

# Contents

<b>1</b>	<b>Resolutions of the ground ring</b>	<b>3</b>
<b>2</b>	<b>Resolutions of modules</b>	<b>5</b>
<b>3</b>	<b>Induced equivariant chain maps</b>	<b>6</b>
<b>4</b>	<b>Functors</b>	<b>7</b>
<b>5</b>	<b>Chain complexes</b>	<b>8</b>
<b>6</b>	<b>Sparse Chain complexes</b>	<b>9</b>
<b>7</b>	<b>Homology and cohomology groups</b>	<b>10</b>
<b>8</b>	<b>Poincare series</b>	<b>11</b>
<b>9</b>	<b>Cohomology ring structure</b>	<b>12</b>
<b>10</b>	<b>Cohomology rings of <math>p</math>-groups (mainly <math>p = 2</math>)</b>	<b>13</b>
<b>11</b>	<b>Commutator and nonabelian tensor computations</b>	<b>14</b>
<b>12</b>	<b>Lie commutators and nonabelian Lie tensors</b>	<b>15</b>
<b>13</b>	<b>Generators and relators of groups</b>	<b>16</b>
<b>14</b>	<b>Orbit polytopes and fundamental domains</b>	<b>17</b>
<b>15</b>	<b>Cocycles</b>	<b>18</b>
<b>16</b>	<b>Words in free <math>ZG</math>-modules</b>	<b>19</b>
<b>17</b>	<b><math>FpG</math>-modules</b>	<b>20</b>
<b>18</b>	<b>Meataxe modules</b>	<b>21</b>
<b>19</b>	<b>G-Outer Groups</b>	<b>22</b>
<b>20</b>	<b>Cat-1-groups</b>	<b>23</b>

<b>21</b>	<b>Simplicial groups</b>	<b>24</b>
<b>22</b>	<b>Coxeter diagrams and graphs of groups</b>	<b>27</b>
<b>23</b>	<b>Torsion subcomplexes</b>	<b>28</b>
<b>24</b>	<b>Simplicial Complexes</b>	<b>29</b>
<b>25</b>	<b>Cubical Complexes</b>	<b>30</b>
<b>26</b>	<b>Regular CW-Complexes</b>	<b>32</b>
<b>27</b>	<b>Knots and Links</b>	<b>33</b>
<b>28</b>	<b>Commutative diagrams and abstract categories</b>	<b>34</b>
<b>29</b>	<b>Arrays and Pseudo lists</b>	<b>35</b>
<b>30</b>	<b>Parallel Computation - Core Functions</b>	<b>36</b>
<b>31</b>	<b>Parallel Computation - Extra Functions</b>	<b>37</b>
<b>32</b>	<b>Some functions for accessing basic data</b>	<b>38</b>
<b>33</b>	<b>Miscellaneous</b>	<b>39</b>

## **Chapter 1**

# **Resolutions of the ground ring**

TietzeReducedResolution( $R$ ) Inputs a  $\mathbb{Z}G$ -resolution  $R$  and returns a  $\mathbb{Z}G$ -resolution  $S$  which is obtained from  $R$  by Tietze transformations.  
 ResolutionArithmeticGroup("PSL(4,Z)",  $n$ ) Inputs a positive integer  $n$  and one of the following strings:

"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" ,

or one of the following strings

"SL(2,Z[sqrt(-2)])" , "SL(2,Z[sqrt(-7)])" , "SL(2,Z[sqrt(-11)])" , "SL(2,Z[sqrt(-19)])" , "SL(2,Z[sqrt(-43)])" , "SL(2,Z[sqrt(-67)])" ,

It returns  $n$  terms of a free  $\mathbb{Z}G$ -resolution for the group  $G$  described by the string. (Subscripts  $_b$  ,  $_c$  ,  $_d$  denote alternative resolutions.)

Data for the first list of resolutions was provided by MATHIEU DUTOIR. Data for the second list was provided by MATHIEU DUTOIR.

FreeGResolution( $P$ ,  $n$ ) FreeGResolution( $P$ ,  $n$ ,  $p$ ) Inputs a non-free  $\mathbb{Z}G$ -resolution  $P$  with finite stabilizer group  $G$  and integer  $n$  and  $p$ .

ResolutionGTree( $P$ ,  $n$ ) Inputs a non-free  $\mathbb{Z}G$ -resolution  $P$  of dimension 1 (i.e. a  $G$ -tree) with finite stabilizer group  $G$  and integer  $n$ .

ResolutionSL2Z( $p$ ,  $n$ ) Inputs positive integers  $m, n$  and returns  $n$  terms of a  $\mathbb{Z}G$ -resolution for the group  $G = SL(2, \mathbb{Z})$ .

ResolutionAbelianGroup( $L$ ,  $n$ ) ResolutionAbelianGroup( $G$ ,  $n$ ) Inputs a list  $L := [m_1, m_2, \dots, m_d]$  of nonnegative integers and integer  $n$ .

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

ResolutionAlmostCrystalQuotient( $G$ ,  $n$ ,  $c$ ) ResolutionAlmostCrystalQuotient( $G$ ,  $n$ ,  $c$ ,  $false$ ) An almost crystallographic pcg group  $G$  and integer  $n$  and  $c$ .

ResolutionArtinGroup( $D$ ,  $n$ ) Inputs a Coxeter diagram  $D$  and an integer  $n > 1$ . It returns  $n$  terms of a free  $\mathbb{Z}G$ -resolution for the Artin group  $A(D)$ .

ResolutionAsphericalPresentation( $F$ ,  $R$ ,  $n$ ) Inputs a free group  $F$ , a set  $R$  of words in  $F$  which constitute an aspherical presentation of  $F/R$  and integer  $n$ .

ResolutionBieberbachGroup( $G$ ) ResolutionBieberbachGroup( $G$ ,  $v$ ) Inputs a torsion free crystallographic group  $G$  and integer  $v$ .

ResolutionCoxeterGroup( $D$ ,  $n$ ) Inputs a Coxeter diagram  $D$  and an integer  $n > 1$ . It returns  $k$  terms of a free  $\mathbb{Z}G$ -resolution for the Coxeter group  $C(D)$ .

ResolutionDirectProduct( $R$ ,  $S$ ) ResolutionDirectProduct( $R$ ,  $S$ , "internal") Inputs a  $\mathbb{Z}G$ -resolution  $R$  and a  $\mathbb{Z}H$ -resolution  $S$ .

ResolutionExtension( $g$ ,  $R$ ,  $S$ ) ResolutionExtension( $g$ ,  $R$ ,  $S$ , "TestFiniteness") ResolutionExtension( $g$ ,  $R$ ,  $S$ ,  $n$ ) Inputs a  $\mathbb{Z}G$ -resolution  $R$ , a  $\mathbb{Z}H$ -resolution  $S$  and integer  $n$ .

ResolutionFiniteDirectProduct( $R$ ,  $S$ ) ResolutionFiniteDirectProduct( $R$ ,  $S$ , "internal") Inputs a  $\mathbb{Z}G$ -resolution  $R$  and a  $\mathbb{Z}H$ -resolution  $S$ .

ResolutionFiniteExtension( $gensE$ ,  $gensG$ ,  $R$ ,  $n$ ) ResolutionFiniteExtension( $gensE$ ,  $gensG$ ,  $R$ ,  $n$ ,  $true$ ) Inputs a list of generators  $gensE$  for a group  $E$ , a list of generators  $gensG$  for a group  $G$ , a  $\mathbb{Z}G$ -resolution  $R$  and integer  $n$ .

ResolutionFiniteGroup( $gens$ ,  $n$ ) ResolutionFiniteGroup( $gens$ ,  $n$ ,  $true$ ) ResolutionFiniteGroup( $gens$ ,  $n$ ,  $gensG$ ) Inputs a list of generators  $gens$  for a group  $G$  and integer  $n$ .

ResolutionFiniteSubgroup( $R$ ,  $K$ ) ResolutionFiniteSubgroup( $R$ ,  $gensG$ ,  $gensK$ ) Inputs a  $\mathbb{Z}G$ -resolution  $R$  for a group  $G$  and a subgroup  $K$  of  $G$ .

ResolutionGraphOfGroups( $D$ ,  $n$ ) ResolutionGraphOfGroups( $D$ ,  $n$ ,  $L$ ) Inputs a graph of groups  $D$  and a positive integer  $n$ .

ResolutionNilpotentGroup( $G$ ,  $n$ ) ResolutionNilpotentGroup( $G$ ,  $n$ , "TestFiniteness") Inputs a nilpotent group  $G$  and integer  $n$ .

ResolutionNormalSeries( $L$ ,  $n$ ) ResolutionNormalSeries( $L$ ,  $n$ ,  $true$ ) ResolutionNormalSeries( $L$ ,  $n$ ,  $false$ ) Inputs a list of subgroups  $L_i$  of a group  $G$  and integer  $n$ .

ResolutionPrimePowerGroup( $P$ ,  $n$ ) ResolutionPrimePowerGroup( $G$ ,  $n$ ,  $p$ ) Inputs a  $p$ -group  $P$  and integer  $n > 0$ .

ResolutionSmallFpGroup( $G$ ,  $n$ ) ResolutionSmallFpGroup( $G$ ,  $n$ ,  $p$ ) Inputs a small finitely presented group  $G$  and integer  $n$ .

ResolutionSubgroup( $R$ ,  $K$ ) Inputs a  $\mathbb{Z}G$ -resolution for an (infinite) group  $G$  and a subgroup  $K$  of finite index  $|G : K|$ .

ResolutionSubnormalSeries( $L$ ,  $n$ ) Inputs a positive integer  $n$  and a list  $L = [L_1, \dots, L_k]$  of subgroups  $L_i$  of a finite group  $G$ .

TwistedTensorProduct( $R$ ,  $S$ ,  $EhomG$ ,  $GmapE$ ,  $NhomE$ ,  $NEhomN$ ,  $EltSE$ ,  $Mult$ ,  $InvE$ ) Inputs a  $\mathbb{Z}G$ -resolution  $R$ , a  $\mathbb{Z}H$ -resolution  $S$ , and a map  $E$  from  $G$  to  $H$ .

ConjugatedResolution( $R$ ,  $x$ ) Inputs a  $\mathbb{Z}G$ -resolution  $R$  and an element  $x$  from some group containing  $G$ . It returns a  $\mathbb{Z}G$ -resolution  $S$  such that  $S = R \cdot x$ .

RecalculateIncidenceNumbers( $R$ ) Inputs a  $\mathbb{Z}G$ -resolution  $R$  which arises as the cellular chain complex of a regular  $n$ -gon.

