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
`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

It returns n terms of a free ZG-resolution for the group G described by the string. (Subscripts _b , _c , _d denote alter

Data for the first list of resolutions was provided provided by MATHIEU DUTOUR. Data for the second list was provi
`FreeGResolution(P,n)` `FreeGResolution(P,n,p)` Inputs a non-free ZG-resolution P with finite stabilizer group
`ResolutionGTree(P,n)` Inputs a non-free ZG-resolution P of dimension 1 (i.e. a G-tree) with finite stabilizer grou
`ResolutionSL2Z(p,n)` Inputs positive integers m,n and returns n terms of a ZG-resolution for the group $G = SL(2$
`ResolutionAbelianGroup(L,n)` `ResolutionAbelianGroup(G,n)` Inputs a list $L := [m_1, m_2, ..., m_d]$ of nonnegati
`ResolutionAlmostCrystalGroup(G,n)` Inputs a positive integer n and an almost crystallographic pcp group G. It
`ResolutionAlmostCrystalQuotient(G,n,c)` `ResolutionAlmostCrystalQuotient(G,n,c,false)` An almos
`ResolutionArtinGroup(D,n)` Inputs a Coxeter diagram D and an integer $n > 1$. It returns n terms of a free ZG-res
`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 crystallogr
`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 an
`ResolutionExtension(g,R,S)` `ResolutionExtension(g,R, S,"TestFiniteness")` `ResolutionExtensio`
`ResolutionFiniteDirectProduct(R,S)` `ResolutionFiniteDirectProduct(R,S, "internal")` Inputs a Z
`ResolutionFiniteExtension(gensE,gensG,R,n)` `ResolutionFiniteExtension(gensE,gensG,R,n,true)`
`ResolutionFiniteGroup(gens,n)` `ResolutionFiniteGroup(gens,n,true)` `ResolutionFiniteGroup(gens`
`ResolutionFiniteSubgroup(R,K)` `ResolutionFiniteSubgroup(R,gensG,gensK)` Inputs a ZG-resolution for a
`ResolutionGraphOfGroups(D,n)` `ResolutionGraphOfGroups(D,n,L)` Inputs a graph of groups D and a posi
`ResolutionNilpotentGroup(G,n)` `ResolutionNilpotentGroup(G,n,"TestFiniteness")` Inputs a nilpoten
`ResolutionNormalSeries(L,n)` `ResolutionNormalSeries(L,n,true)` `ResolutionNormalSeries(L,n,fa`
`ResolutionPrimePowerGroup(P,n)` `ResolutionPrimePowerGroup(G,n,p)` Inputs a p-group P and integer n>
`ResolutionSmallFpGroup(G,n)` `ResolutionSmallFpGroup(G,n,p)` Inputs a small finitely presented group G
`ResolutionSubgroup(R,K)` Inputs a ZG-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 $L = [L_1, \ldots, L_k]$ of subgroups L_i of a fini
`TwistedTensorProduct(R,S,EhomG,GmapE,NhomE,NEhomN,EltsE,Mult,InvE)` Inputs a ZG-resolution R, a ZN
`ConjugatedResolution(R,x)` Inputs a ZG-resoluton R and an element x from some group containing G. It returns
`RecalculateIncidenceNumbers(R)` Inputs a ZG-resoluton R which arises as the cellular chain complex of a regul

