Presentations-ByProjectiveGrad-edModules

Graded module presentations for CAP over a graded ring

2019.03.15

15 March 2019

Martin Bies

Martin Bies
Email: martin.bies@alumni.uni-heidelberg.de
Homepage: https://www.ulb.ac.be/sciences/ptm/pmif/people.html
Address: Physique Théorique et Mathématique
Université Libre de Bruxelles
Campus Plaine - CP 231
Building NO - Level 6 - Office O.6.111
1050 Brussels
Belgium

Contents

1 The CAP category of graded module presentations for CAP **4**
- 1.1 The GAP categories for graded module presentations for CAP . . . 4
- 1.2 The GAP categories for graded module presentation morphisms for CAP . . . 5
- 1.3 CAP categories . . . 5

2 Graded submodules of projective graded modules over a graded ring **6**
- 2.1 GAP category of graded submodules for CAP . . . 6
- 2.2 GAP category of graded ideals for CAP . . . 6
- 2.3 Constructors for graded submodules from a list list and a graded ring . . . 7
- 2.4 Constructors for graded submodules from a list of lists and a specified superobject . . . 7
- 2.5 Constructors for graded submodules from a morphism . . . 8
- 2.6 Attributes for graded submodules . . . 8
- 2.7 Full information of a submodule . . . 9
- 2.8 Submodule powers . . . 9

3 Functors for graded module presentations for CAP **10**
- 3.1 Functor less generators for S-fpgrmod . . . 10
- 3.2 Functor StandardModule for S-fpgrmod . . . 11
- 3.3 Functor ByASmallerPresentation for S-fpgrmod . . . 11
- 3.4 The Frobenius-power functor . . . 12

4 Natural transformations for graded module presentations for CAP **14**
- 4.1 Natural isomorphism from identity functor to the standard module functor . . . 14

5 Tools **15**
- 5.1 Saturation . . . 15
- 5.2 Embeddings in projective modules . . . 16
- 5.3 Minimal free resolutions . . . 16
- 5.4 Betti tables . . . 16
- 5.5 Extension modules . . . 16
- 5.6 Twisting graded module presentations . . . 17

6 Examples and Tests **18**
- 6.1 The category SfpgrmodLeft . . . 18
- 6.2 The category SfpgrmodRight . . . 18
- 6.3 Graded left ideals . . . 19
- 6.4 Graded right ideals . . . 21

6.5 Graded left submodules 23
6.6 Graded right submodules 25
6.7 The Frobenius functor 28
6.8 Minimal free resolutions and Betti tables 30

Index **32**

Chapter 1

The CAP category of graded module presentations for CAP

1.1 The GAP categories for graded module presentations for CAP

1.1.1 IsGradedLeftOrRightModulePresentationForCAP (for IsCAPPresentationCategoryObject)

▷ `IsGradedLeftOrRightModulePresentationForCAP(`*object*`)` (filter)

Returns: `true` or `false`

The GAP category of graded left and right module presentations.

1.1.2 IsGradedLeftModulePresentationForCAP (for IsGradedLeftOrRightModulePresentationForCAP)

▷ `IsGradedLeftModulePresentationForCAP(`*object*`)` (filter)

Returns: `true` or `false`

The GAP category of objects in the presentation category over the category of projective graded left modules.

1.1.3 IsGradedRightModulePresentationForCAP (for IsGradedLeftOrRightModulePresentationForCAP)

▷ `IsGradedRightModulePresentationForCAP(`*object*`)` (filter)

Returns: `true` or `false`

The GAP category of objects in the presentation category over the category of projective graded right modules.

1.2 The GAP categories for graded module presentation morphisms for CAP

1.2.1 IsGradedLeftOrRightModulePresentationMorphismForCAP (for IsCAPPresentationCategoryMorphism)

▷ `IsGradedLeftOrRightModulePresentationMorphismForCAP(`*object*`)` (filter)

Returns: `true` or `false`

The GAP category of left or right module presentation morphisms

1.2.2 IsGradedLeftModulePresentationMorphismForCAP (for IsGradedLeftOrRightModulePresentationMorphismForCAP)

▷ `IsGradedLeftModulePresentationMorphismForCAP(`*object*`)` (filter)

Returns: `true` or `false`

The GAP category of morphisms in the presentation category over the category of projective graded left modules.

