# Contents

# 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 $R$ b
`FreeGResolution(P,n)` `FreeGResolution(P,n,p)` Inputs a non-free $ZG$-resolution $P$ with finite stabilizer grou
`ResolutionAbelianGroup(L,n)` `ResolutionAbelianGroup(G,n)` Inputs a list $L := [m_1, m_2, ..., m_d]$ of nonneg
`ResolutionAlmostCrystalGroup(G,n)` Inputs a positive integer $n$ and an almost crystallographic pcp group $G$. I
`ResolutionAlmostCrystalQuotient(G,n,c)` `ResolutionAlmostCrystalQuotient(G,n,c,false)` An alm
`ResolutionArtinGroup(D,n)` Inputs a Coxeter diagram $D$ and an integer $n > 1$. It returns $n$ terms of a free $ZG$-re
`ResolutionAsphericalPresentation(F,R,n)` Inputs a free group $F$, a set $R$ of words in $F$ which constitute an a
`ResolutionBieberbachGroup( G )`
`ResolutionBieberbachGroup( G, v )` Inputs a torsion free crystallographic group $G$, also known as a Bieberb
`ResolutionCoxeterGroup(D,n)` Inputs a Coxeter diagram $D$ and an integer $n > 1$. It returns $k$ terms of a free $ZG$
`ResolutionDirectProduct(R,S)` `ResolutionDirectProduct(R,S,"internal")` Inputs a $ZG$-resolution $R$
`ResolutionExtension(g,R,S)` `ResolutionExtension(g,R, S,"TestFiniteness")` `ResolutionExter`
`ResolutionFiniteDirectProduct(R,S)` `ResolutionFiniteDirectProduct(R,S, "internal")` Inputs a
`ResolutionFiniteExtension(gensE,gensG,R,n)` `ResolutionFiniteExtension(gensE,gensG,R,n,true`
`ResolutionFiniteGroup(gens,n)` `ResolutionFiniteGroup(gens,n,true)` `ResolutionFiniteGroup(g`
`ResolutionFiniteSubgroup(R,K)` `ResolutionFiniteSubgroup(R,gensG,gensK)` Inputs a $ZG$-resolution fo
`ResolutionGraphOfGroups(D,n)` `ResolutionGraphOfGroups(D,n,L)` Inputs a graph of groups $D$ and a p
`ResolutionNilpotentGroup(G,n)` `ResolutionNilpotentGroup(G,n,"TestFiniteness")` Inputs a nilpot
`ResolutionNormalSeries(L,n)` `ResolutionNormalSeries(L,n,true)` `ResolutionNormalSeries(L,n`
`ResolutionPrimePowerGroup(P,n)` `ResolutionPrimePowerGroup(G,n,p)` Inputs a $p$-group $P$ and integer
`ResolutionSmallFpGroup(G,n)` `ResolutionSmallFpGroup(G,n,p)` Inputs a small finitely presented group
`ResolutionSubgroup(R,K)` Inputs a $ZG$-resolution for an (infinite) group $G$ and a subgroup $K$ of finite index $|G :$
`ResolutionSubnormalSeries(L,n)` Inputs a positive integer n and a list $L = [L_1, \ldots, L_k]$ of subgroups $L_i$ of a fin
`TwistedTensorProduct(R,S,EhomG,GmapE,NhomE,NEhomN,EltsE,Mult,InvE)` Inputs a $ZG$-resolution $R$, a $ZN$

# 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 fr

# Chapter 3

# Induced equivariant chain maps

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

# Chapter 4

# Functors

•

`ExtendScalars(R,G,EltsG)` Inputs a $ZH$-resolution $R$, a group $G$ containing $H$ as a subgroup, and a list $EltsG$ o
`HomToIntegers(X)` Inputs either a $ZG$-resolution $X = R$, or an equivariant chain map $X = (F : R \longrightarrow S)$. It retur
`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 gro
`HomToGModule(R,A)` Inputs a $ZG$-resolution $R$ and an abelian G-outer group A. It returns the G-cocomplex obtai
`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
`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 NamesOfCo
`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
`TensorWithRationals(R)` Inputs a $ZG$-resolution $R$ and returns the chain complex obtained by tensoring with th

