplyWord(n, w)` (function)

Inputs a word w and integer n. It returns the scalar multiple $n \cdot w$.

EXAMPLES:

20.1.5 Negate

▷ `Negate([i, j])` (function)

Inputs a pair $[i, j]$ of integers and returns $[-i, j]$.
EXAMPLES:

20.1.6 NegateWord

▷ `NegateWord(w)` (function)

Inputs a word w in a free ZG-module and returns the negated word $-w$.
EXAMPLES:

20.1.7 PrintZGword

▷ `PrintZGword(w, elts)` (function)

Inputs a word w in a free ZG-module and a (possibly partial but sufficient) listing elts of the elements of G. The function prints the word w to the screen in the form
$r_1E_1 + \ldots + r_nE_n$
where r_i are elements in the group ring ZG, and E_i denotes the i-th free generator of the module.
EXAMPLES: 1

20.1.8 TietzeReduction

▷ `TietzeReduction(S, w)` (function)

Inputs a set S of words in a free ZG-module, and a word w in the module. The function returns a word w' such that $\{S, w'\}$ generates the same abelian group as $\{S, w\}$. The word w' is possibly shorter (and certainly no longer) than w. This function needs to be improved!
EXAMPLES:

20.1.9 ResolutionBoundaryOfWord

▷ `ResolutionBoundaryOfWord(R, n, w)` (function)

Inputs a resolution R, a positive integer n and a list w representing a word in the free module R_n. It returns the image of w under the n-th boundary homomorphism.
EXAMPLES: 1

Chapter 21

FpG-modules

21.1

21.1.1 CompositionSeriesOfFpGModules

▷ `CompositionSeriesOfFpGModules` (global variable)

Inputs an FpG-module M and returns a list of FpG-modules that constitute a composition series for M.

EXAMPLES:

21.1.2 DirectSumOfFpGModules

▷ `DirectSumOfFpGModules(M, N)` (function)
▷ `DirectSumOfFpGModules([M[, 1], M[, 2], ..., M[, k]])` (function)

Inputs two FpG-modules M and N with common group and characteristic. It returns the direct sum of M and N as an FpG-Module.

Alternatively, the function can input a list of FpG-modules with common group G. It returns the direct sum of the list.

EXAMPLES:

21.1.3 FpGModule

▷ `FpGModule(A, P)` (function)
▷ `FpGModule(A, G, p)` (function)

Inputs a p-group P and a matrix A whose rows have length a multiple of the order of G. It returns the "canonical" FpG-module generated by the rows of A.

A small non-prime-power group G can also be input, provided the characteristic p is entered as a third input variable.

EXAMPLES: 1 , 2 , 3 , 4

21.1.4 FpGModuleDualBasis

▷ `FpGModuleDualBasis(M)` (function)

Inputs an FpG-module M. It returns a record R with two components:

- $R.freeModule$ is the free module FG of rank one.
- $R.basis$ is a list representing an F-basis for the module $Hom_{FG}(M,FG)$. Each term in the list is a matrix A whose rows are vectors in FG such that $M!.generators[i] \longrightarrow A[i]$ extends to a module homomorphism $M \longrightarrow FG$.

EXAMPLES:

21.1.5 FpGModuleHomomorphism

▷ `FpGModuleHomomorphism(M, N, A)` (function)
▷ `FpGModuleHomomorphismNC(M, N, A)` (function)

Inputs FpG-modules M and N over a common p-group G. Also inputs a list A of vectors in the vector space spanned by $N!.matrix$. It tests that the function
$M!.generators[i] \longrightarrow A[i]$
extends to a homomorphism of FpG-modules and, if the test is passed, returns the corresponding FpG-module homomorphism. If the test is failed it returns fail.

The "NC" version of the function assumes that the input defines a homomorphism and simply returns the FpG-module homomorphism.

EXAMPLES:

21.1.6 DesuspensionFpGModule

▷ `DesuspensionFpGModule(M, n)` (function)
▷ `DesuspensionFpGModule(R, n)` (function)

Inputs a positive integer n and and FpG-module M. It returns an FpG-module D^nM which is mathematically related to M via an exact sequence $0 \longrightarrow D^nM \longrightarrow R_n \longrightarrow \ldots \longrightarrow R_0 \longrightarrow M \longrightarrow 0$ where R_* is a free resolution. (If $G = Group(M)$ is of prime-power order then the resolution is minimal.)

Alternatively, the function can input a positive integer n and at least n terms of a free resolution R of M.

EXAMPLES:

21.1.7 RadicalOfFpGModule

▷ `RadicalOfFpGModule(M)` (function)

Inputs an FpG-module M with G a p-group, and returns the Radical of M as an FpG-module. (Ig G is not a p-group then a submodule of the radical is returned.

EXAMPLES:

21.1.8 RadicalSeriesOfFpGModule

▷ `RadicalSeriesOfFpGModule(M)` (function)

Inputs an FpG-module M and returns a list of FpG-modules that constitute the radical series for M.

EXAMPLES:

21.1.9 GeneratorsOfFpGModule

▷ `GeneratorsOfFpGModule(M)` (function)

Inputs an FpG-module M and returns a matrix whose rows correspond to a minimal generating set for M.

EXAMPLES:

21.1.10 ImageOfFpGModuleHomomorphism

▷ `ImageOfFpGModuleHomomorphism(f)` (function)

Inputs an FpG-module homomorphism $f : M \longrightarrow N$ and returns its image $f(M)$ as an FpG-module.

EXAMPLES:

21.1.11 GroupAlgebraAsFpGModule

▷ `GroupAlgebraAsFpGModule(G)` (function)

Inputs a p-group G and returns its mod p group algebra as an FpG-module.

EXAMPLES:

21.1.12 IntersectionOfFpGModules

▷ `IntersectionOfFpGModules(M, N)` (function)

Inputs two FpG-modules M,N arising as submodules in a common free module $(FG)^n$ where G is a finite group and F the field of p-elements. It returns the FpG-module arising as the intersection of M and N.

EXAMPLES:

21.1.13 IsFpGModuleHomomorphismData

▷ `IsFpGModuleHomomorphismData(M, N, A)` (function)

Inputs FpG-modules M and N over a common p-group G. Also inputs a list A of vectors in the vector space spanned by $N!.matrix$. It returns true if the function

$M!.generators[i] \longrightarrow A[i]$

extends to a homomorphism of FpG-modules. Otherwise it returns false.

EXAMPLES:

21.1.14 MaximalSubmoduleOfFpGModule

▷ `MaximalSubmoduleOfFpGModule(M)` (function)

Inputs an FpG-module M and returns one maximal FpG-submodule of M.
EXAMPLES:

21.1.15 MaximalSubmodulesOfFpGModule

▷ `MaximalSubmodulesOfFpGModule(M)` (function)

Inputs an FpG-module M and returns the list of maximal FpG-submodules of M.
EXAMPLES:

21.1.16 MultipleOfFpGModule

▷ `MultipleOfFpGModule(w, M)` (function)

Inputs an FpG-module M and a list $w := [g_1, ..., g_t]$ of elements in the group $G = M!.group$. The list w can be thought of as representing the element $w = g_1 + \ldots + g_t$ in the group algebra FG, and the function returns a semi-echelon matrix B which is a basis for the vector subspace wM .
EXAMPLES:

21.1.17 ProjectedFpGModule

▷ `ProjectedFpGModule(M, k)` (function)

Inputs an FpG-module M of ambient dimension $n|G|$, and an integer k between 1 and n. The module M is a submodule of the free module $(FG)^n$. Let M_k denote the intersection of M with the last k summands of $(FG)^n$. The function returns the image of the projection of M_k onto the k-th summand of $(FG)^n$. This image is returned an FpG-module with ambient dimension $|G|$.
EXAMPLES:

21.1.18 RandomHomomorphismOfFpGModules

▷ `RandomHomomorphismOfFpGModules(M, N)` (function)

Inputs two FpG-modules M and N over a common group G. It returns a random matrix A whose rows are vectors in N such that the function

$M!.generators[i] \longrightarrow A[i]$

extends to a homomorphism $M \longrightarrow N$ of FpG-modules. (There is a problem with this function at present.)
EXAMPLES:

21.1.19 Rank

▷ `Rank(f)` (function)

Inputs an FpG-module homomorphism $f : M \longrightarrow N$ and returns the dimension of the image of f as a vector space over the field F of p elements.

EXAMPLES: 1 , 2 , 3

21.1.20 SumOfFpGModules

▷ `SumOfFpGModules(M, N)` (function)

Inputs two FpG-modules M,N arising as submodules in a common free module $(FG)^n$ where G is a finite group and F the field of p-elements. It returns the FpG-Module arising as the sum of M and N.

EXAMPLES:

21.1.21 SumOp

▷ `SumOp(f, g)` (function)

Inputs two FpG-module homomorphisms $f,g : M \longrightarrow N$ with common sorce and common target. It returns the sum $f+g : M \longrightarrow N$. (This operation is also available using "+".

EXAMPLES:

21.1.22 VectorsToFpGModuleWords

▷ `VectorsToFpGModuleWords(M, L)` (function)

Inputs an FpG-module M and a list $L = [v_1, \ldots, v_k]$ of vectors in M. It returns a list $L' = [x_1, ..., x_k]$. Each $x_j = [[W_1, G_1], ..., [W_t, G_t]]$ is a list of integer pairs corresponding to an expression of v_j as a word

$v_j = g_1 * w_1 + g_2 * w_1 + ... + g_t * w_t$

where

$g_i = Elements(M!.group)[G_i]$

$w_i = GeneratorsOfFpGModule(M)[W_i]$.

EXAMPLES:

Chapter 22

Meataxe modules

22.1

22.1.1 DesuspensionMtxModule

▷ `DesuspensionMtxModule(`*M*`)` (function)

Inputs a meataxe module *M* over the field of *p* elements and returns an FpG-module DM. The two modules are related mathematically by the existence of a short exact sequence $DM \longrightarrow FM \longrightarrow M$ with FM a free module. Thus the homological properties of DM are equal to those of M with a dimension shift.

(If $G := Group(M.generators)$ is a p-group then FM is a projective cover of M in the sense that the homomorphism $FM \longrightarrow M$ does not factor as $FM \longrightarrow P \longrightarrow M$ for any projective module P.)

EXAMPLES: 1 , 2 , 3

22.1.2 FpG_to_MtxModule

▷ `FpG_to_MtxModule(`*M*`)` (function)

Inputs an FpG-module *M* and returns an isomorphic meataxe module.

EXAMPLES:

22.1.3 GeneratorsOfMtxModule

▷ `GeneratorsOfMtxModule(`*M*`)` (function)

Inputs a meataxe module *M* acting on, say, the vector space *V*. The function returns a minimal list of row vectors in *V* which generate *V* as a *G*-module (where G=Group(M.generators)).

EXAMPLES:

Chapter 23

G-Outer Groups

23.1

23.1.1 GOuterGroup

▷ `GOuterGroup(E, N)` (function)
▷ `GOuterGroup()` (function)

Inputs a group E and normal subgroup N. It returns N as a G-outer group where $G = E/N$.

The function can be used without an argument. In this case an empty outer group C is returned. The components must be set using SetActingGroup(C,G), SetActedGroup(C,N) and SetOuterAction(C,alpha).

EXAMPLES: 1 , 2 , 3 , 4

23.1.2 GOuterGroupHomomorphismNC

▷ `GOuterGroupHomomorphismNC` (global variable)
▷ `GOuterGroupHomomorphismNC` (global variable)

Inputs G-outer groups A and B with common acting group, and a group homomorphism phi:ActedGroup(A) --> ActedGroup(B). It returns the corresponding G-outer homomorphism PHI:A--> B. No check is made to verify that phi is actually a group homomorphism which preserves the G-action.

The function can be used without an argument. In this case an empty outer group homomorphism *PHI* is returned. The components must then be set.

EXAMPLES:

23.1.3 GOuterHomomorphismTester

▷ `GOuterHomomorphismTester(A, B, phi)` (function)

Inputs G-outer groups A and B with common acting group, and a group homomorphism phi:ActedGroup(A) --> ActedGroup(B). It tests whether phi is a group homomorphism which preserves the G-action.

The function can be used without an argument. In this case an empty outer group homomorphism *PHI* is returned. The components must then be set.

EXAMPLES:

23.1.4 Centre

▷ `Centre(A)` (function)

Inputs G-outer group *A* and returns the group theoretic centre of ActedGroup(A) as a G-outer group.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6

23.1.5 DirectProductGog

▷ `DirectProductGog(A, B)` (function)
▷ `DirectProductGog(Lst)` (function)

Inputs G-outer groups *A* and *B* with common acting group, and returns their group-theoretic direct product as a G-outer group. The outer action on the direct product is the diagonal one.

The function also applies to a list Lst of G-outer groups with common acting group.

For a direct product D constructed using this function, the embeddings and projections can be obtained (as G-outer group homomorphisms) using the functions Embedding(D,i) and Projection(D,i).

EXAMPLES:

Chapter 24

Cat-1-groups

24.1

24.1.1 AutomorphismGroupAsCatOneGroup

▷ `AutomorphismGroupAsCatOneGroup(G)` (function)

Inputs a group G and returns the Cat-1-group C corresponding to the crossed module $G \to Aut(G)$.
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6

24.1.2 HomotopyGroup

▷ `HomotopyGroup(C, n)` (function)

Inputs a cat-1-group C and an integer n. It returns the nth homotopy group of C.
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8

24.1.3 HomotopyModule

▷ `HomotopyModule(C, 2)` (function)

Inputs a cat-1-group C and an integer n=2. It returns the second homotopy group of C as a G-module (i.e. abelian G-outer group) where G is the fundamental group of C.
EXAMPLES: 1 , 2

24.1.4 QuasiIsomorph

▷ `QuasiIsomorph(C)` (function)

Inputs a cat-1-group C and returns a cat-1-group D for which there exists some homomorphism $C \to D$ that induces isomorphisms on homotopy groups.
This function was implemented by LE VAN LUYEN.
EXAMPLES: 1 , 2 , 3

24.1.5 ModuleAsCatOneGroup

▷ `ModuleAsCatOneGroup` (global variable)

Inputs a group G, an abelian group M and a homomorphism $\alpha: G \to Aut(M)$. It returns the Cat-1-group C corresponding th the zero crossed module $0: M \to G$.

EXAMPLES:

24.1.6 MooreComplex

▷ `MooreComplex(C)` (function)

Inputs a cat-1-group C and returns its Moore complex as a G-complex (i.e. as a complex of groups considered as 1-outer groups).

EXAMPLES:

24.1.7 NormalSubgroupAsCatOneGroup

▷ `NormalSubgroupAsCatOneGroup(G, N)` (function)

Inputs a group G with normal subgroup N. It returns the Cat-1-group C corresponding th the inclusion crossed module $N \to G$.

EXAMPLES:

24.1.8 XmodToHAP

▷ `XmodToHAP(C)` (function)

Inputs a cat-1-group C obtained from the Xmod package and returns a cat-1-group D for which IsHapCatOneGroup(D) returns true.

It returns "fail" id C has not been produced by the Xmod package.

EXAMPLES: 1

Chapter 25

Simplicial groups

25.1

25.1.1 NerveOfCatOneGroup

▷ `NerveOfCatOneGroup(G, n)` (function)

Inputs a cat-1-group G and a positive integer n. It returns the low-dimensional part of the nerve of G as a simplicial group of length n.

This function applies both to cat-1-groups for which IsHapCatOneGroup(G) is true, and to cat-1-groups produced using the Xmod package.

This function was implemented by VAN LUYEN LE.
EXAMPLES: 1 , 2 , 3

25.1.2 EilenbergMacLaneSimplicialGroup

▷ `EilenbergMacLaneSimplicialGroup(G, n, dim)` (function)

Inputs a group G, a positive integer n, and a positive integer dim. The function returns the first $1+dim$ terms of a simplicial group with $n-1$st homotopy group equal to G and all other homotopy groups equal to zero.

This function was implemented by VAN LUYEN LE.
EXAMPLES: 1 , 2 , 3 , 4 , 5

25.1.3 EilenbergMacLaneSimplicialGroupMap

▷ `EilenbergMacLaneSimplicialGroupMap` (global variable)

Inputs a group homomorphism $f : G \to Q$, a positive integer n, and a positive integer dim. The function returns the first $1+dim$ terms of a simplicial group homomorphism $f : K(G,n) \to K(Q,n)$ of Eilenberg-MacLane simplicial groups.

This function was implemented by VAN LUYEN LE.
EXAMPLES:

25.1.4 MooreComplex

▷ `MooreComplex(G)` (function)

Inputs a simplicial group G and returns its Moore complex as a G-complex.

This function was implemented by VAN LUYEN LE.
EXAMPLES:

25.1.5 ChainComplexOfSimplicialGroup

▷ `ChainComplexOfSimplicialGroup(G)` (function)

Inputs a simplicial group G and returns the cellular chain complex C of a CW-space X represented by the homotopy type of the simplicial group. Thus the homology groups of C are the integral homology groups of X.

This function was implemented by VAN LUYEN LE.
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6

25.1.6 SimplicialGroupMap

▷ `SimplicialGroupMap` (global variable)

Inputs a homomorphism $f : G \to Q$ of simplicial groups. The function returns an induced map $f : C(G) \to C(Q)$ of chain complexes whose homology is the integral homology of the simplicial group G and Q respectively.

This function was implemented by VAN LUYEN LE.
EXAMPLES:

25.1.7 HomotopyGroup

▷ `HomotopyGroup(G, n)` (function)

Inputs a simplicial group G and a positive integer n. The integer n must be less than the length of G. It returns, as a group, the (n)-th homology group of its Moore complex. Thus HomotopyGroup(G,0) returns the "fundamental group" of G.
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8

25.1.8 Representation of elements in the bar resolution

▷ `Representation of elements in the bar resolution` (global variable)

For a group G we denote by $B_n(G)$ the free $\mathbb{Z}G$-module with basis the lists $[g_1|g_2|...|g_n]$ where the g_i range over G.

We represent a word

$w = h_1.[g_{11}|g_{12}|...|g_{1n}] - h_2.[g_{21}|g_{22}|...|g_{2n}] + ... + h_k.[g_{k1}|g_{k2}|...|g_{kn}]$

in $B_n(G)$ as a list of lists:

$[[+1, h_1, g_{11}, g_{12}, ..., g_{1n}], [-1, h_2, g_{21}, g_{22}, ...|g_{2n}] + ... + [+1, h_k, g_{k1}, g_{k2}, ..., g_{kn}]$.

EXAMPLES:

25.1.9 BarResolutionBoundary

▷ `BarResolutionBoundary` (global variable)

This function inputs a word w in the bar resolution module $B_n(G)$ and returns its image under the boundary homomorphism $d_n: B_n(G) \to B_{n-1}(G)$ in the bar resolution.

This function was implemented by VAN LUYEN LE.

EXAMPLES:

25.1.10 BarResolutionHomotopy

▷ `BarResolutionHomotopy` (global variable)

This function inputs a word w in the bar resolution module $B_n(G)$ and returns its image under the contracting homotopy $h_n: B_n(G) \to B_{n+1}(G)$ in the bar resolution.

This function is currently being implemented by VAN LUYEN LE.

EXAMPLES:

25.1.11 Representation of elements in the bar complex

▷ `Representation of elements in the bar complex` (global variable)

For a group G we denote by $BC_n(G)$ the free abelian group with basis the lists $[g_1|g_2|...|g_n]$ where the g_i range over G.

We represent a word

$w = [g_{11}|g_{12}|...|g_{1n}] - [g_{21}|g_{22}|...|g_{2n}] + ... + [g_{k1}|g_{k2}|...|g_{kn}]$

in $BC_n(G)$ as a list of lists:

$[[+1, g_{11}, g_{12}, ..., g_{1n}], [-1, g_{21}, g_{22}, ...|g_{2n}] + ... + [+1, g_{k1}, g_{k2}, ..., g_{kn}]$.

EXAMPLES:

25.1.12 BarComplexBoundary

▷ `BarComplexBoundary` (global variable)

This function inputs a word w in the n-th term of the bar complex $BC_n(G)$ and returns its image under the boundary homomorphism $d_n: BC_n(G) \to BC_{n-1}(G)$ in the bar complex.

This function was implemented by VAN LUYEN LE.

EXAMPLES:

25.1.13 BarResolutionEquivalence

▷ `BarResolutionEquivalence(R)` (function)

This function inputs a free ZG-resolution R. It returns a component object HE with components

- HE!.phi(n,w) is a function which inputs a non-negative integer n and a word w in $B_n(G)$. It returns the image of w in R_n under a chain equivalence $\phi: B_n(G) \to R_n$.
- HE!.psi(n,w) is a function which inputs a non-negative integer n and a word w in R_n. It returns the image of w in $B_n(G)$ under a chain equivalence $\psi: R_n \to B_n(G)$.
- HE!.equiv(n,w) is a function which inputs a non-negative integer n and a word w in $B_n(G)$. It returns the image of w in $B_{n+1}(G)$ under a ZG-equivariant homomorphism

  $equiv(n,-): B_n(G) \to B_{n+1}(G)$

  satisfying

  $$w - \psi(\phi(w)) = d(n+1, equiv(n,w)) + equiv(n-1, d(n,w)).$$

  where $d(n,-): B_n(G) \to B_{n-1}(G)$ is the boundary homomorphism in the bar resolution.

This function was implemented by VAN LUYEN LE.

EXAMPLES:

25.1.14 BarComplexEquivalence

▷ `BarComplexEquivalence(R)` (function)

This function inputs a free ZG-resolution R. It first constructs the chain complex $T = TensorWithIntegerts(R)$. The function returns a component object HE with components

- HE!.phi(n,w) is a function which inputs a non-negative integer n and a word w in $BC_n(G)$. It returns the image of w in T_n under a chain equivalence $\phi: BC_n(G) \to T_n$.
- HE!.psi(n,w) is a function which inputs a non-negative integer n and an element w in T_n. It returns the image of w in $BC_n(G)$ under a chain equivalence $\psi: T_n \to BC_n(G)$.
- HE!.equiv(n,w) is a function which inputs a non-negative integer n and a word w in $BC_n(G)$. It returns the image of w in $BC_{n+1}(G)$ under a homomorphism

$equiv(n,-): BC_n(G) \to BC_{n+1}(G)$

satisfying

$$w - \psi(\phi(w)) = d(n+1, equiv(n,w)) + equiv(n-1, d(n,w)).$$

where $d(n,-): BC_n(G) \to BC_{n-1}(G)$ is the boundary homomorphism in the bar complex.

This function was implemented by VAN LUYEN LE.

EXAMPLES:

25.1.15 Representation of elements in the bar cocomplex

▷ `Representation of elements in the bar cocomplex` (global variable)

For a group G we denote by $BC^n(G)$ the free abelian group with basis the lists $[g_1|g_2|...|g_n]$ where the g_i range over G.

We represent a word

$w = [g_{11}|g_{12}|...|g_{1n}] - [g_{21}|g_{22}|...|g_{2n}] + ... + [g_{k1}|g_{k2}|...|g_{kn}]$

in $BC^n(G)$ as a list of lists:

$[[+1, g_{11}, g_{12}, ..., g_{1n}], [-1, g_{21}, g_{22}, ...|g_{2n}] + ... + [+1, g_{k1}, g_{k2}, ..., g_{kn}].$

EXAMPLES:

25.1.16 BarCocomplexCoboundary

▷ `BarCocomplexCoboundary` (global variable)

This function inputs a word w in the n-th term of the bar cocomplex $BC^n(G)$ and returns its image under the coboundary homomorphism $d^n: BC^n(G) \to BC^{n+1}(G)$ in the bar cocomplex.

This function was implemented by VAN LUYEN LE.