1.2.3 IsGradedRightModulePresentationMorphismForCAP (for IsGradedLeftOrRightModulePresentationMorphismForCAP)

▷ `IsGradedRightModulePresentationMorphismForCAP(`*object*`)` (filter)

Returns: `true` or `false`

The GAP category of morphisms in the presentation category over the category of projective graded right modules.

1.3 CAP categories

1.3.1 SfpgrmodLeft (for IsHomalgGradedRing)

▷ `SfpgrmodLeft(`*S*`)` (attribute)

Returns: a CapCategory

Given a graded ring *S*, one can consider the category of f.p. graded left *S*-modules, which is captured by this attribute.

1.3.2 SfpgrmodRight (for IsHomalgGradedRing)

▷ `SfpgrmodRight(`*S*`)` (attribute)

Returns: a CapCategory

Given a graded ring *S*, one can consider the category of f.p. graded right *S*-modules, which is captured by this attribute.

Chapter 2

Graded submodules of projective graded modules over a graded ring

2.1 GAP category of graded submodules for CAP

2.1.1 IsGradedLeftSubmoduleForCAP (for IsGradedLeftModulePresentationForCAP)

▷ `IsGradedLeftSubmoduleForCAP(`*object*`)` (filter)

Returns: true or false

The GAP category of graded left submodules for CAP.

2.1.2 IsGradedRightSubmoduleForCAP (for IsGradedRightModulePresentationForCAP)

▷ `IsGradedRightSubmoduleForCAP(`*object*`)` (filter)

Returns: true or false

The GAP category of graded right submodules for CAP.

2.1.3 IsGradedLeftOrRightSubmoduleForCAP (for IsGradedLeftOrRightModulePresentationForCAP)

▷ `IsGradedLeftOrRightSubmoduleForCAP(`*object*`)` (filter)

Returns: true or false

The GAP category of graded left or right submodules for CAP.

2.2 GAP category of graded ideals for CAP

2.2.1 IsGradedLeftIdealForCAP (for IsGradedLeftSubmoduleForCAP)

▷ `IsGradedLeftIdealForCAP(`*object*`)` (filter)

Returns: true or false

The GAP category of graded left ideals for CAP.

2.2.2 IsGradedRightIdealForCAP (for IsGradedRightSubmoduleForCAP)

▷ IsGradedRightIdealForCAP(*object*) (filter)

Returns: true or false

The GAP category of graded right ideals for CAP.

2.2.3 IsGradedLeftOrRightIdealForCAP (for IsGradedLeftOrRightSubmoduleForCAP)

▷ IsGradedLeftOrRightIdealForCAP(*object*) (filter)

Returns: true or false

The GAP category of graded left or right ideals for CAP.

2.3 Constructors for graded submodules from a list list and a graded ring

2.3.1 GradedLeftSubmoduleForCAP (for IsList, IsHomalgGradedRing)

▷ GradedLeftSubmoduleForCAP(*L*, *R*) (operation)

Returns: a graded left submodule for CAP

The arguments are a graded ring *R* and a list of lists *L* of homogeneous elements of *R* which generate the submodule. The method then returns the corresponding graded left submodule.

2.3.2 GradedRightSubmoduleForCAP (for IsList, IsHomalgGradedRing)

▷ GradedRightSubmoduleForCAP(*L*, *R*) (operation)

Returns: a graded right submodule for CAP

The arguments are a graded ring *R* and a list of lists *L* of homogeneous elements of *R* which generate the submodule. The method then returns the corresponding graded right submodule.

2.4 Constructors for graded submodules from a list of lists and a specified superobject

2.4.1 GradedLeftSubmoduleForCAP (for IsList, IsCAPCategoryOfProjectiveGradedLeftModulesObject)

▷ GradedLeftSubmoduleForCAP(*L*, *M*) (operation)

Returns: a graded left submodule for CAP

The arguments are a projective graded left module *M* defined over a graded ring *R* and a list of lists *L* of homogeneous elements from *R* which generate the submodule. The method then returns the corresponding graded left submodule of *M*.