# Chapter 5

# Chain complexes

`ChainComplex(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex $T$ and returns the (ofte
`ChainComplexOfPair(T,S)` Inputs a pure cubical complex or cubical complex $T$ and contractible subcomplex $S$. It
`ChevalleyEilenbergComplex(X,n)` Inputs either a Lie algebra $X = A$ (over the ring of integers $Z$ or over a field
`LeibnizComplex(X,n)` Inputs either a Lie or Leibniz algebra $X = A$ (over the ring of integers $Z$ or over a field $K$)
`SuspendedChainComplex(C)` Inputs a chain complex $C$ and returns the chain complex $S$ defined by applying the de
`ReducedSuspendedChainComplex(C)` Inputs a chain complex $C$ and returns the chain complex $S$ defined by applyi
`CoreducedChainComplex(C)` `CoreducedChainComplex(C,2)` Inputs a chain complex $C$ and returns a quasi-isomo
`LefschetzNumber(F)` Inputs a chain map $F: C \to C$ with common source and target. It returns the Lefschetz numbe

# Chapter 6

# 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 cohomolo
`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 eithera finite group $X = G$ or
`GroupHomology(X,n)`
`GroupHomology(X,n,p)` Inputs a positive integer $n$ and eithera finite group $X = G$ or a nilpotent Pcp-group $X = G$ c
`PersistentHomologyOfQuotientGroupSeries(S,n)`
`PersistentHomologyOfQuotientGroupSeries(S,n,p,Resolution_Algorithm)` Inputs a positive integer $n$ and
`PersistentCohomologyOfQuotientGroupSeries(S,n)`
`PersistentCohomologyOfQuotientGroupSeries(S,n,p,Resolution_Algorithm)` Inputs a positive integer $n$ ar
`PersistentHomologyOfSubGroupSeries(S,n)`
`PersistentHomologyOfSubGroupSeries(S,n,p,Resolution_Algorithm)` Inputs a positive integer $n$ and a decre
`PersistentHomologyOfCommutativeDiagramOfPGroups(D,n)` Inputs a commutative diagram $D$ of finite $p$-gro
`PersistentHomologyOfPureCubicalComplex(L,n,p)`
`PersistentHomologyOfPureCubicalComplex(M,n,p)` Inputs a positive integer $n$, a prime $p$ and an increasing cha
`RipsHomology(G,n)` `RipsHomology(G,n,p)` Inputs a graph $G$, a non-negative integer $n$ (and optionally a prime nu
`BarCode(P)` Inputs an integer persistence matrix P and returns the same information in the form of a binary matrix (
`BarCodeDisplay(P)` `BarCodeDisplay(P,"mozilla")` Inputs an integer persistence matrix P, and an optional strin
`Homology(X,n)` Inputs either a chain complex $X = C$ or a chain map $X = (C \longrightarrow D)$.If $X = C$ then the torsion coeffi
`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 characte
`HomologyPrimePart(C,n,p)` Inputs a chain complex $C$ in characteristic 0, a positive integer $n$, and a prime $p$. It ret
`LeibnizAlgebraHomology(A,n)` Inputs a Lie or Leibniz algebra $X = A$ (over the ring of integers $Z$ or over a field $K$
`LieAlgebraHomology(A,n)` Inputs a Lie algebra $A$ (over the integers or a field) and a positive integer $n$. It returns th
`PrimePartDerivedFunctor(G,R,F,n)` Inputs a finite group $G$, a positive integer $n$, at least $n+1$ terms of a $ZP$-reso
`RankHomologyPGroup(G,n)` `RankHomologyPGroup(R,n)` `RankHomologyPGroup(G,n,"empirical")` Inputs a
`RankPrimeHomology(G,n)` Inputs a (smallish) $p$-group $G$ together with a positive integer $n$. It returns a function *dir*

