

ModulePresentationsForCAP

Category R-pres for CAP
2017.07.25

25 July 2017

Sebastian Gutsche

Sebastian Posur

Sebastian Gutsche

Email: gutsche@mathematik.uni-siegen.de

Homepage: <http://www.uni-siegen.de/fb6/rmi/>

Address: Department Mathematik
Universität Siegen
Walter-Flex-Straße 3
57068 Siegen
Germany

Sebastian Posur

Email: sebastian.posur@uni-siegen.de

Homepage: <http://www.uni-siegen.de/fb6/rmi/>

Address: Department Mathematik
Universität Siegen
Walter-Flex-Straße 3
57068 Siegen
Germany

Contents

1	Module Presentations	3
1.1	Functors	3
1.2	GAP Categories	4
1.3	Constructors	5
1.4	Attributes	7
1.5	Non-Categorical Operations	8
1.6	Natural Transformations	8
2	Examples and Tests	10
2.1	Annihilator	10
2.2	Intersection of Submodules	10
2.3	Koszul Complex	11
2.4	Closed Monoidal Structure	13
	Index	14

Chapter 1

Module Presentations

1.1 Functors

1.1.1 `FunctorStandardModuleLeft` (for `IsHomalgRing`)

▷ `FunctorStandardModuleLeft(R)` (attribute)

Returns: a functor

The argument is a homalg ring R . The output is a functor which takes a left presentation as input and computes its standard presentation.

1.1.2 `FunctorStandardModuleRight` (for `IsHomalgRing`)

▷ `FunctorStandardModuleRight(R)` (attribute)

Returns: a functor

The argument is a homalg ring R . The output is a functor which takes a right presentation as input and computes its standard presentation.

1.1.3 `FunctorGetRidOfZeroGeneratorsLeft` (for `IsHomalgRing`)

▷ `FunctorGetRidOfZeroGeneratorsLeft(R)` (attribute)

Returns: a functor

The argument is a homalg ring R . The output is a functor which takes a left presentation as input and gets rid of the zero generators.

1.1.4 `FunctorGetRidOfZeroGeneratorsRight` (for `IsHomalgRing`)

▷ `FunctorGetRidOfZeroGeneratorsRight(R)` (attribute)

Returns: a functor

The argument is a homalg ring R . The output is a functor which takes a right presentation as input and gets rid of the zero generators.

1.1.5 `FunctorLessGeneratorsLeft` (for `IsHomalgRing`)

▷ `FunctorLessGeneratorsLeft(R)` (attribute)

Returns: a functor

The argument is a homalg ring R . The output is functor which takes a left presentation as input and computes a presentation having less generators.

1.1.6 FunctorLessGeneratorsRight (for IsHomalgRing)

▷ `FunctorLessGeneratorsRight(R)` (attribute)

Returns: a functor

The argument is a homalg ring R . The output is functor which takes a right presentation as input and computes a presentation having less generators.

1.1.7 FunctorDualLeft (for IsHomalgRing)

▷ `FunctorDualLeft(R)` (attribute)

Returns: a functor

The argument is a homalg ring R that has an involution function. The output is functor which takes a left presentation M as input and computes its $\text{Hom}(M, R)$ as a left presentation.

1.1.8 FunctorDualRight (for IsHomalgRing)

▷ `FunctorDualRight(R)` (attribute)

Returns: a functor

The argument is a homalg ring R that has an involution function. The output is functor which takes a right presentation M as input and computes its $\text{Hom}(M, R)$ as a right presentation.

1.1.9 FunctorDoubleDualLeft (for IsHomalgRing)

▷ `FunctorDoubleDualLeft(R)` (attribute)

Returns: a functor

The argument is a homalg ring R that has an involution function. The output is functor which takes a left presentation M as input and computes its $\text{Hom}(\text{Hom}(M, R), R)$ as a left presentation.

1.1.10 FunctorDoubleDualRight (for IsHomalgRing)

▷ `FunctorDoubleDualRight(R)` (attribute)

Returns: a functor

The argument is a homalg ring R that has an involution function. The output is functor which takes a right presentation M as input and computes its $\text{Hom}(\text{Hom}(M, R), R)$ as a right presentation.

1.2 GAP Categories

1.2.1 IsLeftOrRightPresentationMorphism (for IsCapCategoryMorphism)

▷ `IsLeftOrRightPresentationMorphism($object$)` (filter)

Returns: true or false

The GAP category of morphisms in the category of left or right presentations.