2.4.2 GradedRightSubmoduleForCAP (for IsList, IsCAPCategoryOfProjectiveGradedRightModulesObject)

▷ GradedRightSubmoduleForCAP(*L*, *M*) (operation)

Returns: a graded right submodule for CAP

The arguments are a projective graded right module `M` defined over a graded ring `R` and a list of lists `L` of homogeneous elements from `R` which generate the submodule. The method then returns the corresponding graded right submodule of `M`.

2.5 Constructors for graded submodules from a morphism

2.5.1 GradedLeftSubmoduleForCAP (for IsCAPCategoryOfProjectiveGradedLeftModulesMorphism)

▷ `GradedLeftSubmoduleForCAP(a)` (operation)

Returns: a graded left submodule for CAP

The argument is a morphism of projective graded left modules `a`. The kernel embedding of a is then used to define a left presentation that we embed into the projective module Range(a). Thereby we constructed a graded left submodule.

2.5.2 GradedRightSubmoduleForCAP (for IsCAPCategoryOfProjectiveGradedRightModulesMorphism)

▷ `GradedRightSubmoduleForCAP(a)` (operation)

Returns: a graded right submodule for CAP

The argument is a morphism of projective graded right modules `a`. The kernel embedding of a is then used to define a right presentation that we embed into the projective module Range(a). Thereby we constructed a graded left submodule.

2.6 Attributes for graded submodules

2.6.1 PresentationForCAP (for IsGradedLeftOrRightSubmoduleForCAP)

▷ `PresentationForCAP(M)` (attribute)

Returns: a graded left presentation for CAP

The argument is a graded left or right submodule `M` over a graded ring. We then return a left or right presentation of this submodule, respectively.

2.6.2 Generators (for IsGradedLeftOrRightSubmoduleForCAP)

▷ `Generators(M)` (attribute)

Returns: a list

The argument is a graded left or right submodule `M` over a graded ring. We then return the list of generators of this submodule.

2.6.3 HomalgGradedRing (for IsGradedLeftOrRightSubmoduleForCAP)

▷ `HomalgGradedRing(M)` (attribute)

Returns: a graded homalg ring

The argument is a graded left or right submodule `M` over a graded ring. We then return this graded ring.

2.6.4 EmbeddingInSuperObjectForCAP (for IsGradedLeftOrRightSubmoduleForCAP)

▷ EmbeddingInSuperObjectForCAP(*I*) (attribute)

Returns: a CAP presentation category morphism

The argument is a graded left or right submodule *M* over a graded ring. We return the embedding of this module into the corresponding projective graded module.

2.6.5 SuperObjectForCAP (for IsGradedLeftOrRightSubmoduleForCAP)

▷ SuperObjectForCAP(*I*) (attribute)

Returns: a CAP presentation category object

The argument is a graded left or right submodule *M* in a graded ring. We return the superobject.

2.7 Full information of a submodule

The method 'FullInformation' is also avaiable to display all information about a graded submodule.

2.8 Submodule powers

2.8.1 * (for IsGradedLeftSubmoduleForCAP, IsGradedLeftSubmoduleForCAP)

▷ *(*arg1*, *arg2*) (operation)

2.8.2 * (for IsGradedRightSubmoduleForCAP, IsGradedRightSubmoduleForCAP)

▷ *(*arg1*, *arg2*) (operation)

2.8.3 \^ (for IsGradedLeftSubmoduleForCAP, IsInt)

▷ \^(*arg1*, *arg2*) (operation)

2.8.4 \^ (for IsGradedRightSubmoduleForCAP, IsInt)

▷ \^(*arg1*, *arg2*) (operation)

Chapter 3

Functors for graded module presentations for CAP

3.1 Functor less generators for S-fpgrmod

3.1.1 LessGradedGenerators (for IsGradedLeftOrRightModulePresentationForCAP)

▷ `LessGradedGenerators(M)` (operation)

Returns: a graded left or right module presentation for CAP

The argument is a graded left or right module presentation M for CAP. We then return a presentation of this module which uses less generators.

3.1.2 LessGradedGenerators (for IsGradedLeftOrRightModulePresentationMorphismForCAP)

▷ `LessGradedGenerators(a)` (operation)

Returns: a graded left or right module presentation morphism for CAP

The argument is a graded left or right module presentation morphism a for CAP. We then return a presentation of this morphism which uses less generators.