EXAMPLES:

Chapter 26

Coxeter diagrams and graphs of groups

26.1

26.1.1 CoxeterDiagramComponents

▷ `CoxeterDiagramComponents(D)` (function)

Inputs a Coxeter diagram D and returns a list $[D_1, ..., D_d]$ of the maximal connected subgraphs D_i.
EXAMPLES:

26.1.2 CoxeterDiagramDegree

▷ `CoxeterDiagramDegree(D, v)` (function)

Inputs a Coxeter diagram D and vertex v. It returns the degree of v (i.e. the number of edges incident with v).
EXAMPLES:

26.1.3 CoxeterDiagramDisplay

▷ `CoxeterDiagramDisplay(D)` (function)
▷ `CoxeterDiagramDisplay(D, str)` (function)

Inputs a Coxeter diagram D and displays it as a .gif file. It uses the Mozilla web browser as a default to view the diagram. An alternative browser can be set using a second argument *str*="mozilla".
This function requires Graphviz software.
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7

26.1.4 CoxeterDiagramFpArtinGroup

▷ `CoxeterDiagramFpArtinGroup(D)` (function)

Inputs a Coxeter diagram D and returns the corresponding finitely presented Artin group.
EXAMPLES: 1

26.1.5 CoxeterDiagramFpCoxeterGroup

▷ `CoxeterDiagramFpCoxeterGroup(D)` (function)

Inputs a Coxeter diagram D and returns the corresponding finitely presented Coxeter group.
EXAMPLES: 1

26.1.6 CoxeterDiagramIsSpherical

▷ `CoxeterDiagramIsSpherical(D)` (function)

Inputs a Coxeter diagram D and returns "true" if the associated Coxeter groups is finite, and returns "false" otherwise.
EXAMPLES: 1

26.1.7 CoxeterDiagramMatrix

▷ `CoxeterDiagramMatrix(D)` (function)

Inputs a Coxeter diagram D and returns a matrix representation of it. The matrix is given as a function $DiagramMatrix(D)(i,j)$ where i,j can range over the vertices.
EXAMPLES:

26.1.8 CoxeterSubDiagram

▷ `CoxeterSubDiagram(D, V)` (function)

Inputs a Coxeter diagram D and a subset V of its vertices. It returns the full sub-diagram of D with vertex set V.
EXAMPLES:

26.1.9 CoxeterDiagramVertices

▷ `CoxeterDiagramVertices(D)` (function)

Inputs a Coxeter diagram D and returns its set of vertices.
EXAMPLES:

26.1.10 EvenSubgroup

▷ `EvenSubgroup(G)` (function)

Inputs a group G and returns a subgroup G^+. The subgroup is that generated by all products xy where x and y range over the generating set for G stored by GAP. The subgroup is probably only meaningful when G is an Artin or Coxeter group.
EXAMPLES: 1 , 2

26.1.11 GraphOfGroupsDisplay

▷ `GraphOfGroupsDisplay(D)` (function)
▷ `GraphOfGroupsDisplay(D, str)` (function)

Inputs a graph of groups *D* and displays it as a .gif file. It uses the Mozilla web browser as a default to view the diagram. An alternative browser can be set using the second argument *str*="mozilla".

This function requires Graphviz software.

EXAMPLES: 1 , 2 , 3 , 4

26.1.12 GraphOfResolutions

▷ `GraphOfResolutions(D, n)` (function)

Inputs a graph of groups *D* and a positive integer *n*. It returns a graph of resolutions, each resolution being of length *n*. It uses the function ResolutionGenericGroup() to produce the resolutions.

EXAMPLES:

26.1.13 GraphOfGroups

▷ `GraphOfGroups(D)` (function)

Inputs a graph of resolutions *D* and returns the corresponding graph of groups.

EXAMPLES: 1 , 2 , 3 , 4

26.1.14 GraphOfResolutionsDisplay

▷ `GraphOfResolutionsDisplay(D)` (function)

Inputs a graph of resolutions *D* and displays it as a .gif file. It uses the Mozilla web browser as a default to view the diagram.

This function requires Graphviz software.

EXAMPLES:

26.1.15 GraphOfGroupsTest

▷ `GraphOfGroupsTest(D)` (function)

Inputs an object *D* and itries to test whether it is a Graph of Groups. However, it DOES NOT test the injectivity of any homomorphisms. It returns true if *D* passes the test, and false otherwise.

Note that there is no function *IsHapGraphOfGroups*() because no special data type has been created for these graphs.

EXAMPLES:

26.1.16 TreeOfGroupsToContractibleGcomplex

▷ `TreeOfGroupsToContractibleGcomplex(D, G)` (function)

Inputs a graph of groups D which is a tree, and also inputs the fundamental group G of the tree in a form which contains each of the groups in the graph as subgroups. It returns a corresponding contractible G-complex.

EXAMPLES:

26.1.17 TreeOfResolutionsToContractibleGcomplex

▷ `TreeOfResolutionsToContractibleGcomplex(D, G)` (function)

Inputs a graph of resolutions D which is a tree, and also inputs the fundamental group G of the tree in a form which contains each of the groups in the graph as subgroups. It returns a corresponding contractible G-complex. The resolutions are stored as a component of the contractible G-complex.

EXAMPLES:

Chapter 27

Torsion Subcomplexes

The Torsion Subcomplex subpackage has been conceived and implemented by BUI ANH TUAN and ALEXANDER D. RAHM.

27.1

27.1.1 RigidFacetsSubdivision

▷ `RigidFacetsSubdivision(X)` (function)

It inputs an n-dimensional G-equivariant CW-complex X on which all the cell stabilizer subgroups in G are finite. It returns an n-dimensional G-equivariant CW-complex Y which is topologically the same as X, but equipped with a G-CW-structure which is rigid.

EXAMPLES:

27.1.2 IsPNormal

▷ `IsPNormal(G, p)` (function)

Inputs a finite group G and a prime p. Checks if the group G is p-normal for the prime p. Zassenhaus defines a finite group to be p-normal if the center of one of its Sylow p-groups is the center of every Sylow p-group in which it is contained.

EXAMPLES:

27.1.3 TorsionSubcomplex

▷ `TorsionSubcomplex(C, p)` (function)

Inputs either a cell complex with action of a group as a variable or a group name. In HAP, presently the following cell complexes with stabilisers fixing their cells pointwise are available, specified by the following "groupName" strings:

"SL(2,O-2)" , "SL(2,O-7)" , "SL(2,O-11)" , "SL(2,O-19)" , "SL(2,O-43)" , "SL(2,O-67)" , "SL(2,O-163)",

where the symbol O[-m] stands for the ring of integers in the imaginary quadratic number field Q(sqrt(-m)), the latter being the extension of the field of rational numbers by the square root of minus the square-free positive integer m. The additive structure of this ring O[-m] is given as the module Z[omega] over the natural integers Z with basis {1, omega}, and omega being the square root of minus m if m is congruent to 1 or 2 modulo four; else, in the case m congruent 3 modulo 4, the element omega is the arithmetic mean with 1, namely $(1+sqrt(-m))/2$.

The function TorsionSubcomplex prints the cells with p-torsion in their stabilizer on the screen and returns the incidence matrix of the 1-skeleton of this cellular subcomplex, as well as a Boolean value on whether the cell complex has its cell stabilisers fixing their cells pointwise.

It is also possible to input the cell complexes

"SL(2,Z)" , "SL(3,Z)" , "PGL(3,Z[i])" , "PGL(3,Eisenstein_Integers)" , "PSL(4,Z)" , "PSL(4,Z)_b" , "PSL(4,Z)_c" , "PSL(4,Z)_d" , "Sp(4,Z)"

provided by MATHIEU DUTOUR.

EXAMPLES: 1 , 2 , 3

27.1.4 DisplayAvailableCellComplexes

▷ `DisplayAvailableCellComplexes()` (function)

Displays the cell complexes that are available in HAP.

EXAMPLES:

27.1.5 VisualizeTorsionSkeleton

▷ `VisualizeTorsionSkeleton(`*groupName, p*`)` (function)

Executes the function TorsionSubcomplex(groupName, p) and visualizes its output, namely the incidence matrix of the 1-skeleton of the p-torsion subcomplex, as a graph.

EXAMPLES:

27.1.6 ReduceTorsionSubcomplex

▷ `ReduceTorsionSubcomplex(`*C, p*`)` (function)

This function start with the same operations as the function TorsionSubcomplex(C, p), and if the cell stabilisers are fixing their cells pointwise, it continues as follows.

It prints on the screen which cells to merge and which edges to cut off in order to reduce the p-torsion subcomplex without changing the equivariant Farrell cohomology. Finally, it prints the representative cells, their stabilizers and the Abelianization of the latter.

EXAMPLES:

27.1.7 EquivariantEulerCharacteristic

▷ `EquivariantEulerCharacteristic(X)` (function)

It inputs an n-dimensional Γ-equivariant CW-complex X all the cell stabilizer subgroups in Γ are finite. It returns the equivariant euler characteristic obtained by using mass formula $\sum_\sigma (-1)^{dim\sigma} \frac{1}{card(\Gamma_\sigma)}$

EXAMPLES:

27.1.8 CountingCellsOfACellComplex

▷ `CountingCellsOfACellComplex(X)` (function)

It inputs an n-dimensional Γ-equivariant CW-complex X on which all the cell stabilizer subgroups in Γ are finite. It returns the number of cells in X

EXAMPLES:

27.1.9 CountingControlledSubdividedCells

▷ `CountingControlledSubdividedCells(X)` (function)

It inputs an n-dimensional Γ-equivariant CW-complex X on which all the cell stabilizer subgroups in Γ are finite. It returns the number of cells in X appear during the subdivision process using the RigidFacetsSubdivision.

EXAMPLES:

27.1.10 CountingBaryCentricSubdividedCells

▷ `CountingBaryCentricSubdividedCells(X)` (function)

It inputs an n-dimensional Γ-equivariant CW-complex X on which all the cell stabilizer subgroups in Γ are finite. It returns the number of cells in X appear during the subdivision process using the barycentric subdivision.

EXAMPLES:

27.1.11 EquivariantSpectralSequencePage

▷ `EquivariantSpectralSequencePage(C, m, n)` (function)

It inputs a triple (C,m,n) where C is either a groupName explained as in TorsionSubcomplex, m is the dimension of the reduced torsion subcomplex, and n is the highest vertical degree in the spectral sequence page. At the moment, the function works only when m=1,i.e, after reduction the torsion subcomplex has degree 1. It returns a component object R consists of the first page of spectral sequence, and i-th cohomology groups for i less than n.

EXAMPLES:

27.1.12 ExportHapCellcomplexToDisk

▷ `ExportHapCellcomplexToDisk(`*C, groupName*`)` (function)

It inputs a cell complex *C* which is stored as a variable in the memory, together with a user's desire name. In case, the input is a torsion cell complex then the user's desire name should be in the form "group_ptorsion" in order to use the function EquivariantSpectralSequencePage. The function will export C to the hard disk.

EXAMPLES:

Chapter 28

Simplicial Complexes

28.1

28.1.1 Homology

▷ `Homology(T, n)` (function)
▷ `Homology(T)` (function)

Inputs a pure cubical complex, or cubical complex, or simplicial complex T and a non-negative integer n. It returns the n-th integral homology of T as a list of torsion integers. If no value of n is input then the list of all homologies of T in dimensions 0 to Dimension(T) is returned .

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45

28.1.2 RipsHomology

▷ `RipsHomology(G, n)` (function)
▷ `RipsHomology(G, n, p)` (function)

Inputs a graph G, a non-negative integer n (and optionally a prime number p). It returns the integral homology (or mod p homology) in degree n of the Rips complex of G.

EXAMPLES:

28.1.3 Bettinumbers

▷ `Bettinumbers(T, n)` (function)
▷ `Bettinumbers(T)` (function)

Inputs a pure cubical complex, or cubical complex, simplicial complex or chain complex T and a non-negative integer n. The rank of the n-th rational homology group $H_n(T,\mathbb{Q})$ is returned. If no value for n is input then the list of Betti numbers in dimensions 0 to Dimension(T) is returned .

EXAMPLES: 1 , 2

28.1.4 ChainComplex

▷ `ChainComplex(T)` (function)

Inputs a pure cubical complex, or cubical complex, or simplicial complex T and returns the (often very large) cellular chain complex of T.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13

28.1.5 CechComplexOfPureCubicalComplex

▷ `CechComplexOfPureCubicalComplex(T)` (function)

Inputs a d-dimensional pure cubical complex T and returns a simplicial complex S. The simplicial complex S has one vertex for each d-cube in T, and an n-simplex for each collection of n+1 d-cubes with non-trivial common intersection. The homotopy types of T and S are equal.

EXAMPLES: 1 , 2

28.1.6 PureComplexToSimplicialComplex

▷ `PureComplexToSimplicialComplex(T, k)` (function)

Inputs either a d-dimensional pure cubical complex T or a d-dimensional pure permutahedral complex T together with a non-negative integer k. It returns the first k dimensions of a simplicial complex S. The simplicial complex S has one vertex for each d-cell in T, and an n-simplex for each collection of n+1 d-cells with non-trivial common intersection. The homotopy types of T and S are equal.

For a pure cubical complex T this uses a slightly different algorithm to the function CechComplexOfPureCubicalComplex(T) but constructs the same simplicial complex.

EXAMPLES: 1

28.1.7 RipsChainComplex

▷ `RipsChainComplex(G, n)` (function)

Inputs a graph G and a non-negative integer n. It returns $n+1$ terms of a chain complex whose homology is that of the nerve (or Rips complex) of the graph in degrees up to n.

EXAMPLES: 1

28.1.8 VectorsToSymmetricMatrix

▷ `VectorsToSymmetricMatrix(M)` (function)
▷ `VectorsToSymmetricMatrix(M, distance)` (function)

Inputs a matrix M of rational numbers and returns a symmetric matrix S whose (i, j) entry is the distance between the i-th row and j-th rows of M where distance is given by the sum of the absolute values of the coordinate differences.

Optionally, a function distance(v,w) can be entered as a second argument. This function has to return a rational number for each pair of rational vectors v, w of length Length(M[1]).

EXAMPLES: 1 , 2 , 3

28.1.9 EulerCharacteristic

▷ `EulerCharacteristic(T)` (function)

Inputs a pure cubical complex, or cubical complex, or simplicial complex T and returns its Euler characteristic.

EXAMPLES: 1

28.1.10 MaximalSimplicesToSimplicialComplex

▷ `MaximalSimplicesToSimplicialComplex(L)` (function)

Inputs a list L whose entries are lists of vertices representing the maximal simplices of a simplicial complex. The simplicial complex is returned. Here a "vertex" is a GAP object such as an integer or a subgroup.

EXAMPLES: 1 , 2 , 3 , 4

28.1.11 SkeletonOfSimplicialComplex

▷ `SkeletonOfSimplicialComplex(S, k)` (function)

Inputs a simplicial complex S and a positive integer k less than or equal to the dimension of S. It returns the truncated k-dimensional simplicial complex S^k (and leaves S unchanged).

EXAMPLES:

28.1.12 GraphOfSimplicialComplex

▷ `GraphOfSimplicialComplex(S)` (function)

Inputs a simplicial complex S and returns the graph of S.

EXAMPLES: 1 , 2 , 3

28.1.13 ContractibleSubcomplexOfSimplicialComplex

▷ `ContractibleSubcomplexOfSimplicialComplex(S)` (function)

Inputs a simplicial complex S and returns a (probably maximal) contractible subcomplex of S.

EXAMPLES:

28.1.14 PathComponentsOfSimplicialComplex

▷ `PathComponentsOfSimplicialComplex(S, n)` (function)

Inputs a simplicial complex S and a nonnegative integer n. If $n = 0$ the number of path components of S is returned. Otherwise the n-th path component is returned (as a simplicial complex).

EXAMPLES:

28.1.15 QuillenComplex

▷ `QuillenComplex(G)` (function)

Inputs a finite group G and returns, as a simplicial complex, the order complex of the poset of non-trivial elementary abelian subgroups of G.

EXAMPLES: 1 , 2 , 3 , 4

28.1.16 SymmetricMatrixToIncidenceMatrix

▷ `SymmetricMatrixToIncidenceMatrix(S, t)` (function)
▷ `SymmetricMatrixToIncidenceMatrix(S, t, d)` (function)

Inputs a symmetric integer matrix S and an integer t. It returns the matrix M with $M_{ij} = 1$ if I_{ij} is less than t and $I_{ij} = 1$ otherwise.

An optional integer d can be given as a third argument. In this case the incidence matrix should have roughly at most d entries in each row (corresponding to the d smallest entries in each row of S).

EXAMPLES:

28.1.17 IncidenceMatrixToGraph

▷ `IncidenceMatrixToGraph(M)` (function)

Inputs a symmetric 0/1 matrix M. It returns the graph with one vertex for each row of M and an edges between vertices i and j if the (i, j) entry in M equals 1.

EXAMPLES:

28.1.18 CayleyGraphOfGroup

▷ `CayleyGraphOfGroup(G, A)` (function)

Inputs a group G and a set A of generators. It returns the Cayley graph.

EXAMPLES:

28.1.19 PathComponentsOfGraph

▷ `PathComponentsOfGraph(G, n)` (function)

Inputs a graph G and a nonnegative integer n. If $n = 0$ the number of path components is returned. Otherwise the n-th path component is returned (as a graph).

EXAMPLES:

28.1.20 ContractGraph

▷ `ContractGraph(G)` (function)

Inputs a graph G and tries to remove vertices and edges to produce a smaller graph G' such that the indlusion $G' \to G$ induces a homotopy equivalence $RG \to RG'$ of Rips complexes. If the graph G is modified the function returns true, and otherwise returns false.

EXAMPLES: 1 , 2

28.1.21 GraphDisplay

▷ `GraphDisplay(G)` (function)

This function uses GraphViz software to display a graph G.

EXAMPLES: 1 , 2

28.1.22 SimplicialMap

▷ `SimplicialMap(K, L, f)` (function)
▷ `SimplicialMapNC(K, L, f)` (function)

Inputs simplicial complexes K , L and a function $f: K!.vertices \to L!.vertices$ representing a simplicial map. It returns a simplicial map $K \to L$. If f does not happen to represent a simplicial map then SimplicialMap(K,L,f) will return fail; SimplicialMapNC(K,L,f) will not do any check and always return something of the data type "simplicial map".

EXAMPLES:

28.1.23 ChainMapOfSimplicialMap

▷ `ChainMapOfSimplicialMap(f)` (function)

Inputs a simplicial map $f: K \to L$ and returns the corresponding chain map $C_*(f): C_*(K) \to C_*(L)$ of the simplicial chain complexes..

EXAMPLES:

28.1.24 SimplicialNerveOfGraph

▷ `SimplicialNerveOfGraph(G, d)` (function)

Inputs a graph G and returns a d-dimensional simplicial complex K whose 1-skeleton is equal to G. There is a simplicial inclusion $K \to RG$ where: (i) the inclusion induces isomorphisms on homotopy groups in dimensions less than d; (ii) the complex RG is the Rips complex (with one n-simplex for each complete subgraph of G on $n+1$ vertices).

EXAMPLES: 1 , 2

Chapter 29

Cubical Complexes

29.1

29.1.1 ArrayToPureCubicalComplex

▷ `ArrayToPureCubicalComplex(A, n)` (function)

Inputs an integer array A of dimension d and an integer n. It returns a d-dimensional pure cubical complex corresponding to the black/white "image" determined by the threshold n and the values of the entries in A. (Integers below the threshold correspond to a black pixel, and higher integers correspond to a white pixel.)

EXAMPLES:

29.1.2 PureCubicalComplex

▷ `PureCubicalComplex(A, n)` (function)

Inputs a binary array A of dimension d. It returns the corresponding d-dimensional pure cubical complex.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12

29.1.3 FramedPureCubicalComplex

▷ `FramedPureCubicalComplex(M)` (function)

Inputs a pure cubical complex M and returns the pure cubical complex with a border of zeros attached the each face of the boundary array M!.boundaryArray. This function just adds a bit of space for performing operations such as thickenings to M.

EXAMPLES: 1

29.1.4 RandomCubeOfPureCubicalComplex

▷ `RandomCubeOfPureCubicalComplex(M)` (function)

Inputs a pure cubical complex M and returns a pure cubical complex R with precisely the same dimensions as M. The complex R consist of one cube selected at random from M.

EXAMPLES: 1

29.1.5 PureCubicalComplexIntersection

▷ `PureCubicalComplexIntersection(S, T)` (function)

Inputs two pure cubical complexes with common dimension and array size. It returns the intersection of the two complexes. (An entry in the binary array of the intersection has value 1 if and only if the corresponding entries in the binary arrays of S and T both have value 1.)

EXAMPLES: 1

29.1.6 PureCubicalComplexUnion

▷ `PureCubicalComplexUnion(S, T)` (function)

Inputs two pure cubical complexes with common dimension and array size. It returns the union of the two complexes. (An entry in the binary array of the union has value 1 if and only if at least one of the corresponding entries in the binary arrays of S and T has value 1.)

EXAMPLES: 1

29.1.7 PureCubicalComplexDifference

▷ `PureCubicalComplexDifference(S, T)` (function)

Inputs two pure cubical complexes with common dimension and array size. It returns the difference S-T. (An entry in the binary array of the difference has value 1 if and only if the corresponding entry in the binary array of S is 1 and the corresponding entry in the binary array of T is 0.)

EXAMPLES: 1 , 2

29.1.8 ReadImageAsPureCubicalComplex

▷ `ReadImageAsPureCubicalComplex(str, n)` (function)

Reads an image file *str* (= "file.png", "file.eps", "file.bmp" etc) and an integer n between 0 and 765. It returns a 2-dimensional pure cubical complex based on the black/white version of the image determined by the threshold n.

EXAMPLES: 1 , 2 , 3 , 4 , 5

29.1.9 ReadLinkImageAsPureCubicalComplex

▷ `ReadLinkImageAsPureCubicalComplex(str)` (function)
▷ `ReadLinkImageAsPureCubicalComplex(str, n)` (function)

Reads an image file *str* (= "file.png", "file.eps", "file.bmp" etc) containing a knot or link diagram, and optionally a positive integer n. The integer n should be a little larger than the line thickness in the link diagram, and if not provided then n is set equal to 10. The function tries to output the

corresponding knot or link as a 3-dimensional pure cubical complex. Ideally the link diagram should be produced with line thickness 6 in Xfig, and the under-crossing spaces should not be too large or too small or too near one another. The function does not always succeed: it applies several checks, and if one of these checks fails then the function returns "fail".

EXAMPLES: 1

29.1.10 ReadImageSequenceAsPureCubicalComplex

▷ `ReadImageSequenceAsPureCubicalComplex(dir, n)` (function)

Reads the name of a directory *dir* containing a sequence of image files (ordered alphanumerically), and an integer *n* between 0 and 765. It returns a 3-dimensional pure cubical complex based on the black/white version of the images determined by the threshold *n*.

EXAMPLES: 1

29.1.11 Size

▷ `Size(T)` (function)

This returns the number of non-zero entries in the binary array of the cubical complex, or pure cubical complex T.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22

29.1.12 Dimension

▷ `Dimension(T)` (function)

This returns the dimension of the cubical complex, or pure cubical complex T.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

29.1.13 WritePureCubicalComplexAsImage

▷ `WritePureCubicalComplexAsImage(T, str1, str2)` (function)

Inputs a 2-dimensional pure cubical complex T, and a filename *str*1 followed by its extension *str*2 (e.g. *str*1="myfile" followed by *str*2="png"). A black/white image is saved to the file.

EXAMPLES:

29.1.14 ViewPureCubicalComplex

▷ `ViewPureCubicalComplex(T)` (function)
▷ `ViewPureCubicalComplex(T, str)` (function)

Inputs a 2-dimensional pure cubical complex T, and optionally a command such as *str*="mozilla" for viewing image files. A black/white image is displayed.

EXAMPLES: 1 , 2 , 3

29.1.15 Homology

▷ `Homology(T, n)` (function)
▷ `Homology(T)` (function)

Inputs a pure cubical complex, or cubical complex, or simplicial complex T and a non-negative integer n. It returns the n-th integral homology of T as a list of torsion integers. If no value of n is input then the list of all homologies of T in dimensions 0 to Dimension(T) is returned .

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45

29.1.16 Bettinumbers

▷ `Bettinumbers(T, n)` (function)
▷ `Bettinumbers(T)` (function)

Inputs a pure cubical complex, or cubical complex, simplicial complex or chain complex T and a non-negative integer n. The rank of the n-th rational homology group $H_n(T,\mathbb{Q})$ is returned. If no value for n is input then the list of Betti numbers in dimensions 0 to Dimension(T) is returned .

EXAMPLES: 1 , 2

29.1.17 DirectProductOfPureCubicalComplexes

▷ `DirectProductOfPureCubicalComplexes(M, N)` (function)

Inputs two pure cubical complexes M,N and returns their direct product D as a pure cubical complex. The dimension of D is the sum of the dimensions of M and N.

EXAMPLES: 1 , 2

29.1.18 SuspensionOfPureCubicalComplex

▷ `SuspensionOfPureCubicalComplex(M)` (function)

Inputs a pure cubical complex M and returns a pure cubical complex with the homotopy type of the suspension of M.

EXAMPLES: 1

29.1.19 EulerCharacteristic

▷ `EulerCharacteristic(T)` (function)

Inputs a pure cubical complex, or cubical complex, or simplicial complex T and returns its Euler characteristic.

EXAMPLES: 1

29.1.20 PathComponentOfPureCubicalComplex

▷ `PathComponentOfPureCubicalComplex(T, n)` (function)

Inputs a pure cubical complex T and an integer n in the rane 1, ..., Bettinumbers(T)[1] . It returns the n-th path component of T as a pure cubical complex. The value $n = 0$ is also allowed, in which case the number of path components is returned.

EXAMPLES: 1

29.1.21 ChainComplex

▷ `ChainComplex(T)` (function)

Inputs a pure cubical complex, or cubical complex, or simplicial complex T and returns the (often very large) cellular chain complex of T.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13

29.1.22 ChainComplexOfPair

▷ `ChainComplexOfPair(T, S)` (function)

Inputs a pure cubical complex or cubical complex T and subcomplex S. It returns the quotient $C(T)/C(S)$ of cellular chain complexes.