# Chapter 7

# Poincare series

•

`EfficientNormalSubgroups(G)`
`EfficientNormalSubgroups(G,k)` Inputs a prime-power group $G$ and, optionally, a positive integer $k$. The default
`ExpansionOfRationalFunction(f,n)` Inputs a positive integer $n$ and a rational function $f(x) = p(x)/q(x)$ where t
`PoincareSeries(G,n)` `PoincareSeries(R,n)`
`PoincareSeries(L,n)`
`PoincareSeries(G)` Inputs a finite $p$-group $G$ and a positive integer $n$. It returns a quotient of polynomials $f(x) =$
`PoincareSeriesPrimePart(G,p,n)` Inputs a finite group $G$, a prime $p$, and a positive integer $n$. It returns a quotie
`PoincareSeriesLHS(G)` Inputs a finite 2-group $G$ and returns a quotient of polynomials $f(x) = P(x)/Q(x)$ whose
`Prank(G)` Inputs a $p$-group $G$ and returns the rank of the largest elementary abelian subgroup.

# Chapter 8

# Cohomology ring structure

`IntegralCupProduct(R,u,v,p,q)`
`IntegralCupProduct(R,u,v,p,q,P,Q,N)` (Various functions used to construct the cup product are also availabl
`IntegralRingGenerators(R,n)` Inputs at least $n+1$ terms of a $ZG$-resolution and integer $n>0$. It returns a min
`ModPCohomologyGenerators(G,n)`
`ModPCohomologyGenerators(R)` Inputs either a $p$-group $G$ and positive integer $n$, or else $n$ terms of a minimal $Z$
`ModPCohomologyRing(G,n)`
`ModPCohomologyRing(G,n,level)`
`ModPCohomologyRing(R)`
`ModPCohomologyRing(R,level)` Inputs either a $p$-group $G$ and positive integer $n$, or else $n$ terms of a minimal $Z$
`ModPRingGenerators(A)` Inputs a mod $p$ cohomology ring $A$ (created using the preceeding function). It returns a
`Mod2CohomologyRingPresentation(G)`
`Mod2CohomologyRingPresentation(G,n)`
`Mod2CohomologyRingPresentation(A)`
`Mod2CohomologyRingPresentation(R)` When applied to a finite 2-group $G$ this function returns a presentation f

# Chapter 9

# 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)`
  `Mod2CohomologyRingPresentation(A)`
  `Mod2CohomologyRingPresentation(R)` When applied to a finite 2-group $G$ this function returns a presentation f

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

# Chapter 10

# 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 a
`Coclass(G)` Inputs a group $G$ of prime-power order $p^n$ and nilpotency class $c$ say. It returns the integer $r = n - c$
`EpiCentre(G,N)`
`EpiCentre(G)` Inputs a finite group $G$ and normal subgroup $N$ and returns a subgroup $Z^*(G,N)$ of the centre of $N$.
`NonabelianExteriorProduct(G,N)` Inputs a finite group $G$ and normal subgroup $N$. It returns a record $E$ with t
`NonabelianSymmetricKernel(G)`
`NonabelianSymmetricKernel(G,m)` Inputs a finite or nilpotent infinite group $G$ and returns the abelian invariant
`NonabelianSymmetricSquare(G)`
`NonabelianSymmetricSquare(G,m)` Inputs a finite or nilpotent infinite group $G$ and returns a record $T$ with the
`NonabelianTensorProduct(G,N)` Inputs a finite group $G$ and normal subgroup $N$. It returns a record $E$ with the
`NonabelianTensorSquare(G)`
`NonabelianTensorSquare(G,m)` Inputs a finite or nilpotent infinite group $G$ and returns a record $T$ with the follo
`RelativeSchurMultiplier(G,N)` Inputs a finite group $G$ and normal subgroup $N$. It returns the homology grou
`TensorCentre(G)` Inputs a group $G$ and returns the largest central subgroup $N$ such that the induced homomorph
`ThirdHomotopyGroupOfSuspensionB(G)`
`ThirdHomotopyGroupOfSuspensionB(G,m)` Inputs a finite or nilpotent infinite group $G$ and returns the abelian i
`UpperEpicentralSeries(G,c)` Inputs a nilpotent group $G$ and an integer $c$. It returns the $c$-th term of the upper