3.1.3 FunctorLessGradedGeneratorsLeft (for IsHomalgGradedRing)

▷ `FunctorLessGradedGeneratorsLeft(R)` (attribute)

Returns: a functor

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

3.1.4 FunctorLessGradedGeneratorsRight (for IsHomalgGradedRing)

▷ `FunctorLessGradedGeneratorsRight(R)` (attribute)

Returns: a functor

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

3.2 Functor StandardModule for S-fpgrmod

3.2.1 GradedStandardModule (for IsGradedLeftOrRightModulePresentationForCAP)

▷ `GradedStandardModule(M)` (operation)

Returns: a graded left or right module presentation for CAP

The argument is a graded left or right module presentation M for CAP. We then try to reduce the relations and thereby return a new presentation - the Standard module.

3.2.2 GradedStandardModule (for IsGradedLeftOrRightModulePresentationMorphismForCAP)

▷ `GradedStandardModule(a)` (operation)

Returns: a graded left or right module presentation morphism for CAP

The argument is a graded left or right module presentation morphism a for CAP. We then try to reduce the relations and thereby return a new presentation, which we term the Standard module morphism.

3.2.3 FunctorGradedStandardModuleLeft (for IsHomalgGradedRing)

▷ `FunctorGradedStandardModuleLeft(R)` (attribute)

Returns: a functor

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

3.2.4 FunctorGradedStandardModuleRight (for IsHomalgGradedRing)

▷ `FunctorGradedStandardModuleRight(R)` (attribute)

Returns: a functor

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

3.3 Functor ByASmallerPresentation for S-fpgrmod

3.3.1 ByASmallerPresentation (for IsGradedLeftOrRightModulePresentationForCAP)

▷ `ByASmallerPresentation(M)` (operation)

Returns: a graded left or right module presentation for CAP

The argument is a graded left or right module presentation M for CAP. We then return a smaller presentation of this module. This is obtained by first applying 'LessGenerators' and then 'StandardModule'.

3.3.2 ByASmallerPresentation (for IsGradedLeftOrRightModulePresentationMorphismForCAP)

▷ `ByASmallerPresentation(a)` (operation)

Returns: a graded left or right module presentation morphism for CAP

The argument is a graded left or right module presentation morphism a for CAP. We then return a smaller presentation of this morphism. This is obtained by first applying 'LessGenerators' and then 'StandardModule'.

3.3.3 FunctorByASmallerPresentationLeft (for IsHomalgGradedRing)

▷ `FunctorByASmallerPresentationLeft(R)` (attribute)

Returns: a functor

The argument is a homalg graded ring R. The output is functor which takes a left presentation in S-fpgrmodL as input and computes a smaller presentation. The latter is achieved by first applying 'LessGenerators' and then acting with 'StandardModule'.

3.3.4 FunctorByASmallerPresentationRight (for IsHomalgGradedRing)

▷ `FunctorByASmallerPresentationRight(R)` (attribute)

Returns: a functor

The argument is a homalg graded ring R. The output is functor which takes a right presentation in S-fpgrmodR as input and computes a smaller presentation. The latter is achieved by first applying 'LessGenerators' and then acting with 'StandardModule'.

3.4 The Frobenius-power functor

3.4.1 FrobeniusPower (for IsGradedLeftOrRightModulePresentationForCAP, IsInt)

▷ `FrobeniusPower(M, p)` (operation)

Returns: a presentation category object

The arguments are a CAPPresentationCategoryObject M and a non-negative integer p. This method then computes the p-th Frobenius power of M.

3.4.2 FrobeniusPower (for IsGradedLeftOrRightModulePresentationMorphismForCAP, IsInt)

▷ `FrobeniusPower(M, p)` (operation)

Returns: a presentation category morphism

The arguments are a CAPPresentationCategoryMorphism M and a non-negative integer p. This method then computes the p-th Frobenius power of M.

3.4.3 FrobeniusPowerWithGivenSourceAndRangePowers (for IsGradedLeftOrRightModulePresentationMorphismForCAP, IsInt,IsGradedLeftOrRightModulePresentationForCAP, IsGradedLeftOrRightModulePresentationForCAP)

▷ `FrobeniusPowerWithGivenSourceAndRangePowers(m, p, s', r')` (operation)

Returns: a presentation category morphism

The arguments are a CAPPresentationCategoryMorphism m, a non-negative integer p, the p-th Frobenius power of the source of m, s', and the p-th Frobenius power of the range of m, r'. This method then computes the p-th Frobenius power of m by use of s' and r'.