## Chapter 2

# Resolutions of modules

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

## Chapter 3

# Induced equivariant chain maps

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

## Chapter 4

# Functors

•

`ExtendScalars(R,G,EltsG)` Inputs a  $ZH$ -resolution  $R$ , a group  $G$  containing  $H$  as a subgroup, and a list  $EltsG$  of  
`HomToIntegers(X)` Inputs either a  $ZG$ -resolution  $X = R$ , or an equivariant chain map  $X = (F : R \longrightarrow S)$ . It returns  
`HomToIntegersModP(R)` Inputs a  $ZG$ -resolution  $R$  and returns the cochain complex obtained by applying  $HomZG$   
`HomToIntegralModule(R,f)` Inputs a  $ZG$ -resolution  $R$  and a group homomorphism  $f : G \longrightarrow GL_n(Z)$  to the group  
`TensorWithIntegralModule(R,f)` Inputs a  $ZG$ -resolution  $R$  and a group homomorphism  $f : G \longrightarrow GL_n(Z)$  to the group  
`HomToGModule(R,A)` Inputs a  $ZG$ -resolution  $R$  and an abelian  $G$ -outer group  $A$ . It returns the  $G$ -cocomplex obtained  
`InduceScalars(R,hom)` Inputs a  $ZQ$ -resolution  $R$  and a surjective group homomorphism  $hom : G \rightarrow Q$ . It returns  
`LowerCentralSeriesLieAlgebra(G)` `LowerCentralSeriesLieAlgebra(f)` Inputs a pcp group  $G$ . If each qu  
`TensorWithIntegers(X)` Inputs either a  $ZG$ -resolution  $X = R$ , or an equivariant chain map  $X = (F : R \longrightarrow S)$ . It  
`FilteredTensorWithIntegers(R)` Inputs a  $ZG$ -resolution  $R$  for which "filteredDimension" lies in `NamesOfCom`  
`TensorWithTwistedIntegers(X,rho)` Inputs either a  $ZG$ -resolution  $X = R$ , or an equivariant chain map  $X = (F$   
`TensorWithIntegersModP(X,p)` Inputs either a  $ZG$ -resolution  $X = R$ , or a characteristics 0 chain complex, or an  
`TensorWithTwistedIntegersModP(X,p,rho)` Inputs either a  $ZG$ -resolution  $X = R$ , or an equivariant chain map  $X$   
`TensorWithRationals(R)` Inputs a  $ZG$ -resolution  $R$  and returns the chain complex obtained by tensoring with the

## Chapter 5

# Chain complexes

`ChainComplex(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex  $T$  and returns the (often) chain complex of  $T$ .

`ChainComplexOfPair(T,S)` Inputs a pure cubical complex or cubical complex  $T$  and contractible subcomplex  $S$ . It returns the chain complex of the pair  $(T,S)$ .

`ChevalleyEilenbergComplex(X,n)` Inputs either a Lie algebra  $X = A$  (over the ring of integers  $Z$  or over a field  $K$ ) and an integer  $n$ . It returns the Chevalley-Eilenberg complex of  $X$  in degree  $n$ .

`LeibnizComplex(X,n)` Inputs either a Lie or Leibniz algebra  $X = A$  (over the ring of integers  $Z$  or over a field  $K$ ) and an integer  $n$ . It returns the Leibniz complex of  $X$  in degree  $n$ .

`SuspendedChainComplex(C)` Inputs a chain complex  $C$  and returns the chain complex  $S$  defined by applying the suspension operator to  $C$ .

`ReducedSuspendedChainComplex(C)` Inputs a chain complex  $C$  and returns the chain complex  $S$  defined by applying the reduced suspension operator to  $C$ .

`CoreducedChainComplex(C)` `CoreducedChainComplex(C,2)` Inputs a chain complex  $C$  and returns a quasi-isomorphic coreduced chain complex.

`TensorProductOfChainComplexes(C,D)` Inputs two chain complexes  $C$  and  $D$  of the same characteristic and returns their tensor product.

`LefschetzNumber(F)` Inputs a chain map  $F:C \rightarrow C$  with common source and target. It returns the Lefschetz number of  $F$ .



## Chapter 6

# Sparse Chain complexes

`SparseMat(A)` Inputs a matrix  $A$  and returns the matrix in sparse format.

`SparseRowMult(A,i,k)` Multiplies the  $i$ -th row of a sparse matrix  $A$  by  $k$ . The sparse matrix  $A$  is modified but not

`SparseRowInterchange(A,i,j,k)` Interchanges the  $i$ -th and  $j$ -th rows of a sparse matrix  $A$  by  $k$ . The sparse matrix  $A$

`SparseRowAdd(A,i,j,k)` Adds  $k$  times the  $j$ -th row to the  $i$ -th row of a sparse matrix  $A$ . The sparse matrix  $A$  is mod

`SparseSemiEchelon(A)` Converts a sparse matrix  $A$  to semi-echelon form (which means echelon form up to a perm

`RankMatDestructive(A)` Returns the rank of a sparse matrix  $A$ . The sparse matrix  $A$  is modified during the calcula

`RankMat(A)` Returns the rank of a sparse matrix  $A$ .

`SparseChainComplex(Y)` Inputs a regular CW-complex  $Y$  and returns a sparse chain complex which is chain homo

`SparseChainComplexOfRegularCWComplex(Y)` Inputs a regular CW-complex  $Y$  and returns its cellular chain com

`SparseBoundaryMatrix(C,n)` Inputs a sparse chain complex  $C$  and integer  $n$ . Returns the  $n$ -th boundary matrix of

`Bettinnumbers(C,n)` Inputs a sparse chain complex  $C$  and integer  $n$ . Returns the  $n$ -th Netti number of the chain con

## Chapter 7

# Homology and cohomology groups

`Cohomology(X,n)` Inputs either a cochain complex  $X = C$  (or  $G$ -cocomplex  $C$ ) or a cochain map  $X = (C \longrightarrow D)$  in  
`CohomologyModule(C,n)` Inputs a  $G$ -cocomplex  $C$  together with a non-negative integer  $n$ . It returns the cohomology  
`CohomologyPrimePart(C,n,p)` Inputs a cochain complex  $C$  in characteristic 0, a positive integer  $n$ , and a prime  $p$ .  
`GroupCohomology(X,n)` `GroupCohomology(X,n,p)` Inputs a positive integer  $n$  and either a finite group  $X = G$  or a nilpotent  
`GroupHomology(X,n)` `GroupHomology(X,n,p)` Inputs a positive integer  $n$  and either a finite group  $X = G$  or a nilpotent  
`PersistentHomologyOfQuotientGroupSeries(S,n)` `PersistentHomologyOfQuotientGroupSeries(S,n,p)`  
`PersistentCohomologyOfQuotientGroupSeries(S,n)` `PersistentCohomologyOfQuotientGroupSeries(S,n,p)`  
`UniversalBarCode("UpperCentralSeries",n,d)` `UniversalBarCode("UpperCentralSeries",n,d,k)` Inputs  
`PersistentHomologyOfSubGroupSeries(S,n)` `PersistentHomologyOfSubGroupSeries(S,n,p,Resolution)`  
`PersistentHomologyOfFilteredChainComplex(C,n,p)` Inputs a filtered chain complex  $C$  (of characteristic 0 or  
`PersistentHomologyOfCommutativeDiagramOfPGroups(D,n)` Inputs a commutative diagram  $D$  of finite  $p$ -groups  
`PersistentHomologyOfPureCubicalComplex(L,n,p)` Inputs a positive integer  $n$ , a prime  $p$  and an increasing chain  
`ZZPersistentHomologyOfPureCubicalComplex(L,n,p)` Inputs a positive integer  $n$ , a prime  $p$  and any sequence  
`RipsHomology(G,n)` `RipsHomology(G,n,p)` Inputs a graph  $G$ , a non-negative integer  $n$  (and optionally a prime number  
`BarCode(P)` Inputs an integer persistence matrix  $P$  and returns the same information in the form of a binary matrix (or  
`BarCodeDisplay(P)` `BarCodeDisplay(P,"mozilla")` `BarCodeCompactDisplay(P)` `BarCodeCompactDisplay(P,"mozilla")`  
`Homology(X,n)` Inputs either a chain complex  $X = C$  or a chain map  $X = (C \longrightarrow D)$ . If  $X = C$  then the torsion coefficients  
`HomologyPb(C,n)` This is a back-up function which might work in some instances where `Homology(C,n)` fails. It is  
`HomologyVectorSpace(X,n)` Inputs either a chain complex  $X = C$  or a chain map  $X = (C \longrightarrow D)$  in prime characteristic  
`HomologyPrimePart(C,n,p)` Inputs a chain complex  $C$  in characteristic 0, a positive integer  $n$ , and a prime  $p$ . It returns  
`LeibnizAlgebraHomology(A,n)` Inputs a Lie or Leibniz algebra  $X = A$  (over the ring of integers  $Z$  or over a field  $K$ ) and a positive integer  
`LieAlgebraHomology(A,n)` Inputs a Lie algebra  $A$  (over the integers or a field) and a positive integer  $n$ . It returns the  
`PrimePartDerivedFunctor(G,R,F,n)` Inputs a finite group  $G$ , a positive integer  $n$ , at least  $n+1$  terms of a  $ZP$ -resolution  
`RankHomologyPGroup(G,n)` `RankHomologyPGroup(R,n)` `RankHomologyPGroup(G,n,"empirical")` Inputs a (smallish)  $p$ -group  
`RankPrimeHomology(G,n)` Inputs a (smallish)  $p$ -group  $G$  together with a positive integer  $n$ . It returns a function `dim`

## Chapter 8

# Poincare series

•  
EfficientNormalSubgroups( $G$ ) EfficientNormalSubgroups( $G, k$ ) Inputs a prime-power group  $G$  and, optionally, a positive integer  $k$ . It returns a list of normal subgroups of  $G$  of index at most  $k$ .  
ExpansionOfRationalFunction( $f, n$ ) Inputs a positive integer  $n$  and a rational function  $f(x) = p(x)/q(x)$  where  $p(x)$  and  $q(x)$  are polynomials with integer coefficients. It returns the expansion of  $f(x)$  as a power series in  $x$  up to order  $n$ .  
PoincareSeries( $G, n$ ) PoincareSeries( $R, n$ ) PoincareSeries( $L, n$ ) PoincareSeries( $G$ ) Inputs a finite group  $G$ , a finite ring  $R$ , a finite lattice  $L$ , or a finite group  $G$ , and a positive integer  $n$ . It returns the Poincare series of  $G$  up to order  $n$ .  
PoincareSeriesPrimePart( $G, p, n$ ) Inputs a finite group  $G$ , a prime  $p$ , and a positive integer  $n$ . It returns a quotient of polynomials whose numerator and denominator are polynomials with integer coefficients.  
PoincareSeriesLHS( $G$ ) Inputs a finite 2-group  $G$  and returns a quotient of polynomials  $f(x) = P(x)/Q(x)$  whose numerator and denominator are polynomials with integer coefficients.  
Prank( $G$ ) Inputs a  $p$ -group  $G$  and returns the rank of the largest elementary abelian subgroup.