# Chapter 11

# 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

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

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

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

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

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

# Chapter 12

# Generators and relators of groups

•

`CayleyGraphDisplay(G,X)`
`CayleyGraphDisplay(G,X,"mozilla")` Inputs a finite group $G$ together with a subset $X$ of $G$. It displays the co
`IdentityAmongRelatorsDisplay(R,n)` `IdentityAmongRelatorsDisplay(R,n,"mozilla")` Inputs a free
`IsAspherical(F,R)` Inputs a free group $F$ and a set $R$ of words in $F$. It performs a test on the 2-dimensional CW
`PresentationOfResolution(R)` Inputs at least two terms of a reduced $ZG$-resolution $R$ and returns a record $P$ w
`TorsionGeneratorsAbelianGroup(G)` Inputs an abelian group $G$ and returns a generating set $[x_1,\ldots,x_n]$ where

# Chapter 13

# Orbit polytopes and fundamental domains

•

`CoxeterComplex(D)` `CoxeterComplex(D,n)` Inputs a Coxeter diagram $D$ of finite type. It returns a non-free ZV
`ContractibleGcomplex("PSL(4,Z)")` Inputs one of the following strings: "SL(3,Z)", "PSL(4,Z)", "PSL(4,Z)_b"
`FundamentalDomainStandardSpaceGroup(v,G)` Inputs a crystallographic group G (represented using AffineCrys
`OrbitPolytope(G,v,L)` Inputs a permutation group or matrix group $G$ of degree $n$ and a rational vector $v$ of leng
`PolytopalComplex(G,v)`
`PolytopalComplex(G,v,n)` Inputs a permutation group or matrix group $G$ of degree $n$ and a rational vector $v$ of
`PolytopalGenerators(G,v)` Inputs a permutation group or matrix group $G$ of degree $n$ and a rational vector $v$ o
`VectorStabilizer(G,v)` Inputs a permutation group or matrix group $G$ of degree $n$ and a rational vector of degr

# Chapter 14

# Cocycles

- `CcGroup(A,f)` Inputs a $G$-module $A$ (i.e. an abelian $G$-outer group) and a standard 2-cocycle f $GxG--->A$. It
- `CocycleCondition(R,n)` Inputs a resolution $R$ and an integer $n$>0. It returns an integer matrix $M$ with the follow
- `StandardCocycle(R,f,n)`
- `StandardCocycle(R,f,n,q)` Inputs a $ZG$-resolution $R$ (with contracting homotopy), a positive integer $n$ and an i
- `Syzygy(R,g)` Inputs a $ZG$-resolution $R$ (with contracting homotopy) and a list $g=[g[1],...,g[n]]$ of elements in $G$

# Chapter 15

# 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
`AddFreeWordsModP(v,w,p)` Inputs two words $v, w$ in a free $ZG$-module and the characteristic $p$ of $Z$. It returns th
`AlgebraicReduction(w)`
`AlgebraicReduction(w,p)` Inputs a word $w$ in a free $ZG$-module and returns a reduced version of the word in w
`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 t
`TietzeReduction(S,w)` Inputs a set $S$ of words in a free $ZG$-module, and a word $w$ in the module. The function r

