GradedModulePresentationsForCAP

Presentations for graded modules

2019.08.07

7 August 2019

Sebastian Gutsche

Sebastian Gutsche
Email: gutsche@mathematik.uni-siegen.de
Homepage: https://sebasguts.github.io
Address: Department Mathematik
Universität Siegen
Walter-Flex-Straße 3
57068 Siegen
Germany

Contents

Chapter 1

Graded Module Presentations

1.1 GAP Categories

1.1.1 IsGradedLeftOrRightPresentationMorphism (for IsCapCategoryMorphism)

▷ `IsGradedLeftOrRightPresentationMorphism(`*object*`)` (filter)

Returns: `true` or `false`

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

1.1.2 IsGradedLeftPresentationMorphism (for IsGradedLeftOrRightPresentationMorphism)

▷ `IsGradedLeftPresentationMorphism(`*object*`)` (filter)

Returns: `true` or `false`

The GAP category of morphisms in the category of graded left presentations.

1.1.3 IsGradedRightPresentationMorphism (for IsGradedLeftOrRightPresentationMorphism)

▷ `IsGradedRightPresentationMorphism(`*object*`)` (filter)

Returns: `true` or `false`

The GAP category of morphisms in the category of graded right presentations.

1.1.4 IsGradedLeftOrRightPresentation (for IsCapCategoryObject)

▷ `IsGradedLeftOrRightPresentation(`*object*`)` (filter)

Returns: `true` or `false`

The GAP category of objects in the category of graded left presentations or graded right presentations.

1.1.5 IsGradedLeftPresentation (for IsGradedLeftOrRightPresentation)

▷ `IsGradedLeftPresentation(`*object*`)` (filter)

Returns: `true` or `false`

The GAP category of objects in the category of graded left presentations.

1.1.6 IsGradedRightPresentation (for IsGradedLeftOrRightPresentation)

▷ `IsGradedRightPresentation(object)` (filter)

Returns: `true` or `false`

The GAP category of objects in the category of graded right presentations.

1.2 Constructors

1.2.1 GradedPresentationMorphism (for IsGradedLeftOrRightPresentation, IsHomalgMatrix, IsGradedLeftOrRightPresentation)

▷ `GradedPresentationMorphism(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 graded left presentations or both be objects in the category of graded right presentations. The output is a morphism $A \to B$ in the the category of graded left or right presentations whose underlying matrix is given by M.

1.2.2 AsGradedLeftPresentation (for IsHomalgMatrix)

▷ `AsGradedLeftPresentation(M)` (attribute)

Returns: an object

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

1.2.3 AsGradedRightPresentation (for IsHomalgMatrix)

▷ `AsGradedRightPresentation(M)` (attribute)

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.

1.2.4 AsGradedLeftOrRightPresentation

▷ `AsGradedLeftOrRightPresentation(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.

1.2.5 GradedFreeLeftPresentation (for IsInt, IsHomalgRing)

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

Returns: an object

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

1.2.6 GradedFreeRightPresentation (for IsInt, IsHomalgRing)

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

Returns: an object

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

1.2.7 UnderlyingPresentationObject (for IsGradedLeftOrRightPresentation)

▷ `UnderlyingPresentationObject(A)` (attribute)

Returns: a left or right presentation

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

1.2.8 UnderlyingHomalgRing (for IsGradedLeftOrRightPresentation)

▷ `UnderlyingHomalgRing(A)` (attribute)

Returns: a homalg ring

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

1.2.9 GeneratorDegrees (for IsGradedLeftOrRightPresentation)

▷ `GeneratorDegrees(A)` (attribute)

Returns: a list

The argument is an object A in the category of graded left of right presentations over a ring R. The output is a list of elements of the degree group of R, the weights of the generators of A.

1.2.10 AffineDimension (for IsGradedLeftOrRightPresentation)

▷ `AffineDimension(A)` (attribute)

Returns: an integer

Returns the Krull dimension (of the annihilator ideal) of the underlying nongraded module `A`. The underlying ring must be commutative.

1.2.11 GradedLeftPresentations (for IsHomalgRing)

▷ `GradedLeftPresentations(R)` (attribute)

Returns: a category

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

1.2.12 GradedRightPresentations (for IsHomalgRing)

▷ `GradedRightPresentations(R)` (attribute)

Returns: a category

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

1.3 Attributes

1.3.1 UnderlyingHomalgRing (for IsGradedLeftOrRightPresentationMorphism)

▷ `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.