## Chapter 9

# Cohomology ring structure

`IntegralCupProduct(R,u,v,p,q)`    `IntegralCupProduct(R,u,v,p,q,P,Q,N)` (Various functions used to compute cup products in integral cohomology rings.)  
`IntegralRingGenerators(R,n)` Inputs at least  $n+1$  terms of a  $ZG$ -resolution and integer  $n > 0$ . It returns a minimal set of generators for the integral cohomology ring.  
`ModPCohomologyGenerators(G,n)` `ModPCohomologyGenerators(R)` Inputs either a  $p$ -group  $G$  and positive integer  $n$ , or a finite group  $G$  and a prime  $p$ . It returns a minimal set of generators for the mod  $p$  cohomology ring.  
`ModPCohomologyRing(G,n)` `ModPCohomologyRing(G,n,level)` `ModPCohomologyRing(R)` `ModPCohomologyRing(R,n)` Inputs either a  $p$ -group  $G$  and positive integer  $n$ , or a finite group  $G$  and a prime  $p$ , or a finite group  $G$  and a prime  $p$  and a positive integer  $n$ . It returns the mod  $p$  cohomology ring.  
`ModPRingGenerators(A)` Inputs a mod  $p$  cohomology ring  $A$  (created using the preceding function). It returns a minimal set of generators for the mod  $p$  cohomology ring.  
`Mod2CohomologyRingPresentation(G)` `Mod2CohomologyRingPresentation(G,n)` `Mod2CohomologyRingPresentation(G,n,level)` Inputs either a 2-group  $G$  and positive integer  $n$ , or a finite group  $G$  and a prime  $p=2$ . It returns a presentation for the mod 2 cohomology ring.

## Chapter 10

# 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.)

•  
| `Mod2CohomologyRingPresentation(G)` `Mod2CohomologyRingPresentation(G,n)` `Mod2CohomologyRingP`  
| `PoincareSeriesLHS(G)` Inputs a finite 2-group  $G$  and returns a quotient of polynomials  $f(x) = P(x)/Q(x)$  whose

## Chapter 11

# Commutator and nonabelian tensor computations

`BaerInvariant(G,c)` Inputs a nilpotent group  $G$  and integer  $c>0$ . It returns the Baer invariant  $M^{(c)}(G)$  defined as  
`Coclass(G)` Inputs a group  $G$  of prime-power order  $p^n$  and nilpotency class  $c$  say. It returns the integer  $r = n - c$ .  
`EpiCentre(G,N)` `EpiCentre(G)` Inputs a finite group  $G$  and normal subgroup  $N$  and returns a subgroup  $Z^*(G,N)$   
`NonabelianExteriorProduct(G,N)` Inputs a finite group  $G$  and normal subgroup  $N$ . It returns a record  $E$  with the f  
`NonabelianSymmetricKernel(G)` `NonabelianSymmetricKernel(G,m)` Inputs a finite or nilpotent infinite gro  
`NonabelianSymmetricSquare(G)` `NonabelianSymmetricSquare(G,m)` Inputs a finite or nilpotent infinite gro  
`NonabelianTensorProduct(G,N)` Inputs a finite group  $G$  and normal subgroup  $N$ . It returns a record  $E$  with the f  
`NonabelianTensorSquare(G)` `NonabelianTensorSquare(G,m)` Inputs a finite or nilpotent infinite group  $G$  and  
`RelativeSchurMultiplier(G,N)` Inputs a finite group  $G$  and normal subgroup  $N$ . It returns the homology group  
`TensorCentre(G)` Inputs a group  $G$  and returns the largest central subgroup  $N$  such that the induced homomorphis  
`ThirdHomotopyGroupOfSuspensionB(G)` `ThirdHomotopyGroupOfSuspensionB(G,m)` Inputs a finite or nilpo  
`UpperEpicentralSeries(G,c)` Inputs a nilpotent group  $G$  and an integer  $c$ . It returns the  $c$ -th term of the upper e

## Chapter 12

# Lie commutators and nonabelian Lie tensors

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

`LieCoveringHomomorphism(L)` Inputs a finite dimensional Lie algebra  $L$  over a field, and returns a surjective Lie h

`LeibnizQuasiCoveringHomomorphism(L)` Inputs a finite dimensional Lie algebra  $L$  over a field, and returns a surj

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

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

`LieTensorSquare(L)` Inputs a finite dimensional Lie algebra  $L$  over a field and returns a record  $T$  with the followi

`LieTensorCentre(L)` Inputs a finite dimensional Lie algebra  $L$  over a field and returns the largest ideal  $N$  such tha

## Chapter 13

# Generators and relators of groups

•

<code>CayleyGraphOfGroupDisplay(G,X)</code>	<code>CayleyGraphOfGroupDisplay(G,X,"mozilla")</code>	Inputs a finite group $G$ and a set $X$ of generators of $G$ .
<code>IdentityAmongRelatorsDisplay(R,n)</code>	<code>IdentityAmongRelatorsDisplay(R,n,"mozilla")</code>	Inputs a free $Z$ -module $R$ and a natural number $n$ .
<code>IsAspherical(F,R)</code>		Inputs a free group $F$ and a set $R$ of words in $F$ . It performs a test on the 2-dimensional CW-complex with presentation $\langle F, R \rangle$ .
<code>PresentationOfResolution(R)</code>		Inputs at least two terms of a reduced $ZG$ -resolution $R$ and returns a record $P$ with the presentation of the resolution.
<code>TorsionGeneratorsAbelianGroup(G)</code>		Inputs an abelian group $G$ and returns a generating set $[x_1, \dots, x_n]$ where $n$ is the rank of $G$ .



## Chapter 14

# Orbit polytopes and fundamental domains

`CoxeterComplex(D)` `CoxeterComplex(D,n)` Inputs a Coxeter diagram  $D$  of finite type. It returns a non-free  $ZW$ -resolution of the Coxeter group  $W(D)$ .  
`ContractibleGcomplex("PSL(4,Z)")` Inputs one of the following strings:

"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" ,

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. (

Data for the first list of non-free resolutions was provided provided by MATHIEU DUTOIR. Data for the second list is from [1].  
`QuotientOfContractibleGcomplex(C,D)` Inputs a non-free  $ZG$ -resolution  $C$  and a finite subgroup  $D$  of  $G$  which is normal in  $G$ . It returns the non-free  $ZG$ -resolution of the quotient group  $G/D$ .  
`TruncatedGComplex(R,m,n)` Inputs a non-free  $ZG$ -resolution  $R$  and two positive integers  $m$  and  $n$ . It returns the non-free  $ZG$ -resolution of the quotient group  $G/D$ .  
`FundamentalDomainStandardSpaceGroup(v,G)` Inputs a crystallographic group  $G$  (represented using `AffineCrys`) and a rational vector  $v$  of length  $n$ . It returns the fundamental domain of the standard space group  $G$ .  
`OrbitPolytope(G,v,L)` Inputs a permutation group or matrix group  $G$  of degree  $n$  and a rational vector  $v$  of length  $n$ . It returns the orbit polytope of  $v$  under the action of  $G$ .  
`PolytopalComplex(G,v)` `PolytopalComplex(G,v,n)` Inputs a permutation group or matrix group  $G$  of degree  $n$  and a rational vector  $v$  of length  $n$ . It returns the polytopal complex of  $v$  under the action of  $G$ .  
`PolytopalGenerators(G,v)` Inputs a permutation group or matrix group  $G$  of degree  $n$  and a rational vector  $v$  of length  $n$ . It returns the generators of the polytopal complex of  $v$  under the action of  $G$ .  
`VectorStabilizer(G,v)` Inputs a permutation group or matrix group  $G$  of degree  $n$  and a rational vector  $v$  of length  $n$ . It returns the stabilizer of  $v$  in  $G$ .

## Chapter 15

# Cocycles

•

<code>CcGroup(A,f)</code>	Inputs a $G$ -module $A$ (i.e. an abelian $G$ -outer group) and a standard 2-cocycle $f: G \times G \rightarrow A$ . It returns a $G$ -module $A$ with the standard 2-cocycle $f$ .
<code>CocycleCondition(R,n)</code>	Inputs a resolution $R$ and an integer $n > 0$ . It returns an integer matrix $M$ with the following property: $M \cdot \text{cocycles}(R, n) = 0$ .
<code>StandardCocycle(R,f,n)</code>	Inputs a resolution $R$ and a standard 2-cocycle $f: G \times G \rightarrow A$ . It returns a standard 2-cocycle $f$ on $R$ .
<code>StandardCocycle(R,f,n,q)</code>	Inputs a $ZG$ -resolution $R$ (with contracting homotopy), a positive integer $n$ and an integer $q$ . It returns a standard 2-cocycle $f$ on $R$ .
<code>Syzygy(R,g)</code>	Inputs a $ZG$ -resolution $R$ (with contracting homotopy) and a list $g = [g[1], \dots, g[n]]$ of elements in $G$ . It returns a syzygy of $g$ .

## Chapter 16

# Words in free $ZG$ -modules

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

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

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

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

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

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

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

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

`TietzeReduction(S,w)` Inputs a set  $S$  of words in a free  $ZG$ -module, and a word  $w$  in the module. The function returns a reduced version of  $w$  modulo  $S$ .

`ResolutionBoundaryOfWord(R,n,w)` Inputs a resolution  $R$ , a positive integer  $n$  and a list  $w$  representing a word in the free  $ZG$ -module. It returns the boundary of the word  $w$  in the resolution  $R$  at level  $n$ .

## Chapter 17

### $FpG$ -modules

`CompositionSeriesOfFpGModules(M)` Inputs an  $FpG$ -module  $M$  and returns a list of  $FpG$ -modules that constitute a composition series for  $M$ .

`DirectSumOfFpGModules(M,N)` `DirectSumOfFpGModules([ M[1], M[2], ..., M[k] ])` Inputs two  $FpG$ -modules  $M$  and  $N$  or a list of  $FpG$ -modules  $M_1, M_2, \dots, M_k$  and returns their direct sum.

`FpGModule(A,P)` `FpGModule(A,G,p)` Inputs a  $p$ -group  $P$  and a matrix  $A$  whose rows have length a multiple of the order of  $P$  and returns the  $FpG$ -module defined by  $A$  and  $P$ .