# Chapter 16

# $FpG$-modules

`CompositionSeriesOfFpGModules(M)` Inputs an $FpG$-module $M$ and returns a list of $FpG$-modules that constit
`DirectSumOfFpGModules(M,N)`
`DirectSumOfFpGModules([ M[1], M[2], ..., M[k] ]))` Inputs two $FpG$-modules $M$ and $N$ with common g
`FpGModule(A,P)`
`FpGModule(A,G,p)` Inputs a $p$-group $P$ and a matrix $A$ whose rows have length a multiple of the order of $G$. It ret
`FpGModuleDualBasis(M)` Inputs an $FpG$-module $M$. It returns a record $R$ with two components:$R.freeModule$ i
`FpGModuleHomomorphism(M,N,A)`
`FpGModuleHomomorphismNC(M,N,A)` Inputs $FpG$-modules $M$ and $N$ over a common $p$-group $G$. Also inputs a lis
`DesuspensionFpGModule(M,n)`
`DesuspensionFpGModule(R,n)` Inputs a positive integer $n$ and and FpG-module $M$. It returns an FpG-module $D'$
`RadicalOfFpGModule(M)` Inputs an $FpG$-module $M$ with $G$ a $p$-group, and returns the Radical of $M$ as an $FpG$-n
`RadicalSeriesOfFpGModule(M)` Inputs an $FpG$-module $M$ and returns a list of $FpG$-modules that constitute the
`GeneratorsOfFpGModule(M)` Inputs an $FpG$-module $M$ and returns a matrix whose rows correspond to a minima
`ImageOfFpGModuleHomomorphism(f)` Inputs an $FpG$-module homomorphism $f : M \longrightarrow N$ 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 mod
`IsFpGModuleHomomorphismData(M,N,A)` Inputs $FpG$-modules $M$ and $N$ over a common $p$-group $G$. Also input
`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 $FpG$-submodule
`MultipleOfFpGModule(w,M)` Inputs an $FpG$-module $M$ and a list $w := [g_1, ..., g_t]$ of elements in the group $G = M$
`ProjectedFpGModule(M,k)` Inputs an $FpG$-module $M$ of ambient dimension $n|G|$, and an integer $k$ between 1 an
`RandomHomomorphismOfFpGModules(M,N)` Inputs two $FpG$-modules $M$ and $N$ over a common group $G$. It return
`Rank(f)` Inputs an $FpG$-module homomorphism $f : M \longrightarrow N$ and returns the dimension of the image of $f$ as a ve
`SumOfFpGModules(M,N)` Inputs two $FpG$-modules $M,N$ arising as submodules in a common free module $(FG)^n$
`SumOp(f,g)` Inputs two $FpG$-module homomorphisms $f,g : M \longrightarrow N$ with common sorce and common target. It
`VectorsToFpGModuleWords(M,L)` Inputs an $FpG$-module $M$ and a list $L = [v_1, \ldots, v_k]$ of vectors in $M$. It returns

# Chapter 17

# Meataxe modules

•

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

# Chapter 18

# G-Outer Groups

`GOuterGroup(E,N)`
`GOuterGroup()` Inputs a group $E$ and normal subgroup $N$. It returns $N$ as a $G$-outer group where $G = E/N$. The fun
`GOuterGroupHomomorphismNC(A,B,phi)`
`GOuterGroupHomomorphismNC()` Inputs G-outer groups $A$ and $B$ with common acting group, and a group homomor
`GOuterHomomorphismTester(A,B,phi)` Inputs G-outer groups $A$ and $B$ with common acting group, and a group ho
`Centre(A)` Inputs G-outer group $A$ and returns the group theoretic centre of ActedGroup(A) as a G-outer group.
`DirectProductGog(A,B)`
`DirectProductGog(Lst)` Inputs G-outer groups $A$ and $B$ with common acting group, and returns their group-theore

# Chapter 19

# Cat-1-groups

`AutomorphismGroupAsCatOneGroup(G)` Inputs a group $G$ and returns the Cat-1-group $C$ corresponding th the cro
`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 G
`ModuleAsCatOneGroup(G,alpha,M)` Inputs a group $G$, an abelian group $M$ and a homomorphism $\alpha\colon G \to Aut(M)$
`MooreComplex(C)` Inputs a cat-1-group $C$ and returns its Moore complex $[M_1 \to M_0]$ as a list whose single entry is
`NormalSubgroupAsCatOneGroup(G,N)` Inputs a group $G$ with normal subgroup $N$. It returns the Cat-1-group $C$ co