1.3.2 UnderlyingMatrix (for IsGradedLeftOrRightPresentationMorphism)

▷ `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.

1.4 Non-Categorical Operations

1.4.1 StandardGeneratorMorphism (for IsGradedLeftOrRightPresentation, 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.

1.4.2 CoverByProjective (for IsGradedLeftOrRightPresentation)

▷ `CoverByProjective(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.

1.5 Examples

Example

```
gap> LoadPackage( "GradedModulePresentationsForCAP" );
true
gap> Q := HomalgFieldOfRationalsInSingular( );
Q
gap> S := GradedRing( Q["x,y"] );
Q[x,y]
(weights: yet unset)
gap> Sgrmod := GradedLeftPresentations( S );
The category of graded left f.p. modules over Q[x,y] (with weights [ 1, 1 ])
gap> InfoOfInstalledOperationsOfCategory( Sgrmod );
40 primitive operations were used to derive 179 operations for this category which
* IsAbCategory
```

```
* IsMonoidalCategory
* IsAbelianCategoryWithEnoughProjectives
gap> #ListPrimitivelyInstalledOperationsOfCategory( Sgrmod );
gap> M := GradedFreeLeftPresentation( 2, S, [ 1, 1 ] );
<An object in The category of graded left f.p. modules over Q[x,y]
 (with weights [ 1, 1 ])>
gap> N := GradedFreeLeftPresentation( 1, S, [ 0 ] );
<An object in The category of graded left f.p. modules over Q[x,y]
 (with weights [ 1, 1 ])>
gap> mat := HomalgMatrix( "[x,y]", 2, 1, S );
<A 2 x 1 matrix over a graded ring>
gap> Display( mat );
x,
y
(over a graded ring)
gap> phi := GradedPresentationMorphism( M, mat, N );
<A morphism in The category of graded left f.p. modules over Q[x,y]
 (with weights [ 1, 1 ])>
gap> IsWellDefined( phi );
true
gap> IsMonomorphism( phi );
false
gap> IsEpimorphism( phi );
false
gap> iota := ImageEmbedding( phi );
<A monomorphism in The category of graded left f.p. modules over Q[x,y]
 (with weights [ 1, 1 ])>
gap> IsMonomorphism( iota );
true
gap> IsIsomorphism( iota );
false
gap> coker_mod := CokernelObject( phi );
<An object in The category of graded left f.p. modules over Q[x,y]
 (with weights [ 1, 1 ])>
gap> Display( coker_mod );
x,
y
(over a graded ring)

An object in The category of graded left f.p. modules over Q[x,y]
(with weights [ 1, 1 ])

(graded, degree of generator:[ 0 ])
gap> IsZero( coker_mod );
false
gap> is_artinian := M -> AffineDimension( M ) <= 0;
function( M ) ... end
gap> C := FullSubcategoryByMembershipFunction( Sgrmod, is_artinian );
<Subcategory of The category of graded left f.p. modules over Q[x,y]
 (with weights [ 1, 1 ]) by is_artinian>
gap> CohP1 := Sgrmod / C;
The Serre quotient category of The category of graded left f.p. modules
```

```
over Q[x,y] (with weights [ 1, 1 ]) by test function with name: is_artinian
gap> InfoOfInstalledOperationsOfCategory( CohP1 );
21 primitive operations were used to derive 146 operations for this category which
* IsAbCategory
* IsAbelianCategory
gap> Sh := CanonicalProjection( CohP1 );
Localization functor of The Serre quotient category of The category of graded left
f.p. modules over Q[x,y] (with weights [ 1, 1 ]) by test function with name:
is_artinian
gap> InstallFunctor( Sh, "Sheafification" );
gap> psi := ApplyFunctor( Sh, phi );
<A morphism in The Serre quotient category of The category of graded left
 f.p. modules over Q[x,y] (with weights [ 1, 1 ]) by test function with name:
 is_artinian>
gap> IsMonomorphism( psi );
false
gap> IsEpimorphism( psi );
true
gap> coker_shv := CokernelObject( psi );
<A zero object in The Serre quotient category of The category of graded left
 f.p. modules over Q[x,y] (with weights [ 1, 1 ]) by test function with name:
 is_artinian>
gap> IsZero( coker_shv );
true
gap> epsilon := ApplyFunctor( Sh, iota );
<A morphism in The Serre quotient category of The category of graded left
 f.p. modules over Q[x,y] (with weights [ 1, 1 ]) by test function with name:
 is_artinian>
gap> IsIsomorphism( epsilon );
true
```

Index