`FpGModuleDualBasis(M)` Inputs an  $FpG$ -module  $M$ . It returns a record  $R$  with two components:  $R.freeModule$  is a free  $FpG$ -module isomorphic to  $M$  and  $R.dualBasis$  is a dual basis for  $M$ .

`FpGModuleHomomorphism(M,N,A)` `FpGModuleHomomorphismNC(M,N,A)` Inputs  $FpG$ -modules  $M$  and  $N$  over a common group  $G$  and a matrix  $A$  and returns a homomorphism from  $M$  to  $N$ .

`DesuspensionFpGModule(M,n)` `DesuspensionFpGModule(R,n)` Inputs a positive integer  $n$  and an  $FpG$ -module  $M$  or a record  $R$  and returns the desuspension of  $M$  by  $n$ .

`RadicalOfFpGModule(M)` Inputs an  $FpG$ -module  $M$  with  $G$  a  $p$ -group, and returns the Radical of  $M$  as an  $FpG$ -module.

`RadicalSeriesOfFpGModule(M)` Inputs an  $FpG$ -module  $M$  and returns a list of  $FpG$ -modules that constitute the radical series of  $M$ .

`GeneratorsOfFpGModule(M)` Inputs an  $FpG$ -module  $M$  and returns a matrix whose rows correspond to a minimal generating set for  $M$ .

`ImageOfFpGModuleHomomorphism(f)` Inputs an  $FpG$ -module homomorphism  $f : M \rightarrow N$  and returns its image.

`GroupAlgebraAsFpGModule(G)` Inputs a  $p$ -group  $G$  and returns its mod  $p$  group algebra as an  $FpG$ -module.

`IntersectionOfFpGModules(M,N)` Inputs two  $FpG$ -modules  $M, N$  arising as submodules in a common free module and returns their intersection.

`IsFpGModuleHomomorphismData(M,N,A)` Inputs  $FpG$ -modules  $M$  and  $N$  over a common  $p$ -group  $G$ . Also inputs a matrix  $A$  and returns a record with fields `isHomomorphism` and `data`.

`MaximalSubmoduleOfFpGModule(M)` Inputs an  $FpG$ -module  $M$  and returns one maximal  $FpG$ -submodule of  $M$ .

`MaximalSubmodulesOfFpGModule(M)` Inputs an  $FpG$ -module  $M$  and returns the list of maximal  $FpG$ -submodules of  $M$ .

`MultipleOfFpGModule(w,M)` Inputs an  $FpG$ -module  $M$  and a list  $w := [g_1, \dots, g_t]$  of elements in the group  $G = M$  and returns the multiple of  $M$  by  $w$ .

`ProjectedFpGModule(M,k)` Inputs an  $FpG$ -module  $M$  of ambient dimension  $n|G|$ , and an integer  $k$  between 1 and  $n$  and returns the projection of  $M$  onto a subspace of dimension  $k|G|$ .

`RandomHomomorphismOfFpGModules(M,N)` Inputs two  $FpG$ -modules  $M$  and  $N$  over a common group  $G$ . It returns a random homomorphism from  $M$  to  $N$ .

`Rank(f)` Inputs an  $FpG$ -module homomorphism  $f : M \rightarrow N$  and returns the dimension of the image of  $f$  as a vector space over  $F$ .

`SumOfFpGModules(M,N)` Inputs two  $FpG$ -modules  $M, N$  arising as submodules in a common free module  $(FG)^n$  and returns their sum.

`SumOp(f,g)` Inputs two  $FpG$ -module homomorphisms  $f, g : M \rightarrow N$  with common source and common target. It returns the sum  $f + g$ .

`VectorsToFpGModuleWords(M,L)` Inputs an  $FpG$ -module  $M$  and a list  $L = [v_1, \dots, v_k]$  of vectors in  $M$ . It returns a list of words representing the vectors in  $L$ .

## Chapter 18

# Meataxe modules

•  
DesuspensionMtxModule( $M$ ) Inputs a meataxe module  $M$  over the field of  $p$  elements and returns an FpG-module.  
FpG\_to\_MtxModule( $M$ ) Inputs an FpG-module  $M$  and returns an isomorphic meataxe module.  
GeneratorsOfMtxModule( $M$ ) Inputs a meataxe module  $M$  acting on, say, the vector space  $V$ . The function returns a

## Chapter 19

# G-Outer Groups

`GOuterGroup(E,N)` `GOuterGroup()` Inputs a group  $E$  and normal subgroup  $N$ . It returns  $N$  as a  $G$ -outer group where  $G = E/N$ .  
`GOuterGroupHomomorphismNC(A,B,phi)` `GOuterGroupHomomorphismNC()` Inputs  $G$ -outer groups  $A$  and  $B$  with common acting group  $G$ , and a group homomorphism  $\phi: A \rightarrow B$ . It returns a  $G$ -outer group homomorphism from  $A$  to  $B$ .  
`GOuterHomomorphismTester(A,B,phi)` Inputs  $G$ -outer groups  $A$  and  $B$  with common acting group  $G$ , and a group homomorphism  $\phi: A \rightarrow B$ . It returns a boolean value indicating whether  $\phi$  is a  $G$ -outer group homomorphism.  
`Centre(A)` Inputs  $G$ -outer group  $A$  and returns the group theoretic centre of  $\text{ActedGroup}(A)$  as a  $G$ -outer group.  
`DirectProductGog(A,B)` `DirectProductGog(Lst)` Inputs  $G$ -outer groups  $A$  and  $B$  with common acting group  $G$ , and a list of  $G$ -outer groups. It returns the direct product of the  $G$ -outer groups as a  $G$ -outer group.

## Chapter 20

# Cat-1-groups

`AutomorphismGroupAsCatOneGroup(G)` Inputs a group  $G$  and returns the Cat-1-group  $C$  corresponding to the cross

`HomotopyGroup(C,n)` Inputs a cat-1-group  $C$  and an integer  $n$ . It returns the  $n$ th homotopy group of  $C$ .

`HomotopyModule(C,2)` Inputs a cat-1-group  $C$  and an integer  $n=2$ . It returns the second homotopy group of  $C$  as a  $C$ -

`QuasiIsomorph(C)` Inputs a cat-1-group  $C$  and returns a cat-1-group  $D$  for which there exists some homomorphism

`ModuleAsCatOneGroup(G,alpha,M)` Inputs a group  $G$ , an abelian group  $M$  and a homomorphism  $\alpha: G \rightarrow \text{Aut}(M)$ .

`MooreComplex(C)` Inputs a cat-1-group  $C$  and returns its Moore complex as a  $G$ -complex (i.e. as a complex of group

`NormalSubgroupAsCatOneGroup(G,N)` Inputs a group  $G$  with normal subgroup  $N$ . It returns the Cat-1-group  $C$  cor

`XmodToHAP(C)` Inputs a cat-1-group  $C$  obtained from the Xmod package and returns a cat-1-group  $D$  for which `IsHa`

## **Chapter 21**

# **Simplicial groups**



`NerveOfCatOneGroup(G, n)` Inputs a cat-1-group  $G$  and a positive integer  $n$ . It returns the low-dimensional part of  $f$

This function applies both to cat-1-groups for which `IsHapCatOneGroup(G)` is true, and to cat-1-groups produced us

This function was implemented by VAN LUYEN LE.

`EilenbergMacLaneSimplicialGroup(G, n, dim)` Inputs a group  $G$ , a positive integer  $n$ , and a positive integer  $dim$

This function was implemented by VAN LUYEN LE.

`EilenbergMacLaneSimplicialGroupMap(f, n, dim)` Inputs a group homomorphism  $f : G \rightarrow Q$ , a positive integer

This function was implemented by VAN LUYEN LE.

`MooreComplex(G)` Inputs a simplicial group  $G$  and returns its Moore complex as a  $G$ -complex.

This function was implemented by VAN LUYEN LE.

`ChainComplexOfSimplicialGroup(G)` Inputs a simplicial group  $G$  and returns the cellular chain complex  $C$  of a  $C$

This function was implemented by VAN LUYEN LE.

`SimplicialGroupMap(f)` Inputs a homomorphism  $f : G \rightarrow Q$  of simplicial groups. The function returns an induced

This function was implemented by VAN LUYEN LE.

`HomotopyGroup(G, n)` Inputs a simplicial group  $G$  and a positive integer  $n$ . The integer  $n$  must be less than the length

Representation of elements in the bar resolution For a group  $G$  we denote by  $B_n(G)$  the free  $\mathbb{Z}G$ -modu

We represent a word

$$w = h_1 \cdot [g_{11} | g_{12} | \dots | g_{1n}] - h_2 \cdot [g_{21} | g_{22} | \dots | g_{2n}] + \dots + h_k \cdot [g_{k1} | g_{k2} | \dots | g_{kn}]$$

in  $B_n(G)$  as a list of lists:

$$[[+1, h_1, g_{11}, g_{12}, \dots, g_{1n}], [-1, h_2, g_{21}, g_{22}, \dots, g_{2n}] + \dots + [+1, h_k, g_{k1}, g_{k2}, \dots, g_{kn}].$$

`BarResolutionBoundary(w)` This function inputs a word  $w$  in the bar resolution module  $B_n(G)$  and returns its ima

This function was implemented by VAN LUYEN LE.

`BarResolutionHomotopy(w)` This function inputs a word  $w$  in the bar resolution module  $B_n(G)$  and returns its ima

This function is currently being implemented by VAN LUYEN LE.

Representation of elements in the bar complex For a group  $G$  we denote by  $BC_n(G)$  the free abelian group

We represent a word

$$w = [g_{11} | g_{12} | \dots | g_{1n}] - [g_{21} | g_{22} | \dots | g_{2n}] + \dots + [g_{k1} | g_{k2} | \dots | g_{kn}]$$

in  $BC_n(G)$  as a list of lists:

$$[[+1, g_{11}, g_{12}, \dots, g_{1n}], [-1, g_{21}, g_{22}, \dots, g_{2n}] + \dots + [+1, g_{k1}, g_{k2}, \dots, g_{kn}].$$

`BarComplexBoundary(w)` This function inputs a word  $w$  in the  $n$ -th term of the bar complex  $BC_n(G)$  and returns its

This function was implemented by VAN LUYEN LE.

`BarResolutionEquivalence(R)` This function inputs a free  $ZG$ -resolution  $R$ . It returns a component object HE wi

$$equiv(n, -): B_n(G) \rightarrow B_{n+1}(G)$$

satisfying  $w - \psi(\phi(w)) = d(n+1, equiv(n, w)) + equiv(n-1, d(n, w))$ . where  $d(n, -): B_n(G) \rightarrow B_{n-1}(G)$  is the boundar

This function was implemented by VAN LUYEN LE.