3.4.4 FrobeniusPowerFunctorLeft (for IsHomalgGradedRing, IsInt)

▷ `FrobeniusPowerFunctorLeft(R, p)` (operation)

Returns: a functor

The argument is a homalg graded ring R and a non-negative integers p. The output is the functor which takes graded left-presentations and -morphisms to their p-th Frobenius power.

3.4.5 FrobeniusPowerFunctorRight (for IsHomalgGradedRing, IsInt)

▷ `FrobeniusPowerFunctorRight(R, p)` (operation)

Returns: a functor

The argument is a homalg graded ring R and a non-negative integers p. The output is the functor which takes graded right-presentations and -morphisms to their p-th Frobenius power.

Chapter 4

Natural transformations for graded module presentations for CAP

4.1 Natural isomorphism from identity functor to the standard module functor

4.1.1 NaturalIsomorphismFromIdentityToGradedStandardModuleLeft (for IsHomalgGradedRing)

▷ `NaturalIsomorphismFromIdentityToGradedStandardModuleLeft(`*S*`)` (attribute)

Returns: a natural transformation Id ⇒ StandardModuleLeft

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

4.1.2 NaturalIsomorphismFromIdentityToGradedStandardModuleRight (for IsHomalgGradedRing)

▷ `NaturalIsomorphismFromIdentityToGradedStandardModuleRight(`*S*`)` (attribute)

Returns: a natural transformation Id ⇒ StandardModuleRight

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

Chapter 5

Tools

5.1 Saturation

5.1.1 Saturate (for IsGradedLeftModulePresentationForCAP, IsGradedLeftIdealForCAP)

▷ `Saturate(M, I)` (operation)

Returns: a presentation category object

The arguments are two CAPPresentationCategoryObject M and a graded left ideal I. We then compute the saturation of M with respect to I.

5.1.2 Saturate (for IsGradedRightModulePresentationForCAP, IsGradedRightIdealForCAP)

▷ `Saturate(M, I)` (operation)

Returns: a presentation category object

The arguments are two CAPPresentationCategoryObject M and a graded right ideal I. We then compute the saturation of M with respect to I.

5.1.3 EmbeddingInSaturationOfGradedModulePresentation (for IsGradedLeftModulePresentationForCAP, IsGradedLeftIdealForCAP)

▷ `EmbeddingInSaturationOfGradedModulePresentation(M, I)` (operation)

Returns: a presentation category morphism

The arguments are two CAPPresentationCategoryObject M and a graded left idea lI. We then compute the embedding of M into its saturation with respect to I.

5.1.4 EmbeddingInSaturationOfGradedModulePresentation (for IsGradedRightModulePresentationForCAP, IsGradedRightIdealForCAP)

▷ `EmbeddingInSaturationOfGradedModulePresentation(M, I)` (operation)

Returns: a presentation category morphism

The arguments are two CAPPresentationCategoryObject M and a graded right ideal I. We then compute the embedding of M into its saturation with respect to I.

5.2 Embeddings in projective modules

5.2.1 EmbeddingInProjectiveObject (for IsGradedLeftOrRightModulePresentationForCAP)

▷ `EmbeddingInProjectiveObject(M)` (operation)

Returns: a presentation category morphism

The argument is a CAPPresentationCategoryObject `M`, which is represented by a morphism *m* in the underlying category of projective modules. In this category we can compute the cokernel projection *m* (although this need not be possible in more general proj-categories). The range of this morphism is a projective module. The zero morphism into this very projective module defines an object of the presentation category, which allows us to embed `M` into a projective module presentation. The corresponding presentation category morphism is returned.

5.3 Minimal free resolutions

5.3.1 MinimalFreeResolutionForCAP (for IsGradedLeftOrRightModulePresentationForCAP)

▷ `MinimalFreeResolutionForCAP(M)` (attribute)

Returns: a complex of projective graded module morphisms

The argument is a graded left or right module presentation `M`. We then compute a minimal free resolution of `M`.

5.4 Betti tables

5.4.1 BettiTableForCAP (for IsGradedLeftOrRightModulePresentationForCAP)

▷ `BettiTableForCAP(M)` (attribute)

Returns: a list of lists

The argument is a graded left or right module presentation `M`. We then compute the Betti table of `M`.