# Chapter 20

# Coxeter diagrams and graphs of groups

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

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

`CoxeterDiagramDisplay(D)`

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

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

`CoxeterDiagramFpCoxeterGroup(D)` Inputs a Coxeter diagram $D$ and returns the corresponding finitely present

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

`CoxeterDiagramMatrix(D)` Inputs a Coxeter diagram $D$ and returns a matrix representation of it. The matrix is g

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

`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 $xy$

`GraphOfGroupsDisplay(D)`

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

`GraphOfGroupsTest(D)` Inputs an object $D$ and itries to test whether it is a Graph of Groups. However, it DOES

# Chapter 21

# Simplicial Complexes

`Homology(T,n)` `Homology(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex $T$ and a
`RipsHomology(G,n)` `RipsHomology(G,n,p)` Inputs a graph $G$, a non-negative integer $n$ (and optionally a prime nu
`Bettinumbers(T,n)` `Bettinumbers(T)` Inputs a pure cubical complex, or cubical complex, simplicial complex c
`ChainComplex(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex $T$ and returns the (ofte
`CechComplexOfPureCubicalComplex(T)` Inputs a d-dimensional pure cubical complex $T$ and returns a simplicial c
`RipsChainComplex(S,epsilon)` `RipsChainComplex(S,epsilon,true)` Inputs an $n \times n$ symmetric matrix $S$ with
`VectorsToSymmetricMatrix(M)` `VectorsToSymmetricMatrix(M,distance)` Inputs a matrix $M$ of rational numb
`EulerCharacteristic(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex $T$ and returns
`MaximalSimplicesToSimplicialComplex(L)` Inputs a list L whose entries are lists of vertices representing the ma
`SkeletonOfSimplicialComplex(S,k)` Inputs a simplicial complex $S$ and a positive integer $k$ less than or equal to t
`GraphOfSimplicialComplex(S)` Inputs a simplicial complex $S$ and returns the graph of $S$.
`ContractibleSubcomplexOfSimplicialComplex(S)` Inputs a simplicial complex $S$ and returns a (probably maxir
`PathComponentsOfSimplicialComplex(S,n)` Inputs a simplicial complex $S$ and a nonnegative integer $n$. If $n = 0$
`QuillenComplex(G)` Inputs a finite group $G$ and returns, as a simplicial complex, the order complex of the poset of
`SymmetricMatrixToIncidenceMatrix(S,t)` Inputs a symmetric integer matrix S and an integer t. It returns the m
`IncidenceMatrixToGraph(M)` Inputs a symmetric 0/1 matrix M. It returns the graph with one vertex for each row c
`PathComponentsOfGraph(G,n)` Inputs a graph $G$ and a nonnegative integer $n$. If $n = 0$ the number of path compone
`ContractGraph(G)` Inputs a graph $G$ and tries to remove vertices and edges to produce a smaller graph $G'$ such that
`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 $f\colon K!.vertic$
`ChainMapOfSimplicialMap(f)` Inputs a simplicial map $f\colon K \to L$ and returns the corresponding chain map $C_*(f)\colon C$
`SimplicialNerveOfGraph(G,d)` Inputs a graph $G$ and returns a $d$-dimensional simplicial complex $K$ whose 1-skelo