1.2.2 IsLeftPresentationMorphism (for IsLeftOrRightPresentationMorphism)

- ▷ IsLeftPresentationMorphism(*object*) (filter)
Returns: true or false
 The GAP category of morphisms in the category of left presentations.

1.2.3 IsRightPresentationMorphism (for IsLeftOrRightPresentationMorphism)

- ▷ IsRightPresentationMorphism(*object*) (filter)
Returns: true or false
 The GAP category of morphisms in the category of right presentations.

1.2.4 IsLeftOrRightPresentation (for IsCapCategoryObject)

- ▷ IsLeftOrRightPresentation(*object*) (filter)
Returns: true or false
 The GAP category of objects in the category of left presentations or right presentations.

1.2.5 IsLeftPresentation (for IsLeftOrRightPresentation)

- ▷ IsLeftPresentation(*object*) (filter)
Returns: true or false
 The GAP category of objects in the category of left presentations.

1.2.6 IsRightPresentation (for IsLeftOrRightPresentation)

- ▷ IsRightPresentation(*object*) (filter)
Returns: true or false
 The GAP category of objects in the category of right presentations.

1.3 Constructors

1.3.1 PresentationMorphism (for IsLeftOrRightPresentation, IsHomalgMatrix, IsLeftOrRightPresentation)

- ▷ PresentationMorphism(*A*, *M*, *B*) (operation)
Returns: a morphism in $\text{Hom}(A, B)$
 The arguments are an object *A*, a homalg matrix *M*, and another object *B*. *A* and *B* shall either both be objects in the category of left presentations or both be objects in the category of right presentations. The output is a morphism $A \rightarrow B$ in the the category of left or right presentations whose underlying matrix is given by *M*.

1.3.2 AsMorphismBetweenFreeLeftPresentations (for IsHomalgMatrix)

- ▷ AsMorphismBetweenFreeLeftPresentations(*m*) (attribute)
Returns: a morphism in $\text{Hom}(F^r, F^c)$
 The argument is a homalg matrix *m*. The output is a morphism $F^r \rightarrow F^c$ in the the category of left presentations whose underlying matrix is given by *m*, where F^r and F^c are free left presentations of ranks given by the number of rows and columns of *m*.

1.3.3 AsMorphismBetweenFreeRightPresentations (for IsHomalgMatrix)

▷ `AsMorphismBetweenFreeRightPresentations(m)` (attribute)

Returns: a morphism in $\text{Hom}(F^c, F^r)$

The argument is a homalg matrix m . The output is a morphism $F^c \rightarrow F^r$ in the the category of right presentations whose underlying matrix is given by m , where F^r and F^c are free right presentations of ranks given by the number of rows and columns of m .

1.3.4 AsLeftPresentation (for IsHomalgMatrix)

▷ `AsLeftPresentation(M)` (operation)

Returns: an object

The argument is a homalg matrix M over a ring R . The output is an object in the category of left presentations over R . This object has M as its underlying matrix.

1.3.5 AsRightPresentation (for IsHomalgMatrix)

▷ `AsRightPresentation(M)` (operation)

Returns: an object

The argument is a homalg matrix M over a ring R . The output is an object in the category of right presentations over R . This object has M as its underlying matrix.

1.3.6 AsLeftOrRightPresentation

▷ `AsLeftOrRightPresentation(M, l)` (function)

Returns: an object

The arguments are a homalg matrix M and a boolean l . If l is true, the output is an object in the category of left presentations. If l is false, the output is an object in the category of right presentations. In both cases, the underlying matrix of the result is M .

1.3.7 FreeLeftPresentation (for IsInt, IsHomalgRing)

▷ `FreeLeftPresentation(r, R)` (operation)

Returns: an object

The arguments are a non-negative integer r and a homalg ring R . The output is an object in the category of left presentations over R . It is represented by the $0 \times r$ matrix and thus it is free of rank r .

1.3.8 FreeRightPresentation (for IsInt, IsHomalgRing)

▷ `FreeRightPresentation(r, R)` (operation)

Returns: an object

The arguments are a non-negative integer r and a homalg ring R . The output is an object in the category of right presentations over R . It is represented by the $r \times 0$ matrix and thus it is free of rank r .

1.3.9 UnderlyingMatrix (for IsLeftOrRightPresentation)

▷ UnderlyingMatrix(A) (attribute)

Returns: a homalg matrix

The argument is an object A in the category of left or right presentations over a homalg ring R . The output is the underlying matrix which presents A .