`BarComplexEquivalence(R)`

This function inputs a free  $ZG$ -resolution  $R$ . It first constructs the chain complex  $T = \text{TensorWithIntegers}(R)$ . The function returns a component object HE with components

- $\text{HE}!.phi(n, w)$  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  $\phi: BC_n(G) \rightarrow T_n$ .
- $\text{HE}!.psi(n, w)$  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  $\psi: T_n \rightarrow BC_n(G)$ .
- $\text{HE}!.equiv(n, w)$  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  $equiv(n, -): BC_n(G) \rightarrow BC_{n+1}(G)$  satisfying

$$w - \psi(\phi(w)) = d(n+1, equiv(n, w)) + equiv(n-1, d(n, w)).$$

where  $d(n, -): BC_n(G) \rightarrow BC_{n-1}(G)$  is the boundary homomorphism in the bar complex.

This function was implemented by VAN LUYEN LE.

`Representation of elements in the bar cocomplex`

For a group  $G$  we denote by  $BC^n(G)$  the free abelian group with basis the lists  $[g_1|g_2|\dots|g_n]$  where the  $g_i$  range over  $G$ .

We represent a word

$$w = [g_{11}|g_{12}|\dots|g_{1n}] - [g_{21}|g_{22}|\dots|g_{2n}] + \dots + [g_{k1}|g_{k2}|\dots|g_{kn}]$$

in  $BC^n(G)$  as a list of lists:

$$[[+1, g_{11}, g_{12}, \dots, g_{1n}], [-1, g_{21}, g_{22}, \dots, g_{2n}] + \dots + [+1, g_{k1}, g_{k2}, \dots, g_{kn}].$$

`BarCocomplexCoboundary(w)`

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  $d^n: BC^n(G) \rightarrow BC^{n+1}(G)$  in the bar cocomplex.

This function was implemented by VAN LUYEN LE.

## Chapter 22

# Coxeter diagrams and graphs of groups

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

CoxeterDiagramDegree(D, v) Inputs a Coxeter diagram  $D$  and vertex  $v$ . It returns the degree of  $v$  (i.e. the number of edges incident to  $v$ ).

CoxeterDiagramDisplay(D) CoxeterDiagramDisplay(D, "web browser") Inputs a Coxeter diagram  $D$  and displays it in a web browser.

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

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

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

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

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

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

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

GraphOfGroupsDisplay(D) GraphOfGroupsDisplay(D, "web browser") Inputs a graph of groups  $D$  and displays it in a web browser.

GraphOfResolutions(D, n) Inputs a graph of groups  $D$  and a positive integer  $n$ . It returns a graph of resolutions of  $D$  of order  $n$ .

GraphOfGroups(D) Inputs a graph of resolutions  $D$  and returns the corresponding graph of groups.

GraphOfResolutionsDisplay(D) Inputs a graph of resolutions  $D$  and displays it as a .gif file. It uses the Mozilla browser.

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

TreeOfGroupsToContractibleGcomplex(D, G) Inputs a graph of groups  $D$  which is a tree, and also inputs the fundamental group  $G$ .

TreeOfResolutionsToContractibleGcomplex(D, G) Inputs a graph of resolutions  $D$  which is a tree, and also inputs the fundamental group  $G$ .

#

## Chapter 23

# Torsion subcomplexes

The torsion subcomplexes subpackage has been conceived and implemented by ALEXANDER D. RAHM.

`IsPnormal( G, p)` Inputs a finite group  $G$  and a prime  $p$ . Checks if the group  $G$  is  $p$ -normal for the prime  $p$ . Zassenhaus.

`TorsionSubcomplex( groupName, p)` Inputs a cell complex with action of a group. In HAP, presently the following

"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  $\mathbb{Q}(\sqrt{-m})$ , the latter

The function `TorsionSubcomplex` prints the cells with  $p$ -torsion in their stabilizer on the screen and returns the incidence

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" ,

provided by MATHIEU DUTOUR.

`DisplayAvailableCellComplexes()` ; Displays the cell complexes that are available in HAP.

`VisualizeTorsionSkeleton( groupName, p)` Executes the function `TorsionSubcomplex( groupName, p)` and visualizes

`ReduceTorsionSubcomplex( groupName, p)` This function start with the same operations as the function `TorsionSubcomplex`

It prints on the screen which cells to merge and which edges to cut off in order to reduce the  $p$ -torsion subcomplex w

## Chapter 24

# Simplicial Complexes

`Homology(T,n)` `Homology(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex  $T$  and a non-negative integer  $n$ . It returns the  $n$ -th homology group of  $T$ .

`RipsHomology(G,n)` `RipsHomology(G,n,p)` Inputs a graph  $G$ , a non-negative integer  $n$  (and optionally a prime number  $p$ ). It returns the  $n$ -th Rips homology group of  $G$ .

`Bettinnumbers(T,n)` `Bettinnumbers(T)` Inputs a pure cubical complex, or cubical complex, simplicial complex  $T$  and a non-negative integer  $n$ . It returns the  $n$ -th Bettin number of  $T$ .

`ChainComplex(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex  $T$  and returns the (often infinite) chain complex of  $T$ .

`CechComplexOfPureCubicalComplex(T)` Inputs a  $d$ -dimensional pure cubical complex  $T$  and returns a simplicial complex whose simplices are the non-empty intersections of the  $d$ -cubes of  $T$ .

`PureComplexToSimplicialComplex(T,k)` Inputs either a  $d$ -dimensional pure cubical complex  $T$  or a  $d$ -dimensional cubical complex  $T$  and a non-negative integer  $k$ . It returns the simplicial complex of  $T$  of dimension at most  $k$ .

`RipsChainComplex(G,n)` Inputs a graph  $G$  and a non-negative integer  $n$ . It returns  $n+1$  terms of a chain complex whose homology is the  $n$ -th Rips homology group of  $G$ .

`VectorsToSymmetricMatrix(M)` `VectorsToSymmetricMatrix(M,distance)` Inputs a matrix  $M$  of rational numbers and a non-negative integer  $distance$ . It returns a symmetric matrix  $S$  such that  $S_{ij} = M_{ij} - distance$ .

`EulerCharacteristic(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex  $T$  and returns the Euler characteristic of  $T$ .

`MaximalSimplicesToSimplicialComplex(L)` Inputs a list  $L$  whose entries are lists of vertices representing the maximal simplices of a simplicial complex. It returns the simplicial complex.

`SkeletonOfSimplicialComplex(S,k)` Inputs a simplicial complex  $S$  and a positive integer  $k$  less than or equal to the dimension of  $S$ . It returns the  $k$ -skeleton of  $S$ .

`GraphOfSimplicialComplex(S)` Inputs a simplicial complex  $S$  and returns the graph of  $S$ .

`ContractibleSubcomplexOfSimplicialComplex(S)` Inputs a simplicial complex  $S$  and returns a (probably maximal) contractible subcomplex of  $S$ .

`PathComponentsOfSimplicialComplex(S,n)` Inputs a simplicial complex  $S$  and a nonnegative integer  $n$ . If  $n=0$  it returns the number of path components of  $S$ . If  $n>0$  it returns the number of path components of the  $n$ -skeleton of  $S$ .

`QuillenComplex(G)` Inputs a finite group  $G$  and returns, as a simplicial complex, the order complex of the poset of proper subgroups of  $G$ .

`SymmetricMatrixToIncidenceMatrix(S,t)` `SymmetricMatrixToIncidenceMatrix(S,t,d)` Inputs a symmetric 0/1 matrix  $S$  and a non-negative integer  $t$  (and optionally a non-negative integer  $d$ ). It returns a matrix  $M$  such that  $M_{ij} = S_{ij} - t$  (or  $M_{ij} = S_{ij} - t + d$  if  $d>0$ ).

`IncidenceMatrixToGraph(M)` Inputs a symmetric 0/1 matrix  $M$ . It returns the graph with one vertex for each row of  $M$  and edges between vertices  $i$  and  $j$  if  $M_{ij} = 1$ .

`CayleyGraphOfGroup(G,A)` Inputs a group  $G$  and a set  $A$  of generators. It returns the Cayley graph.

`PathComponentsOfGraph(G,n)` Inputs a graph  $G$  and a nonnegative integer  $n$ . If  $n=0$  it returns the number of path components of  $G$ . If  $n>0$  it returns the number of path components of the  $n$ -skeleton of  $G$ .

`ContractGraph(G)` Inputs a graph  $G$  and tries to remove vertices and edges to produce a smaller graph  $G'$  such that  $G$  and  $G'$  have the same homology.

`GraphDisplay(G)` This function uses GraphViz software to display a graph  $G$ .

`SimplicialMap(K,L,f)` `SimplicialMapNC(K,L,f)` Inputs simplicial complexes  $K$ ,  $L$  and a function  $f:K \rightarrow L$ . It returns a simplicial map  $f:K \rightarrow L$ .

`ChainMapOfSimplicialMap(f)` Inputs a simplicial map  $f:K \rightarrow L$  and returns the corresponding chain map  $C_*(f):C_*(K) \rightarrow C_*(L)$ .

`SimplicialNerveOfGraph(G,d)` Inputs a graph  $G$  and returns a  $d$ -dimensional simplicial complex  $K$  whose 1-skeleton is  $G$ .