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 fre

Chapter 3

Induced equivariant chain maps

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

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 return
`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 grou
`TensorWithIntegralModule(R,f)` Inputs a ZG-resolution R and a group homomorphism $f : G \longrightarrow GL_n(Z)$ to th
`HomToGModule(R,A)` Inputs a ZG-resolution R and an abelian G-outer group A. It returns the G-cocomplex obtain
`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 (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 fiel
`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-isom
`TensorProductOfChainComplexes(C,D)` Inputs two chain complexes C and D of the same characteristic and retur
`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 $X = (C \longrightarrow D)$ i
`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 nilp
`PersistentHomologyOfQuotientGroupSeries(S,n)` `PersistentHomologyOfQuotientGroupSeries(S,n,p,`
`PersistentCohomologyOfQuotientGroupSeries(S,n)` `PersistentCohomologyOfQuotientGroupSeries(S,`
`UniversalBarCode("UpperCentralSeries",n,d)` `UniversalBarCode("UpperCentralSeries",n,d,k)` Inpu
`PersistentHomologyOfSubGroupSeries(S,n)` `PersistentHomologyOfSubGroupSeries(S,n,p,Resolution`
`PersistentHomologyOfFilteredChainComplex(C,n,p)` Inputs a filtered chain complex C (of characteristic 0 c
`PersistentHomologyOfCommutativeDiagramOfPGroups(D,n)` Inputs a commutative diagram D of finite p-gro
`PersistentHomologyOfPureCubicalComplex(L,n,p)` Inputs a positive integer n, a prime p and an increasing ch
`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 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 coeff
`HomologyPb(C,n)` This is a back-up function which might work in some instances where $Homology(C,n)$ fails. It i
`HomologyVectorSpace(X,n)` Inputs either a chain complex $X = C$ or a chain map $X = (C \longrightarrow D)$ in prime charact
`HomologyPrimePart(C,n,p)` Inputs a chain complex C in characteristic 0, a positive integer n, and a prime p. It re
`LeibnizAlgebraHomology(A,n)` Inputs a Lie or Leibniz algebra $X = A$ (over the ring of integers Z or over a field
`LieAlgebraHomology(A,n)` Inputs a Lie algebra A (over the integers or a field) and a positive integer n. It returns t
`PrimePartDerivedFunctor(G,R,F,n)` Inputs a finite group G, a positive integer n, at least $n+1$ terms of a ZP-res
`RankHomologyPGroup(G,n)` `RankHomologyPGroup(R,n)` `RankHomologyPGroup(G,n,"empirical")` Inputs a (s
`RankPrimeHomology(G,n)` Inputs a (smallish) p-group G together with a positive integer n. It returns a function di

Chapter 7

Poincare series

•

`EfficientNormalSubgroups(G)` `EfficientNormalSubgroups(G,k)` Inputs a prime-power group G and, optiona
`ExpansionOfRationalFunction(f,n)` Inputs a positive integer n and a rational function $f(x) = p(x)/q(x)$ where
`PoincareSeries(G,n)` `PoincareSeries(R,n)` `PoincareSeries(L,n)` `PoincareSeries(G)` Inputs a f
`PoincareSeriesPrimePart(G,p,n)` Inputs a finite group G, a prime p, and a positive integer n. It returns a quoti
`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 co
`IntegralRingGenerators(R,n)` Inputs at least $n+1$ terms of a ZG-resolution and integer n> 0. It returns a mini
`ModPCohomologyGenerators(G,n)` `ModPCohomologyGenerators(R)` Inputs either a p-group G and positive in
`ModPCohomologyRing(G,n)` `ModPCohomologyRing(G,n,level)` `ModPCohomologyRing(R)` `ModPCohomolog`
`ModPRingGenerators(A)` Inputs a mod p cohomology ring A (created using the preceeding function). It returns a
`Mod2CohomologyRingPresentation(G)` `Mod2CohomologyRingPresentation(G,n)` `Mod2CohomologyRingP`

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)` `Mod2CohomologyRingP`
`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 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 th
`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 an
`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 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 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
`LieExteriorSquare(L)` Inputs a finite dimensional Lie algebra L over a field. It returns a record E with the follov
`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

•

`CayleyGraphOfGroupDisplay(G,X)` `CayleyGraphOfGroupDisplay(G,X,"mozilla")` Inputs a finite group G
`IdentityAmongRelatorsDisplay(R,n)` `IdentityAmongRelatorsDisplay(R,n,"mozilla")` Inputs a free Z
`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 r

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 ZW-r
`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,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

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 DUTOUR. Data for the second list
`QuotientOfContractibleGcomplex(C,D)` Inputs a non-free ZG-resolution C and a finite subgroup D of G which
`TruncatedGComplex(R,m,n)` Inputs a non-free ZG-resolution R and two positive integers m and n. It returns the no
`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 lengt
`PolytopalComplex(G,v)` `PolytopalComplex(G,v,n)` Inputs a permutation group or matrix group G of degree
`PolytopalGenerators(G,v)` Inputs a permutation group or matrix group G of degree n and a rational vector v of
`VectorStabilizer(G,v)` Inputs a permutation group or matrix group G of degree n and a rational vector of degre

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 followi

`StandardCocycle(R,f,n)`

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

`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 c
`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 wh
`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 th
`TietzeReduction(S,w)` Inputs a set S of words in a free ZG-module, and a word w in the module. The function re
`ResolutionBoundaryOfWord(R,n,w)` Inputs a resolution R, a positive integer n and a list w representing a word in