1.3.10 UnderlyingHomalgRing (for IsLeftOrRightPresentation)

▷ UnderlyingHomalgRing(A) (attribute)

Returns: a homalg ring

The argument is an object A in the category of left or right presentations over a homalg ring R . The output is R .

1.3.11 Annihilator (for IsLeftOrRightPresentation)

▷ Annihilator(A) (attribute)

Returns: a morphism in $\text{Hom}(I, F)$

The argument is an object A in the category of left or right presentations. The output is the embedding of the annihilator I of A into the free module F of rank 1. In particular, the annihilator itself is seen as a left or right presentation.

1.3.12 LeftPresentations (for IsHomalgRing)

▷ LeftPresentations(R) (attribute)

Returns: a category

The argument is a homalg ring R . The output is the category of free left presentations over R .

1.3.13 RightPresentations (for IsHomalgRing)

▷ RightPresentations(R) (attribute)

Returns: a category

The argument is a homalg ring R . The output is the category of free right presentations over R .

1.4 Attributes

1.4.1 UnderlyingHomalgRing (for IsLeftOrRightPresentationMorphism)

▷ UnderlyingHomalgRing(R) (attribute)

Returns: a homalg ring

The argument is a morphism α in the category of left or right presentations over a homalg ring R . The output is R .

1.4.2 UnderlyingMatrix (for IsLeftOrRightPresentationMorphism)

▷ UnderlyingMatrix(α) (attribute)

Returns: a homalg matrix

The argument is a morphism α in the category of left or right presentations. The output is its underlying homalg matrix.

1.5 Non-Categorical Operations

1.5.1 StandardGeneratorMorphism (for IsLeftOrRightPresentation, IsInt)

▷ `StandardGeneratorMorphism(A, i)` (operation)

Returns: a morphism in $\text{Hom}(F, A)$

The argument is an object A in the category of left or right presentations over a homalg ring R with underlying matrix M and an integer i . The output is a morphism $F \rightarrow A$ given by the i -th row or column of M , where F is a free left or right presentation of rank 1.

1.5.2 CoverByFreeModule (for IsLeftOrRightPresentation)

▷ `CoverByFreeModule(A)` (attribute)

Returns: a morphism in $\text{Hom}(F, A)$

The argument is an object A in the category of left or right presentations. The output is a morphism from a free module F to A , which maps the standard generators of the free module to the generators of A .

1.6 Natural Transformations

1.6.1 NaturalIsomorphismFromIdentityToStandardModuleLeft (for IsHomalgRing)

▷ `NaturalIsomorphismFromIdentityToStandardModuleLeft(R)` (attribute)

Returns: a natural transformation $\text{Id} \rightarrow \text{StandardModuleLeft}$

The argument is a homalg ring R . The output is the natural isomorphism from the identity functor to the left standard module functor.

1.6.2 NaturalIsomorphismFromIdentityToStandardModuleRight (for IsHomalgRing)

▷ `NaturalIsomorphismFromIdentityToStandardModuleRight(R)` (attribute)

Returns: a natural transformation $\text{Id} \rightarrow \text{StandardModuleRight}$

The argument is a homalg ring R . The output is the natural isomorphism from the identity functor to the right standard module functor.

1.6.3 NaturalIsomorphismFromIdentityToGetRidOfZeroGeneratorsLeft (for IsHomalgRing)

▷ `NaturalIsomorphismFromIdentityToGetRidOfZeroGeneratorsLeft(R)` (attribute)

Returns: a natural transformation $\text{Id} \rightarrow \text{GetRidOfZeroGeneratorsLeft}$

The argument is a homalg ring R . The output is the natural isomorphism from the identity functor to the functor that gets rid of zero generators of left modules.

1.6.4 NaturalIsomorphismFromIdentityToGetRidOfZeroGeneratorsRight (for IsHomalgRing)

▷ `NaturalIsomorphismFromIdentityToGetRidOfZeroGeneratorsRight(R)` (attribute)

Returns: a natural transformation $\text{Id} \rightarrow \text{GetRidOfZeroGeneratorsRight}$

The argument is a homalg ring R . The output is the natural isomorphism from the identity functor to the functor that gets rid of zero generators of right modules.