## **Chapter 25**

# **Cubical Complexes**

`ArrayToPureCubicalComplex(A,n)` Inputs an integer array  $A$  of dimension  $d$  and an integer  $n$ . It returns a  $d$ -dimensional pure cubical complex of dimension  $n$ .  
`PureCubicalComplex(A,n)` Inputs a binary array  $A$  of dimension  $d$ . It returns the corresponding  $d$ -dimensional pure cubical complex.  
`PureCubicalComplexIntersection(S,T)` Inputs two pure cubical complexes with common dimension and array size. It returns their intersection.  
`PureCubicalComplexUnion(S,T)` Inputs two pure cubical complexes with common dimension and array size. It returns their union.  
`PureCubicalComplexDifference(S,T)` Inputs two pure cubical complexes with common dimension and array size. It returns their difference.  
`ReadImageAsPureCubicalComplex("file.png",n)` Reads an image file ("file.png", "file.eps", "file.bmp" etc) and returns a pure cubical complex of dimension  $n$ .  
`ReadLinkImageAsPureCubicalComplex("file.png")` Reads a link image file ("file.png") and returns a pure cubical complex of dimension  $n$ .  
`ReadImageSequenceAsPureCubicalComplex("directory",n)` Reads the name of a directory containing a sequence of image files and returns a pure cubical complex of dimension  $n$ .  
`Size(T)` This returns the number of non-zero entries in the binary array of the cubical complex, or pure cubical complex  $T$ .  
`Dimension(T)` This returns the dimension of the cubical complex, or pure cubical complex  $T$ .  
`WritePureCubicalComplexAsImage(T,"filename","ext")` Inputs a 2-dimensional pure cubical complex  $T$ , and a filename and extension. It writes the image to the file.  
`ViewPureCubicalComplex(T)` `ViewPureCubicalComplex(T,"mozilla")` Inputs a 2-dimensional pure cubical complex  $T$ , and a browser name. It opens the image in the browser.  
`Homology(T,n)` `Homology(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex  $T$  and a non-negative integer  $n$ . It returns the  $n$ -th homology group.  
`Bettinnumbers(T,n)` `Bettinnumbers(T)` Inputs a pure cubical complex, or cubical complex, simplicial complex or chain complex  $T$  and a non-negative integer  $n$ . It returns the  $n$ -th Bettin number.  
`DirectProductOfPureCubicalComplexes(M,N)` Inputs two pure cubical complexes  $M, N$  and returns their direct product.  
`SuspensionOfPureCubicalComplex(M)` Inputs a pure cubical complex  $M$  and returns a pure cubical complex with one higher dimension.  
`EulerCharacteristic(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex  $T$  and returns the Euler characteristic.  
`PathComponentOfPureCubicalComplex(T,n)` Inputs a pure cubical complex  $T$  and an integer  $n$  in the range  $1, \dots, \dim T$ . It returns the  $n$ -th component.  
`ChainComplex(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex  $T$  and returns the (often infinite) chain complex.  
`ChainComplexOfPair(T,S)` Inputs a pure cubical complex or cubical complex  $T$  and subcomplex  $S$ . It returns the chain complex of the pair.  
`ExcisedPureCubicalPair(T,S)` Inputs a pure cubical complex  $T$  and subcomplex  $S$ . It returns the pair  $[T \setminus \text{int} S, S]$ .  
`ChainInclusionOfPureCubicalPair(S,T)` Inputs a pure cubical complex  $T$  and subcomplex  $S$ . It returns the chain inclusion.  
`ChainMapOfPureCubicalPairs(M,S,N,T)` Inputs a pure cubical complex  $N$  and subcomplexes  $M, T$  and  $S$  in  $T$ . It returns a chain map.  
`ContractPureCubicalComplex(T)` Inputs a pure cubical complex  $T$  of dimension  $d$  and removes  $d$ -dimensional cells.  
`ContractedComplex(T)` Inputs a pure cubical complex  $T$  and returns a structural copy of the complex obtained from `ContractPureCubicalComplex(T)`.  
`ZigZagContractedPureCubicalComplex(T)` Inputs a pure cubical complex  $T$  and returns a homotopy equivalent complex.  
`ContractCubicalComplex(T)` Inputs a cubical complex  $T$  and removes cells without changing the homotopy type.  
`DVFRducedCubicalComplex(T)` Inputs a cubical complex  $T$  and returns a non-regular cubical complex  $R$  by construction.  
`SkeletonOfCubicalComplex(T,n)` Inputs a cubical complex, or pure cubical complex  $T$  and positive integer  $n$ . It returns the  $n$ -skeleton.  
`ContractibleSubomplexOfPureCubicalComplex(T)` Inputs a pure cubical complex  $T$  and returns a maximal contractible subcomplex.  
`AcyclicSubomplexOfPureCubicalComplex(T)` Inputs a pure cubical complex  $T$  and returns a (not necessarily contractible) acyclic subcomplex.  
`HomotopyEquivalentMaximalPureCubicalSubcomplex(T,S)` Inputs a pure cubical complex  $T$  together with a pure cubical complex  $S$ . It returns a maximal subcomplex homotopy equivalent to  $S$ .  
`HomotopyEquivalentMinimalPureCubicalSubcomplex(T,S)` Inputs a pure cubical complex  $T$  together with a pure cubical complex  $S$ . It returns a minimal subcomplex homotopy equivalent to  $S$ .  
`BoundaryOfPureCubicalComplex(T)` Inputs a pure cubical complex  $T$  and returns its boundary as a pure cubical complex.  
`SingularitiesOfPureCubicalComplex(T,radius,tolerance)` Inputs a pure cubical complex  $T$  together with a radius and tolerance. It returns the singularities.  
`ThickenedPureCubicalComplex(T)` Inputs a pure cubical complex  $T$  and returns a pure cubical complex  $S$ . If a euclidean space is embedded in  $\mathbb{R}^n$ ,  $S$  is the  $\epsilon$ -neighborhood of  $T$ .  
`CropPureCubicalComplex(T)` Inputs a pure cubical complex  $T$  and returns a pure cubical complex  $S$  obtained from  $T$  by cropping.  
`BoundingPureCubicalComplex(T)` Inputs a pure cubical complex  $T$  and returns a contractible pure cubical complex  $S$  containing  $T$ .  
`MorseFiltration(M,i,t,bool)` `MorseFiltration(M,i,t)` Inputs a pure cubical complex  $M$  of dimension  $d$ , an integer  $i$ , a real number  $t$ , and a boolean. It returns the  $i$ -th Morse filtration.  
`ComplementOfPureCubicalComplex(T)` Inputs a pure cubical complex  $T$  and returns a pure cubical complex  $S$ . A euclidean space is embedded in  $\mathbb{R}^n$ .  
`PureCubicalComplexToTextFile(file,M)` Inputs a pure cubical complex  $M$  and a string containing the address of a file. It writes the complex to the file.  
`ThickeningFiltration(M,n)` `ThickeningFiltration(M,n,k)` Inputs a pure cubical complex  $M$  and a positive integer  $n$  and  $k$ . It returns the  $n$ -th thickening filtration.  
`Dendrogram(M)` Inputs a filtered pure cubical complex  $M$  and returns data that specifies the dendrogram (or phylogenetic tree).  
`DendrogramDisplay(M)` Inputs a filtered pure cubical complex  $M$ , or alternatively inputs the output from the command `Dendrogram(M)`. It displays the dendrogram.  
`DendrogramToPersistentceMat(D)` Inputs the output of the function `Dendrogram(M)` and returns the corresponding persistence matrix.  
`ReadImageAsFilteredCubicalComplex(file,n)` Inputs a string containing the path to an image file, together with a dimension  $n$ . It returns a filtered cubical complex.  
`ComplementOfFilteredCubicalComplex(M)` Inputs a filtered pure cubical complex  $M$  and returns the complement of  $M$ .

## Chapter 26

# Regular CW-Complexes

`SimplicialComplexToRegularCWComplex(K)` Inputs a simplicial complex  $K$  and returns the corresponding regular CW-complex.  
`CubicalComplexToRegularCWComplex(K)` `CubicalComplexToRegularCWComplex(K,n)` Inputs a pure cubical complex  $K$  and returns the corresponding regular CW-complex.  
`CriticalCellsOfRegularCWComplex(Y)` `CriticalCellsOfRegularCWComplex(Y,n)` Inputs a regular CW-complex  $Y$  and returns the critical cells of  $Y$ .  
`ChainComplex(Y)` Inputs a regular CW-complex  $Y$  and returns the cellular chain complex of a CW-complex  $W$  whose universal cover is  $Y$ .  
`ChainComplexOfRegularCWComplex(Y)` Inputs a regular CW-complex  $Y$  and returns the cellular chain complex of  $Y$ .  
`FundamentalGroup(Y)` `FundamentalGroup(Y,n)` Inputs a regular CW-complex  $Y$  and, optionally, the number of generators.



## Chapter 27

# Knots and Links

`PureCubicalKnot(L)` `PureCubicalKnot(n,i)` Inputs a list  $L = [[m_1, n_1], [m_2, n_2], \dots, [m_k, n_k]]$  of pairs of integers  
`ViewPureCubicalKnot(L)` Inputs a pure cubical link  $L$  and displays it.  
`KnotSum(K,L)` Inputs two pure cubical knots  $K, L$  and returns their sum as a pure cubical knot. This function is not  
`KnotGroup(K)` Inputs a pure cubical link  $K$  and returns the fundamental group of its complement. The group is returned  
`AlexanderMatrix(G)` Inputs a finitely presented group  $G$  whose abelianization is infinite cyclic. It returns the Alexander  
`AlexanderPolynomial(K)` `AlexanderPolynomial(G)` Inputs either a pure cubical knot  $K$  or a finitely presented group  
`ProjectionOfPureCubicalComplex(K)` Inputs an  $n$ -dimensional pure cubical complex  $K$  and returns an  $n-1$ -dimensional  
`ReadPDBfileAsPureCubicalComplex(file)` `ReadPDBfileAsPureCubicalComplex(file,m,c)` Inputs a protein structure file

## Chapter 28

# Commutative diagrams and abstract categories

### COMMUTATIVE DIAGRAMS

`HomomorphismChainToCommutativeDiagram(H)` Inputs a list  $H = [h_1, h_2, \dots, h_n]$  of mappings such that the composition of consecutive mappings is the identity mapping.  
`NormalSeriesToQuotientDiagram(L)` `NormalSeriesToQuotientDiagram(L, M)` Inputs an increasing (or decreasing) normal series  $L$  of a group  $G$  and a normal subgroup  $M$  of  $G$ .  
`NerveOfCommutativeDiagram(D)` Inputs a commutative diagram  $D$  and returns the commutative diagram  $ND$  corresponding to  $D$ .  
`GroupHomologyOfCommutativeDiagram(D, n)` `GroupHomologyOfCommutativeDiagram(D, n, prime)` Group homology of a commutative diagram  $D$  of finite  $p$ -groups.  
`PersistentHomologyOfCommutativeDiagramOfPGroups(D, n)` Inputs a commutative diagram  $D$  of finite  $p$ -groups.