EXAMPLES: 1 , 2

29.1.23 ExcisedPureCubicalPair

▷ `ExcisedPureCubicalPair(T, S)` (function)

Inputs a pure cubical complex T and subcomplex S. It returns the pair $[T \setminus intS, S \setminus intS])$ of pure cubical complexes where $intS$ is the pure cubical complex obtained from S by removing its boundary.

EXAMPLES:

29.1.24 ChainInclusionOfPureCubicalPair

▷ `ChainInclusionOfPureCubicalPair(S, T)` (function)

Inputs a pure cubical complex T and subcomplex S. It returns the chain inclusion $C(S) \to C(T)$ of cellular chain complexes.

EXAMPLES:

29.1.25 ChainMapOfPureCubicalPairs

▷ `ChainMapOfPureCubicalPairs` (global variable)

Inputs a pure cubical complex N and subcomplexes M, T and S in T. It returns the chain map $C(M/S) \to C(N/T)$ of quotient cellular chain complexes.

EXAMPLES:

29.1.26 ContractPureCubicalComplex

▷ `ContractPureCubicalComplex(T)` (function)

Inputs a pure cubical complex T of dimension d and removes d-dimensional cells from T without changing the homotopy type of T. When the function has been applied, no further d-cells can be removed from T without changing its homotopy type. This function modifies T.

EXAMPLES: 1

29.1.27 ContractedComplex

▷ `ContractedComplex(T)` (function)

Inputs a pure cubical complex T and returns a structural copy of the complex obtained from T by applying the function ContractPureCubicalComplex(T).

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13

29.1.28 ZigZagContractedPureCubicalComplex

▷ `ZigZagContractedPureCubicalComplex(T)` (function)

Inputs a pure cubical complex T and returns a homotopy equivalent pure cubical complex S. The aim is for S to involve fewer cells than T and certainly to involve no more cells than T.

EXAMPLES: 1 , 2 , 3

29.1.29 ContractCubicalComplex

▷ `ContractCubicalComplex(T)` (function)

Inputs a cubical complex T and removes cells without changing the homotopy type of T. It changes T. In particular, it adds the components T.vectors and T.rewrite of a discrete vector field.

At present this function only works for cubical complexes of dimension 2 or 3.

EXAMPLES: 1

29.1.30 DVFReducedCubicalComplex

▷ `DVFReducedCubicalComplex(T)` (function)

Inputs a cubical complex T and returns a non-regular cubical complex R by constructing a discrete vector field. The vector field is designed to minimize the number of critical cells in R at the cost of allowing cell attaching maps that are not homeomorphisms on boundaries.

At present this function works only for 2- and 3-dimensional cubical complexes.

The function ChainComplex(R) can be used to obtain the cellular chain complex of R.

EXAMPLES: 1

29.1.31 SkeletonOfCubicalComplex

▷ `SkeletonOfCubicalComplex(T, n)` (function)

Inputs a cubical complex, or pure cubical complex T and positive integer n. It returns the n-skeleton of T as a cubical complex.

EXAMPLES:

29.1.32 ContractibleSubomplexOfPureCubicalComplex

▷ `ContractibleSubomplexOfPureCubicalComplex` (global variable)

Inputs a pure cubical complex T and returns a maximal contractible pure cubical subcomplex.

EXAMPLES:

29.1.33 AcyclicSubomplexOfPureCubicalComplex

▷ `AcyclicSubomplexOfPureCubicalComplex` (global variable)

Inputs a pure cubical complex T and returns a (not necessarily connected) pure cubical subcomplex having trivial homology in all degrees greater than 0.

EXAMPLES:

29.1.34 HomotopyEquivalentMaximalPureCubicalSubcomplex

▷ `HomotopyEquivalentMaximalPureCubicalSubcomplex(T, S)` (function)

Inputs a pure cubical complex T together with a pure cubical subcomplex S. It returns a pure cubical subcomplex H of T which contains S and is maximal with respect to the property that it is homotopy equivalent to S.

EXAMPLES:

29.1.35 HomotopyEquivalentMinimalPureCubicalSubcomplex

▷ `HomotopyEquivalentMinimalPureCubicalSubcomplex(T, S)` (function)

Inputs a pure cubical complex T together with a pure cubical subcomplex S. It returns a pure cubical subcomplex H of T which contains S and is minimal with respect to the property that it is homotopy equivalent to T.

EXAMPLES: 1

29.1.36 BoundaryOfPureCubicalComplex

▷ `BoundaryOfPureCubicalComplex(T)` (function)

Inputs a pure cubical complex T and returns its boundary as a pure cubical complex. The boundary consists of all cubes which have one or more facets that lie in just the one cube.

EXAMPLES: 1

29.1.37 SingularitiesOfPureCubicalComplex

▷ `SingularitiesOfPureCubicalComplex(T, radius, tolerance)` (function)

Inputs a pure cubical complex T together with a positive integer "radius" and an integer "tolerance" in the range 1..100. It returns the pure cubical subcomplex of those cells in the boundary where the boundary is not differentiable. (The method for deciding differentiability at a point is crude/discrete, prone to errors and depends on the radius and tolerance.)

EXAMPLES: 1

29.1.38 ThickenedPureCubicalComplex

▷ `ThickenedPureCubicalComplex(T)` (function)

Inputs a pure cubical complex T and returns a pure cubical complex S. If a euclidean cube is in T then this cube and all its neighbouring cubes are included in S.

EXAMPLES: 1 , 2 , 3

29.1.39 CropPureCubicalComplex

▷ `CropPureCubicalComplex(T)` (function)

Inputs a pure cubical complex T and returns a pure cubical complex S obtained from T by removing any "zero boundary sheets" of the binary array. Thus S and T are isometric as euclidean spaces but there may be fewer zero entries in the binary array for S.

EXAMPLES:

29.1.40 BoundingPureCubicalComplex

▷ `BoundingPureCubicalComplex(T)` (function)

Inputs a pure cubical complex T and returns a contractible pure cubical complex S containing T.

EXAMPLES:

29.1.41 MorseFiltration

▷ `MorseFiltration(M, i, t, bool)` (function)
▷ `MorseFiltration(M, i, t)` (function)

Inputs a pure cubical complex M of dimension d, an integer i between 1 and d, a positive integer t and a boolean value True or False. The function returns a list $[M_1, M_2, ..., M_t]$ of pure cubical complexes with M_k a subcomplex of M_{k+1}. The list is constructed by setting all slices of M perpendicular to the i-th axis equal to zero if they meet the ith axis at a sufficiently high coordinate (if bool=True) or sufficiently low coordinate (if bool=False).

If the variable bool is not specified then it is assumed to have the value True.

EXAMPLES:

29.1.42 ComplementOfPureCubicalComplex

▷ `ComplementOfPureCubicalComplex(T)` (function)

Inputs a pure cubical complex T and returns a pure cubical complex S. A euclidean cube is in S precisely when the cube is not in T.

EXAMPLES: 1 , 2 , 3

29.1.43 PureCubicalComplexToTextFile

▷ `PureCubicalComplexToTextFile(file, M)` (function)

Inputs a pure cubical complex M and a string containing the address of a file. A representation of this complex is written to the file in a format that can be read by the CAPD (Computer Assisted Proofs in Dynamics) software developed by Marian Mrozek and others.

EXAMPLES:

29.1.44 ThickeningFiltration

▷ `ThickeningFiltration(M, n)` (function)
▷ `ThickeningFiltration(M, n, k)` (function)

Inputs a pure cubical complex M and a positive integer n. It returns a filtered pure cubical complex constructed frim n thickenings of M. If a positive integer k is supplied as an optional third argument, then each step of the filtration is obtained from a k-fold thickening.

EXAMPLES: 1 , 2

29.1.45 Dendrogram

▷ `Dendrogram(M)` (function)

Inputs a filtered pure cubical complex M and returns data that specifies the dendrogram (or phylogenetic tree) describing how path components are born and then merge during the filtration.

EXAMPLES:

29.1.46 DendrogramDisplay

▷ `DendrogramDisplay` (global variable)

Inputs a filtered pure cubical complex M, or alternatively inputs the out from the command Dendrogram(M), and then uses GraphViz software to display the path component dendrogram of M.

EXAMPLES:

29.1.47 DendrogramToPersistenceMat

▷ `DendrogramToPersistenceMat(D)` (function)

Inputs the output of the function Dendrogram(M) and returns the corresponding degree 0 Betti bar code.

EXAMPLES:

29.1.48 ReadImageAsFilteredPureCubicalComplex

▷ `ReadImageAsFilteredPureCubicalComplex(file, n)` (function)

Inputs a string containing the path to an image file, together with a positive integer n. It returns a filtered pure cubical complex of filtration length *n*.

EXAMPLES: 1

29.1.49 ComplementOfFilteredPureCubicalComplex

▷ `ComplementOfFilteredPureCubicalComplex(M)` (function)

Inputs a filtered pure cubical complex *M* and returns the complement as a filtered pure cubical complex.

EXAMPLES: 1

29.1.50 PersistentHomologyOfFilteredPureCubicalComplex

▷ `PersistentHomologyOfFilteredPureCubicalComplex(M, n)` (function)

Inputs a filtered pure cubical complex *M* and a non-negative integer *n*. It returns the degree *n* persistent homology of *M* with rational coefficients.

EXAMPLES:

Chapter 30

Regular CW-Complexes

30.1

30.1.1 SimplicialComplexToRegularCWComplex

▷ `SimplicialComplexToRegularCWComplex(K)` (function)

Inputs a simplicial complex K and returns the corresponding regular CW-complex.
EXAMPLES: 1 , 2

30.1.2 CubicalComplexToRegularCWComplex

▷ `CubicalComplexToRegularCWComplex(K)` (function)
▷ `CubicalComplexToRegularCWComplex(K, n)` (function)

Inputs a pure cubical complex (or cubical complex) K and returns the corresponding regular CW-complex. If a positive integer n is entered as an optional second argument, then just the n-skeleton of K is returned.
EXAMPLES: 1

30.1.3 CriticalCellsOfRegularCWComplex

▷ `CriticalCellsOfRegularCWComplex(Y)` (function)
▷ `CriticalCellsOfRegularCWComplex(Y, n)` (function)

Inputs a regular CW-complex Y and returns the critical cells of Y with respect to some discrete vector field. If Y does not initially have a discrete vector field then one is constructed.

If a positive integer n is given as a second optional input, then just the critical cells in dimensions up to and including n are returned.

The function $CriticalCellsOfRegularCWComplex(Y)$ works by homotopy reducing cells starting at the top dimension. The function $CriticalCellsOfRegularCWComplex(Y,n)$ works by homotopy coreducing cells starting at dimension 0. The two methods may well return different numbers of cells.
EXAMPLES: 1 , 2 , 3

30.1.4 ChainComplex

▷ `ChainComplex(Y)` (function)

Inputs a regular CW-complex Y and returns the cellular chain complex of a CW-complex W whose cells correspond to the critical cells of Y with respect to some discrete vector field. If Y does not initially have a discrete vector field then one is constructed.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13

30.1.5 ChainComplexOfRegularCWComplex

▷ `ChainComplexOfRegularCWComplex(Y)` (function)

Inputs a regular CW-complex Y and returns the cellular chain complex of Y.

EXAMPLES: 1 , 2

30.1.6 FundamentalGroup

▷ `FundamentalGroup(Y)` (function)
▷ `FundamentalGroup(Y, n)` (function)

Inputs a regular CW-complex Y and, optionally, the number of some 0-cell. It returns the fundamental group of Y based at the 0-cell n. The group is returned as a finitely presented group. If n is not specified then it is set $n = 1$. The algorithm requires a discrete vector field on Y. If Y does not initially have a discrete vector field then one is constructed.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13

Chapter 31

Knots and Links

31.1

31.1.1 PureCubicalKnot

▷ `PureCubicalKnot(L)` (function)
▷ `PureCubicalKnot(n, i)` (function)

Inputs a list $L = [[m1,n1],[m2,n2],...,[mk,nk]]$ of pairs of integers describing a cubical arc presentation of a link with all vertical lines at the front and all horizontal lines at the back. The bottom horizontal line extends from the m1-th column to the n1-th column. The second to bottom horizontal line extends from the m2-th column to the n2-th column. And so on. The link is returned as a 3-dimensional pure cubical complex.

Alternatively the function inputs two integers n, i and returns the i-th prime knot on n crossings.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11

31.1.2 ViewPureCubicalKnot

▷ `ViewPureCubicalKnot(L)` (function)

Inputs a pure cubical link L and displays it.

EXAMPLES: 1 , 2

31.1.3 KnotSum

▷ `KnotSum(K, L)` (function)

Inputs two pure cubical knots K, L and returns their sum as a pure cubical knot. This function is not defined for links with more than one component.

EXAMPLES: 1 , 2 , 3 , 4 , 5

31.1.4 KnotGroup

▷ `KnotGroup(K)` (function)

Inputs a pure cubical link *K* and returns the fundamental group of its complement. The group is returned as a finitely presented group.

EXAMPLES: 1

31.1.5 AlexanderMatrix

▷ `AlexanderMatrix(G)` (function)

Inputs a finitely presented group *G* whose abelianization is infinite cyclic. It returns the Alexander matrix of the presentation.

EXAMPLES:

31.1.6 AlexanderPolynomial

▷ `AlexanderPolynomial(K)` (function)
▷ `AlexanderPolynomial(G)` (function)

Inputs either a pure cubical knot *K* or a finitely presented group *G* whose abelianization is infinite cyclic. The Alexander Polynomial is returned.

EXAMPLES: 1 , 2 , 3 , 4

31.1.7 ProjectionOfPureCubicalComplex

▷ `ProjectionOfPureCubicalComplex(K)` (function)

Inputs an n-dimensional pure cubical complex *K* and returns an n-1-dimensional pure cubical complex K'. The returned complex is obtained by projecting Euclidean n-space onto Euclidean n-1-space.

EXAMPLES:

31.1.8 ReadPDBfileAsPureCubicalComplex

▷ `ReadPDBfileAsPureCubicalComplex(file)` (function)
▷ `ReadPDBfileAsPureCubicalComplex(file, m, c)` (function)

Inputs a protein database file describing a protein, and optionally inputs a positive integer m and character string c. The default values for the optional inputs are m=5 and c="A". It loads the chain of amino acids labelled by c in the file as a 3-dimensional pure cubical complex of the homotopy type of a circle.

It might happen that the function fails to construct a pure cubical complex of the homotopy type of a circle. In this case retry with a larger integer m.

EXAMPLES: 1 , 2 , 3

Chapter 32

Knots and Quandles

32.1

Knots

32.1.1 PresentationKnotQuandle

▷ `PresentationKnotQuandle(`*gaussCode*`)` (function)

Inputs a Gauss Code of a knot (with the orientations; see *GaussCodeOfPureCubicalKnot* in HAP package) and outputs the generators and relators of the knot quandle associated (in the form of a record).

EXAMPLES: 1 , 2 , 3 , 4

32.1.2 PD2GC

▷ `PD2GC(`*PD*`)` (function)

Inputs a Planar Diagram of a knot; outputs the Gauss Code associated (with the orientations).

EXAMPLES: 1 , 2 , 3

32.1.3 PlanarDiagramKnot

▷ `PlanarDiagramKnot(`*n*`, `*k*`)` (function)

Returns a Planar Diagram for the k-th knot with n crossings ($n \leq 12$) if it exists; fail otherwise.

EXAMPLES: 1 , 2 , 3

32.1.4 GaussCodeKnot

▷ `GaussCodeKnot(`*n*`, `*k*`)` (function)

Returns a Gauss Code (with orientations) for the k-th knot with n crossings ($n \leq 12$) if it exists; fail otherwise.

EXAMPLES:

32.1.5 PresentationKnotQuandleKnot

▷ `PresentationKnotQuandleKnot(n, k)` (function)

Returns generators and relators (in the form of a record) for the k-th knot with n crossings ($n \leq 12$) if it exists; fail otherwise.

EXAMPLES: 1 , 2 , 3

32.1.6 NumberOfHomomorphisms

▷ `NumberOfHomomorphisms(genRelQ, finiteQ)` (function)

Inputs generators and relators *genRelQ* of a knot quandle (in the form of a record, see above) and a finite quandle *finiteQ*; outputs the number of homomorphisms from the former to the latter.

EXAMPLES: 1 , 2 , 3

32.1.7 PartitionedNumberOfHomomorphisms

▷ `PartitionedNumberOfHomomorphisms(genRelQ, finiteQ)` (function)

Inputs generators and relators *genRelQ* of a knot quandle (in the form of a record, see above) and a finite connected quandle *finiteQ*; outputs a partition of the number of homomorphisms from the former to the latter.

EXAMPLES: 1

Quandles

32.1.8 ConjugationQuandle

▷ `ConjugationQuandle(G, n)` (function)

Inputs a finite group G and an integer n; outputs the associated n-fold conjugation quandle.

EXAMPLES: 1 , 2

32.1.9 FirstQuandleAxiomIsSatisfied

▷ `FirstQuandleAxiomIsSatisfied(M)` (function)
▷ `SecondQuandleAxiomIsSatisfied(M)` (function)
▷ `ThirdQuandleAxiomIsSatisfied(M)` (function)

Inputs a finite magma M; returns true if M satisfy the first/second/third axiom of a quandle, false otherwise.

EXAMPLES:

32.1.10 IsQuandle

▷ `IsQuandle(M)` (function)

Inputs a finite magma M; returns true if M is a quandle, false otherwise.

EXAMPLES: 1 , 2 , 3

32.1.11 Quandles

▷ `Quandles(n)` (function)

Returns a list of all quandles of size n, $n \leq 6$. If $n \geq 7$, it returns fail.
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6

32.1.12 Quandle

▷ `Quandle(n, k)` (function)

Returns the k-th quandle of size n ($n \leq 6$) if such a quandle exists, fail otherwise.
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7

32.1.13 IdQuandle

▷ `IdQuandle(Q)` (function)

Inputs a quandle Q; and outputs a list of integers [n,k] such that Q is isomorphic to $Quandle(n,k)$. If $n \geq 7$, then it returns [n,fail] (where n is the size of Q).
EXAMPLES:

32.1.14 IsLatin

▷ `IsLatin` (global variable)

Inputs a finite quandle Q; returns true if Q is latin, false otherwise.
EXAMPLES:

32.1.15 IsConnectedQuandle

▷ `IsConnectedQuandle` (global variable)

Inputs a finite quandle Q; returns true if Q is connected, false otherwise.
EXAMPLES:

32.1.16 ConnectedQuandles

▷ `ConnectedQuandles(n)` (function)

Returns a list of all connected quandles of size n.
EXAMPLES: 1 , 2 , 3

32.1.17 ConnectedQuandle

▷ `ConnectedQuandle(n, k)` (function)

Returns the k-th quandle of size n if such a quandle exists, fail otherwise.
EXAMPLES: 1 , 2 , 3 , 4

32.1.18 IdConnectedQuandle

▷ `IdConnectedQuandle(Q)` (function)

Inputs a connected quandle Q; and outputs a list of integers [n,k] such that Q is isomorphic to $ConnectedQuandle(n,k)$.
EXAMPLES: 1

32.1.19 IsQuandleEnvelope

▷ `IsQuandleEnvelope(Q, G, e, stigma)` (function)

Inputs a set Q, a permutation group G, an element $e \in Q$ and an element $stigma \in G$; returns true if this structure describes a quandle envelope, false otherwise.
EXAMPLES: 1 , 2 , 3

32.1.20 QuandleQuandleEnvelope

▷ `QuandleQuandleEnvelope(Q, G, e, stigma)` (function)

Inputs a set Q, a permutation group G, an element $e \in Q$ and an element $stigma \in G$. If this structure describes a quandle envelope, the function returns the quandle from this quandle envelope; and fail otherwise. Nb: this quandle is a connected quandle.
EXAMPLES: 1 , 2 , 3

32.1.21 KnotInvariantCedric

▷ `KnotInvariantCedric(genRelQ, n, m)` (function)

Inputs generators and relators of a knot quandle (in the form of a record, see above) and two integers n and m; outputs a list [n1,n2,...,nk] where nj is a partition of the number of homomorphisms from the considered knot quandle to the j-th connected quandle of size $n \leq i \leq m$.
EXAMPLES:

32.1.22 RightMultiplicationGroupAsPerm

▷ `RightMultiplicationGroupAsPerm` (global variable)

Inputs a connected quandle Q; output its right multiplication group whose elements are permutations.
EXAMPLES:

32.1.23 RightMultiplicationGroup

▷ `RightMultiplicationGroup` (global variable)

Inputs a connected quandle Q; output its right multiplication group whose elements are mappings from Q to Q.

EXAMPLES:

32.1.24 AutomorphismGroupQuandleAsPerm

▷ `AutomorphismGroupQuandleAsPerm(Q)` (function)

Inputs a connected quandle Q; outputs its automorphism group whose elements are permutations.

EXAMPLES:

32.1.25 AutomorphismGroupQuandle

▷ `AutomorphismGroupQuandle(Q)` (function)

Inputs a connected quandle Q; outputs its automorphism group whose elements are mappings from Q to Q.

EXAMPLES: 1 , 2 , 3

Chapter 33

Finite metric spaces and their filtered complexes

33.1

33.1.1 CayleyMetric

▷ `CayleyMetric(g, h, N)` (function)
▷ `CayleyMetric(g, h)` (function)

Inputs two permutations g,h and optionally the degree N of a symmetric group containing them. It returns the minimum number of transpositions needed to express $g*h^-1$ as a product of transpositions.

EXAMPLES: 1

33.1.2 HammingMetric

▷ `HammingMetric(g, h, N)` (function)
▷ `HammingMetric(g, h)` (function)

Inputs two permutations g,h and optionally the degree N of a symmetric group containing them. It returns the number of integers moved by the permutation $g*h^-1$.

EXAMPLES:

33.1.3 KendallMetric

▷ `KendallMetric(g, h, N)` (function)
▷ `KendallMetric(g, h)` (function)

Inputs two permutations g,h and optionally the degree N of a symmetric group containing them. It returns the minimum number of adjacent transpositions needed to express g^-1*h as a product of adjacent transpositions. An adjacent transposition has the form $(i,i+1)$.

EXAMPLES:

33.1.4 EuclideanSquaredMetric

▷ `EuclideanSquaredMetric(v, w)` (function)

Inputs two vectors v,w of equal length and returns the sum of the squares of the components of $v-w$. In other words, it returns the square of the Euclidean distance between v and w.

EXAMPLES:

33.1.5 EuclideanApproximatedMetric

▷ `EuclideanApproximatedMetric(v, w)` (function)

Inputs two vectors v,w of equal length and returns a rational approximation to the square root of the sum of the squares of the components of $v-w$. In other words, it returns an approximation to the Euclidean distance between v and w.

EXAMPLES: 1 , 2

33.1.6 ManhattanMetric

▷ `ManhattanMetric(v, w)` (function)

Inputs two vectors v,w of equal length and returns the sum of the absolute values of the components of $v-w$. This is often referred to as the taxi-cab distance between v and w.

EXAMPLES: 1

33.1.7 VectorsToSymmetricMatrix

▷ `VectorsToSymmetricMatrix(L)` (function)
▷ `VectorsToSymmetricMatrix(L, D)` (function)

Inputs a list L of vectors and optionally a metric D. The default is $D = ManhattanMetric$. It returns the symmetric matrix whose i-j-entry is $S[i][j] = D(L[i],L[j])$.

EXAMPLES: 1 , 2 , 3

33.1.8 SymmetricMatDisplay

▷ `SymmetricMatDisplay(S)` (function)
▷ `SymmetricMatDisplay(L, V)` (function)

Inputs an $n \times n$ symmetric matrix S of non-negative integers and an integer t in $[0..100]$. Optionally it inputs a list $V = [V_1,...,V_k]$ of disjoint subsets of $[1..n]$. It displays the graph with vertex set $[1..n]$ and with an edge between i and j if $S[i][j] < t$. If the optional list V is input then the vertices in V_i will be given a common colour distinct from other vertices.

EXAMPLES: 1

33.1.9 SymmetricMatrixToFilteredGraph

▷ `SymmetricMatrixToFilteredGraph(S, t, m)` (function)

Inputs an integer symmetric matrix S, a positive integer t and a positive integer m. The function returns a filtered graph of filtration length t. The k-th term of the filtration is a graph with one vertex for each row of S. There is an edge in this graph between the i-th and j-th vertices if the entry $S[i][j]$ is less than or equal to $k*m/t$.

EXAMPLES: 1 , 2 , 3

33.1.10 PermGroupToFilteredGraph

▷ `PermGroupToFilteredGraph(S, D)` (function)

Inputs a permutation group G and a metric D defined on permutations. The function returns a filtered graph. The k-th term of the filtration is a graph with one vertex for each element of the group G. There is an edge in this graph between vertices g and h if $D(g,h)$ is less than some integer threshold t_k. The thresholds $t_1 < t_2 < ... < t_N$ are chosen to form as long a sequence as possible subject to each term of the filtration being a distinct graph.