1.6.5 NaturalIsomorphismFromIdentityToLessGeneratorsLeft (for IsHomalgRing)

▷ `NaturalIsomorphismFromIdentityToLessGeneratorsLeft` (R) (attribute)

Returns: a natural transformation $\text{Id} \rightarrow \text{LessGeneratorsLeft}$

The argument is a homalg ring R . The output is the natural morphism from the identity functor to the left less generators functor.

1.6.6 NaturalIsomorphismFromIdentityToLessGeneratorsRight (for IsHomalgRing)

▷ `NaturalIsomorphismFromIdentityToLessGeneratorsRight` (R) (attribute)

Returns: a natural transformation $\text{Id} \rightarrow \text{LessGeneratorsRight}$

The argument is a homalg ring R . The output is the natural morphism from the identity functor to the right less generator functor.

1.6.7 NaturalTransformationFromIdentityToDoubleDualLeft (for IsHomalgRing)

▷ `NaturalTransformationFromIdentityToDoubleDualLeft` (R) (attribute)

Returns: a natural transformation $\text{Id} \rightarrow \text{FunctorDoubleDualLeft}$

The argument is a homalg ring R . The output is the natural morphism from the identity functor to the double dual functor in left Presentations category.

1.6.8 NaturalTransformationFromIdentityToDoubleDualRight (for IsHomalgRing)

▷ `NaturalTransformationFromIdentityToDoubleDualRight` (R) (attribute)

Returns: a natural transformation $\text{Id} \rightarrow \text{FunctorDoubleDualRight}$

The argument is a homalg ring R . The output is the natural morphism from the identity functor to the double dual functor in right Presentations category.

Chapter 2

Examples and Tests

2.1 Annihilator

Example

```
gap> ZZ := HomalgRingOfIntegersInSingular();
gap> M1 := AsLeftPresentation( HomalgMatrix( [ [ "2" ] ], ZZ ) );
gap> M2 := AsLeftPresentation( HomalgMatrix( [ [ "3" ] ], ZZ ) );
gap> M3 := AsLeftPresentation( HomalgMatrix( [ [ "4" ] ], ZZ ) );
gap> M := DirectSum( M1, M2, M3 );
gap> Display( Annihilator( M ) );
12
```

A monomorphism in Category of left presentations of Z

```
gap> M1 := AsRightPresentation( HomalgMatrix( [ [ "2" ] ], ZZ ) );
gap> M2 := AsRightPresentation( HomalgMatrix( [ [ "3" ] ], ZZ ) );
gap> M3 := AsRightPresentation( HomalgMatrix( [ [ "4" ] ], ZZ ) );
gap> M := DirectSum( M1, M2, M3 );
gap> Display( Annihilator( M ) );
12
```

A monomorphism in Category of right presentations of Z

2.2 Intersection of Submodules

Example

```
gap> Q := HomalgFieldOfRationalsInSingular();
gap> R := Q * "x,y";
Q[x,y]
gap> F := AsLeftPresentation( HomalgMatrix( [ [ 0 ] ], R ) );
<An object in Category of left presentations of Q[x,y]>
gap> I1 := AsLeftPresentation( HomalgMatrix( [ [ "x" ] ], R ) );
gap> I2 := AsLeftPresentation( HomalgMatrix( [ [ "y" ] ], R ) );
gap> Display( I1 );
x
```

An object in Category of left presentations of Q[x,y]

```
gap> Display( I2 );
y
```

An object in Category of left presentations of $Q[x,y]$

```
gap> eps1 := PresentationMorphism( F, HomalgMatrix( [ [ 1 ] ], R ), I1 );
<A morphism in Category of left presentations of Q[x,y]>
gap> eps2 := PresentationMorphism( F, HomalgMatrix( [ [ 1 ] ], R ), I2 );
<A morphism in Category of left presentations of Q[x,y]>
gap> kernelemb1 := KernelEmbedding( eps1 );
<A monomorphism in Category of left presentations of Q[x,y]>
gap> kernelemb2 := KernelEmbedding( eps2 );
<A monomorphism in Category of left presentations of Q[x,y]>
gap> P := FiberProduct( kernelemb1, kernelemb2 );;
gap> Display( P );
(an empty 0 x 1 matrix)
```

An object in Category of left presentations of $Q[x,y]$

```
gap> pi1 := ProjectionInFactorOfFiberProduct( [ kernelemb1, kernelemb2 ], 1 );
<A monomorphism in Category of left presentations of Q[x,y]>
gap> composite := PreCompose( pi1, kernelemb1 );
<A monomorphism in Category of left presentations of Q[x,y]>
gap> Display( composite );
x*y
```