### ABSTRACT CATEGORIES

`CategoricalEnrichment(X, Name)` Inputs a structure  $X$  such as a group or group homomorphism, together with a name  $Name$  for the category.  
`IdentityArrow(X)` Inputs an object  $X$  in some category, and returns the identity arrow on the object  $X$ .  
`InitialArrow(X)` Inputs an object  $X$  in some category, and returns the arrow from the initial object in the category to the object  $X$ .  
`TerminalArrow(X)` Inputs an object  $X$  in some category, and returns the arrow from  $X$  to the terminal object in the category.  
`HasInitialObject(Name)` Inputs the name of a category and returns true or false depending on whether the category has an initial object.  
`HasTerminalObject(Name)` Inputs the name of a category and returns true or false depending on whether the category has a terminal object.  
`Source(f)` Inputs an arrow  $f$  in some category, and returns its source.  
`Target(f)` Inputs an arrow  $f$  in some category, and returns its target.  
`CategoryName(X)` Inputs an object or arrow  $X$  in some category, and returns the name of the category.  
`"*", "=", "+", "-"` Composition of suitable arrows  $f, g$  is given by  $f * g$  when the source of  $f$  equals the target of  $g$ .  
`Object(X)` Inputs an object  $X$  in some category, and returns the GAP structure  $Y$  such that  $X = \text{CategoricalEnrichment}(Y)$ .  
`Mapping(X)` Inputs an arrow  $f$  in some category, and returns the GAP structure  $Y$  such that  $f = \text{CategoricalEnrichment}(Y)$ .  
`IsCategoryObject(X)` Inputs  $X$  and returns true if  $X$  is an object in some category.  
`IsCategoryArrow(X)` Inputs  $X$  and returns true if  $X$  is an arrow in some category.

## Chapter 29

# Arrays and Pseudo lists

`Array(A,f)` Inputs an array  $A$  and a function  $f$ . It returns the array obtained by applying  $f$  to each entry of  $A$  (and  $f$  to each entry of  $A$ ).

`PermuteArray(A,f)` Inputs an array  $A$  of dimension  $d$  and a permutation  $f$  of degree at most  $d$ . It returns the array obtained by permuting the entries of  $A$  according to  $f$ .

`ArrayDimension(A)` Inputs an array  $A$  and returns its dimension.

`ArrayDimensions(A)` Inputs an array  $A$  and returns its dimensions.

`ArraySum(A)` Inputs an array  $A$  and returns the sum of its entries.

`ArrayValue(A,x)` Inputs an array  $A$  and a coordinate vector  $x$ . It returns the value of the entry in  $A$  with coordinate  $x$ .

`ArrayValueFunctions(d)` Inputs a positive integer  $d$  and returns an efficient version of the function `ArrayValue` for arrays of dimension  $d$ .

`ArrayAssign(A,x,n)` Inputs an array  $A$  and a coordinate vector  $x$  and an integer  $n$ . It sets the entry of  $A$  with coordinate  $x$  to  $n$ .

`ArrayAssignFunctions(d)` Inputs a positive integer  $d$  and returns an efficient version of the function `ArrayAssign` for arrays of dimension  $d$ .

`ArrayIterate(d)` Inputs a positive integer  $d$  and returns a function `ArrayIt(Dimensions,f)`. This function inputs a list of coordinates of length  $d$  and returns the value of the entry in  $A$  with that coordinate.

`BinaryArrayToTextFile(file,A)` Inputs a string containing the address of a file, and an array  $A$  of 0s and 1s. The file is created if it does not exist. The file contains the binary representation of the entries of  $A$ .

`FrameArray(A)` Inputs an array  $A$  and returns the array obtained by appending a 0 to the beginning and end of each "row" of  $A$ .

`UnframeArray(A)` Inputs an array  $A$  and returns the array obtained by removing the first and last entry in each "row" of  $A$ .

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

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

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

## Parallel Computation - Core Functions

- 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 user@remotehost:~/
- 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

`ChildCommand("cmd;", s)` This runs a GAP command "cmd;" on the child process accessed by the stream s. Here  
`NextAvailableChild(L)` Inputs a list L of child processes and returns a child in L which is ready for computation.  
`IsAvailableChild(s)` Inputs a child process s and returns true if s is currently available for computations, and false otherwise.  
`ChildPut(A, "B", s)` This copies a GAP object A on the parent process to an object B on the child process s. (The object B must already exist.)  
`ChildGet("A", s)` This function copies a GAP object A on the child process s and returns it on the parent process.  
`HAPPrintTo("file", R)` Inputs a name "file" of a new text file and a HAP object R. It writes the object R to "file".  
`HAPRead("file", R)` Inputs a name "file" containing a HAP object R and returns the object. Currently this is only implemented for files.

## Chapter 31

# Parallel Computation - Extra Functions

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

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

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

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

## Chapter 32

### Some functions for accessing basic data

`BoundaryMap(C)` Inputs a resolution, chain complex or cochain complex  $C$  and returns the function  $C!.boundary$ .  
`BoundaryMatrix(C,n)` Inputs a chain or cochain complex  $C$  and integer  $n>0$ . It returns the  $n$ -th boundary map of  $C$ .  
`Dimension(C)`  
`Dimension(M)` Inputs a resolution, chain complex or cochain complex  $C$  and returns the function  $C!.dimension$ .  
`EvaluateProperty(X,"name")` Inputs a component object  $X$  (such as a  $ZG$ -resolution or chain map) and a string `name`.  
`GroupOfResolution(R)` Inputs a  $ZG$ -resolution  $R$  and returns the group  $G$ .  
`Length(R)` Inputs a resolution  $R$  and returns its length (i.e. the number of terms of  $R$  that HAP has computed).  
`Map(f)` Inputs a chain map, or cochain map or equivariant chain map  $f$  and returns the mapping function (as opposed to the map itself).  
`Source(f)` Inputs a chain map, or cochain map, or equivariant chain map, or  $FpG$ -module homomorphism  $f$  and returns the source module.  
`Target(f)` Inputs a chain map, or cochain map, or equivariant chain map, or  $FpG$ -module homomorphism  $f$  and returns the target module.

## Chapter 33

### Miscellaneous

`SL2Z(p)` `SL2Z(1/m)` Inputs a prime  $p$  or the reciprocal  $1/m$  of a square free integer  $m$ . In the first case the function returns a list of all subgroups of  $SL(2, \mathbb{Z})$  of index  $p$ . In the second case it returns a list of all subgroups of  $SL(2, \mathbb{Z})$  of index  $m$ .

`BigStepLCS(G,n)` Inputs a group  $G$  and a positive integer  $n$ . It returns a subseries  $G = L_1 > L_2 > \dots L_k = 1$  of the lower central series of  $G$  such that  $|G/L_i| \leq n$  for all  $i$ .

`Classify(L,Inv)` Inputs a list of objects  $L$  and a function  $Inv$  which computes an invariant of each object. It returns a list of objects  $L$  grouped by their invariant.

`RefineClassification(C,Inv)` Inputs a list  $C := Classify(L, OldInv)$  and returns a refined classification according to the function  $Inv$ .

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

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

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

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

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

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

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

`LinearHomomorphismsPersistenceMat(L)` Inputs a composable sequence  $L$  of vector space homomorphisms. It returns a matrix of persistence.

`NormalSeriesToQuotientHomomorphisms(L)` Inputs an (increasing or decreasing) chain  $L$  of normal subgroups. It returns a list of quotient homomorphisms.

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

# Index

- AcyclicSubomplexOfPureCubicalComplex, 31
- Add, 35
- AddFreeWords, 19
- AddFreeWordsModP, 19
- AlexanderMatrix, 33
- AlexanderPolynomial, 33
- AlgebraicReduction, 19
- Append, 35
- Array, 35
- ArrayAssign, 35
- ArrayAssignFunctions, 35
- ArrayDimension, 35
- ArrayDimensions, 35
- ArrayIterate, 35
- ArraySum, 35
- ArrayToPureCubicalComplex, 31
- ArrayValue, 35
- ArrayValueFunctions, 35
- AutomorphismGroupAsCatOneGroup, 23
- BaerInvariant, 14
- Bar Cocomplex, 26
- Bar Complex, 25
- Bar Resolution, 25
- BarCocomplexCoboundary, 26
- BarCode, 10
- BarCodeCompactDisplay, 10
- BarCodeDisplay, 10
- BarComplexBoundary, 25
- BarComplexEquivalence, 26
- BarResolutionBoundary, 25
- BarResolutionEquivalence, 25
- BarResolutionHomotopy, 25
- Bettinnumbers, 9, 29, 31
- BigStepLCS, 39
- BinaryArrayToTextFile, 35
- BoundaryMap, 38
- BoundaryMatrix, 38
- BoundaryOfPureCubicalComplex, 31
- BoundingPureCubicalComplex, 31
- CategoricalEnrichment, 34
- CategoryName, 34
- CayleyGraphOfGroup, 29
- CayleyGraphOfGroupDisplay, 16
- CcGroup (HAPcocyclic), 18
- CechComplexOfPureCubicalComplex, 29
- Centre, 22
- ChainComplex, 8, 32
- ChainComplexOfPair, 8
- ChainComplexOfRegularCWComplex, 32
- ChainComplexOfSimplicialGroup, 25
- ChainInclusionOfPureCubicalPair, 31
- ChainMapOfPureCubicalPairs, 31
- ChainMapOfSimplicialMap, 29
- ChevalleyEilenbergComplex, 8
- ChildClose, 36
- ChildCommand, 36
- ChildFunction, 37
- ChildGet, 36
- ChildProcess, 36
- ChildPut, 36
- ChildRead, 37
- ChildReadEval, 37
- Classify, 39
- Coclass, 14
- CocycleCondition, 18
- Cohomology, 10
- CohomologyModule, 10
- CohomologyPrimePart, 10
- ComplementOfFilteredCubicalComplex, 31
- ComplementOfPureCubicalComplex, 31
- Compose(f,g), 39
- CompositionSeriesOfFpGModules, 20
- ConjugatedResolution, 4
- ContractCubicalComplex, 31
- ContractedComplex, 31
- ContractGraph, 29



- ContractibleGcomplex, 17
- ContractibleSubcomplexOfSimplicialComplex, 29
- ContractibleSubcomplexOfPureCubicalComplex, 31
- ContractPureCubicalComplex, 31
- CoreducedChainComplex, 8
- CoxeterComplex, 17
- CoxeterDiagramComponents, 27
- CoxeterDiagramDegree, 27
- CoxeterDiagramDisplay, 27
- CoxeterDiagramFpArtinGroup, 27
- CoxeterDiagramFpCoxeterGroup, 27
- CoxeterDiagramIsSpherical, 27
- CoxeterDiagramMatrix, 27
- CoxeterDiagramVertices, 27
- CoxeterSubDiagram, 27
- CriticalCellsOfRegularCWComplex, 32
- CropPureCubicalComplex, 31
- CubicalComplexToRegularCWComplex, 32
  
- Dendrogram, 31
- DendrogramDisplay, 31
- DendrogramToPersistenceMat, 31
- DesuspensionFpGModule, 20
- DesuspensionMtxModule, 21
- Dimension, 38
- DirectProductGog, 22
- DirectProductOfPureCubicalComplexes, 31
- DirectSumOfFpGModules, 20
- DisplayAvailableCellComplexes, 28
- DVFRducedCubicalComplex, 31
  