# Chapter 22

# Cubical Complexes

`ArrayToPureCubicalComplexA,n)` Inputs an integer array $A$ of dimension $d$ and an integer $n$. It returns a d-dimen
`PureCubicalComplexA,n)` Inputs a binary array $A$ of dimension $d$. It returns the corresponding d-dimensional pure
`PureCubicalComplexIntersection(S,T)` Inputs two pure cubical complexes with common dimension and array s
`PureCubicalComplexUnion(S,T)` Inputs two pure cubical complexes with common dimension and array size. It re
`PureCubicalComplexDifference(S,T)` Inputs two pure cubical complexes with common dimension and array siz
`ReadImageAsPureCubicalComplex("file.png",n)` Reads an image file (”file.png”, ”file.eps”, ”file.bmp” etc) ar
`ReadImageSequenceAsPureCubicalComplex("directory",n)` Reads the name of a directory containing a seque
`Size(T)` This returns the number of non-zero entries in the binary array of the pure cubical complex T.
`WritePureCubicalComplexAsImage(T,"filename","ext")` Inputs a 2-dimensional pure cubical complex T, an
`ViewPureCubicalComplex(T)` `ViewPureCubicalComplex(T,"mozilla")` Inputs a 2-dimensional pure cubical
`Homology(T,n)` `Homology(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex $T$ and a
`Bettinumbers(T,n)` `Bettinumbers(T)` Inputs a pure cubical complex, or cubical complex, simplicial complex o
`DirectProductOfPureCubicalComplexes(M,N)` Inputs two cubical complexes $M,N$ and returns their direct produ
`EulerCharacteristic(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex $T$ and returns
`PathComponentOfPureCubicalComplex(T,n)` Inputs a pure cubical complex $T$ and an integer $n$ in the rane 1, ...,
`ChainComplex(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex $T$ and returns the (ofte
`ChainComplexOfPair(T,S)` Inputs a pure cubical complex or cubical complex $T$ and contractible subcomplex $S$. It
`ChainInclusionOfPureCubicalPair(S,T)` Inputs a pure cubical complex $T$ and subcomplex $S$. It returns the chai
`ChainMapOfPureCubicalPairs(M,S,N,T)` Inputs a pure cubical complex $N$ and subcomplexes $M$, $T$ and $S$ in $T$. It
`ContractPureCubicalComplex(T)` Inputs a pure cubical complex $T$ of dimension $d$ and removes $d$-dimensional c
`ContractedComplex(T)` Inputs a pure cubical complex $T$ and returns a structural copy of the complex obtained fro
`ContractibleSubomplexOfPureCubicalComplex(T)` Inputs a pure cubical complex $T$ and returns a maximal co
`AlmostContractibleSubomplexOfPureCubicalComplex(T)` Inputs a pure cubical complex $T$ and returns a (not
`HomotopyEquivalentMaximalPureCubicalSubcomplex(T,S)` Inputs a pure cubical complex $T$ together with a s
`HomotopyEquivalentMinimalPureCubicalSubcomplex(T,S)` Inputs a pure cubical complex $T$ together with a s
`BoundaryOfPureCubicalComplex(T)` Inputs a pure cubical complex $T$ and returns its boundary as a pure cubical
`SingularitiesOfPureCubicalComplex(T,radius,tolerance)` Inputs a pure cubical complex $T$ together with a
`ThickenedPureCubicalComplex(T)` Inputs a pure cubical complex $T$ and returns a pure cubical complex $S$. If a e
`ComplementOfPureCubicalComplex(T)` Inputs a pure cubical complex $T$ and returns a pure cubical complex $S$. A

# Chapter 23

# Commutative diagrams and abstract categories

COMMUTATIVE DIAGRAMS

`HomomorphismChainToCommutativeDiagram(H)` Inputs a list $H = [h_1, h_2, ..., h_n]$ of mappings such that the comp
`NormalSeriesToQuotientDiagram(L)` `NormalSeriesToQuotientDiagram(L,M)` Inputs an increasing (or decr
`NerveOfCommutativeDiagram(D)` Inputs a commutative diagram $D$ and returns the commutative diagram $ND$ co
`GroupHomologyOfCommutativeDiagram(D,n)` `GroupHomologyOfCommutativeDiagram(D,n,prime)` Grou
`PersistentHomologyOfCommutativeDiagramOfPGroups(D,n)` Inputs a commutative diagram $D$ of finite $p$-gro