A monomorphism in Category of left presentations of $Q[x,y]$

2.3 Koszul Complex

Example

```
gap> Q := HomalgFieldOfRationalsInSingular();;
gap> R := Q * "x,y,z";;
gap> M := HomalgMatrix( [ [ "x" ], [ "y" ], [ "z" ] ], 3, 1, R );;
gap> M1 := AsLeftPresentation( M );;
gap> eps := CoverByFreeModule( M1 );;
gap> iota1 := KernelEmbedding( eps );;
gap> Display( iota1 );
x,
y,
z
```

A monomorphism in Category of left presentations of $Q[x,y,z]$

```
gap> Display( Source( iota1 ) );
0, -z,y,
-y,x, 0,
-z,0, x
```

An object in Category of left presentations of $Q[x,y,z]$

```
gap> pi1 := CoverByFreeModule( Source( iota1 ) );;
gap> d1 := PreCompose( pi1, iota1 );;
gap> Display( d1 );
x,
y,
z
```

A morphism in Category of left presentations of $Q[x,y,z]$

```
gap> iota2 := KernelEmbedding( d1 );;
gap> Display( iota2 );
0, -z,y,
-y,x, 0,
-z,0, x
```

A monomorphism in Category of left presentations of $Q[x,y,z]$

```
gap> Display( Source( iota2 ) );;
x,z,-y
```

An object in Category of left presentations of $Q[x,y,z]$

```
gap> pi2 := CoverByFreeModule( Source( iota2 ) );;
gap> d2 := PreCompose( pi2, iota2 );;
gap> Display( d2 );
0, -z,y,
-y,x, 0,
-z,0, x
```

A morphism in Category of left presentations of $Q[x,y,z]$

```
gap> iota3 := KernelEmbedding( d2 );;
gap> Display( iota3 );
x,z,-y
```

A monomorphism in Category of left presentations of $Q[x,y,z]$

```
gap> Display( Source( iota3 ) );
(an empty 0 x 1 matrix)
```

An object in Category of left presentations of $Q[x,y,z]$

```
gap> pi3 := CoverByFreeModule( Source( iota3 ) );;
gap> d3 := PreCompose( pi3, iota3 );;
gap> Display( d3 );
x,z,-y
```

A morphism in Category of left presentations of $Q[x,y,z]$

```
gap> N := HomalgMatrix( [ [ "x" ] ], 1, 1, R );;
gap> N1 := AsLeftPresentation( N );;
gap> d2N1 := TensorProductOnMorphisms( d2, IdentityMorphism( N1 ) );;
gap> d1N1 := TensorProductOnMorphisms( d1, IdentityMorphism( N1 ) );;
gap> IsZero( PreCompose( d2N1, d1N1 ) );
true
gap> cycles := KernelEmbedding( d1N1 );;
gap> boundaries := ImageEmbedding( d2N1 );;
gap> boundaries_in_cycles := LiftAlongMonomorphism( cycles, boundaries );;
gap> homology := CokernelObject( boundaries_in_cycles );;
gap> LessGenFunctor := FunctorLessGeneratorsLeft( R );;
gap> homology := ApplyFunctor( LessGenFunctor, homology );;
gap> StdBasisFunctor := FunctorStandardModuleLeft( R );;
gap> homology := ApplyFunctor( StdBasisFunctor, homology );;
gap> Display( homology );
z,
y,
x
```