- EilenbergMacLaneSimplicialGroup, 25
- EilenbergMacLaneSimplicialGroupMap, 25
- EpiCentre, 14
- EquivariantChainMap, 6
- EulerCharacteristic, 29
- EvaluateProperty, 38
- EvenSubgroup, 27
- ExpansionOfRationalFunction, 11
- ExtendScalars, 7
  
- FilteredTensorWithIntegers, 7
- FpGModule, 20
- FpGModuleDualBasis, 20
- FpGModuleHomomorphism, 20
- FpG\_to\_MtxModule, 21
  
- FrameArray, 35
- FreeGResolution, 4
- FundamentalDomainStandardSpaceGroup (HAPcryst), 17
- FundamentalGroup, 32
- FundamentalGroupOfRegularCWComplex, 32
  
- GeneratorsOfFpGModule, 20
- GeneratorsOfMtxModule, 21
- GOuterGroup, 22
- GOuterGroupHomomorphismNC, 22
- GOuterHomomorphismTester, 22
- GraphDisplay, 29
- GraphOfGroups, 27
- GraphOfGroupsDisplay, 27
- GraphOfGroupsTest, 27
- GraphOfResolutions, 27
- GraphOfResolutionsDisplay, 27
- GraphOfSimplicialComplex, 29
- GroupAlgebraAsFpGModule, 20
- GroupCohomology, 10
- GroupHomology, 10
- GroupHomologyOfCommutativeDiagram, 34
- GroupOfResolution, 38
  
- HAPcopyright, 39
- HAPPrintTo, 36
- HAPRead, 36
- HasInitialObject, 34
- HasTerminalObject, 34
- Homology, 10, 31
- HomologyPb, 10
- HomologyPrimePart, 10
- HomologyVectorSpace, 10
- HomomorphismChainToCommutativeDiagram, 34
- HomotopyEquivalentMaximalPureCubicalSubcomplex, 31
- HomotopyEquivalentMinimalPureCubicalSubcomplex, 31
- HomotopyGroup, 23, 25
- HomotopyModule, 23
- HomToGModule, 7
- HomToIntegers, 7
- HomToIntegersModP, 7
- HomToIntegralModule, 7
  
- IdentityAmongRelatorsDisplay, 16

- IdentityArrow, 34
- ImageOfFpGModuleHomomorphism, 20
- IncidenceMatrixToGraph, 29
- InduceScalars, 7
- InitialArrow, 34
- IntegralCupProduct, 12
- IntegralRingGenerators, 12
- IntersectionOfFpGModules, 20
- IsAspherical, 16
- IsAvailableChild, 36
- IsCategoryArrow, 34
- IsCategoryObject, 34
- IsFpGModuleHomomorphismData, 20
- IsLieAlgebraHomomorphism, 39
- IsPnormal, 28
- IsSuperperfect, 39
  
- KnotGroup, 33
- KnotSum, 33
  
- LefschetzNumber, 8
- LeibnizAlgebraHomology, 10
- LeibnizComplex, 8
- LeibnizQuasiCoveringHomomorphism, 15
- Length, 38
- LieAlgebraHomology, 10
- LieCoveringHomomorphism, 15
- LieEpiCentre, 15
- LieExteriorSquare, 15
- LieTensorCentre, 15
- LieTensorSquare, 15
- LinearHomomorphismsPersistenceMat, 39
- ListToPseudoList, 35
- LowerCentralSeriesLieAlgebra, 7
  
- MakeHAPManual, 39
- Map, 38
- Mapping, 34
- MaximalSimplexesToSimplicialComplex, 29
- MaximalSubmoduleOfFpGModule, 20
- MaximalSubmodulesOfFpGModule, 20
- Mod2CohomologyRingPresentation (HAP-prime), 13
- ModPCohomologyGenerators, 12
- ModPCohomologyRing, 12
- ModPRingGenerators, 12
- ModuleAsCatOneGroup, 23
- MooreComplex, 23, 25
  
- MorseFiltration, 31
- MultipleOfFpGModule, 20
- MultiplyWord, 19
  
- Negate, 19
- NegateWord, 19
- NerveOfCatOneGroup, 25
- NerveOfCommutativeDiagram, 34
- NextAvailableChild, 36
- NonabelianExteriorProduct, 14
- NonabelianSymmetricKernel, 14
- NonabelianSymmetricSquare, 14
- NonabelianTensorProduct, 14
- NonabelianTensorSquare, 14
- NormalSeriesToQuotientDiagram, 34
- NormalSeriesToQuotientHomomorphisms, 39
- NormalSubgroupAsCatOneGroup, 23
  
- Object, 34
- OrbitPolytope, 17
  
- ParallelList, 37
- PathComponentOfPureCubicalComplex, 31
- PathComponentsOfGraph, 29
- PathComponentsOfSimplicialComplex, 29
- PermToMatrixGroup, 39
- PermuteArray, 35
- PersistentCohomologyOfQuotientGroupSeries, 10
- PersistentHomologyOfCommutativeDiagramOfPGroups, 34
- PersistentHomologyOfFilteredChainComplex, 10
- PersistentHomologyOfPureCubicalComplex, 10
- PersistentHomologyOfQuotientGroupSeries, 10
- PersistentHomologyOfSubGroupSeries, 10
- PoincareSeries, 11
- PoincareSeriesLHS (HAPprime), 13
- PoincareSeriesPrimePart, 11
- PolytopalComplex, 17
- PolytopalGenerators, 17
- Prank, 11
- PresentationOfResolution, 16
- PrimePartDerivedFunctor, 10
- PrintZGword, 19
- ProjectedFpGModule, 20
- ProjectionOfPureCubicalComplex, 33
- PureComplexToSimplicialComplex, 29

- PureCubicalComplex, 31
- PureCubicalComplexDifference, 31
- PureCubicalComplexIntersection, 31
- PureCubicalComplexToTextFile, 31
- PureCubicalComplexUnion, 31
- PureCubicalKnot, 33
- QuasiIsomorph, 23
- QuillenComplex, 29
- QuotientOfContractibleGcomplex, 17
- RadicalOfFpGModule, 20
- RadicalSeriesOfFpGModule, 20
- RandomHomomorphismOfFpGModules, 20
- Rank, 20
- RankHomologyPGroup, 10
- RankMat, 9
- RankMatDestructive, 9
- RankPrimeHomology, 10
- ReadImageAsFilteredCubicalComplex, 31
- ReadImageAsPureCubicalComplex, 31
- ReadImageSequenceAsPureCubicalComplex, 31
- ReadLinkImageAsPureCubicalComplex, 31
- ReadPDBfileAsPureCubicalComplex, 33
- RecalculateIncidenceNumbers, 4
- ReducedSuspendedChainComplex, 8
- ReduceTorsionSubcomplex, 28
- RefineClassification, 39
- RelativeSchurMultiplier, 14
- ResolutionAbelianGroup, 4
- ResolutionAlmostCrystalGroup, 4
- ResolutionAlmostCrystalQuotient, 4
- ResolutionArithmeticGroup, 4
- ResolutionArtinGroup, 4
- ResolutionAsphericalPresentation, 4
- ResolutionBieberbachGroup (HAPcryst), 4
- ResolutionBoundaryOfWord, 19
- ResolutionCoxeterGroup, 4
- ResolutionDirectProduct, 4
- ResolutionExtension, 4
- ResolutionFiniteDirectProduct, 4
- ResolutionFiniteExtension, 4
- ResolutionFiniteGroup, 4
- ResolutionFiniteSubgroup, 4
- ResolutionFpGModule, 5
- ResolutionGraphOfGroups, 4
- ResolutionGTree, 4
- ResolutionNilpotentGroup, 4
- ResolutionNormalSeries, 4
- ResolutionPrimePowerGroup, 4
- ResolutionSL2Z, 4
- ResolutionSmallFpGroup, 4
- ResolutionSubgroup, 4
- ResolutionSubnormalSeries, 4
- RipsChainComplex, 29
- RipsHomology, 10
- SimplicialComplexToRegularCWComplex, 32
- SimplicialGroupMap, 25
- SimplicialMap, 29
- SimplicialMapNC, 29
- SimplicialNerveOfGraph, 29
- SingularitiesOfPureCubicalComplex, 31
- SkeletonOfCubicalComplex, 31
- SkeletonOfSimplicialComplex, 29
- SL2Z, 39
- SolutionsMatDestructive, 39
- Source, 34, 38
- SparseBoundaryMatrix, 9
- SparseChainComplex, 9
- SparseChainComplexOfRegularCWComplex, 9
- SparseMat, 9
- SparseRowAdd, 9
- SparseRowInterchange, 9
- SparseRowMult, 9
- SparseSemiEchelon, 9
- StandardCocycle, 18
- SumOfFpGModules, 20
- SumOp, 20
- SuspendedChainComplex, 8
- SuspensionOfPureCubicalComplex, 31
- SymmetricMatrixToIncidenceMatrix, 29
- Syzygy, 18
- Target, 34, 38
- TensorCentre, 14
- TensorProductOfChainComplexes, 8
- TensorWithIntegers, 7
- TensorWithIntegersModP, 7
- TensorWithIntegralModule, 7
- TensorWithRationals, 7
- TensorWithTwistedIntegers, 7
- TensorWithTwistedIntegersModP, 7
- TerminalArrow, 34

TestHap, [39](#)  
 ThickenedPureCubicalComplex, [31](#)  
 ThickeningFiltration, [31](#)  
 ThirdHomotopyGroupOfSuspensionB, [14](#)  
 TietzeReducedResolution, [4](#)  
 TietzeReduction, [19](#)  
 TorsionGeneratorsAbelianGroup, [16](#)  
 TorsionSubcomplex, [28](#)  
 TreeOfGroupsToContractibleGcomplex, [27](#)  
 TreeOfResolutionsToContractibleGcomplex, [27](#)  
 TruncatedGComplex, [17](#)  
 TwistedTensorProduct, [4](#)  
  
 UnframeArray, [35](#)  
 UniversalBarCode, [10](#)  
 UpperEpicentralSeries, [14](#)  
  
 VectorStabilizer, [17](#)  
 VectorsToFpGModuleWords, [20](#)  
 VectorsToSymmetricMatrix, [29](#)  
 ViewPureCubicalComplex, [31](#)  
 ViewPureCubicalKnot, [33](#)  
 VisualizeTorsionSkeleton, [28](#)  
  
 WritePureCubicalComplexAsImage, [31](#)  
  
 XmodToHAP, [23](#)  
  
 ZigZagContractedPureCubicalComplex, [31](#)  
 ZZPersistentHomologyOfPureCubicalComplex,  
[10](#)