Chapter 16

FpG-modules

`CompositionSeriesOfFpGModules(M)` Inputs an FpG-module M and returns a list of FpG-modules that constitu

`DirectSumOfFpGModules(M,N)` `DirectSumOfFpGModules([ M[1], M[2], ..., M[k] ]))` Inputs two FpG

`FpGModule(A,P)` `FpGModule(A,G,p)` Inputs a p-group P and a matrix A whose rows have length a multiple of th

`FpGModuleDualBasis(M)` Inputs an FpG-module M. It returns a record R with two components:$R.freeModule$ is

`FpGModuleHomomorphism(M,N,A)` `FpGModuleHomomorphismNC(M,N,A)` Inputs FpG-modules M and N over a

`DesuspensionFpGModule(M,n)` `DesuspensionFpGModule(R,n)` Inputs a positive integer n and and FpG-modu

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

`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 minimal

`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 inputs

`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 and

`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 vec

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

`SumOp(f,g)` Inputs two FpG-module homomorphisms $f,g : M \longrightarrow N$ with common sorce and common target. It r

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

Chapter 17

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 18

G-Outer Groups

`GOuterGroup(E,N)` `GOuterGroup()` Inputs a group E and normal subgroup N. It returns N as a G-outer group whe
`GOuterGroupHomomorphismNC(A,B,phi)` `GOuterGroupHomomorphismNC()` Inputs G-outer groups A and B with
`GOuterHomomorphismTester(A,B,phi)` Inputs G-outer groups A and B with common acting group, and a group h
`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, a

Chapter 19

Cat-1-groups

`AutomorphismGroupAsCatOneGroup(G)` Inputs a group G and returns the Cat-1-group C corresponding to the cros
`HomotopyGroup(C,n)` Inputs a cat-1-group C and an integer n. It returns the nth 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
`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 $\alpha\colon G \to Aut(M)$.
`MooreComplex(C)` Inputs a cat-1-group C and returns its Moore complex as a G-complex (i.e. as a complex of grou
`NormalSubgroupAsCatOneGroup(G,N)` Inputs a group G with normal subgroup N. It returns the Cat-1-group C co
`XmodToHAP(C)` Inputs a cat-1-group C obtained from the Xmod package and returns a cat-1-group D for which IsHa

Chapter 20

Simplicial groups

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

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 $f: G \to Q$, 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 $f: G \to Q$ 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 lengt
`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

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

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

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

We represent a word

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

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

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

$equiv(n,-)\colon B_n(G) \to B_{n+1}(G)$

satisfying w - $\psi(\phi(w)) = d(n+1, equiv(n,w)) + equiv(n-1, d(n,w)).where$d(n,-):$B_n(G) \to B_{n-1}(G)$ is the 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 $T = TensorWithIntegerts(R)$. The function returns a component object HE with components

- HE!.phi(n,w) is a function which inputs a non-negative integer n and a word w in $BC_n(G)$. It returns the image of w in T_n under a chain equivalence $\phi\colon BC_n(G) \to T_n$.
- HE!.psi(n,w) is a function which inputs a non-negative integer n and an element w in T_n. It returns the image of w in $BC_n(G)$ under a chain equivalence $\psi\colon T_n \to BC_n(G)$.
- HE!.equiv(n,w) is a function which inputs a non-negative integer n and a word w in $BC_n(G)$. It returns the image of w in $BC_{n+1}(G)$ under a homomorphism $equiv(n,-)\colon BC_n(G) \to BC_{n+1}(G)$ satisfying
$$w - \psi(\phi(w)) = d(n+1, equiv(n,w)) + equiv(n-1, d(n,w)).$$
where $d(n,-)\colon BC_n(G) \to BC_{n-1}(G)$ 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 $[g_1|g_2|...|g_n]$ where the g_i range over G.

We represent a word
$$w = [g_{11}|g_{12}|...|g_{1n}] - [g_{21}|g_{22}|...|g_{2n}] + ... + [g_{k1}|g_{k2}|...|g_{kn}]$$
in $BC^n(G)$ as a list of lists:
$$[[+1, g_{11}, g_{12}, ..., g_{1n}], [-1, g_{21}, g_{22}, ...|g_{2n}] + ... + [+1, g_{k1}, g_{k2}, ..., g_{kn}].$$

`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 $d^n\colon BC^n(G) \to BC^{n+1}(G)$ in the bar cocomplex.