ABSTRACT CATEGORIES

`CategoricalEnrichment(X,Name)` Inputs a structure $X$ such as a group or group homomorphism, together with
`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 catego
`TerminalArrow(X)` Inputs an object $X$ in some category, and returns the arrow from $X$ to the terminal object in th
`HasInitialObject(Name)` Inputs the name of a category and returns true or false depending on whether the cate
`HasTerminalObject(Name)` Inputs the name of a category and returns true or false depending on whether the cat
`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 targe
`Object(X)` Inputs an object $X$ in some category, and returns the GAP structure $Y$ such that $X = CategoricalEnric$
`Mapping(X)` Inputs an arrow $f$ in some category, and returns the GAP structure $Y$ such that $f = CategoricalEnric$
`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 24

# Arrays and Pseudo lists

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

`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

`ArrayValueFunctions(d)` Inputs a positive integer $d$ and returns an efficient version of the function ArrayValue f

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

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

`ArrayIterate(d)` Inputs a positive integer $d$ and returns a function ArrayIt(Dimensions,f). This function inputs a

`BinaryArrayToTextFile(file,A)` Inputs a string containing the address of a file, and an array $A$ of 0s and 1s. Th

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

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

`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 th

`Append(L,K)` Let $L$ be a pseudo list and $K$ a list whose objects are compatible with those in $L$. This operation appli

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

# Chapter 25

# Parallel Computation - Core Functions

`ChildProcess()`
`ChildProcess("computer.ac.wales")`
`ChildProcess(["-m", "100000M", "-T"])`
`ChildProcess("computer.ac.wales", ["-m", "100000M", "-T"])` This starts a GAP session as a child proce

- 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

You should now be able to connect from "thishost" to "remotehost" without a password prompt.)
`ChildClose(s)` This closes the stream s to a child GAP process.
`ChildCommand("cmd;",s)` This runs a GAP command "cmd;" on the child process accessed by the stream s. Here
`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 fal
`ChildPut(A,"B",s)` This copies a GAP object A on the parent process to an object B on the child process s. (The
`ChildGet("A",s)` This functions 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 i

# Chapter 26

# Parallel Computation - Extra Functions

`ChildFunction("function(arg);",s)` This runs the GAP function "function(arg);" on a child process accessed b
`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 $ChildFunction("functi$
`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 27

# 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 o

`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

`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 oppo

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

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

# Chapter 28

# Miscellaneous

`BigStepLCS(G,n)` Inputs a group $G$ and a positive integer $n$. It returns a subseries $G = L_1 > L_2 > \ldots L_k = 1$ of the l
`Classify(L,Inv)` Inputs a list of objects $L$ and a function $Inv$ which computes an invariant of each object. It retu
`RefineClassification(C,Inv)` Inputs a list $C := Classify(L, OldInv)$ and returns a refined classification accor
`Compose(f,g)` Inputs two $FpG$-module homomorphisms $f : M \longrightarrow N$ and $g : L \longrightarrow M$ with $Source(f) = Target($
`HAPcopyright()` This function provides details of HAP'S GNU public copyright licence.
`IsLieAlgebraHomomorphism(f)` Inputs an object $f$ and returns true if $f$ is a homomorphism $f : A \longrightarrow B$ of Lie a
`IsSuperperfect(G)` Inputs a group $G$ and returns "true" if both the first and second integral homology of $G$ is tri
`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 $f$
`SolutionsMatDestructive(M,B)` Inputs an $m\tilde{A}n$ matrix $M$ and a $k\tilde{A}n$ matrix $B$ over a field. It returns a k×m ma
`LinearHomomorphismsPersistenceMat(L)` Inputs a composable sequence $L$ of vector space homomorphisms. I
`NormalSeriesToQuotientHomomorphisms(L)` Inputs an (increasing or decreasing) chain $L$ of normal subgroups
`TestHap()` This runs a representative sample of HAP functions and checks to see that they produce the correct ou

# Index