An object in Category of left presentations of $\mathbb{Q}[x,y,z]$

2.4 Closed Monoidal Structure

Example

```
gap> R := HomalgRingOfIntegers( );
gap> M := AsLeftPresentation( HomalgMatrix( [ [ 2 ] ], 1, 1, R ) );
<An object in Category of left presentations of Z>
gap> N := AsLeftPresentation( HomalgMatrix( [ [ 3 ] ], 1, 1, R ) );
<An object in Category of left presentations of Z>
gap> T := TensorProductOnObjects( M, N );
<An object in Category of left presentations of Z>
gap> Display( T );
[ [ 3 ],
  [ 2 ] ]

An object in Category of left presentations of Z
gap> IsZero( T );
true
gap> H := InternalHomOnObjects( DirectSum( M, M ), DirectSum( M, N ) );
<An object in Category of left presentations of Z>
gap> Display( H );
[ [ -4, -2 ],
  [ 2, 2 ] ]

An object in Category of left presentations of Z
gap> alpha := StandardGeneratorMorphism( H, 1 );
<A morphism in Category of left presentations of Z>
gap> l := LambdaElimination( DirectSum( M, M ), DirectSum( M, N ), alpha );
<A morphism in Category of left presentations of Z>
gap> IsZero( l );
false
gap> Display( l );
[ [ 0, 0 ],
  [ 1, 0 ] ]

A morphism in Category of left presentations of Z
gap> alpha2 := StandardGeneratorMorphism( H, 2 );
<A morphism in Category of left presentations of Z>
gap> l2 := LambdaElimination( DirectSum( M, M ), DirectSum( M, N ), alpha2 );
<A morphism in Category of left presentations of Z>
gap> IsZero( l2 );
false
gap> Display( l2 );
[ [ 1, 0 ],
  [ 0, 0 ] ]

A morphism in Category of left presentations of Z
```

Index

- Annihilator
 - for IsLeftOrRightPresentation, 7
- AsLeftOrRightPresentation, 6
- AsLeftPresentation
 - for IsHomalgMatrix, 6
- AsMorphismBetweenFreeLeftPresentations
 - for IsHomalgMatrix, 5
- AsMorphismBetweenFreeRightPresentations
 - for IsHomalgMatrix, 6
- AsRightPresentation
 - for IsHomalgMatrix, 6
- CoverByFreeModule
 - for IsLeftOrRightPresentation, 8
- FreeLeftPresentation
 - for IsInt, IsHomalgRing, 6
- FreeRightPresentation
 - for IsInt, IsHomalgRing, 6
- FunctorDoubleDualLeft
 - for IsHomalgRing, 4
- FunctorDoubleDualRight
 - for IsHomalgRing, 4
- FunctorDualLeft
 - for IsHomalgRing, 4
- FunctorDualRight
 - for IsHomalgRing, 4
- FunctorGetRidOfZeroGeneratorsLeft
 - for IsHomalgRing, 3
- FunctorGetRidOfZeroGeneratorsRight
 - for IsHomalgRing, 3
- FunctorLessGeneratorsLeft
 - for IsHomalgRing, 3
- FunctorLessGeneratorsRight
 - for IsHomalgRing, 4
- FunctorStandardModuleLeft
 - for IsHomalgRing, 3
- FunctorStandardModuleRight
 - for IsHomalgRing, 3
- IsLeftOrRightPresentation
 - for IsCapCategoryObject, 5
- IsLeftOrRightPresentationMorphism
 - for IsCapCategoryMorphism, 4
- IsLeftPresentation
 - for IsLeftOrRightPresentation, 5
- IsLeftPresentationMorphism
 - for IsLeftOrRightPresentationMorphism, 5
- IsRightPresentation
 - for IsLeftOrRightPresentation, 5
- IsRightPresentationMorphism
 - for IsLeftOrRightPresentationMorphism, 5
- LeftPresentations
 - for IsHomalgRing, 7
- NaturalIsomorphismFromIdentityToGetRidOfZeroGeneratorsLeft
 - for IsHomalgRing, 8
- NaturalIsomorphismFromIdentityToGetRidOfZeroGeneratorsRight
 - for IsHomalgRing, 8
- NaturalIsomorphismFromIdentityToLessGeneratorsLeft
 - for IsHomalgRing, 9
- NaturalIsomorphismFromIdentityToLessGeneratorsRight
 - for IsHomalgRing, 9
- NaturalIsomorphismFromIdentityToStandardModuleLeft
 - for IsHomalgRing, 8
- NaturalIsomorphismFromIdentityToStandardModuleRight
 - for IsHomalgRing, 8
- NaturalTransformationFromIdentityToDoubleDualLeft
 - for IsHomalgRing, 9

NaturalTransformationFromIdentityTo-
DoubleDualRight
for IsHomalgRing, 9

PresentationMorphism
for IsLeftOrRightPresentation, IsHomalgMatrix, IsLeftOrRightPresentation,
5

RightPresentations
for IsHomalgRing, 7

StandardGeneratorMorphism
for IsLeftOrRightPresentation, IsInt, 8

UnderlyingHomalgRing
for IsLeftOrRightPresentation, 7
for IsLeftOrRightPresentationMorphism, 7

UnderlyingMatrix
for IsLeftOrRightPresentation, 7
for IsLeftOrRightPresentationMorphism, 7