This function was implemented by VAN LUYEN LE.

Chapter 21

Coxeter diagrams and graphs of groups

`CoxeterDiagramComponents(D)` Inputs a Coxeter diagram D and returns a list $[D_1, ..., D_d]$ of the maximal connec
`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
`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 presente
`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 gi
`CoxeterSubDiagram(D,V)` Inputs a Coxeter diagram D and a subset V of its vertices. It returns the full sub-diagra
`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 di
`GraphOfResolutions(D,n)` Inputs a graph of groups D and a positive integer n. It returns a graph of resolutions
`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 Mozill
`GraphOfGroupsTest(D)` Inputs an object D and itries to test whether it is a Graph of Groups. However, it DOES N
`TreeOfGroupsToContractibleGcomplex(D,G)` Inputs a graph of groups D which is a tree, and also inputs the fu
`TreeOfResolutionsToContractibleGcomplex(D,G)` Inputs a graph of resolutions D which is a tree, and also in

#

Chapter 22

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

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

where the symbol O[-m] stands for the ring of integers in the imaginary quadratic number field Q(sqrt(-m)), the latte

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

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, only there will be some warnings printed on the screen regarding the function redu
`DisplayAvailableCellComplexes();` Displays the cell complexes that are available in HAP.
`VisualizeTorsionSkeleton( groupName, p)` Executes the function TorsionSubcomplex(groupName, p) and vi
`ReduceTorsionSubcomplex( groupName, p)` This function may be applied to the cell complexes for which the fu

Chapter 23

Simplicial Complexes

`Homology(T,n)` `Homology(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex T and a no
`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
`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
`PureComplexToSimplicialComplex(T,k)` Inputs either a d-dimensional pure cubical complex T or a d-dimensior
`RipsChainComplex(G,n)` Inputs a graph G and a non-negative integer n. It returns $n+1$ terms of a chain complex
`VectorsToSymmetricMatrix(M)` `VectorsToSymmetricMatrix(M,distance)` Inputs a matrix M of rational num
`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 m
`SkeletonOfSimplicialComplex(S,k)` Inputs a simplicial complex S and a positive integer k less than or equal to
`GraphOfSimplicialComplex(S)` Inputs a simplicial complex S and returns the graph of S.
`ContractibleSubcomplexOfSimplicialComplex(S)` Inputs a simplicial complex S and returns a (probably maxi
`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)` `SymmetricMatrixToIncidenceMatrix(S,t,d)` Inputs a symmetr
`IncidenceMatrixToGraph(M)` Inputs a symmetric 0/1 matrix M. It returns the graph with one vertex for each row
`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 $n=0$ the number of path compon
`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 $C_*(f)\colon$
`SimplicialNerveOfGraph(G,d)` Inputs a graph G and returns a d-dimensional simplicial complex K whose 1-skel

Chapter 24

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

`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) a

`ReadLinkImageAsPureCubicalComplex("file.png")` `ReadLinkImageAsPureCubicalComplex("file.png`

`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 cubical complex, or pure cubical com

`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

`ViewPureCubicalComplex(T)` `ViewPureCubicalComplex(T,"mozilla")` Inputs a 2-dimensional pure cubical co

`Homology(T,n)` `Homology(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex T and a no

`Bettinumbers(T,n)` `Bettinumbers(T)` Inputs a pure cubical complex, or cubical complex, simplicial complex or

`DirectProductOfPureCubicalComplexes(M,N)` Inputs two pure cubical complexes M,N and returns their direct

`SuspensionOfPureCubicalComplex(M)` Inputs a pure cubical complex M and returns a pure cubical complex with

`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 subcomplex S. It returns the

`ExcisedPureCubicalPair(T,S)` Inputs a pure cubical complex T and subcomplex S. It returns the pair $[T \setminus intS, S$

`ChainInclusionOfPureCubicalPair(S,T)` Inputs a pure cubical complex T and subcomplex S. It returns the cha

`ChainMapOfPureCubicalPairs(M,S,N,T)` Inputs a pure cubical complex N and subcomplexes M, T and S in T. I

`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

`ZigZagContractedPureCubicalComplex(T)` Inputs a pure cubical complex T and returns a homotopy equivalent

`ContractCubicalComplex(T)` Inputs a cubical complex T and removes cells without changing the homotopy type

`DVFReducedCubicalComplex(T)` Inputs a cubical complex T and returns a non-regular cubical complex R by const

