ModulePresentationsForCAP

Category R-pres for CAP

2015.12.09

09/12/2015

Sebastian Gutsche

Sebastian Posur

Sebastian Gutsche
Email: gutsche@mathematik.uni-kl.de
Homepage: http://wwwb.math.rwth-aachen.de/~gutsche/
Address: Department of Mathematics
University of Kaiserslautern
67653 Kaiserslautern
Germany

Sebastian Posur
Email: sposur@momo.math.rwth-aachen.de
Homepage: http://wwwb.math.rwth-aachen.de/Mitarbeiter/posur.php
Address: Lehrstuhl B für Mathematik RWTH - Aachen
Templergraben 64
52062 Aachen
Germany

Contents

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 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 functor which takes a right presentation as input and computes its standard presentation.

1.1.3 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 it a presentation having less generators.

1.1.4 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 it a presentation having less generators.

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 $\mathrm{Hom}(A,B)$
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 $A \to B$ in the the category of left or right presentations whose underlying matrix is given by M.

1.3.2 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.

1.3.3 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.

1.3.4 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.

1.3.5 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.

1.3.6 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.

1.3.7 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.

1.3.8 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.

1.3.9 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.

1.3.10 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(alpha)` (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 $\mathrm{Hom}(F,A)$

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 $F \to A$ 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 $\mathrm{Hom}(F,A)$

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 Id → StandardModuleLeft

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

1.6.2 NaturalIsomorphismFromIdentityToStandardModuleRight (for IsHomalgRing)

▷ `NaturalIsomorphismFromIdentityToStandardModuleRight(`*R*`)` (attribute)

Returns: a natural transformation Id → StandardModuleRight

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

Chapter 2

Examples and Tests

2.1 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 mono morphism in Category of left presentations of Q[x,y]>
gap> kernelemb2 := KernelEmbedding( eps2 );
<A mono morphism 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 := ProjectionInFactor( P, 1 );
<A mono morphism in Category of left presentations of Q[x,y]>
gap> composite := PreCompose( pi1, kernelemb1 );
<A mono morphism in Category of left presentations of Q[x,y]>
gap> Display( composite );
x*y
```

```
A mono morphism in Category of left presentations of Q[x,y]
```

2.2 Koszul Complex

Example

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

A mono morphism 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 mono morphism 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 mono morphism 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> Nl := AsLeftPresentation( N );;
gap> d2Nl := TensorProductOnMorphisms( d2, IdentityMorphism( Nl ) );;
gap> d1Nl := TensorProductOnMorphisms( d1, IdentityMorphism( Nl ) );;
gap> IsZero( PreCompose( d2Nl, d1Nl ) );
true
gap> cycles := KernelEmbedding( d1Nl );;
gap> boundaries := ImageEmbedding( d2Nl );;
gap> boundaries_in_cyles := LiftAlongMonomorphism( cycles, boundaries );;
gap> homology := CokernelObject( boundaries_in_cyles );;
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 Q[x,y,z]
```

2.3 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 );
[ [   0,   0,   0,  -2 ],
  [   1,   2,   0,   0 ],
  [   0,   2,   2,   0 ],
  [   2,   3,   0,   2 ] ]

An object in Category of left presentations of Z
gap> alpha := StandardGeneratorMorphism( H, 3 );
<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 );
[ [  -2,   6 ],
  [  -1,  -3 ] ]

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

Index