EXAMPLES:

Chapter 34

Commutative diagrams and abstract categories

COMMUTATIVE DIAGRAMS

34.1

34.1.1 HomomorphismChainToCommutativeDiagram

▷ `HomomorphismChainToCommutativeDiagram(H)` (function)

Inputs a list $H = [h_1, h_2, ..., h_n]$ of mappings such that the composite $h_1 h_2 ... h_n$ is defined. It returns the list of composable homomorphism as a commutative diagram.

EXAMPLES:

34.1.2 NormalSeriesToQuotientDiagram

▷ `NormalSeriesToQuotientDiagram(L)` (function)
▷ `NormalSeriesToQuotientDiagram(L, M)` (function)

Inputs an increasing (or decreasing) list $L = [L_1, L_2, ..., L_n]$ of normal subgroups of a group G with $G = L_n$. It returns the chain of quotient homomorphisms $G/L_i \to G/L_{i+1}$ as a commutative diagram.

Optionally a subseries M of L can be entered as a second variable. Then the resulting diagram of quotient groups has two rows of horizontal arrows and one row of vertical arrows.

EXAMPLES:

34.1.3 NerveOfCommutativeDiagram

▷ `NerveOfCommutativeDiagram(D)` (function)

Inputs a commutative diagram D and returns the commutative diagram ND consisting of all possible composites of the arrows in D.

EXAMPLES:

34.1.4 GroupHomologyOfCommutativeDiagram

▷ `GroupHomologyOfCommutativeDiagram(D, n)` (function)
▷ `GroupHomologyOfCommutativeDiagram(D, n, prime)` (function)
▷ `GroupHomologyOfCommutativeDiagram(D, n, prime, Resolution_Algorithm)` (function)

Inputs a commutative diagram D of p-groups and positive integer n. It returns the commutative diagram of vector spaces obtained by applying mod p homology.

Non-prime power groups can also be handled if a prime p is entered as the third argument. Integral homology can be obtained by setting $p = 0$. For $p = 0$ the result is a diagram of groups.

A particular resolution algorithm, such as ResolutionNilpotentGroup, can be entered as a fourth argument. For positive p the default is ResolutionPrimePowerGroup. For $p = 0$ the default is ResolutionFiniteGroup.

EXAMPLES:

34.1.5 PersistentHomologyOfCommutativeDiagramOfPGroups

▷ `PersistentHomologyOfCommutativeDiagramOfPGroups(D, n)` (function)

Inputs a commutative diagram D of finite p-groups and a positive integer n. It returns a list containing, for each homomorphism in the nerve of D, a triple $[k,l,m]$ where k is the dimension of the source of the induced mod p homology map in degree n, l is the dimension of the image, and m is the dimension of the cokernel.

EXAMPLES:

ABSTRACT CATEGORIES

34.2

34.2.1 CategoricalEnrichment

▷ `CategoricalEnrichment(X, Name)` (function)

Inputs a structure X such as a group or group homomorphism, together with the name of some existing category such as Name:=Category_of_Groups or Category_of_Abelian_Groups. It returns, as appropriate, an object or arrow in the named category.

EXAMPLES: 1

34.2.2 IdentityArrow

▷ `IdentityArrow(X)` (function)

Inputs an object X in some category, and returns the identity arrow on the object X.

EXAMPLES: 1

34.2.3 InitialArrow

▷ `InitialArrow(X)` (function)

Inputs an object X in some category, and returns the arrow from the initial object in the category to X.

EXAMPLES: 1

34.2.4 TerminalArrow

▷ `TerminalArrow(X)` (function)

Inputs an object X in some category, and returns the arrow from X to the terminal object in the category.

EXAMPLES: 1

34.2.5 HasInitialObject

▷ `HasInitialObject(Name)` (function)

Inputs the name of a category and returns true or false depending on whether the category has an initial object.

EXAMPLES: 1

34.2.6 HasTerminalObject

▷ `HasTerminalObject(Name)` (function)

Inputs the name of a category and returns true or false depending on whether the category has a terminal object.

EXAMPLES:

34.2.7 Source

▷ `Source(f)` (function)

Inputs an arrow f in some category, and returns its source.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12

34.2.8 Target

▷ `Target(f)` (function)

Inputs an arrow f in some category, and returns its target.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

34.2.9 CategoryName

▷ `CategoryName(X)` (function)

Inputs an object or arrow X in some category, and returns the name of the category.
EXAMPLES: 1

34.2.10 CompositionEqualityAdditionMinus

▷ `CompositionEqualityAdditionMinus` (global variable)

Composition of suitable arrows f,g is given by $f*g$ when the source of f equals the target of g. (Warning: this differes to the standard GAP convention.)
Equality is tested using $f=g$.
In an additive category the sum and difference of suitable arrows is given by $f+g$ and $f-g$.
EXAMPLES:

34.2.11 Object

▷ `Object(X)` (function)

Inputs an object X in some category, and returns the GAP structure Y such that $X = CategoricalEnrichment(Y, CategoryName(X))$.
EXAMPLES: 1 , 2

34.2.12 Mapping

▷ `Mapping(X)` (function)

Inputs an arrow f in some category, and returns the GAP structure Y such that $f = CategoricalEnrichment(Y, CategoryName(X))$.
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6

34.2.13 IsCategoryObject

▷ `IsCategoryObject(X)` (function)

Inputs X and returns true if X is an object in some category.
EXAMPLES:

34.2.14 IsCategoryArrow

▷ `IsCategoryArrow(X)` (function)

Inputs X and returns true if X is an arrow in some category.
EXAMPLES:

Chapter 35

Arrays and Pseudo lists

35.1

35.1.1 Array

▷ `Array(A, f)` (function)

Inputs an array A and a function f. It returns the the array obtained by applying f to each entry of A (and leaves A unchanged).

EXAMPLES: 1 , 2 , 3 , 4

35.1.2 PermuteArray

▷ `PermuteArray(A, f)` (function)

Inputs an array A of dimension d and a permutation f of degree at most d. It returns the array B defined by $B[i1][i2]...[id] = A[f(i1)][f(i2)]...A[f(id)]$ (and leaves A unchanged).

EXAMPLES:

35.1.3 ArrayDimension

▷ `ArrayDimension(A)` (function)

Inputs an array A and returns its dimension.

EXAMPLES:

35.1.4 ArrayDimensions

▷ `ArrayDimensions(A)` (function)

Inputs an array A and returns its dimensions.

EXAMPLES:

35.1.5 ArraySum

▷ `ArraySum(A)` (function)

Inputs an array A and returns the sum of its entries.
EXAMPLES:

35.1.6 ArrayValue

▷ `ArrayValue(A, x)` (function)

Inputs an array A and a coordinate vector x. It returns the value of the entry in A with coordinate x.
EXAMPLES:

35.1.7 ArrayValueFunctions

▷ `ArrayValueFunctions(d)` (function)

Inputs a positive integer d and returns an efficient version of the function ArrayValue for arrays of dimension d.
EXAMPLES:

35.1.8 ArrayAssign

▷ `ArrayAssign(A, x, n)` (function)

Inputs an array A and a coordinate vector x and an integer n. It sets the entry of A with coordinate x equal to n.
EXAMPLES:

35.1.9 ArrayAssignFunctions

▷ `ArrayAssignFunctions(d)` (function)

Inputs a positive integer d and returns an efficient version of the function ArrayAssign for arrays of dimension d.
EXAMPLES:

35.1.10 ArrayIterate

▷ `ArrayIterate(d)` (function)

Inputs a positive integer d and returns a function ArrayIt(Dimensions,f). This function inputs a list Dimensions of d positive integers and also a function $f(x)$. It applies the function $f(x)$ to each integer list x of length d with entries $x[i]$ in the range [1..Dimension[i]].
EXAMPLES:

35.1.11 BinaryArrayToTextFile

▷ `BinaryArrayToTextFile(file, A)` (function)

Inputs a string containing the address of a file, and an array A of 0s and 1s. The array represents a pure cubical complex. A representation of this complex is written to the file in a format that can be read by the CAPD (Computer Assisted Proofs in Dynamics) software developed by Marian Mrozek and others.

The second input A can also be a pure cubical complex.

EXAMPLES:

35.1.12 FrameArray

▷ `FrameArray(A)` (function)

Inputs an array A and returns the array obtained by appending a 0 to the beginning and end of each "row" of the array.

EXAMPLES: 1

35.1.13 UnframeArray

▷ `UnframeArray(A)` (function)

Inputs an array A and returns the array obtained by removing the first and last entry in each "row" of the array.

EXAMPLES:

35.1.14 Add

▷ `Add(L, x)` (function)

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 converts L into a pseudo list of length n+1 by adding x as the final entry. If x is in L the operation has no effect on L.

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10

35.1.15 Append

▷ `Append(L, K)` (function)

Let L be a pseudo list and K a list whose objects are compatible with those in L. This operation applies Add(L,x) for each x in K.

EXAMPLES: 1 , 2 , 3 , 4 , 5

35.1.16 ListToPseudoList

▷ `ListToPseudoList(L)` (function)

Inputs a list L and returns the pseudo list representation of L.
EXAMPLES:

Chapter 36

Parallel Computation - Core Functions

36.1

36.1.1 ChildProcess

▷ `ChildProcess()` (function)
▷ `ChildProcess(arg)` (function)

This starts a GAP session as a child process and returns a stream to the child process. If no argument is given then the child process is created on the local machine; otherwise the argument should be:
1) *arg*="computer.ac.wales" the address of a remote computer for which ssh has been configured to require no password from the user;
(2) *arg*=["-m", "100000M", "-T"] a list of GAP command line options;
(3) *arg*="computer.ac.wales", ["-m", "100000M", "-T"] the address of a computer followed by a list of command line options.

(To configure ssh so that the user can login without a password prompt from "thishost" to "remotehost" either consult "man ssh" or

- open a shell on thishost
- cd .ssh
- ls
-> if id_dsa, id_rsa etc exists, skip the next two steps!
- ssh-keygen -t rsa
- ssh-keygen -t dsa
- scp *.pub userremotehost:~/
- ssh remotehost -l user
- cat id_rsa.pub >> .ssh/authorized_keys
- cat id_dsa.pub >> .ssh/authorized_keys
- rm id_rsa.pub id_dsa.pub
- exit

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

EXAMPLES: 1 , 2

36.1.2 ChildClose

▷ `ChildClose(s)` (function)

This closes the stream s to a child GAP process.
EXAMPLES: 1

36.1.3 ChildCommand

▷ `ChildCommand(str, s)` (function)

This runs a GAP command *str*="cmd;" on the child process accessed by the stream s. Here "cmd;" is a string representing the command.
EXAMPLES: 1 , 2

36.1.4 NextAvailableChild

▷ `NextAvailableChild(L)` (function)

Inputs a list *L* of child processes and returns a child in *L* which is ready for computation (as soon as such a child is available).
EXAMPLES: 1 , 2

36.1.5 IsAvailableChild

▷ `IsAvailableChild(s)` (function)

Inputs a child process *s* and returns true if s is currently available for computations, and false otherwise.
EXAMPLES: 1

36.1.6 ChildPut

▷ `ChildPut(A, str, s)` (function)

This copies a GAP object A on the parent process to an object B=*str* on the child process s. (The copying relies on the function PrintObj(A);)
EXAMPLES: 1 , 2

36.1.7 ChildGet

▷ `ChildGet(str, s)` (function)

This functions copies a GAP object A="str" on the child process s and returns it on the parent process. (The copying relies on the function PrintObj(A);)
EXAMPLES: 1 , 2

36.1.8 HAPPrintTo

▷ HAPPrintTo(*str*, *R*) (function)

Inputs a string *str*="file" giving the address of a new text file and a HAP object R. It writes the object R to "file". Currently this is only implemented for R equal to a resolution.

EXAMPLES: 1

36.1.9 HAPRead

▷ HAPRead(*str*, *R*) (function)

Inputs an address *str*="file" of a file containing a HAP object R and returns the object. Currently this is only implemented for R equal to a resolution.

EXAMPLES: 1

Chapter 37

Parallel Computation - Extra Functions

37.1

37.1.1 ChildFunction

▷ `ChildFunction(str, s)` (function)

This runs the GAP function *str*="function(arg);" on a child process accessed by the stream s. The output from "func;" can be accessed via the stream.

EXAMPLES:

37.1.2 ChildRead

▷ `ChildRead(s)` (function)

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

EXAMPLES:

37.1.3 ChildReadEval

▷ `ChildReadEval(s)` (function)

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

EXAMPLES:

37.1.4 ParallelList

▷ `ParallelList(I, fn, L)` (function)

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 uses the children in L to compute $List(I,x->fn(x))$. (Obviously the function fn must be defined on all child processes in L.)

EXAMPLES: 1 , 2

Chapter 38

Some functions for accessing basic data

38.1

38.1.1 BoundaryMap

▷ `BoundaryMap(C)` (function)

Inputs a resolution, chain complex or cochain complex C and returns the function C!.*boundary*.
EXAMPLES: 1 , 2 , 3

38.1.2 BoundaryMatrix

▷ `BoundaryMatrix(C, n)` (function)

Inputs a chain or cochain complex C and integer n>0. It returns the n-th boundary map of C as a matrix.
EXAMPLES: 1

38.1.3 Dimension

▷ `Dimension(C)` (function)
▷ `Dimension(M)` (function)

Inputs a resolution, chain complex or cochain complex C and returns the function C!.*dimension* .
Alternatively, inputs an FpG-module M and returns its dimension as a vector space over the field of p elements.
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

38.1.4 EvaluateProperty

▷ `EvaluateProperty(X, str)` (function)

Inputs a component object X (such as a ZG-resolution or chain map) and a string *str*="name" (such as "characteristic" or "type"). It searches X.*property* for the pair ["name",value] and returns value. If X.*property* does not exist, or if ["name",value] does not exist, it returns fail.

EXAMPLES:

38.1.5 GroupOfResolution

▷ `GroupOfResolution(R)` (function)

Inputs a ZG-resolution R and returns the group G.
EXAMPLES:

38.1.6 Length

▷ `Length(R)` (function)

Inputs a resolution R and returns its length (i.e. the number of terms of R that HAP has computed).
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19

38.1.7 Map

▷ `Map(f)` (function)

Inputs a chain map, or cochain map or equivariant chain map f and returns the mapping function (as opposed to the target or the source of f) .
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17

38.1.8 Source

▷ `Source(f)` (function)

Inputs a chain map, or cochain map, or equivariant chain map, or FpG-module homomorphism f and returns it source.
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12

38.1.9 Target

▷ `Target(f)` (function)

Inputs a chain map, or cochain map, or equivariant chain map, or FpG-module homomorphism f and returns its target.
EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

Chapter 39

Miscellaneous

39.1

39.1.1 SL2Z

▷ `SL2Z(p)` (function)
▷ `SL2Z(1/m)` (function)

Inputs a prime p or the reciprocal $1/m$ of a square free integer m. In the first case the function returns the conjugate $SL(2,Z)^P$ of the special linear group $SL(2,Z)$ by the matrix $P = [[1,0],[0,p]]$. In the second case it returns the group $SL(2,Z[1/m])$.

EXAMPLES: 1 , 2 , 3 , 4

39.1.2 BigStepLCS

▷ `BigStepLCS(G, n)` (function)

Inputs a group G and a positive integer n. It returns a subseries $G = L_1 > L_2 > \ldots L_k = 1$ of the lower central series of G such that L_i/L_{i+1} has order greater than n.

EXAMPLES: 1 , 2

39.1.3 Classify

▷ `Classify(L, Inv)` (function)

Inputs a list of objects L and a function Inv which computes an invariant of each object. It returns a list of lists which classifies the objects of L according to the invariant..

EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10

39.1.4 RefineClassification

▷ `RefineClassification(C, Inv)` (function)

Inputs a list $C := Classify(L, OldInv)$ and returns a refined classification according to the invariant Inv.

EXAMPLES: 1 , 2 , 3 , 4

39.1.5 Compose

▷ `Compose(f, g)` (function)

Inputs two FpG-module homomorphisms $f : M \longrightarrow N$ and $g : L \longrightarrow M$ with $Source(f) = Target(g)$. It returns the composite homomorphism $fg : L \longrightarrow N$.

This also applies to group homomorphisms f, g.

EXAMPLES: 1

39.1.6 HAPcopyright

▷ `HAPcopyright()` (function)

This function provides details of HAP'S GNU public copyright licence.

EXAMPLES:

39.1.7 IsLieAlgebraHomomorphism

▷ `IsLieAlgebraHomomorphism(f)` (function)

Inputs an object f and returns true if f is a homomorphism $f : A \longrightarrow B$ of Lie algebras (preserving the Lie bracket).

EXAMPLES:

39.1.8 IsSuperperfect

▷ `IsSuperperfect(G)` (function)

Inputs a group G and returns "true" if both the first and second integral homology of G is trivial. Otherwise, it returns "false".

EXAMPLES:

39.1.9 MakeHAPManual

▷ `MakeHAPManual()` (function)

This function creates the manual for HAP from an XML file.

EXAMPLES:

39.1.10 PermToMatrixGroup

▷ `PermToMatrixGroup(G, n)` (function)

Inputs a permutation group G and its degree n. Returns a bijective homomorphism $f : G \longrightarrow M$ where M is a group of permutation matrices.

EXAMPLES: 1 , 2

39.1.11 SolutionsMatDestructive

▷ `SolutionsMatDestructive(M, B)` (function)

Inputs an $m \times n$ matrix M and a $k \times n$ matrix B over a field. It returns a k×m matrix S satisfying $SM = B$.

The function will leave matrix M unchanged but will probably change matrix B.

(This is a trivial rewrite of the standard GAP function *SolutionMatDestructive*(<*mat*>,<*vec*>) .)

EXAMPLES:

39.1.12 LinearHomomorphismsPersistenceMat

▷ `LinearHomomorphismsPersistenceMat(L)` (function)

Inputs a composable sequence L of vector space homomorphisms. It returns an integer matrix containing the dimensions of the images of the various composites. The sequence L is determined up to isomorphism by this matrix.

EXAMPLES:

39.1.13 NormalSeriesToQuotientHomomorphisms

▷ `NormalSeriesToQuotientHomomorphisms(L)` (function)

Inputs an (increasing or decreasing) chain L of normal subgroups in some group G. This G is the largest group in the chain. It returns the sequence of composable group homomorphisms $G/L[i] \to G/L[i+/-1]$.

EXAMPLES:

39.1.14 TestHap

▷ `TestHap()` (function)

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

EXAMPLES:

Chapter 40

HAP variables that are not yet documented

40.1

2CoreducedChainComplex EXAMPLES:

2x2matrix EXAMPLES:

AbelianGOuterGroupToCatOneGroup EXAMPLES:

AbelianInvariantsToTorsionCoefficients EXAMPLES:

AcyclicSubcomplexOfPureCubicalComplex EXAMPLES: 1

AddFirst EXAMPLES:

AdjointGroupOfQuandle EXAMPLES: 1

AlgebraicReduction_alt EXAMPLES:

AmbientTreeDisplay EXAMPLES:

AppendFreeWord EXAMPLES:

ArcDiagramToTubularSurface EXAMPLES:

ArcPresentation EXAMPLES: 1 , 2 , 3 , 4

ArcPresentationToKnottedOneComplex EXAMPLES:

AreIsoclinic EXAMPLES:

AreStrictlyFundamentalCoordinates EXAMPLES:

ArrayIterateBreak EXAMPLES:

ArrayValueKD EXAMPLES:

AsWordInSL2Z EXAMPLES: 1

AutomorphismGroupQuandleAsPerm_nonconnected EXAMPLES:

AverageInnerProduct EXAMPLES:

BarCodeOfFilteredPureCubicalComplex EXAMPLES:

BarCodeOfSymmetricMatrix EXAMPLES:

BarComplexOfMonoid EXAMPLES: 1

BarycentricallySimplifiedComplex EXAMPLES: 1

BarycentricallySubdivideCell EXAMPLES:

BettinumbersOfPureCubicalComplex_dim_2 EXAMPLES:

BianchiGcomplex EXAMPLES: 1

BianchiPolyhedron EXAMPLES: 1

BigStepUCS EXAMPLES: 1

BocksteinHomology EXAMPLES:

BogomolovMultiplier_viaTensorSquare EXAMPLES:

BoundariesOfFilteredChainComplex EXAMPLES:

BoundaryOfPureComplex EXAMPLES: 1

BoundaryOfPureRegularCWComplex EXAMPLES: 1

BoundaryOfRegularCWCell EXAMPLES:

BoundaryPairOfPureRegularCWComplex EXAMPLES:

BoundingPureComplex EXAMPLES:

CR_ChainMapFromCocycle EXAMPLES:

CR_CocyclesAndCoboundaries EXAMPLES:

CR_IntegralClassToCocycle EXAMPLES:

CR_IntegralCocycleToClass EXAMPLES:

CR_IntegralCohomology EXAMPLES:

CR_IntegralCycleToClass EXAMPLES:

CWMap2ChainMap EXAMPLES:

CWSubcomplexToRegularCWMap EXAMPLES: 1

CanonicalRightCountableCosetElement EXAMPLES:

CatOneGroupByCrossedModule EXAMPLES: 1

CatOneGroupsByGroup EXAMPLES: 1

CcElement EXAMPLES:

Cedric_CheckThirdAxiomRow EXAMPLES:

Cedric_ConjugateQuandleElement EXAMPLES:

Cedric_FromAutGeReToAutQe EXAMPLES:

Cedric_IsHomomorphism EXAMPLES:

Cedric_Permute EXAMPLES:

Cedric_Quandle1 EXAMPLES:

Cedric_Quandle2 EXAMPLES:

Cedric_Quandle3 EXAMPLES:

Cedric_Quandle4 EXAMPLES:

Cedric_Quandle5 EXAMPLES:

Cedric_Quandle6 EXAMPLES:

CellComplexBoundaryCheck EXAMPLES:

ChainComplexEquivalenceOfRegularCWComplex EXAMPLES: 1

ChainComplexHomeomorphismEquivalenceOfRegularCWComplex EXAMPLES:

ChainComplexOfCubicalComplex EXAMPLES:

ChainComplexOfCubicalPair EXAMPLES:

ChainComplexOfRegularCWComplexWithVectorField EXAMPLES:

ChainComplexOfSimplicialComplex EXAMPLES:

ChainComplexOfSimplicialPair EXAMPLES:

ChainComplexOfUniversalCover EXAMPLES: 1 , 2 , 3 , 4

ChainComplexToSparseChainComplex EXAMPLES:

ChainComplexWithChainHomotopy EXAMPLES:

ChainMapOfCubicalPairs EXAMPLES:

ChainMapOfRegularCWMap EXAMPLES:

ChevalleyEilenbergComplexOfModule EXAMPLES:

ChildRestart EXAMPLES:

ChildTransfer EXAMPLES:

ClassifyingSpaceFiniteGroup EXAMPLES: 1

ClosureCWCell EXAMPLES:

CoClass EXAMPLES:

CocriticalCellsOfRegularCWComplex EXAMPLES:

CocyclicHadamardMatrices EXAMPLES: 1

CocyclicMatrices EXAMPLES:

CohomologicalData EXAMPLES: 1

CohomologyHomomorphism EXAMPLES: 1 , 2 , 3

CohomologyHomomorphismOfRepresentation EXAMPLES:

CohomologyModule_AsAutModule EXAMPLES:

CohomologyModule_Gmap EXAMPLES:

CohomologyRingOfSimplicialComplex EXAMPLES:

CohomologySimplicialFreeAbelianGroup EXAMPLES:

CombinationDisjointSets EXAMPLES:

CommonEndomorphisms EXAMPLES:

ComplementOfPureComplex EXAMPLES: 1

ComplementaryBasis EXAMPLES:

ComposeCWMaps EXAMPLES:

CompositionOfFpGModuleHomomorphisms EXAMPLES:

CompositionSeriesOfFpGModule EXAMPLES:

ConcentricallyFilteredPureCubicalComplex EXAMPLES: 1

ConjugateSL2ZGroup EXAMPLES:

ConnectingCohomologyHomomorphism EXAMPLES: 1 , 2

ContractArray EXAMPLES:

ContractCubicalComplex_dim2 EXAMPLES:

ContractCubicalComplex_dim3 EXAMPLES:

ContractMatrix EXAMPLES:

ContractPermArray EXAMPLES:

ContractPermMatrix EXAMPLES:

ContractPureComplex EXAMPLES:

ContractSimplicialComplex EXAMPLES:

ContractSimplicialComplex_alt EXAMPLES:

ContractedComplexViaChild EXAMPLES: 1

ContractedFilteredPureCubicalComplex EXAMPLES: 1

ContractedFilteredRegularCWComplex EXAMPLES:

ContractedRegularCWComplex EXAMPLES:

ContractibleSL2ZComplex EXAMPLES:

ContractibleSL2ZComplex_alt EXAMPLES:

ContractibleSubArray EXAMPLES:

ContractibleSubMatrix EXAMPLES:

ContractibleSubcomplexOfPureCubicalComplex EXAMPLES: 1

ConvertTorsionComplexToGcomplex EXAMPLES:

CosetsQuandle EXAMPLES:

CountingCellsOfBaryCentricSubdivision EXAMPLES:

CountingNumberOfCellsInBaryCentricSubdivision EXAMPLES:

CoverOfUnimodularPairs EXAMPLES:

CoxeterComplex_alt EXAMPLES: 1

CoxeterDiagramMatCoxeterGroup EXAMPLES: 1

CoxeterWythoffComplex EXAMPLES:

CreateCoxeterMatrix EXAMPLES: 1

CriticalBoundaryCells EXAMPLES: 1

CropPureComplex EXAMPLES:

CrossedInvariant EXAMPLES:

CrossedModuleByAutomorphismGroup EXAMPLES:

`CrossedModuleByCatOneGroup` EXAMPLES:

`CrossedModuleByNormalSubgroup` EXAMPLES: 1

`CrystCubicalTiling` EXAMPLES:

`CrystFinitePartOfMatrix` EXAMPLES:

`CrystGFullBasis` EXAMPLES: 1 , 2 , 3

`CrystGcomplex` EXAMPLES: 1 , 2 , 3

`CrystMatrix` EXAMPLES:

`CrystTranslationMatrixToVector` EXAMPLES:

`CrystallographicComplex` EXAMPLES: 1

`CubicalToPermutahedralArray` EXAMPLES:

`CupProductMatrix` EXAMPLES:

`CupProductOfRegularCWComplex` EXAMPLES: 1

`CupProductOfRegularCWComplexModP` EXAMPLES:

`CupProductOfRegularCWComplex_alt` EXAMPLES: 1

`CyclesOfFilteredChainComplex` EXAMPLES:

`DavisComplex` EXAMPLES: 1 , 2 , 3 , 4

`DeformationRetract` EXAMPLES:

`DensityMat` EXAMPLES:

`DerivedGroupOfQuandle` EXAMPLES: 1

`DiagonalChainMap` EXAMPLES: 1

`DijkgraafWittenInvariant` EXAMPLES: 1

`DirectProductOfGroupHomomorphisms` EXAMPLES:

`DirectProductOfRegularCWComplexes` EXAMPLES:

`DirectProductOfRegularCWComplexesLazy` EXAMPLES:

`DirectProductOfSimplicialComplexes` EXAMPLES:

`Display3DUnimodularPairs` EXAMPLES:

`Display3DUnimodularPairs_extra` EXAMPLES:

`DisplayCSVknotFile` EXAMPLES:

`DisplayUnimodularPairs` EXAMPLES:

`DisplayVectorField` EXAMPLES:

`DualComplex_FreeResolution` EXAMPLES:

`DualComplex_NonFreeResolution` EXAMPLES:

`E1CohomologyPage` EXAMPLES:

`E1HomologyPage` EXAMPLES:

`EilenbergMacLaneSimplicialFreeAbelianGroup` EXAMPLES: 1

`ElementsLazy` EXAMPLES:

`EquivariantCWComplexToRegularCWComplex` EXAMPLES: 1 , 2 , 3 , 4

`EquivariantCWComplexToRegularCWMap` EXAMPLES: 1 , 2 , 3

`EquivariantCWComplexToResolution` EXAMPLES:

`ExcisedPureCubicalPair_dim_2` EXAMPLES:

`ExtractTorsionSubcomplex` EXAMPLES:

`FactorizationNParts` EXAMPLES:

`FilteredChainComplexToFilteredSparseChainComplex` EXAMPLES:

`FilteredCubicalComplexToFilteredRegularCWComplex` EXAMPLES: 1

`FilteredPureCubicalComplex` EXAMPLES: 1 , 2

`FilteredPureCubicalComplexToCubicalComplex` EXAMPLES: 1

FiltrationTermOfGraph EXAMPLES:

FiltrationTermOfPureCubicalComplex EXAMPLES:

FiltrationTermOfRegularCWComplex EXAMPLES:

FiltrationTerms EXAMPLES: 1

FirstHomologyCoveringCokernels EXAMPLES: 1 , 2

FirstHomologySimplicialTwoComplex EXAMPLES:

FourthHomotopyGroupOfDoubleSuspensionB EXAMPLES: 1

Fp2PcpAbelianGroupHomomorphism EXAMPLES:

FpGModuleSection EXAMPLES:

FreeZGResolution EXAMPLES:

FundamentalGroupOfRegularCWComplex EXAMPLES: 1

FundamentalGroupOfRegularCWMap EXAMPLES:

FundamentalGroupSimplicialTwoComplex EXAMPLES:

FundamentalMultiplesOfStiefelWhitneyClasses EXAMPLES:

GChainComplex EXAMPLES: 1

GComplexToFiniteRegularCWRegion EXAMPLES: 1

GComplexToRegularCWComplex EXAMPLES: 1 , 2 , 3 , 4

GModuleAsCatOneGroup EXAMPLES:

Gamma0_SL3ZTopRationalHomology EXAMPLES:

GammaSubgroupInSL3Z EXAMPLES:

GaussCodeOfPureCubicalKnot EXAMPLES: 1 , 2 , 3 , 4

GetTorsionPowerSubcomplex EXAMPLES:

GetTorsionSubcomplex EXAMPLES:

GraphOfRegularCWComplex EXAMPLES: 1 , 2 , 3 , 4

GraphOfResolutionsTest EXAMPLES:

GraphOfResolutionsToGroups EXAMPLES:

GroupHomomorphismToMatrix EXAMPLES:

HAPCocontractRegularCWComplex EXAMPLES:

HAPContractFilteredRegularCWComplex EXAMPLES:

HAPContractRegularCWComplex EXAMPLES:

HAPContractRegularCWComplex_Alt EXAMPLES:

HAPPRIME_Algebra2Polynomial EXAMPLES:

HAPPRIME_CohomologyRingWithoutResolution EXAMPLES:

HAPPRIME_CombineIndeterminateMaps EXAMPLES:

HAPPRIME_GradedAlgebraPresentationAvoidingIndeterminates EXAMPLES:

HAPPRIME_LHSSpectralSequence EXAMPLES:

HAPPRIME_MakeEliminationOrdering EXAMPLES:

HAPPRIME_MapPolynomialIndeterminates EXAMPLES:

HAPPRIME_Polynomial2Algebra EXAMPLES:

HAPPRIME_RingHomomorphismsAreComposable EXAMPLES:

HAPPRIME_SModule EXAMPLES:

HAPPRIME_SingularGroebnerBasis EXAMPLES:

HAPPRIME_SingularReducedGroebnerBasis EXAMPLES:

HAPPRIME_SwitchGradedAlgebraRing EXAMPLES:

HAPPRIME_SwitchPolynomialIndeterminates EXAMPLES:

HAPPRIME_VersionWithSVN EXAMPLES:

HAPQuadraticRing EXAMPLES:

HAPRegularCWComplex EXAMPLES:

HAPRegularCWPolytope EXAMPLES:

HAPRemoveCellFromRegularCWComplex EXAMPLES:

HAPRemoveVectorField EXAMPLES:

HAPRingModIdeal EXAMPLES:

HAPRingModIdealObj EXAMPLES:

HAPTietzeReduction_Inf EXAMPLES:

HAPTietzeReduction_OneLevel EXAMPLES:

HAPTietzeReduction_OneStep EXAMPLES:

HAP_4x4MatTo2x2Mat EXAMPLES:

HAP_AddGenerator EXAMPLES:

HAP_AllHomomorphisms EXAMPLES:

HAP_AppendTo EXAMPLES:

HAP_Are3IntersectingUnimodularPairs EXAMPLES:

HAP_AreIntersectingUnimodularPairs EXAMPLES:

HAP_AreStrictlyIntersectingUnimodularPairs EXAMPLES:

HAP_AssociahedronBoundaries EXAMPLES:

HAP_AssociahedronCells EXAMPLES:

HAP_BarCodeCompactDisplayList EXAMPLES:

HAP_BaryCentricSubdivisionGComplex EXAMPLES:

HAP_BaryCentricSubdivisionRegularCWComplex EXAMPLES:

HAP_BettiZeroMonotonic EXAMPLES:

HAP_BianchiFundamentalRectangle EXAMPLES:

HAP_BianchiRegularCWComplex EXAMPLES:

HAP_BianchiTransformations EXAMPLES:

HAP_Binlisttoint EXAMPLES:

HAP_ChainComplexToEquivariantChainComplex EXAMPLES:

HAP_CocyclesAndCoboundaries EXAMPLES:

HAP_CocyclesAndCoboundariesModP EXAMPLES:

HAP_CongruenceSubgroupGamma0 EXAMPLES: 1 , 2 , 3

HAP_CongruenceSubgroupGamma0Ideal EXAMPLES:

HAP_ConjugatedCongruenceSubgroup EXAMPLES:

HAP_ConjugatedCongruenceSubgroupGamma0 EXAMPLES:

HAP_CriticalCellsDirected EXAMPLES:

HAP_CupProductOfPresentation EXAMPLES:

HAP_CupProductOfSimplicialComplex EXAMPLES:

HAP_DisplayPlanarTree EXAMPLES:

HAP_DisplayVectorField EXAMPLES:

HAP_ElementsSL2Zfn EXAMPLES:

HAP_FiniteProjectiveLineIntegers EXAMPLES:

HAP_FiniteProjectiveLineIntegers_alt EXAMPLES:

HAP_FiniteProjectivePlaneIntegers EXAMPLES:

HAP_FunctorialModPCohomologyRing EXAMPLES:

HAP_GenericCongruenceSubgroup EXAMPLES:

HAP_GenericSL2OSubgroup EXAMPLES:

HAP_GenericSL2ZConjugatedSubgroup EXAMPLES:

HAP_GenericSL2ZSubgroup EXAMPLES:

HAP_HeightOfPointOnSphere EXAMPLES:

HAP_HomToIntModP_ChainComplex EXAMPLES:

HAP_HomToIntModP_ChainMap EXAMPLES:

HAP_HomToIntModP_CochainComplex EXAMPLES:

HAP_HomToIntModP_CochainMap EXAMPLES:

HAP_HomeoLinkingForm EXAMPLES:

HAP_Hurewicz1Cycles EXAMPLES:

HAP_IntegralClassToCocycle EXAMPLES:

HAP_IntegralCocycleToClass EXAMPLES:

HAP_IntegralCohomology EXAMPLES:

HAP_IntersectionConjugatedCongruenceSubgroup EXAMPLES:

HAP_InverseOpFunction EXAMPLES:

HAP_IsRedundantUnimodularPair EXAMPLES:

HAP_IsomorphismCcFpGroup EXAMPLES:

HAP_KK_AddCell EXAMPLES:

HAP_KnotGroupInv EXAMPLES:

HAP_MultiplicationTableOfGroup EXAMPLES:

HAP_MyIsBieberbachFpGroup EXAMPLES:

HAP_MyIsFiniteFpGroup EXAMPLES:

HAP_MyIsInfiniteFpGroup EXAMPLES:

HAP_PHI EXAMPLES:

HAP_PermBinlisttoint EXAMPLES:

HAP_PlanarBinaryTrees EXAMPLES:

HAP_PlanarTreeGraft EXAMPLES:

HAP_PlanarTreeJoin EXAMPLES:

HAP_PlanarTreeLeaves EXAMPLES:

HAP_PlanarTreeRemovableEdge EXAMPLES:

HAP_PlanarTreeRemoveEdge EXAMPLES:

HAP_PrimePartModified EXAMPLES:

HAP_PrincipalCongruenceSubgroup EXAMPLES: 1

HAP_PrincipalCongruenceSubgroupIdeal EXAMPLES:

HAP_PrintFloat EXAMPLES:

HAP_PrintTo EXAMPLES:

HAP_PureComplexSubcomplex EXAMPLES:

HAP_PureCubicalPairToCWMap EXAMPLES:

HAP_ResolutionAbelianGroupFromInvariants EXAMPLES:

HAP_RightTransversalSLnZSubgroups EXAMPLES:

HAP_RightTransversalSLnZSubgroups_alt EXAMPLES:

HAP_SL2OSubgroupTree_fast EXAMPLES:

HAP_SL2OSubgroupTree_slow EXAMPLES:

HAP_SL2SubgroupTree EXAMPLES:

HAP_SL2TreeDisplay EXAMPLES: 1

HAP_SL2ZSubgroupTree_fast EXAMPLES:

HAP_SL2ZSubgroupTree_slow EXAMPLES:

HAP_SL3ZSubgroupTree_fast EXAMPLES:

HAP_Sequence2Boundaries EXAMPLES:

HAP_SimplicialPairToCWMap EXAMPLES:

HAP_SimplicialProjectivePlane EXAMPLES:

HAP_SimplicialTorus EXAMPLES:

HAP_SimplifiedGaussCode EXAMPLES:

HAP_SqrtInequality EXAMPLES:

HAP_SqrtStrictInequality EXAMPLES:

HAP_StiefelWhitney EXAMPLES:

HAP_SylowConjugatedHomomorphism EXAMPLES:

HAP_SylowSubgroups EXAMPLES:

HAP_Tensor EXAMPLES:

HAP_TransversalCongruenceSubgroupInAmbientGroup EXAMPLES:

HAP_TransversalCongruenceSubgroupInAmbientGroupSlow EXAMPLES:

HAP_TransversalCongruenceSubgroups EXAMPLES:

HAP_TransversalCongruenceSubgroupsIdeal EXAMPLES:

HAP_TransversalCongruenceSubgroupsIdeal_alt EXAMPLES:

HAP_TransversalGamma0SubgroupsIdeal EXAMPLES:

HAP_Triangulation EXAMPLES:

HAP_TzPair EXAMPLES:

HAP_UnimodularComplements EXAMPLES:

HAP_VertexHeights EXAMPLES:

HAP_WedgeSumOfSimplicialComplexes EXAMPLES:

HAP_bockstein EXAMPLES:

HAP_chain_bockstein EXAMPLES:

HAP_coho_isoms EXAMPLES:

HAP_nxnMatTo2nx2nMat EXAMPLES:

HadamardGraph EXAMPLES:

HapExample EXAMPLES:

HapFile EXAMPLES: 1 , 2 , 3 , 4

Hap_int EXAMPLES:

HasTrivialPostnikovInvariant EXAMPLES:

HeckeComponent EXAMPLES:

HeckeComponentWeight2 EXAMPLES:

HeckeOperator EXAMPLES: 1

HenonOrbit EXAMPLES: 1

HomToGModule_hom EXAMPLES:

HomToInt_ChainComplex EXAMPLES:

HomToInt_ChainMap EXAMPLES:

HomToInt_CochainComplex EXAMPLES:

HomToModPModule EXAMPLES: 1

HomToModuleOverCyclotomicField EXAMPLES: 1

HomToRationals EXAMPLES: 1

HomogeneousPolynomials EXAMPLES: 1 , 2

HomogeneousPolynomials_Bianchi EXAMPLES:

HomologicalGroupDecomposition EXAMPLES: 1 , 2

`HomologyOfPureCubicalComplex` EXAMPLES:

`HomologyPbs` EXAMPLES:

`HomologySimplicialFreeAbelianGroup` EXAMPLES:

`HomomorphismAsMatrix` EXAMPLES:

`HomotopyCatOneGroup` EXAMPLES:

`HomotopyCrossedModule` EXAMPLES:

`HomotopyEquivalentLargerSubArray` EXAMPLES:

`HomotopyEquivalentLargerSubArray3D` EXAMPLES:

`HomotopyEquivalentLargerSubMatrix` EXAMPLES:

`HomotopyEquivalentLargerSubPermArray` EXAMPLES:

`HomotopyEquivalentLargerSubPermArray3D` EXAMPLES:

`HomotopyEquivalentLargerSubPermMatrix` EXAMPLES:

`HomotopyEquivalentMaximalPureSubcomplex` EXAMPLES:

`HomotopyEquivalentMinimalPureSubcomplex` EXAMPLES:

`HomotopyEquivalentSmallerSubArray` EXAMPLES:

`HomotopyEquivalentSmallerSubArray3D` EXAMPLES:

`HomotopyEquivalentSmallerSubMatrix` EXAMPLES:

`HomotopyEquivalentSmallerSubPermArray` EXAMPLES:

`HomotopyEquivalentSmallerSubPermArray3D` EXAMPLES:

`HomotopyEquivalentSmallerSubPermMatrix` EXAMPLES:

`HomotopyLowerCentralSeries` EXAMPLES:

`HomotopyLowerCentralSeriesOfCrossedModule` EXAMPLES:

`HomotopyTruncation` EXAMPLES:

`HopfSatohSurface` EXAMPLES: 1 , 2

`HybridSubdivision` EXAMPLES:

`IdCatOneGroup` EXAMPLES: 1 , 2

`IdCrossedModule` EXAMPLES: 1

`IdQuasiCatOneGroup` EXAMPLES: 1

`IdQuasiCrossedModule` EXAMPLES:

`IdentifyKnot` EXAMPLES: 1

`IdentityAmongRelators` EXAMPLES: 1 , 2 , 3

`ImageOfGOuterGroupHomomorphism` EXAMPLES: 1 , 2

`ImageOfMap` EXAMPLES:

`InducedSteenrodHomomorphisms` EXAMPLES:

`IntegerSimplicialComplex` EXAMPLES: 1

`IntegralCellularHomology` EXAMPLES:

`IntegralCohomology` EXAMPLES:

`IntegralCohomologyOfCochainComplex` EXAMPLES:

`IntegralHomology` EXAMPLES: 1

`IntegralHomologyOfChainComplex` EXAMPLES:

`InteriorCohomologyHomomorphism` EXAMPLES: 1

`IntersectionCWSubcomplex` EXAMPLES:

`IsClosedManifold` EXAMPLES: 1

`IsContractibleCube_higherdims` EXAMPLES:

`IsCrystSameOrbit` EXAMPLES:

`IsCrystSufficientLattice` EXAMPLES:

IsHadamardMatrix EXAMPLES:

IsHapQuadraticInteger EXAMPLES:

IsIntList EXAMPLES:

IsIsomorphismOfAbelianFpGroups EXAMPLES: 1

IsMetricMatrix EXAMPLES:

IsPeriodicSpaceGroup EXAMPLES: 1 , 2

IsPureComplex EXAMPLES:

IsPureRegularCWComplex EXAMPLES:

IsQQUnimodularPair EXAMPLES:

IsQUnimodularPair EXAMPLES:

IsRigid EXAMPLES: 1

IsRigidOnRight EXAMPLES:

IsSphericalCoxeterGroup EXAMPLES:

IsStrictlyFundamentalUnimodularPair EXAMPLES:

IsUnimodularCollection EXAMPLES:

IsoclinismClasses EXAMPLES: 1 , 2

IsomorphismCatOneGroups EXAMPLES: 1 , 2

IsomorphismCrossedModules EXAMPLES:

KernelOfGOuterGroupHomomorphism EXAMPLES: 1 , 2

KernelOfMap EXAMPLES:

KernelWG EXAMPLES:

KinkArc2Presentation EXAMPLES:

KnotComplement EXAMPLES: 1 , 2 , 3

`KnotComplementWithBoundary` EXAMPLES: 1 , 2 , 3

`LazyList` EXAMPLES:

`LefschetzNumberOfChainMap` EXAMPLES:

`Lfunction` EXAMPLES: 1

`LiftColouredSurface` EXAMPLES:

`LiftedRegularCWMap` EXAMPLES:

`LinearHomomorphismsZZPersistenceMat` EXAMPLES:

`LinkingForm` EXAMPLES: 1

`LinkingFormHomeomorphismInvariant` EXAMPLES: 1

`LinkingFormHomotopyInvariant` EXAMPLES: 1

`ListsOfCellsToRegularCWComplex` EXAMPLES:

`LowDimensionalCupProduct` EXAMPLES: 1

`MakeHAPprimeDoc` EXAMPLES:

`ManifoldType` EXAMPLES: 1

`Mapper` EXAMPLES: 1

`Mapper_alt` EXAMPLES:

`MatrixFromWord` EXAMPLES: 1

`MatrixSize` EXAMPLES:

`MaximalSimplicesOfSimplicialComplex` EXAMPLES: 1

`MaximalSphericalCoxeterSubgroupsFromAbove` EXAMPLES:

`MinimizeRingRelations` EXAMPLES:

`Mod2SteenrodAlgebra` EXAMPLES: 1

`ModPCohomologyPresentationBounds` EXAMPLES: 1

ModPCohomologyRing_alt EXAMPLES:

ModPCohomologyRing_part_1 EXAMPLES:

ModPCohomologyRing_part_2 EXAMPLES:

ModPRingGeneratorsAlt EXAMPLES:

ModPSteenrodAlgebra EXAMPLES: 1 , 2

ModularCohomology EXAMPLES:

ModularEquivariantChainMap EXAMPLES:

ModularHomology EXAMPLES:

NeighbourhoodOfUnimodularPairs EXAMPLES:

Nil3TensorSquare EXAMPLES:

NonFreeResolutionFiniteSubgroup EXAMPLES: 1

NonManifoldVertices EXAMPLES:

NonRegularCWBoundary EXAMPLES:

NonabelianSymmetricKernel_alt EXAMPLES: 1

NonabelianSymmetricSquare_inf EXAMPLES:

NonabelianTensorProduct_Inf EXAMPLES:

NonabelianTensorProduct_alt EXAMPLES:

NonabelianTensorSquareAsCatOneGroup EXAMPLES:

NonabelianTensorSquareAsCrossedModule EXAMPLES: 1

NonabelianTensorSquare_inf EXAMPLES:

NoncrossingPartitionsLatticeDisplay EXAMPLES: 1

NullspaceSparseMatDestructive EXAMPLES:

NumberConnectedQuandles EXAMPLES:

NumberGeneratorsOfGroupHomology EXAMPLES:

NumberOfCrossingsInArc2Presentation EXAMPLES:

NumberOfHomomorphisms_connected EXAMPLES:

NumberOfHomomorphisms_groups EXAMPLES:

NumberOfPrimeKnots EXAMPLES: 1 , 2

NumberSmallCatOneGroups EXAMPLES:

NumberSmallCrossedModules EXAMPLES:

NumberSmallQuasiCatOneGroups EXAMPLES:

NumberSmallQuasiCrossedModules EXAMPLES:

OppositeGroup EXAMPLES:

OrthogonalizeBasisByAverageInnerProduct EXAMPLES:

PCentre EXAMPLES:

PSubgroupGChainComplex EXAMPLES:

PSubgroupSimplicialComplex EXAMPLES:

PUpperCentralSeries EXAMPLES:

ParallelPersistentBettiNumbers EXAMPLES:

PartialIsoclinismClasses EXAMPLES: 1

PartsOfQuadraticInteger EXAMPLES:

PathComponentOfPureComplex EXAMPLES: 1

PathComponentsCWSubcomplex EXAMPLES:

PathComponentsOfSimplicialComplex_alt EXAMPLES:

PathObjectForChainComplex EXAMPLES: 1

Pause EXAMPLES:

`PermutahedralComplexToRegularCWComplex` EXAMPLES: 1

`PermutahedralToCubicalArray` EXAMPLES:

`PersistentBettiNumbersViaContractions` EXAMPLES:

`PersistentHomologyOfCrossedModule` EXAMPLES:

`PersistentHomologyOfFilteredPureCubicalComplex_alt` EXAMPLES:

`PersistentHomologyOfFilteredSparseChainComplex` EXAMPLES: 1 , 2

`PersistentHomologyOfPureCubicalComplex_Alt` EXAMPLES:

`PersistentHomologyOfQuotientGroupSeries_Int` EXAMPLES:

`PiZeroOfRegularCWComplex` EXAMPLES:

`PoincareBipyramidCWComplex` EXAMPLES: 1

`PoincareCubeCWComplex` EXAMPLES: 1

`PoincareCubeCWComplexNS` EXAMPLES: 1

`PoincareDodecahedronCWComplex` EXAMPLES: 1 , 2

`PoincareOctahedronCWComplex` EXAMPLES: 1

`PoincarePrismCWComplex` EXAMPLES: 1

`PoincareSeriesApproximation` EXAMPLES:

`PoincareSeries_alt` EXAMPLES:

`PolymakeFaceLattice` EXAMPLES:

`PolytopalRepresentationComplex` EXAMPLES:

`PrankAlt` EXAMPLES:

`PresentationOfResolution_alt` EXAMPLES:

`PrimePartDerivedFunctorHomomorphism` EXAMPLES:

`PrimePartDerivedFunctorViaSubgroupChain` EXAMPLES: 1

PrimePartDerivedTwistedFunctor EXAMPLES:

PrintAlgebraWordAsPolynomial EXAMPLES:

PrintTorsionSubcomplex EXAMPLES:

PureComplex EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

PureCubicalComplexToCubicalComplex EXAMPLES: 1 , 2

PureCubicalLink EXAMPLES: 1 , 2

PushoutOfFpGroups EXAMPLES:

QNeighbourhoodOfUnimodularPairs EXAMPLES:

QQNeighbourhoodOfUnimodularPairs EXAMPLES:

QuadraticCharacter EXAMPLES:

QuadraticIntegersByNorm EXAMPLES: 1

QuadraticNumber EXAMPLES: 1 , 2 , 3

QuadraticNumberConjugate EXAMPLES:

QuadraticNumberField EXAMPLES: 1 , 2 , 3

QuadraticNumberField_NC EXAMPLES:

QuandleIsomorphismRepresentatives EXAMPLES:

QuotientByTorsionSubcomplex EXAMPLES:

QuotientChainMap EXAMPLES:

QuotientGroup EXAMPLES:

QuotientQuasiIsomorph EXAMPLES:

RadicalSeriesOfResolution EXAMPLES:

RandomArc2Presentation EXAMPLES:

RandomCellOfPureComplex EXAMPLES:

ReadLinkImageAsGaussCode EXAMPLES: 1

ReadMatrixAsPureCubicalComplex EXAMPLES:

RecalculateIncidenceNumbers_NonFreeRes EXAMPLES:

ReduceGenerators EXAMPLES:

ReduceGenerators_alt EXAMPLES:

ReflectedCubicalKnot EXAMPLES: 1 , 2 , 3 , 4

RegularCWAssociahedron EXAMPLES: 1

RegularCWCellClosure EXAMPLES:

RegularCWComplexComplement EXAMPLES: 1

RegularCWComplexReordered EXAMPLES:

RegularCWComplexWithRemovedCell EXAMPLES: 1

RegularCWComplex_AttachCellDestructive EXAMPLES: 1

RegularCWCube EXAMPLES: 1

RegularCWMapToCWSubcomplex EXAMPLES:

RegularCWOrbitPolytope EXAMPLES:

RegularCWPermutahedron EXAMPLES: 1

RegularCWPolygon EXAMPLES:

RegularCWSimplex EXAMPLES: 1

RelativeCentralQuotientSpaceGroup EXAMPLES:

RelativeGroupHomology EXAMPLES:

RelativeRightTransversal EXAMPLES:

RemoveStar EXAMPLES:

ResolutionAbelianBianchiSubgroup EXAMPLES:

ResolutionAbelianGroup_alt EXAMPLES:

ResolutionAbelianPcpGroup EXAMPLES:

ResolutionAffineCrystGroup EXAMPLES:

ResolutionArtinGroup_spherical EXAMPLES:

ResolutionBoundaryOfWordOnRight EXAMPLES:

ResolutionDirectProductLazy EXAMPLES:

ResolutionFiniteCcGroup EXAMPLES:

ResolutionFiniteCyclicGroup EXAMPLES:

ResolutionFiniteSubgroup_NonFree EXAMPLES:

ResolutionGL2QuadraticIntegers EXAMPLES: 1

ResolutionGL3QuadraticIntegers EXAMPLES: 1

ResolutionGenericGroup EXAMPLES:

ResolutionInfiniteCcGroup EXAMPLES: 1

ResolutionInfiniteCyclicGroup EXAMPLES:

ResolutionIsomorphismGroup EXAMPLES: 1

ResolutionPGL2QuadraticIntegers EXAMPLES: 1

ResolutionPGL3QuadraticIntegers EXAMPLES: 1

ResolutionPSL2QuadraticIntegers EXAMPLES: 1 , 2

ResolutionPrimePowerGroupSparse EXAMPLES:

ResolutionSL2QuadraticIntegers EXAMPLES: 1 , 2

ResolutionSL2ZConjugated EXAMPLES:

ResolutionSL2Z_alt EXAMPLES: 1

ResolutionSpaceGroup EXAMPLES: 1 , 2

ResolutionToEquivariantCWComplex EXAMPLES:

ResolutionToResolutionOfFpGroup EXAMPLES: 1

SL2QuadraticIntegers EXAMPLES: 1 , 2

SL2ZResolution EXAMPLES:

SL2ZResolution_alt EXAMPLES:

SL2ZTree EXAMPLES:

SL2ZmElementsDecomposition EXAMPLES:

SequentialRegularCWComplexComplement EXAMPLES:

SignatureOfSymmetricMatrix EXAMPLES: 1

SignedPermutationGroup EXAMPLES: 1

SimplicesToSimplicialComplex EXAMPLES: 1 , 2 , 3 , 4

SimplicialComplexOfUnimodularPairs EXAMPLES:

SimplicialComplexToRegularCWComplex_alt EXAMPLES:

SimplicialK3Surface EXAMPLES: 1

SimplicialNerveOfFilteredGraph EXAMPLES: 1 , 2

SimplicialNerveOfTwoComplex EXAMPLES:

SimplifiedQuandlePresentation EXAMPLES:

SimplifiedRegularCWComplex EXAMPLES: 1

SimplifiedSparseChainComplex EXAMPLES:

SmallCatOneGroup EXAMPLES: 1 , 2

SmallCrossedModule EXAMPLES:

SmallQuasiCatOneGroup EXAMPLES:

SmallQuasiCrossedModule EXAMPLES:

`SmoothedFpGroup` EXAMPLES:

`SparseChainComplexOfCubicalComplex` EXAMPLES:

`SparseChainComplexOfCubicalPair` EXAMPLES:

`SparseChainComplexOfFilteredRegularCWComplex` EXAMPLES:

`SparseChainComplexOfRegularCWComplexWithVectorField` EXAMPLES:

`SparseChainComplexOfSimplicialComplex` EXAMPLES:

`SparseChainComplexToChainComplex` EXAMPLES:

`SparseChainMapOfCubicalPairs` EXAMPLES:

`SparseFilteredChainComplexOfFilteredCubicalComplex` EXAMPLES:

`SparseFilteredChainComplexOfFilteredSimplicialComplex` EXAMPLES: 1 , 2

`SparseIdentityMat` EXAMPLES:

`SparseMatAddToEntry` EXAMPLES:

`SparseMatConcatenation` EXAMPLES:

`SparseMattoMat` EXAMPLES: 1

`SparseRowReduce` EXAMPLES:

`SphericalKnotComplement` EXAMPLES: 1

`Spin` EXAMPLES:

`SpunAboutHyperplane` EXAMPLES:

`SpunKnotComplement` EXAMPLES: 1

`SpunLinkComplement` EXAMPLES:

`StrongGeneratorsOfDerivedSubgroup` EXAMPLES:

`StrongGeneratorsOfDerivedSubgroup_alt` EXAMPLES:

`StructuralCopyOfFilteredRegularCWComplex` EXAMPLES:

SubQuasiIsomorph EXAMPLES:

SubdivideCell EXAMPLES:

Suspension_alt EXAMPLES:

SwanBianchiCriterion EXAMPLES: 1

SwanBianchiCriterion_alt EXAMPLES:

SylowSubgroupOfCatOneGroup EXAMPLES:

SymmetricCentre EXAMPLES:

SymmetricCommutativityCommutatorGroup EXAMPLES:

SymmetricCommutativityGroup EXAMPLES:

TensorNonFreeResolutionWithRationals EXAMPLES:

TensorWithBurnsideRing EXAMPLES: 1 , 2

TensorWithComplexRepresentationRing EXAMPLES: 1 , 2

TensorWithComplexRepresentationRingOnRight EXAMPLES:

TensorWithIntegersMod2Torsion EXAMPLES:

TensorWithIntegersModPSparse EXAMPLES:

TensorWithIntegersOverSubgroup EXAMPLES: 1 , 2 , 3 , 4

TensorWithIntegersSparse EXAMPLES: 1

TensorWithModPModule EXAMPLES: 1

TestHapBook EXAMPLES:

TestHapQuick EXAMPLES:

ThickenedHEPureCubicalComplex EXAMPLES:

ThickenedPureComplex EXAMPLES: 1

ThickenedPureCubicalComplex_dim2 EXAMPLES:

ThirdHomotopyGroupOfSuspensionB_alt EXAMPLES: 1

ThreeManifoldViaDehnSurgery EXAMPLES: 1

ThreeManifoldWithBoundary EXAMPLES: 1

TransferChainMap EXAMPLES: 1

TransferCochainMap EXAMPLES: 1

TranslationSubGroup EXAMPLES:

TreeOfResolutionsToSL2Zcomplex EXAMPLES:

TruncatedRegularCWComplex EXAMPLES:

Tube EXAMPLES:

TupleOrbitReps EXAMPLES:

TupleOrbitReps_perm EXAMPLES:

TwistedResolution EXAMPLES:

UnboundedArrayAssign EXAMPLES:

UnimodularIntersectingLine EXAMPLES:

UnimodularIntersectingPoint EXAMPLES:

UnimodularPairCoordinates EXAMPLES:

UnimodularPairStandardForm EXAMPLES:

UnimodularPairs EXAMPLES:

UnimodularPairsReduced EXAMPLES:

UnimodularPairsReduced_NN EXAMPLES:

UnitBall EXAMPLES:

UnitCubicalBall EXAMPLES:

UnitPermutahedralBall EXAMPLES:

`UniversalBarCodeEval` EXAMPLES:

`UniversalCover` EXAMPLES: 1 , 2 , 3 , 4

`VectorToCrystMatrix` EXAMPLES:

`VectorsToOneSkeleton` EXAMPLES: 1

`VerticesOfRegularCWCell` EXAMPLES:

`View3dPureComplex` EXAMPLES:

`ViewArc2Presentation` EXAMPLES:

`ViewPureComplex` EXAMPLES:

`VirtuallySimplicialSubdivision` EXAMPLES:

`VoronoiBaseMatrix` EXAMPLES:

`VoronoiBuildStabilizer` EXAMPLES:

`VoronoiCoordinates` EXAMPLES:

`VoronoiFactorize` EXAMPLES: 1

`VoronoiGenerators` EXAMPLES: 1

`VoronoiReduce` EXAMPLES:

`VoronoiWord` EXAMPLES:

`WeakCommutativityCommutatorGroup` EXAMPLES:

`WeakCommutativityGroup` EXAMPLES:

`WirtingerGroup` EXAMPLES: 1

`WirtingerGroup_gc` EXAMPLES:

`WordModP` EXAMPLES:

`ZigZagContractedFilteredPureCubicalComplex` EXAMPLES:

`ZigZagContractedPureComplex` EXAMPLES: 1

Sq EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14

Category_Of_Groups EXAMPLES: 1

IsBianchiAbelianGroup EXAMPLES:

IsHap2x2matrix EXAMPLES:

IsHap2x2matrixGroup EXAMPLES:

AsFpGroup EXAMPLES:

BarycentricSubdivision EXAMPLES: 1 , 2 , 3 , 4

Bockstein EXAMPLES: 1 , 2 , 3

CategoryArrow EXAMPLES:

CategoryObject EXAMPLES:

ChildGetObj EXAMPLES:

ChildPutObj EXAMPLES:

ClosedSurface EXAMPLES: 1 , 2

CoboundaryMatrix EXAMPLES:

CoefficientsOfPoincareSeries EXAMPLES:

CohomologyClass EXAMPLES: 1 , 2

CohomologyRing EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10

ComplexProjectiveSpace EXAMPLES: 1

CompositionRingHomomorphism EXAMPLES:

ConnectedComponentsQuandle EXAMPLES:

ConnectedSum EXAMPLES: 1 , 2

DegreeOfRepresentative EXAMPLES:

Dimensions EXAMPLES:

Display2D EXAMPLES: 1

Display3D EXAMPLES: 1

DualComplex EXAMPLES: 1

ExcisedPair EXAMPLES:

ExpandedComplex EXAMPLES: 1

FilteredRegularCWComplex EXAMPLES: 1

FiniteProjectiveLine EXAMPLES:

FiniteProjectiveLine_alt EXAMPLES:

FiniteProjectivePlane EXAMPLES:

FundamentalGroupWithPathReps EXAMPLES: 1 , 2

GDerivedSubgroup EXAMPLES:

GModuleAsGOuterGroup EXAMPLES: 1 , 2 , 3 , 4

GOuterGroupHomomorphism EXAMPLES: 1 , 2

GOuterGroupHomomorphism EXAMPLES: 1 , 2

GradedAlgebraPresentation EXAMPLES:

GradedAlgebraPresentationNC EXAMPLES:

HAPCongruenceSubgroupGamma0 EXAMPLES: 1

HAPCongruenceSubgroupGamma0_alt EXAMPLES:

HAPCongruenceSubgroupTree EXAMPLES:

HAPDerivationNC EXAMPLES:

HAPIntersectionConjugatedCongruenceSubgroup EXAMPLES:

HAPRingHomomorphismByIndeterminateMap EXAMPLES:

HAPRingReductionHomomorphism EXAMPLES:

HAPRingReductionHomomorphism EXAMPLES:

HAPRingToSubringHomomorphism EXAMPLES:

HAPSubringToRingHomomorphism EXAMPLES:

HAPSubringToRingHomomorphism EXAMPLES:

HAPZeroRingHomomorphism EXAMPLES:

HAP_EquivalenceClasses EXAMPLES:

HomomorphismsImages EXAMPLES:

ImageOfDerivation EXAMPLES:

ImageOfRingHomomorphism EXAMPLES:

IntersectionWithConjugate EXAMPLES:

IsAssociatedGradedRing EXAMPLES:

KernelOfDerivation EXAMPLES:

LowerGCentralSeries EXAMPLES:

PathComponents EXAMPLES: 1

PersistentBettiNumbersAlt EXAMPLES: 1

PersistentHomology EXAMPLES: 1 , 2

PersistentHomology EXAMPLES: 1 , 2

PersistentHomology EXAMPLES: 1 , 2

PoincareSeriesAutoMem EXAMPLES:

PoincareSeriesAutoMem EXAMPLES:

PoincareSeriesAutoMemStop EXAMPLES:

PolynomialToRModuleRep EXAMPLES:

PreimageOfRingHomomorphism EXAMPLES:

`PureComplexMeet` EXAMPLES:

`PureComplexRandomCell` EXAMPLES: 1

`PureComplexSubcomplex` EXAMPLES:

`Pushout` EXAMPLES:

`QuadraticIdeal` EXAMPLES: 1 , 2 , 3

`QuadraticNumberToCyclotomic` EXAMPLES:

`ReduceIdeal` EXAMPLES:

`ReducedPolynomialRingPresentation` EXAMPLES:

`ReducedPolynomialRingPresentationMap` EXAMPLES:

`RefinedColouring` EXAMPLES:

`Resolution` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36

`RightTransversal_alt` EXAMPLES:

`RingOfIntegers` EXAMPLES: 1 , 2 , 3

`SingularPolynomialNormalForm` EXAMPLES:

`SingularSetNormalFormIdeal` EXAMPLES:

`SingularSetNormalFormIdealNC` EXAMPLES:

`SparseChainComplexOfPair` EXAMPLES:

`Sphere` EXAMPLES: 1

`Sq` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14

`Standard2Cocycle` EXAMPLES:

`Standard2Cocycle` EXAMPLES:

`StandardNCocycle` EXAMPLES:

StandardNCocycle EXAMPLES:

SubspaceBasisRepsByDegree EXAMPLES:

SubspaceDimensionDegree EXAMPLES:

Suspension EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6

TrivialGModuleAsGOuterGroup EXAMPLES: 1 , 2 , 3 , 4

VertexLink EXAMPLES:

VertexStar EXAMPLES:

WedgeSum EXAMPLES: 1

CongruenceSubgroupGamma0 EXAMPLES: 1 , 2 , 3 , 4 , 5

CongruenceSubgroupGamma0 EXAMPLES: 1 , 2 , 3 , 4 , 5

CongruenceSubgroupGamma1 EXAMPLES:

PrincipalCongruenceSubgroup EXAMPLES: 1

StarGraph EXAMPLES:

TensorProductOp EXAMPLES:

AmbientGroupOfCongruenceSubgroup EXAMPLES:

AmbientPosition EXAMPLES:

AmbientRepresentation EXAMPLES:

AmbientTransversal EXAMPLES:

AmbientTree EXAMPLES:

Arity EXAMPLES:

AssociatedNumberField EXAMPLES:

AssociatedQuadraticNumberField EXAMPLES:

AssociatedRing EXAMPLES:

`Base` EXAMPLES: 1 , 2

`BaseElement` EXAMPLES:

`BaseRing` EXAMPLES:

`Cocycle` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7

`CoefficientModule` EXAMPLES:

`CohomologicalPeriod` EXAMPLES: 1 , 2

`CoxeterMatrix` EXAMPLES: 1

`DerivationImages` EXAMPLES:

`DerivationRelations` EXAMPLES:

`DerivationRing` EXAMPLES:

`Fibre` EXAMPLES: 1

`FibreElement` EXAMPLES:

`GeneratorsOfPresentationIdeal` EXAMPLES:

`GradedAlgebraPresentationFamily` EXAMPLES:

`HAPDerivationFamily` EXAMPLES:

`HAPPRIME_HilbertSeries` EXAMPLES:

`HAPRingHomomorphismFamily` EXAMPLES:

`HAP_MultiplicativeGenerators` EXAMPLES:

`HapFibre` EXAMPLES:

`IdentityMap` EXAMPLES:

`ImageGenerators` EXAMPLES:

`ImagePolynomialRing` EXAMPLES:

`ImageRelations` EXAMPLES:

`InCcGroup` EXAMPLES:

`IndeterminateAndExponentOfUnivariateMonomial` EXAMPLES:

`IndeterminateDegrees` EXAMPLES:

`IndeterminatesOfGradedAlgebraPresentation` EXAMPLES:

`IndeterminatesOfPolynomial` EXAMPLES:

`IndexInAmbientGroup` EXAMPLES:

`IndexInSL2O` EXAMPLES: 1 , 2 , 3

`InnerAutomorphismGroupQuandle` EXAMPLES:

`InnerAutomorphismGroupQuandleAsPerm` EXAMPLES:

`InverseRingHomomorphism` EXAMPLES:

`IsConnected` EXAMPLES: 1 , 2 , 3

`IsHomogeneousQuandle` EXAMPLES:

`IsLatinQuandle` EXAMPLES: 1

`MaximumDegreeForPresentation` EXAMPLES:

`ModPRingBasisAsPolynomials` EXAMPLES:

`ModPRingGeneratorDegrees` EXAMPLES:

`ModPRingNiceBasis` EXAMPLES:

`ModPRingNiceBasisAsPolynomials` EXAMPLES:

`Module` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15

`NaturalHomomorphismOntoBase` EXAMPLES:

`NormOfIdeal` EXAMPLES:

`OuterAction` EXAMPLES:

`OuterGroup` EXAMPLES: 1 , 2 , 3 , 4

PresentationIdeal EXAMPLES:

PresentationOfGradedStructureConstantAlgebra EXAMPLES: 1

ProjectiveSpace EXAMPLES: 1

Pullbacks EXAMPLES:

Pushouts EXAMPLES:

RightMultiplicationGroupOfQuandle EXAMPLES: 1 , 2 , 3

RightMultiplicationGroupOfQuandleAsPerm EXAMPLES: 1

SingularGroebnerBasis EXAMPLES:

SingularReducedGroebnerBasis EXAMPLES:

SourceGenerators EXAMPLES:

SourcePolynomialRing EXAMPLES:

SourceRelations EXAMPLES:

StabilizerSubgroup EXAMPLES:

StarGraphAttr EXAMPLES:

TermsOfPolynomial EXAMPLES:

UnivariateMonomialsOfMonomial EXAMPLES:

CoefficientsRing EXAMPLES:

ElementsFamily EXAMPLES:

Generators EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27

IndexInSL2Z EXAMPLES: 1

LevelOfCongruenceSubgroup EXAMPLES:

Name EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6

Order EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 ,

18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26

IsAbelianCategory EXAMPLES:

IsAdditiveCategory EXAMPLES:

IsCategoryName EXAMPLES:

IsCcGroup EXAMPLES:

IsCrystTranslationSubGroup EXAMPLES:

IsGOuterGroup EXAMPLES:

IsGOuterGroupHomomorphism EXAMPLES:

IsGammaSubgroupInSL3Z EXAMPLES:

IsHAPCongruenceSubgroupGamma0 EXAMPLES:

IsHAPCongruenceSubgroupGamma1 EXAMPLES:

IsHAPPrincipalCongruenceSubgroup EXAMPLES:

IsHAPRationalMatrixGroup EXAMPLES:

IsHAPRationalSpecialLinearGroup EXAMPLES:

IsIdealOfQuadraticIntegers EXAMPLES:

IsIdealOfQuadraticIntegers_NC EXAMPLES:

IsPeriodic EXAMPLES: 1 , 2 , 3 , 4

IsPseudoListWithFunction EXAMPLES:

IsQuadraticNumberField EXAMPLES:

IsQuadraticNumberField_NC EXAMPLES:

IsRingOfQuadraticIntegers EXAMPLES:

IsRingOfQuadraticIntegers_NC EXAMPLES:

IsStandard2Cocycle EXAMPLES:

`IsStandardNCocycle` EXAMPLES:

`IsCcElement` EXAMPLES:

`IsGradedAlgebraPresentation` EXAMPLES:

`IsHAPCongruenceSubgroup` EXAMPLES:

`IsHAPConjugatedCongruenceSubgroup` EXAMPLES:

`IsHAPDerivation` EXAMPLES:

`IsHAPRingHomomorphism` EXAMPLES:

`IsHAPRingModIdealObj` EXAMPLES:

`IsHapBianchiPolyhedron` EXAMPLES:

`IsHapCatOneGroup` EXAMPLES:

`IsHapCatOneGroupMorphism` EXAMPLES:

`IsHapChainComplex` EXAMPLES:

`IsHapChainMap` EXAMPLES:

`IsHapCochainComplex` EXAMPLES:

`IsHapCochainMap` EXAMPLES:

`IsHapCommutativeDiagram` EXAMPLES:

`IsHapConjQuandElt` EXAMPLES:

`IsHapCrossedModule` EXAMPLES:

`IsHapCrossedModuleMorphism` EXAMPLES:

`IsHapCubicalComplex` EXAMPLES:

`IsHapEquivariantCWComplex` EXAMPLES:

`IsHapEquivariantChainComplex` EXAMPLES:

`IsHapEquivariantChainMap` EXAMPLES:

`IsHapEquivariantNonFreeChainComplex` EXAMPLES:

`IsHapEquivariantSpectralSequencePage` EXAMPLES:

`IsHapFilteredChainComplex` EXAMPLES:

`IsHapFilteredCubicalComplex` EXAMPLES:

`IsHapFilteredGraph` EXAMPLES:

`IsHapFilteredPureCubicalComplex` EXAMPLES:

`IsHapFilteredRegularCWComplex` EXAMPLES:

`IsHapFilteredSimplicialComplex` EXAMPLES:

`IsHapFilteredSparseChainComplex` EXAMPLES:

`IsHapGCocomplex` EXAMPLES:

`IsHapGComplex` EXAMPLES:

`IsHapGComplexMap` EXAMPLES:

`IsHapGraph` EXAMPLES:

`IsHapOppositeElement` EXAMPLES:

`IsHapPureCubicalComplex` EXAMPLES:

`IsHapPureCubicalLink` EXAMPLES:

`IsHapPurePermutahedralComplex` EXAMPLES:

`IsHapQuadraticNumber` EXAMPLES:

`IsHapQuandlePresentation` EXAMPLES:

`IsHapQuotientElement` EXAMPLES:

`IsHapRegularCWComplex` EXAMPLES:

`IsHapRegularCWMap` EXAMPLES:

`IsHapResolution` EXAMPLES:

`IsHapSimplicialComplex` EXAMPLES:

`IsHapSimplicialFreeAbelianGroup` EXAMPLES:

`IsHapSimplicialGroup` EXAMPLES:

`IsHapSimplicialGroupMorphism` EXAMPLES:

`IsHapSimplicialMap` EXAMPLES:

`IsHapSparseChainComplex` EXAMPLES:

`IsHapSparseChainMap` EXAMPLES:

`IsHapSparseMat` EXAMPLES:

`IsHapTorsionSubcomplex` EXAMPLES:

`IsPseudoList` EXAMPLES:

`IsCcElementRep` EXAMPLES:

`IsGradedAlgebraPresentationRep` EXAMPLES:

`IsHAPDerivationRep` EXAMPLES:

`IsHAPIdealRep` EXAMPLES:

`IsHAPRingHomomorphismIndeterminateMapRep` EXAMPLES:

`IsHAPRingReductionHomomorphismRep` EXAMPLES:

`IsHAPRingToSubringHomomorphismRep` EXAMPLES:

`IsHAPSubringToRingHomomorphismRep` EXAMPLES:

`IsHAPZeroRingHomomorphismRep` EXAMPLES:

`IsHapBianchiPolyhedronRep` EXAMPLES:

`IsHapCatOneGroupMorphismRep` EXAMPLES:

`IsHapCatOneGroupRep` EXAMPLES:

`IsHapChainComplexRep` EXAMPLES:

IsHapChainMapRep EXAMPLES:

IsHapCochainComplexRep EXAMPLES:

IsHapCochainMapRep EXAMPLES:

IsHapCommutativeDiagramRep EXAMPLES:

IsHapConjQuandEltRep EXAMPLES:

IsHapCrossedModuleMorphismRep EXAMPLES:

IsHapCrossedModuleRep EXAMPLES:

IsHapCubicalComplexRep EXAMPLES:

IsHapEquivariantCWComplexRep EXAMPLES:

IsHapEquivariantChainComplexRep EXAMPLES:

IsHapEquivariantChainMapRep EXAMPLES:

IsHapEquivariantNonFreeChainComplexRep EXAMPLES:

IsHapEquivariantSpectralSequencePageRep EXAMPLES:

IsHapFilteredChainComplexRep EXAMPLES:

IsHapFilteredCubicalComplexRep EXAMPLES:

IsHapFilteredGraphRep EXAMPLES:

IsHapFilteredPureCubicalComplexRep EXAMPLES:

IsHapFilteredRegularCWComplexRep EXAMPLES:

IsHapFilteredSimplicialComplexRep EXAMPLES:

IsHapFilteredSparseChainComplexRep EXAMPLES:

IsHapGCocomplexRep EXAMPLES:

IsHapGComplexMapRep EXAMPLES:

IsHapGComplexRep EXAMPLES:

`IsHapGraphRep` EXAMPLES:

`IsHapOppositeElementRep` EXAMPLES:

`IsHapPureCubicalComplexRep` EXAMPLES:

`IsHapPureCubicalLinkRep` EXAMPLES:

`IsHapPurePermutahedralComplexRep` EXAMPLES:

`IsHapQuadraticNumberRep` EXAMPLES:

`IsHapQuandlePresentationRep` EXAMPLES:

`IsHapQuotientElementRep` EXAMPLES:

`IsHapRegularCWComplexRep` EXAMPLES:

`IsHapRegularCWMapRep` EXAMPLES:

`IsHapResolutionRep` EXAMPLES:

`IsHapSimplicialComplexRep` EXAMPLES:

`IsHapSimplicialFreeAbelianGroupRep` EXAMPLES:

`IsHapSimplicialGroupMorphismRep` EXAMPLES:

`IsHapSimplicialGroupRep` EXAMPLES:

`IsHapSimplicialMapRep` EXAMPLES:

`IsHapSparseChainComplexRep` EXAMPLES:

`IsHapSparseChainMapRep` EXAMPLES:

`IsHapSparseMatRep` EXAMPLES:

`IsHapTorsionSubcomplexRep` EXAMPLES:

`IsPseudoListRep` EXAMPLES:

`IdealOfQuadraticIntegers` EXAMPLES:

`IdealOfQuadraticIntegers_NC` EXAMPLES:

`QuadraticNF` EXAMPLES:

`QuadraticNF_NC` EXAMPLES:

`RingOfQuadraticIntegers` EXAMPLES:

`RingOfQuadraticIntegers_NC` EXAMPLES:

`*` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41

`*` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41

`*` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41

`*` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41

`+` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46

`+` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46

`+` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46

EXAMPLES:

EXAMPLES:

`=` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

`=` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

= EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

AdditiveInverseMutable EXAMPLES:

AmbientPosition EXAMPLES:

AmbientPosition EXAMPLES:

AmbientPosition EXAMPLES:

AmbientPosition EXAMPLES:

AmbientPosition EXAMPLES:

AmbientRepresentation EXAMPLES:

AmbientRepresentation EXAMPLES:

AmbientRepresentation EXAMPLES:

AmbientRepresentation EXAMPLES:

AmbientTransversal EXAMPLES:

AmbientTransversal EXAMPLES:

AmbientTransversal EXAMPLES:

AmbientTransversal EXAMPLES:

AmbientTree EXAMPLES:

AmbientTree EXAMPLES:

AsFpGroup EXAMPLES:

AsList EXAMPLES:

AsSSortedList EXAMPLES:

BarycentricSubdivision EXAMPLES: 1 , 2 , 3 , 4

BaseRing EXAMPLES:

`Bockstein` EXAMPLES: 1 , 2 , 3

`CategoryArrow` EXAMPLES:

`CategoryObject` EXAMPLES:

`ChildGetObj` EXAMPLES:

`ChildPutObj` EXAMPLES:

`CoboundaryMatrix` EXAMPLES:

`CoefficientsOfPoincareSeries` EXAMPLES:

`CoefficientsRing` EXAMPLES:

`CohomologicalPeriod` EXAMPLES: 1 , 2

`CohomologyClass` EXAMPLES: 1 , 2

`CompositionRingHomomorphism` EXAMPLES:

`CompositionRingHomomorphism` EXAMPLES:

`CompositionRingHomomorphism` EXAMPLES:

`CompositionRingHomomorphism` EXAMPLES:

`CompositionRingHomomorphism` EXAMPLES:

`CongruenceSubgroupGamma0` EXAMPLES: 1 , 2 , 3 , 4 , 5

`CongruenceSubgroupGamma0` EXAMPLES: 1 , 2 , 3 , 4 , 5

`ConnectedComponentsQuandle` EXAMPLES:

`ConnectedSum` EXAMPLES: 1 , 2

`CoxeterMatrix` EXAMPLES: 1

`DefaultFieldOfMatrixGroup` EXAMPLES:

`DegreeOfRepresentative` EXAMPLES:

`DerivationImages` EXAMPLES:

DerivationRelations EXAMPLES:

DerivationRing EXAMPLES:

Dimensions EXAMPLES:

Display2D EXAMPLES: 1

Display3D EXAMPLES: 1

DualComplex EXAMPLES: 1

Enumerator EXAMPLES:

ExcisedPair EXAMPLES:

Fibre EXAMPLES: 1

FilteredRegularCWComplex EXAMPLES: 1

FiniteProjectiveLine EXAMPLES:

FiniteProjectiveLine_alt EXAMPLES:

FiniteProjectivePlane EXAMPLES:

FundamentalGroupWithPathReps EXAMPLES: 1 , 2

GModuleAsGOuterGroup EXAMPLES: 1 , 2 , 3 , 4

GOuterGroupHomomorphism EXAMPLES: 1 , 2

GOuterGroupHomomorphism EXAMPLES: 1 , 2

Generators EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27

GeneratorsOfMagmaWithInverses EXAMPLES:

GeneratorsOfMagmaWithInverses EXAMPLES:

GeneratorsOfMagmaWithInverses EXAMPLES:

GeneratorsOfPresentationIdeal EXAMPLES:

GradedAlgebraPresentation EXAMPLES:

GradedAlgebraPresentationNC EXAMPLES:

HAPCongruenceSubgroupGamma0 EXAMPLES: 1

HAPCongruenceSubgroupGamma0 EXAMPLES: 1

HAPCongruenceSubgroupGamma0_alt EXAMPLES:

HAPCongruenceSubgroupTree EXAMPLES:

HAPCongruenceSubgroupTree EXAMPLES:

HAPDerivationNC EXAMPLES:

HAPIntersectionConjugatedCongruenceSubgroup EXAMPLES:

HAPPRIME_HilbertSeries EXAMPLES:

HAPRingHomomorphismByIndeterminateMap EXAMPLES:

HAPRingReductionHomomorphism EXAMPLES:

HAPRingReductionHomomorphism EXAMPLES:

HAPRingToSubringHomomorphism EXAMPLES:

HAPSubringToRingHomomorphism EXAMPLES:

HAPSubringToRingHomomorphism EXAMPLES:

HAPZeroRingHomomorphism EXAMPLES:

HAP_MultiplicativeGenerators EXAMPLES:

HomomorphismsImages EXAMPLES:

IdGroup EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8

IdentityMap EXAMPLES:

ImageOfDerivation EXAMPLES:

ImageOfRingHomomorphism EXAMPLES:

ImageOfRingHomomorphism EXAMPLES:

ImageOfRingHomomorphism EXAMPLES:

ImageOfRingHomomorphism EXAMPLES:

ImageOfRingHomomorphism EXAMPLES:

IndeterminateAndExponentOfUnivariateMonomial EXAMPLES:

IndeterminateDegrees EXAMPLES:

IndeterminatesOfGradedAlgebraPresentation EXAMPLES:

IndeterminatesOfPolynomial EXAMPLES:

IndexInAmbientGroup EXAMPLES:

IndexInAmbientGroup EXAMPLES:

IndexInSL2O EXAMPLES: 1 , 2 , 3

IndexInSL2Z EXAMPLES: 1

InnerAutomorphismGroupQuandle EXAMPLES:

InnerAutomorphismGroupQuandleAsPerm EXAMPLES:

Int EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41

IntersectionWithConjugate EXAMPLES:

InverseMutable EXAMPLES:

InverseMutable EXAMPLES:

InverseMutable EXAMPLES:

InverseRingHomomorphism EXAMPLES:

InverseRingHomomorphism EXAMPLES:

InverseRingHomomorphism EXAMPLES:

InverseRingHomomorphism EXAMPLES:

`InverseRingHomomorphism` EXAMPLES:

`InverseSameMutability` EXAMPLES:

`IsAssociatedGradedRing` EXAMPLES:

`IsConnected` EXAMPLES: 1 , 2 , 3

`IsHomogeneousQuandle` EXAMPLES:

`IsLatinQuandle` EXAMPLES: 1

`IsMonomial` EXAMPLES:

`IsOne` EXAMPLES:

`IsPeriodic` EXAMPLES: 1 , 2 , 3 , 4

`IsSubset` EXAMPLES:

`IsomorphismFpGroup` EXAMPLES: 1 , 2 , 3

`Kernel` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12

`KernelOfDerivation` EXAMPLES:

`MaximumDegreeForPresentation` EXAMPLES:

`ModPRingBasisAsPolynomials` EXAMPLES:

`ModPRingGeneratorDegrees` EXAMPLES:

`ModPRingNiceBasis` EXAMPLES:

`ModPRingNiceBasisAsPolynomials` EXAMPLES:

`NaturalHomomorphismOntoBase` EXAMPLES:

`OneImmutable` EXAMPLES:

`OneMutable` EXAMPLES:

`PathComponents` EXAMPLES: 1

`PersistentBettiNumbersAlt` EXAMPLES: 1

`PreimageOfRingHomomorphism` EXAMPLES:

`PresentationIdeal` EXAMPLES:

`PresentationOfGradedStructureConstantAlgebra` EXAMPLES: 1

`PrincipalCongruenceSubgroup` EXAMPLES: 1

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`PrintObj` EXAMPLES:

`Projection` EXAMPLES:

`ProjectiveSpace` EXAMPLES: 1

`ProjectiveSpace` EXAMPLES: 1

`PureComplexMeet` EXAMPLES:

`PureComplexRandomCell` EXAMPLES: 1

`PureComplexSubcomplex` EXAMPLES:

`Pushout` EXAMPLES:

`QuadraticIdeal` EXAMPLES: 1 , 2 , 3

`QuadraticNumberToCyclotomic` EXAMPLES:

`Random` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17

`ReduceIdeal` EXAMPLES:

`ReducedPolynomialRingPresentation` EXAMPLES:

`ReducedPolynomialRingPresentationMap` EXAMPLES:

`RefinedColouring` EXAMPLES:

`Resolution` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36

`RightMultiplicationGroupOfQuandle` EXAMPLES: 1 , 2 , 3

`RightMultiplicationGroupOfQuandleAsPerm` EXAMPLES: 1

`RightTransversal` EXAMPLES:

`RightTransversal` EXAMPLES:

`RingOfIntegers` EXAMPLES: 1 , 2 , 3

`SingularGroebnerBasis` EXAMPLES:

`SingularPolynomialNormalForm` EXAMPLES:

`SingularReducedGroebnerBasis` EXAMPLES:

`SingularSetNormalFormIdeal` EXAMPLES:

`SingularSetNormalFormIdealNC` EXAMPLES:

`SparseChainComplexOfPair` EXAMPLES:

`Sq` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14

`StabilizerSubgroup` EXAMPLES:

`StabilizerSubgroup` EXAMPLES:

`Standard2Cocycle` EXAMPLES:

`Standard2Cocycle` EXAMPLES:

`StandardNCocycle` EXAMPLES:

`StandardNCocycle` EXAMPLES:

`StarGraph` EXAMPLES:

`StarGraphAttr` EXAMPLES:

`SubspaceBasisRepsByDegree` EXAMPLES:

`SubspaceDimensionDegree` EXAMPLES:

`Suspension` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6

`TensorProductOp` EXAMPLES:

`TermsOfPolynomial` EXAMPLES:

`TrivialGModuleAsGOuterGroup` EXAMPLES: 1 , 2 , 3 , 4

`TrivialSubmagmaWithOne` EXAMPLES:

`Units` EXAMPLES:

`Units` EXAMPLES:

`UnivariateMonomialsOfMonomial` EXAMPLES:

`VertexLink` EXAMPLES:

`VertexStar` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`ViewObj` EXAMPLES:

`WedgeSum` EXAMPLES: 1

`ZeroMutable` EXAMPLES:

`^` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43

`^` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43

`^` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43

`in` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

`in` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

`in` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

`in` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 ,

42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

`in` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

`in` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

`mod` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42

`*` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41

`*` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41

`*` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41

`*` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41

`*` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41

`*` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41

`*` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41

`*` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41

+ EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46

+ EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46

+ EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46

+ EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46

+ EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46

+ EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46

- EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

- EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

- EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

- EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 ,

20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

/ EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

/ EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

/ EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

/ EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

EXAMPLES:

EXAMPLES:

EXAMPLES:

EXAMPLES:

EXAMPLES:

EXAMPLES:

EXAMPLES:

= EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

= EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 ,

66 , 67

= EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

= EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

= EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

= EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

= EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

= EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

= EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

`AbelianInvariants` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13

`AbelianInvariants` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13

`AdditiveInverseMutable` EXAMPLES:

`AmbientPosition` EXAMPLES:

`AmbientRepresentation` EXAMPLES:

`AsSSortedList` EXAMPLES:

`BarycentricSubdivision` EXAMPLES: 1 , 2 , 3 , 4

`Bockstein` EXAMPLES: 1 , 2 , 3

`CanonicalRightCosetElement` EXAMPLES:

`CanonicalRightCosetElement` EXAMPLES:

`ChildPutObj` EXAMPLES:

`ChildPutObj` EXAMPLES:

`ChildPutObj` EXAMPLES:

`ChildPutObj` EXAMPLES:

`ChildPutObj` EXAMPLES:

`ClosedSurface` EXAMPLES: 1 , 2

`ClosedSurface` EXAMPLES: 1 , 2

`CohomologyRing` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10

`CohomologyRing` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10

`ComplexProjectiveSpace` EXAMPLES: 1

`ConnectedSum` EXAMPLES: 1 , 2

`ConnectedSum` EXAMPLES: 1 , 2

`ConnectedSum` EXAMPLES: 1 , 2

`Determinant` EXAMPLES: 1 , 2 , 3

`Dimensions` EXAMPLES:

`DirectProductOp` EXAMPLES:

`DirectProductOp` EXAMPLES:

`DirectProductOp` EXAMPLES:

DirectProductOp EXAMPLES:

Discriminant EXAMPLES: 1

Discriminant EXAMPLES: 1

Discriminant EXAMPLES: 1

Discriminant EXAMPLES: 1

Display3D EXAMPLES: 1

DualComplex EXAMPLES: 1

Embedding EXAMPLES:

ExpandedComplex EXAMPLES: 1

ExpandedComplex EXAMPLES: 1

ExpandedComplex EXAMPLES: 1

ExpandedComplex EXAMPLES: 1

GDerivedSubgroup EXAMPLES:

Generators EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27

HAPRingReductionHomomorphism EXAMPLES:

HAPRingReductionHomomorphism EXAMPLES:

HAP_EquivalenceClasses EXAMPLES:

ImageOfRingHomomorphism EXAMPLES:

ImageOfRingHomomorphism EXAMPLES:

IndexNC EXAMPLES:

IndexNC EXAMPLES:

IndexNC EXAMPLES:

`Int` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41

`Intersection2` EXAMPLES:

`InverseMutable` EXAMPLES:

`InverseMutable` EXAMPLES:

`InverseMutable` EXAMPLES:

`InverseMutable` EXAMPLES:

`InverseMutable` EXAMPLES:

`InverseMutable` EXAMPLES:

`InverseSameMutability` EXAMPLES:

`IsEmpty` EXAMPLES:

`IsEmpty` EXAMPLES:

`IsPrime` EXAMPLES: 1 , 2 , 3

`IsPrime` EXAMPLES: 1 , 2 , 3

`IsomorphismFpGroup` EXAMPLES: 1 , 2 , 3

`IsomorphismFpGroup` EXAMPLES: 1 , 2 , 3

`IsomorphismPermGroup` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7

`IsomorphismPermGroup` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7

`Iterator` EXAMPLES:

`KernelOfDerivation` EXAMPLES:

`ListOp` EXAMPLES:

`ListOp` EXAMPLES:

`LowerGCentralSeries` EXAMPLES:

`MultiplicationTable` EXAMPLES: 1 , 2 , 3 , 4

`MultiplicationTable` EXAMPLES: 1 , 2 , 3 , 4

`NaturalHomomorphism` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7

`NiceMonomorphism` EXAMPLES:

`Norm` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17

`Norm` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17

`Norm` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17

`Norm` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17

`Norm` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17

`OneImmutable` EXAMPLES:

`OneImmutable` EXAMPLES:

`OneImmutable` EXAMPLES:

`OneImmutable` EXAMPLES:

`OneMutable` EXAMPLES:

`Order` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26

`Order` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26

`Order` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26

`PersistentBettiNumbersAlt` EXAMPLES: 1

`PersistentBettiNumbersAlt` EXAMPLES: 1

`PersistentBettiNumbersAlt` EXAMPLES: 1

`PersistentBettiNumbersAlt` EXAMPLES: 1

`PersistentBettiNumbersAlt` EXAMPLES: 1

PersistentBettiNumbersAlt EXAMPLES: 1

PersistentBettiNumbersAlt EXAMPLES: 1

PersistentBettiNumbersAlt EXAMPLES: 1

PersistentBettiNumbersAlt EXAMPLES: 1

PersistentBettiNumbersAlt EXAMPLES: 1

PersistentBettiNumbersAlt EXAMPLES: 1

PersistentBettiNumbersAlt EXAMPLES: 1

PersistentHomology EXAMPLES: 1 , 2

PersistentHomology EXAMPLES: 1 , 2

Position EXAMPLES: 1 , 2 , 3

Position EXAMPLES: 1 , 2 , 3

Position EXAMPLES: 1 , 2 , 3

PositionCanonical EXAMPLES:

PositionCanonical EXAMPLES:

PreimageOfRingHomomorphism EXAMPLES:

Projection EXAMPLES:

PureComplexMeet EXAMPLES:

PureComplexRandomCell EXAMPLES: 1

PureComplexSubcomplex EXAMPLES:

QuadraticIdeal EXAMPLES: 1 , 2 , 3

QuadraticIdeal EXAMPLES: 1 , 2 , 3

QuadraticIdeal EXAMPLES: 1 , 2 , 3

QuadraticNumberToCyclotomic EXAMPLES:

QuadraticNumberToCyclotomic EXAMPLES:

Range EXAMPLES: 1 , 2

RankMatrixDestructive EXAMPLES:

ReduceIdeal EXAMPLES:

ReduceIdeal EXAMPLES:

ReducedPolynomialRingPresentation EXAMPLES:

ReducedPolynomialRingPresentation EXAMPLES:

ReducedPolynomialRingPresentationMap EXAMPLES:

ReducedPolynomialRingPresentationMap EXAMPLES:

ReducedPolynomialRingPresentationMap EXAMPLES:

RefinedColouring EXAMPLES:

RightTransversal EXAMPLES:

RightTransversal EXAMPLES:

RightTransversal EXAMPLES:

RightTransversal EXAMPLES:

RightTransversal EXAMPLES:

RightTransversal EXAMPLES:

RightTransversal_alt EXAMPLES:

RingOfIntegers EXAMPLES: 1 , 2 , 3

SingularPolynomialNormalForm EXAMPLES:

SparseChainComplexOfPair EXAMPLES:

Sphere EXAMPLES: 1

Sqrt EXAMPLES: 1 , 2 , 3

`SubspaceBasisRepsByDegree` EXAMPLES:

`SubspaceDimensionDegree` EXAMPLES:

`Suspension` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6

`Suspension` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6

`Suspension` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6

`TensorProductOp` EXAMPLES:

`TensorProductOp` EXAMPLES:

`TensorProductOp` EXAMPLES:

`Trace` EXAMPLES:

`Trace` EXAMPLES:

`Trace` EXAMPLES:

`Units` EXAMPLES:

`Units` EXAMPLES:

`WedgeSum` EXAMPLES: 1

`WedgeSum` EXAMPLES: 1

`WedgeSum` EXAMPLES: 1

`ZeroImmutable` EXAMPLES:

`ZeroImmutable` EXAMPLES:

`[]` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13

`[]` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13

`[]` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13

`[]` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13

`^` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 ,

43

^ EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43

^ EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43

^ EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43

^ EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43

in EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

in EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

in EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

in EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

in EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 ,

65 , 66 , 67

in EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

in EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

in EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

in EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 , 60 , 61 , 62 , 63 , 64 , 65 , 66 , 67

mod EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42

mod EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 , 32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42

PathComponentOfSimplicialComplex EXAMPLES:

ResolutionSL2ZInvertedInteger EXAMPLES: 1

ViewGraph EXAMPLES:

InfoHAPprime EXAMPLES:

ASY_PATH EXAMPLES:

AutomorphismGroupAsCrossedModule EXAMPLES:

BROWSER_PATH EXAMPLES:

CATONEGROUP_DATA_PERM EXAMPLES:

CATONEGROUP_DATA_SIZE EXAMPLES:

Cedric_PlanarDiagram EXAMPLES:

ChildKill EXAMPLES:

DISPLAY_PATH EXAMPLES:

DOT_PATH EXAMPLES:

FilteredSimplicialComplexToFilteredCWComplex EXAMPLES:

GradedAlgebraPresentationType EXAMPLES:

HAPTEMPORARYFUNCTION EXAMPLES:

HAP_Knots EXAMPLES:

HAP_transversalmethodtoggle EXAMPLES:

HapCatOneGroup EXAMPLES:

HapCatOneGroupFamily EXAMPLES:

HapCatOneGroupMorphism EXAMPLES:

HapCatOneGroupMorphismFamily EXAMPLES:

HapChainComplex EXAMPLES:

HapChainComplexFamily EXAMPLES:

HapChainMap EXAMPLES:

HapChainMapFamily EXAMPLES:

HapCochainComplex EXAMPLES:

HapCochainComplexFamily EXAMPLES:

HapCochainMap EXAMPLES:

HapCochainMapFamily EXAMPLES:

HapCommutativeDiagram EXAMPLES:

`HapCommutativeDiagramFamily` EXAMPLES:

`HapCrossedModule` EXAMPLES:

`HapCrossedModuleFamily` EXAMPLES:

`HapCrossedModuleMorphism` EXAMPLES:

`HapCrossedModuleMorphismFamily` EXAMPLES:

`HapCubicalComplex` EXAMPLES:

`HapCubicalComplexFamily` EXAMPLES:

`HapEquivariantCWComplex` EXAMPLES:

`HapEquivariantCWComplexFamily` EXAMPLES:

`HapEquivariantChainMap` EXAMPLES:

`HapEquivariantChainMapFamily` EXAMPLES:

`HapFPGModule` EXAMPLES:

`HapFPGModuleHomomorphism` EXAMPLES:

`HapFilteredChainComplex` EXAMPLES:

`HapFilteredChainComplexFamily` EXAMPLES:

`HapFilteredCubicalComplex` EXAMPLES:

`HapFilteredCubicalComplexFamily` EXAMPLES:

`HapFilteredGraph` EXAMPLES:

`HapFilteredGraphFamily` EXAMPLES:

`HapFilteredPureCubicalComplex` EXAMPLES:

`HapFilteredPureCubicalComplexFamily` EXAMPLES:

`HapFilteredRegularCWComplex` EXAMPLES:

`HapFilteredRegularCWComplexFamily` EXAMPLES:

`HapFilteredSimplicialComplex` EXAMPLES:

`HapFilteredSimplicialComplexFamily` EXAMPLES:

`HapFilteredSparseChainComplex` EXAMPLES:

`HapFilteredSparseChainComplexFamily` EXAMPLES:

`HapGChainComplex` EXAMPLES:

`HapGCocomplex` EXAMPLES:

`HapGCocomplexFamily` EXAMPLES:

`HapGComplex` EXAMPLES:

`HapGComplexFamily` EXAMPLES:

`HapGlobalDeclarationsAreAlreadyLoaded` EXAMPLES:

`HapGraph` EXAMPLES:

`HapGraphFamily` EXAMPLES:

`HapNonFreeResolution` EXAMPLES:

`HapOppositeElement` EXAMPLES:

`HapOppositeElementFamily` EXAMPLES:

`HapPureCubicalComplex` EXAMPLES:

`HapPureCubicalComplexFamily` EXAMPLES:

`HapPureCubicalLink` EXAMPLES:

`HapPureCubicalLinkFamily` EXAMPLES:

`HapPurePermutahedralComplex` EXAMPLES:

`HapPurePermutahedralComplexFamily` EXAMPLES:

`HapQuotientElement` EXAMPLES:

`HapQuotientElementFamily` EXAMPLES:

`HapRegularCWComplex` EXAMPLES:

`HapRegularCWComplexFamily` EXAMPLES:

`HapRegularCWMap` EXAMPLES:

`HapRegularCWMapFamily` EXAMPLES:

`HapResolution` EXAMPLES:

`HapResolutionFamily` EXAMPLES:

`HapSimplicialComplex` EXAMPLES:

`HapSimplicialComplexFamily` EXAMPLES:

`HapSimplicialGroup` EXAMPLES:

`HapSimplicialGroupFamily` EXAMPLES:

`HapSimplicialGroupMorphism` EXAMPLES:

`HapSimplicialGroupMorphismFamily` EXAMPLES:

`HapSimplicialMap` EXAMPLES:

`HapSimplicialMapFamily` EXAMPLES:

`HapSparseChainComplex` EXAMPLES:

`HapSparseChainComplexFamily` EXAMPLES:

`HapSparseChainMap` EXAMPLES:

`HapSparseChainMapFamily` EXAMPLES:

`HapSparseMat` EXAMPLES:

`HapSparseMatFamily` EXAMPLES:

`HomomorphismOfDirectProduct` EXAMPLES:

`IDQUASICATONEGROUP_DATA` EXAMPLES:

`IsHapChain` EXAMPLES:

IsHapCochain EXAMPLES:

IsHapComplex EXAMPLES:

IsHapFPGModule EXAMPLES:

IsHapFPGModuleHomomorphism EXAMPLES:

IsHapGChainComplex EXAMPLES:

IsHapMap EXAMPLES:

IsHapNonFreeResolution EXAMPLES:

NEATO_PATH EXAMPLES:

NerveOfCover EXAMPLES:

PseudoList EXAMPLES:

PseudoListFamily EXAMPLES:

QUASICATONEGROUP_DATA_NOT EXAMPLES:

QUASICATONEGROUP_DATA_SIZE EXAMPLES:

ReadBioData EXAMPLES:

SMALLQUASICATONEGROUP_DATA EXAMPLES:

CATONEGROUP_DATA EXAMPLES:

COMPILED EXAMPLES:

Cedric_XYXYConnQuan EXAMPLES:

Cedric_XYXYQuandles EXAMPLES:

CommutingProbability EXAMPLES:

GroupIsomorphismRepresentatives EXAMPLES:

HAPAAA EXAMPLES:

HAPAlgRedSort EXAMPLES:

HAPBARCODE EXAMPLES:

HAPDerivationType EXAMPLES:

HAPNorm EXAMPLES:

HAPPRIME_LastLHSBicomplexSize EXAMPLES:

HAPPRIME_ShuffleRandomSource EXAMPLES:

HAPQuadratic EXAMPLES:

HAPRIGXXX EXAMPLES:

HAPRightTransversalCongruenceSubgroup EXAMPLES:

HAPSqrt EXAMPLES:

HAP_BianchiAction EXAMPLES:

HAP_BianchiRepresentativesAction EXAMPLES:

HAP_GCOMPLEX_LIST EXAMPLES:

HAP_GCOMPLEX_SETUP EXAMPLES:

HAP_MOVES_DIM_2 EXAMPLES:

HAP_MOVES_DIM_3 EXAMPLES:

HAP_PERMMOVES_DIM_2 EXAMPLES:

HAP_PERMMOVES_DIM_3 EXAMPLES:

HAP_PoincareCubeManifoldEdgeDegrees EXAMPLES:

HAP_QuadToCyclotomic EXAMPLES:

HAP_QuadToCyclotomicGroup EXAMPLES:

HAP_QuadToCyclotomicMat EXAMPLES:

HAP_Test EXAMPLES:

HAP_XYXYXYXY EXAMPLES:

`HAPchildFunctionToggle` EXAMPLES:

`HAPchildToggle` EXAMPLES:

`HAPchildren` EXAMPLES:

`HapBianchiPolyhedron` EXAMPLES:

`HapBianchiPolyhedronFamily` EXAMPLES:

`HapConjQuandElt` EXAMPLES:

`HapConjQuandEltFamily` EXAMPLES:

`HapConstantPolRing` EXAMPLES:

`HapEquivariantChainComplex` EXAMPLES:

`HapEquivariantChainComplexFamily` EXAMPLES:

`HapEquivariantNonFreeChainComplex` EXAMPLES:

`HapEquivariantNonFreeChainComplexFamily` EXAMPLES:

`HapEquivariantSpectralSequencePage` EXAMPLES:

`HapEquivariantSpectralSequencePageFamily` EXAMPLES:

`HapGComplexMap` EXAMPLES:

`HapGComplexMapFamily` EXAMPLES:

`HapQuadraticNumber` EXAMPLES:

`HapQuadraticNumberFamily` EXAMPLES:

`HapQuandlePresentation` EXAMPLES:

`HapQuandlePresentationFamily` EXAMPLES:

`HapRightTransversalSLnZSubgroup` EXAMPLES:

`HapSL2ZConjugatedSubgroup` EXAMPLES:

`HapSL2ZSubgroup` EXAMPLES:

HapSimplicialFreeAbelianGroup EXAMPLES:

HapSimplicialFreeAbelianGroupFamily EXAMPLES:

HapTorsionSubcomplex EXAMPLES:

HapTorsionSubcomplexFamily EXAMPLES:

IntersectionForm EXAMPLES: 1 , 2

IsHAPRightTransversalCongruenceSubgroup EXAMPLES:

IsHapCongruenceSubgroup EXAMPLES:

IsHapRightTransversalSL3ZSubgroup EXAMPLES:

IsHapRightTransversalSLnZSubgroup EXAMPLES:

IsHapSL2ConjugatedSubgroup EXAMPLES:

IsHapSL2OSubgroup EXAMPLES:

IsHapSL2Subgroup EXAMPLES:

IsHapSL2ZConjugatedSubgroup EXAMPLES:

IsHapSL2ZSubgroup EXAMPLES:

QuadraticToCyclotomic EXAMPLES: 1

QuadraticToCyclotomicCoefficients EXAMPLES: 1

RefinedColouring_gc EXAMPLES:

RefinedColouring_group EXAMPLES:

RegularCWAssociahedronWithDiscreteVectorField EXAMPLES:

RegularCWClosedSurface EXAMPLES:

RegularCWComplexWithAttachedRelatorCells EXAMPLES: 1

RegularCWComplex_DisjointUnion EXAMPLES:

RegularCWComplex_WedgeSum EXAMPLES:

`RegularCWDiscreteSpace` EXAMPLES: 1

`RegularCWSphere` EXAMPLES: 1

`SimplicialComplexConnectedSum` EXAMPLES:

`SphericalKnotComplementWithBoundary` EXAMPLES:

`StemGroups` EXAMPLES:

`cat` EXAMPLES: 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31

`cnt` EXAMPLES:

`hap_cr` EXAMPLES:

Index

ActedGroup, 40
ActingGroup, 40
AcyclicSubomplexOfPureCubicalComplex, 131
Add, 153
AddFreeWords, 95
AddFreeWordsModP, 95
AlexanderMatrix, 138
AlexanderPolynomial, 19, 138
AlgebraicReduction, 95
Append, 153
AreIsomorphicGradedAlgebras, 34
Array, 151
ArrayAssign, 152
ArrayAssignFunctions, 152
ArrayDimension, 151
ArrayDimensions, 151
ArrayIterate, 152
ArraySum, 152
ArrayToPureCubicalComplex, 125
ArrayValue, 152
ArrayValueFunctions, 152
AutomorphismGroupAsCatOneGroup, 105
AutomorphismGroupQuandle, 143
AutomorphismGroupQuandleAsPerm, 143

BaerInvariant, 80
BarCocomplexCoboundary, 111
BarCode, 69
BarCodeCompactDisplay, 27, 70
BarCodeDisplay, 27, 70
BarComplexBoundary, 110
BarComplexEquivalence, 110
BarResolutionBoundary, 109
BarResolutionEquivalence, 110
BarResolutionHomotopy, 109
BettiNumber, 19, 20
Bettinumbers, 64, 120, 128
BigStepLCS, 161
BinaryArrayToTextFile, 153
Bogomology, 81
BogomolovMultiplier, 80
BoundaryMap, 13, 159
BoundaryMatrix, 159
BoundaryOfPureCubicalComplex, 131
BoundingPureCubicalComplex, 132

CategoricalEnrichment, 148
CategoryName, 150
CayleyGraphOfGroup, 8, 27, 123
CayleyGraphOfGroupDisplay, 87
CayleyMetric, 12, 144
CcGroup, 39, 93
CechComplexOfPureCubicalComplex, 121
Centre, 40, 104
ChainComplex, 22, 59, 121, 129, 136
ChainComplexEquivalence, 23
ChainComplexOfPair, 59, 129
ChainComplexOfQuotient, 23
ChainComplexOfRegularCWComplex, 136
ChainComplexOfSimplicialGroup, 108
ChainInclusionOfPureCubicalPair, 129
ChainMap, 23
ChainMapOfPureCubicalPairs, 129
ChainMapOfSimplicialMap, 124
ChevalleyEilenbergComplex, 59
ChildClose, 156
ChildCommand, 156
ChildCreate, 42
ChildFunction, 158
ChildGet, 156
ChildProcess, 42, 155
ChildPut, 156
ChildRead, 158
ChildReadEval, 158
Classify, 161
CliqueComplex, 14
CochainComplex, 23

Coclass, 81
CocycleCondition, 39, 93
Cohomology, 25, 26, 65
CohomologyModule, 41, 65
CohomologyPrimePart, 65
ComplementOfFilteredPureCubical-
Complex, 134
ComplementOfPureCubicalComplex, 133
Compose, 162
CompositionEqualityAdditionMinus, 150
CompositionSeriesOfFpGModules, 97
ConcentricFiltration, 14
ConjugatedResolution, 51
ConjugationQuandle, 140
ConnectedQuandle, 142
ConnectedQuandles, 141
ContractCubicalComplex, 130
ContractedComplex, 16, 130
ContractGraph, 123
ContractibleGcomplex, 89
ContractibleSubcomplex, 16
ContractibleSubcomplexOfSimplicial-
Complex, 122
ContractibleSubomplexOfPureCubical-
Complex, 131
ContractPureCubicalComplex, 130
CoreducedChainComplex, 60
CountingBaryCentricSubdividedCells, 118
CountingCellsOfACellComplex, 118
CountingControlledSubdividedCells, 118
CoxeterComplex, 89
CoxeterDiagramComponents, 112
CoxeterDiagramDegree, 112
CoxeterDiagramDisplay, 112
CoxeterDiagramFpArtinGroup, 112
CoxeterDiagramFpCoxeterGroup, 113
CoxeterDiagramIsSpherical, 113
CoxeterDiagramMatrix, 113
CoxeterDiagramVertices, 113
CoxeterSubDiagram, 113
CriticalCells, 24
CriticalCellsOfRegularCWComplex, 135
CropPureCubicalComplex, 132
CubicalComplex, 7
CubicalComplexToRegularCWComplex, 135
CupProduct, 26

Dendrogram, 133
DendrogramDisplay, 133
DendrogramMat, 22
DendrogramToPersistenceMat, 133
DesuspensionFpGModule, 98
DesuspensionMtxModule, 102
DiagonalApproximation, 24
Dimension, 127, 159
DirectProduct, 14
DirectProductGog, 104
DirectProductOfPureCubicalComplexes,
128
DirectSumOfFpGModules, 97
Display, 27
DisplayArcPresentation, 27
DisplayAvailableCellComplexes, 117
DisplayCSVKnotFile, 28
DisplayDendrogram, 28
DisplayDendrogramMat, 28
DisplayPDBfile, 28
DVFReducedCubicalComplex, 130

EfficientNormalSubgroups, 73
EilenbergMacLaneSimplicialGroup, 107
EilenbergMacLaneSimplicialGroupMap, 107
EpiCentre, 81
EquivariantChainMap, 30, 54
EquivariantEuclideanSpace, 8
EquivariantEulerCharacteristic, 118
EquivariantOrbitPolytope, 9
EquivariantSpectralSequencePage, 118
EquivariantTwoComplex, 9
EuclideanApproximatedMetric, 145
EuclideanMetric, 12
EuclideanSquaredMetric, 12, 145
EulerCharacteristic, 20, 122, 128
EulerIntegral, 20
EvaluateProperty, 159
EvenSubgroup, 113
ExcisedPureCubicalPair, 129
ExpansionOfRationalFunction, 73
ExportHapCellcomplexToDisk, 119
ExtendScalars, 55

FilteredTensorWithIntegers, 24, 57
FilteredTensorWithIntegersModP, 25
FiltrationTerm, 14

FirstQuandleAxiomIsSatisfied, 140
FpGModule, 97
FpGModuleDualBasis, 98
FpGModuleHomomorphism, 98
FpGModuleHomomorphismNC, 98
FpG_to_MtxModule, 102
FrameArray, 153
FramedPureCubicalComplex, 125
FreeGResolution, 30, 45
FundamentalDomainStandardSpaceGroup, 90
FundamentalGroup, 20, 136
FundamentalGroupOfQuotient, 21

GaussCodeKnot, 139
GeneratorsOfFpGModule, 99
GeneratorsOfMtxModule, 102
GOuterGroup, 40, 103
GOuterGroupHomomorphismNC, 103
GOuterHomomorphismTester, 103
Graph, 14
GraphDisplay, 124
GraphOfGroups, 114
GraphOfGroupsDisplay, 114
GraphOfGroupsTest, 114
GraphOfResolutions, 114
GraphOfResolutionsDisplay, 114
GraphOfSimplicialComplex, 122
GroupAlgebraAsFpGModule, 38, 99
GroupCohomology, 36, 66
GroupHomology, 36, 66
GroupHomologyOfCommutativeDiagram, 148
GroupOfResolution, 160

HammingMetric, 13, 144
HAPcopyright, 162
HAPDerivation, 34
HAPPrintTo, 157
HAPRead, 157
HasInitialObject, 149
HasTerminalObject, 149
HilbertPoincareSeries, 34
Homology, 26, 70, 120, 128
HomologyOfDerivation, 34
HomologyPb, 70
HomologyPrimePart, 71
HomologyVectorSpace, 71
HomomorphismChainToCommutativeDiagram, 147
HomotopyEquivalentMaximalPureCubical-Subcomplex, 131
HomotopyEquivalentMinimalPureCubical-Subcomplex, 131
HomotopyGraph, 15
HomotopyGroup, 105, 108
HomotopyModule, 105
HomToGModule, 41, 56
HomToIntegers, 25, 32, 33, 55
HomToIntegersModP, 55
HomToIntegralModule, 33, 55

IdConnectedQuandle, 142
IdentityAmongRelatorsDisplay, 87
IdentityArrow, 148
IdQuandle, 141
ImageOfFpGModuleHomomorphism, 99
IncidenceMatrixToGraph, 123
InduceScalars, 56
InitialArrow, 149
IntegralCohomologyGenerators, 34
IntegralCupProduct, 75
IntegralRingGenerators, 75
IntersectionOfFpGModules, 99
IsAspherical, 21, 87
IsAvailableChild, 156
IsCategoryArrow, 150
IsCategoryObject, 150
IsConnectedQuandle, 141
IsFpGModuleHomomorphismData, 99
IsLatin, 141
IsLieAlgebraHomomorphism, 162
IsPNormal, 116
IsQuandle, 140
IsQuandleEnvelope, 142
IsSuperperfect, 162

KendallMetric, 13, 144
KnotGroup, 21, 137
KnotInvariantCedric, 142
KnotReflection, 17
KnotSum, 17, 137

LefschetzNumber, 61
LeibnizAlgebraHomology, 71

LeibnizComplex, 32, 60
LeibnizQuasiCoveringHomomorphism, 85
Length, 160
LHSSpectralSequence, 35
LHSSpectralSequenceLastSheet, 35
LieAlgebraHomology, 71
LieCoveringHomomorphism, 85
LieEpiCentre, 86
LieExteriorSquare, 86
LieTensorCentre, 86
LieTensorSquare, 86
LinearHomomorphismsPersistenceMat, 163
ListToPseudoList, 153
LowerCentralSeriesLieAlgebra, 56

MakeHAPManual, 162
ManhattanMetric, 13, 145
Map, 160
Mapping, 150
MaximalSimplicesToSimplicialComplex, 122
MaximalSubmoduleOfFpGModule, 100
MaximalSubmodulesOfFpGModule, 100
Mod2CohomologyRingPresentation, 36, 76–78
ModPCohomologyGenerators, 35, 76
ModPCohomologyRing , 35, 76
ModPRingGenerators, 76
ModuleAsCatOneGroup, 106
MooreComplex, 106, 108
MorseFiltration, 132
MultipleOfFpGModule, 100
MultiplyWord, 95

Negate, 96
NegateWord, 96
Nerve, 15
NerveOfCatOneGroup, 107
NerveOfCommutativeDiagram, 147
NextAvailableChild, 156
NonabelianExteriorProduct, 81
NonabelianSymmetricKernel, 82
NonabelianSymmetricSquare, 82
NonabelianTensorProduct, 82
NonabelianTensorSquare, 83
NormalSeriesToQuotientDiagram, 147
NormalSeriesToQuotientHomomorphisms, 163
NormalSubgroupAsCatOneGroup, 106
NumberOfHomomorphisms, 140

Object, 150
OrbitPolytope, 28, 90
OrientRegularCWComplex, 17

ParallelList, 158
PartitionedNumberOfHomomorphisms, 140
PathComponent, 17
PathComponentOfPureCubicalComplex, 129
PathComponentsOfGraph, 123
PathComponentsOfSimplicialComplex, 122
PD2GC, 139
PermGroupToFilteredGraph, 146
PermToMatrixGroup, 162
PermuteArray, 151
PersistentBettiNumbers, 21, 22
PersistentCohomologyOfQuotientGroupSeries, 67
PersistentHomologyOfCommutativeDiagramOfPGroups, 68, 148
PersistentHomologyOfFilteredChainComplex, 68
PersistentHomologyOfFilteredPureCubicalComplex, 69, 134
PersistentHomologyOfPureCubicalComplex, 69
PersistentHomologyOfQuotientGroupSeries, 67
PersistentHomologyOfSubGroupSeries, 68
PiZero, 21
PlanarDiagramKnot, 139
PoincareSeries, 37, 73
PoincareSeriesLHS, 74, 78
PoincareSeriesPrimePart, 74
PolytopalComplex, 91
PolytopalGenerators, 91
Prank, 74
PresentationKnotQuandle, 139
PresentationKnotQuandleKnot, 140
PresentationOfResolution, 88
PrimePartDerivedFunctor, 37, 71
PrintZGword, 96
ProjectedFpGModule, 100

ProjectionOfPureCubicalComplex, 138
PureComplexBoundary, 17
PureComplexComplement, 18
PureComplexDifference, 18
PureComplexIntersection, 18
PureComplexInterstection, 18
PureComplexThickened, 18
PureComplexToSimplicialComplex, 121
PureComplexUnion, 18
PureCubicalComplex, 7, 125
PureCubicalComplexDifference, 126
PureCubicalComplexIntersection, 126
PureCubicalComplexToTextFile, 133
PureCubicalComplexUnion, 126
PureCubicalKnot, 8, 137
PurePermutahedralComplex, 8
PurePermutahedralKnot, 8

Quandle, 141
QuandleQuandleEnvelope, 142
Quandles, 141
QuasiIsomorph, 105
QuillenComplex, 9, 123
QuotientOfContractibleGcomplex, 90

Radical, 38
RadicalOfFpGModule, 98
RadicalSeries, 38
RadicalSeriesOfFpGModule, 99
RandomCubeOfPureCubicalComplex, 125
RandomHomomorphismOfFpGModules, 100
RandomSimplicialGraph, 9
RandomSimplicialTwoComplex, 9
Rank, 101
RankHomologyPGroup, 38, 72
RankMat, 63
RankMatDestructive, 63
RankPrimeHomology, 72
ReadCSVfileAsPureCubicalKnot, 10
ReadImageAsFilteredPureCubicalComplex, 10, 134
ReadImageAsPureCubicalComplex, 10, 126
ReadImageAsWeightFunction, 10
ReadImageSequenceAsPureCubicalComplex, 127
ReadLinkImageAsPureCubicalComplex, 126
ReadPDBfileAsPureCubicalComplex, 11, 138
ReadPDBfileAsPurePermutahedralComplex, 11
ReadPDBfileAsPurepermutahedralComplex, 11
RecalculateIncidenceNumbers, 52
ReducedSuspendedChainComplex, 60
ReduceTorsionSubcomplex, 117
RefineClassification, 161
RegularCWComplex, 15
RegularCWMap, 15
RegularCWPolytope, 7, 11
RelativeSchurMultiplier, 83
Representation of elements in the bar cocomplex, 111
Representation of elements in the bar complex, 109
Representation of elements in the bar resolution, 108
ResolutionAbelianGroup, 45
ResolutionAlmostCrystalGroup, 46
ResolutionAlmostCrystalQuotient, 46
ResolutionArithmeticGroup, 44
ResolutionArtinGroup, 46
ResolutionAsphericalPresentation, 47
ResolutionBieberbachGroup, 30, 47
ResolutionBoundaryOfWord, 96
ResolutionCoxeterGroup, 47
ResolutionCubicalCrystGroup, 31
ResolutionDirectProduct, 47
ResolutionExtension, 48
ResolutionFiniteDirectProduct, 48
ResolutionFiniteExtension, 48
ResolutionFiniteGroup, 31, 48, 49
ResolutionFiniteSubgroup, 49
ResolutionFpGModule, 53
ResolutionGraphOfGroups, 49
ResolutionGTree, 45
ResolutionNilpotentGroup, 31, 49
ResolutionNormalSeries, 31, 50
ResolutionPrimePowerGroup, 31, 50
ResolutionSL2Z, 32, 45
ResolutionSmallFpGroup, 50
ResolutionSmallGroup, 32
ResolutionSubgroup, 32, 51
ResolutionSubnormalSeries, 51
RestrictedEquivariantCWComplex, 9

ReverseSparseMat, 62
RightMultiplicationGroup, 143
RightMultiplicationGroupAsPerm, 142
RigidFacetsSubdivision, 116
RipsChainComplex, 121
RipsHomology, 69, 120

ScatterPlot, 29
SecondQuandleAxiomIsSatisfied, 140
SimplicialComplex, 11
SimplicialComplexToRegularCWComplex, 135
SimplicialGroupMap, 108
SimplicialMap, 124
SimplicialMapNC, 124
SimplicialNerveOfGraph, 124
SimplifiedComplex, 18, 19
SingularitiesOfPureCubicalComplex, 132
Size, 24, 127
SkeletonOfCubicalComplex, 131
SkeletonOfSimplicialComplex, 122
SL2Z, 161
SolutionsMatDestructive, 163
Source, 149, 160
SparseBoundaryMatrix, 64
SparseChainComplex, 63
SparseChainComplexOfRegularCWComplex, 64
SparseMat, 62
SparseRowAdd, 63
SparseRowInterchange, 63
SparseRowMult, 62
SparseSemiEchelon, 63
StandardCocycle, 39, 93
SumOfFpGModules, 101
SumOp, 101
SuspendedChainComplex, 60
SuspensionOfPureCubicalComplex, 128
SymmetricMatDisplay, 145
SymmetricMatrixToFilteredGraph, 12, 146
SymmetricMatrixToGraph, 12
SymmetricMatrixToIncidenceMatrix, 123
Syzygy, 94

Target, 149, 160
TensorCentre, 83
TensorProductOfChainComplexes, 60
TensorWithIntegers, 33, 57
TensorWithIntegersModP, 25, 33, 57
TensorWithIntegralModule, 56
TensorWithRationals, 58
TensorWithTwistedIntegers, 57
TensorWithTwistedIntegersModP, 57
TerminalArrow, 149
TestHap, 163
ThickenedPureCubicalComplex, 132
ThickeningFiltration, 16, 133
ThirdHomotopyGroupOfSuspensionB, 84
ThirdQuandleAxiomIsSatisfied, 140
TietzeReducedResolution, 44
TietzeReduction, 96
TorsionGeneratorsAbelianGroup, 88
TorsionSubcomplex, 116
TransposeOfSparseMat, 62
TreeOfGroupsToContractibleGcomplex, 114
TreeOfResolutionsToContractible-Gcomplex, 115
TruncatedGComplex, 90
TwistedTensorProduct, 51

UnframeArray, 153
UniversalBarCode, 68
UpperEpicentralSeries, 84

VectorStabilizer, 92
VectorsToFpGModuleWords, 101
VectorsToSymmetricMatrix, 13, 121, 145
ViewPureCubicalComplex, 127
ViewPureCubicalKnot, 137
VisualizeTorsionSkeleton, 117

WritePureCubicalComplexAsImage, 127

XmodToHAP, 106

ZigZagContractedComplex, 19
ZigZagContractedPureCubicalComplex, 130
ZZPersistentHomologyOfPureCubical-Complex, 69