`SkeletonOfCubicalComplex(T,n)` Inputs a cubical complex, or pure cubical complex T and positive integer n. It

`ContractibleSubomplexOfPureCubicalComplex(T)` Inputs a pure cubical complex T and returns a maximal con

`AcyclicSubomplexOfPureCubicalComplex(T)` Inputs a pure cubical complex T and returns a (not necessarily co

`HomotopyEquivalentMaximalPureCubicalSubcomplex(T,S)` Inputs a pure cubical complex T together with a p

`HomotopyEquivalentMinimalPureCubicalSubcomplex(T,S)` Inputs a pure cubical complex T together with a p

`BoundaryOfPureCubicalComplex(T)` Inputs a pure cubical complex T and returns its boundary as a pure cubical c

`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 eu

`CropPureCubicalComplex(T)` Inputs a pure cubical complex T and returns a pure cubical complex S obtained fro

`BoundingPureCubicalComplex(T)` Inputs a pure cubical complex T and returns a contractible pure cubical compl

`MorseFiltration(M,i,t,bool)` `MorseFiltration(M,i,t)` Inputs a pure cubical complex M of dimension d, an

`ComplementOfPureCubicalComplex(T)` Inputs a pure cubical complex T and returns a pure cubical complex S. A

`PureCubicalComplexToTextFile(file,M)` Inputs a pure cubical complex M and a string containing the address o

Chapter 25

Regular CW-Spaces

`SimplicialComplexToRegularCWSpace(K)` Inputs a simplicial complex K and returns the corresponding regular C
`CubicalComplexToRegularCWSpace(K)` Inputs a pure cubical complex (or cubical complex) K and returns the cor
`CriticalCellsOfRegularCWSpace(Y)` Inputs a regular CW-space Y and returns the critical cells of Y with respect
`ChainComplex(Y)` Inputs a regular CW-space Y and returns the cellular chain complex of a CW-space W whose cel
`ChainComplexOfRegularCWSpace(Y)` Inputs a regular CW-space Y and returns the cellular chain complex of Y.
`FundamentalGroup(Y)` `FundamentalGroup(Y,n)` Inputs a regular CW-space Y and, optionally, the number of sor

Chapter 26

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)` Group
`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 t
`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 categor
`TerminalArrow(X)` Inputs an object X in some category, and returns the arrow from X to the terminal object in the
`HasInitialObject(Name)` Inputs the name of a category and returns true or false depending on whether the categ
`HasTerminalObject(Name)` Inputs the name of a category and returns true or false depending on whether the cate
`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
`Object(X)` Inputs an object X in some category, and returns the GAP structure Y such that $X = CategoricalEnrich$
`Mapping(X)` Inputs an arrow f in some category, and returns the GAP structure Y such that $f = CategoricalEnrich$
`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 27

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
`PermuteArray(A,f)` Inputs an array A of dimension d and a permutation f of degree at most d. It returns the array
`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 fo
`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
`ArrayAssignFunctions(d)` Inputs a positive integer d and returns an efficient version of the function ArrayAssign
`ArrayIterate(d)` Inputs a positive integer d and returns a function ArrayIt(Dimensions,f). This function inputs a li
`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 thi
`Append(L,K)` Let L be a pseudo list and K a list whose objects are compatible with those in L. This operation applie
`ListToPseudoList(L)` Inputs a list L and returns the pseudo list representation of L.

Chapter 28

Parallel Computation - Core Functions

`ChildProcess()` `ChildProcess("computer.ac.wales")` `ChildProcess(["-m", "100000M", "-T"])` Child

- 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 fals
`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 29

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 30

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 $C!.dimension$. A
`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 oppose
`Source(f)` Inputs a chain map, or cochain map, or equivariant chain map, or FpG-module homomorphism f and re
`Target(f)` Inputs a chain map, or cochain map, or equivariant chain map, or FpG-module homomorphism f and re

Chapter 31

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 functic
`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 lov
`Classify(L,Inv)` Inputs a list of objects L and a function Inv which computes an invariant of each object. It retur
`RefineClassification(C,Inv)` Inputs a list $C := Classify(L, OldInv)$ and returns a refined classification accord
`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 triv
`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 $m \times n$ matrix M and a $k \times n$ matrix B over a field. It returns a k×m ma
`LinearHomomorphismsPersistenceMat(L)` Inputs a composable sequence L of vector space homomorphisms. It
`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 out

Index