5.5 Extension modules

5.5.1 GradedExtForCAP (for IsInt, IsGradedLeftOrRightModulePresentationForCAP, IsGradedLeftOrRightModulePresentationForCAP)

▷ `GradedExtForCAP(e, M1, M2)` (operation)

Returns: a f.p. graded module

The arguments are an integer `i` and two f.p. graded modules `M_1` and `M_2`. Then this method computes `\text{Ext}^i \left( M_1, M_2 \right)` .

5.6 Twisting graded module presentations

5.6.1 Twist (for IsGradedLeftOrRightModulePresentationForCAP, IsList)

▷ `Twist(M)` (operation)

Returns: a GradedLeftOrRightModulePresentationForCAP

The argument is a graded left or right module presentation *M* and an element of the class group of the ring over which this module is graded. We then compute the twisted graded module presentation.

5.6.2 Twist (for IsGradedLeftOrRightModulePresentationForCAP, IsHomalgModuleElement)

▷ `Twist(arg1, arg2)` (operation)

Chapter 6

Examples and Tests

6.1 The category SfpgrmodLeft

Example

```
gap> Q := HomalgFieldOfRationalsInSingular();
Q
gap> S := GradedRing( Q * "x_1, x_2, x_3, x_4" );
Q[x_1,x_2,x_3,x_4]
(weights: yet unset)
gap> SetWeightsOfIndeterminates( S, [[1,0],[1,0],[0,1],[0,1]] );

gap> category_left := SfpgrmodLeft( S );
Category of graded left module presentations over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
gap> functor1_left := FunctorLessGradedGeneratorsLeft( S );
Less generators for Category of graded left module presentations over
Q[x_1,x_2,x_3,x_4] (with weights
[ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
gap> functor2_left := FunctorGradedStandardModuleLeft( S );
Graded standard module for Category of graded left module presentations over
Q[x_1,x_2,x_3,x_4] (with weights
[ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
gap> natural_transformation_left :=
> NaturalIsomorphismFromIdentityToGradedStandardModuleLeft( S );
Natural isomorphism from Id to Graded standard module for Category of graded
left module presentations over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
```

6.2 The category SfpgrmodRight

Example

```
gap> category_right := SfpgrmodRight( S );
Category of graded right module presentations over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
gap> functor1_right := FunctorLessGradedGeneratorsRight( S );
Less generators for Category of graded right module presentations over
Q[x_1,x_2,x_3,x_4] (with weights
[ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
```

```
gap> functor2_right := FunctorGradedStandardModuleRight( S );
Graded standard module for Category of graded right module presentations over
Q[x_1,x_2,x_3,x_4] (with weights
[ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
gap> natural_transformation_right :=
> NaturalIsomorphismFromIdentityToGradedStandardModuleRight( S );
Natural isomorphism from Id to Graded standard module for Category of graded
right module presentations over Q[x_1,x_2,x_3,x_4] (with weights
[ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
```

6.3 Graded left ideals

Example

```
gap> IdealLeft := GradedLeftSubmoduleForCAP( [ [ "x_1*x_3" ],
>              [ "x_1*x_4" ], [ "x_2*x_3" ] , [ "x_2*x_4" ] ], S );
<A graded left ideal of Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> Generators( IdealLeft );
[ [ "x_1*x_3" ], [ "x_1*x_4" ], [ "x_2*x_3" ], [ "x_2*x_4" ] ]
gap> HomalgGradedRing( IdealLeft );
Q[x_1,x_2,x_3,x_4]
(weights: [ ( 1, 0 ), ( 1, 0 ), ( 0, 1 ), ( 0, 1 ) ])
gap> FullInformation( SuperObjectForCAP( IdealLeft ) );

================================================================================

A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 0 and degrees:
[  ]

A morphism in the category of projective graded left modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
(an empty 0 x 1 matrix)

A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 1 and degrees:
[ [ 0, 1 ] ]

================================================================================
gap> FullInformation( EmbeddingInSuperObjectForCAP( IdealLeft ) );

================================================================================

Source:
-------
A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 4 and degrees:
[ [ ( 1, 2 ), 2 ], [ ( 2, 1 ), 2 ] ]

A morphism in the category of projective graded left modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
```

```
with matrix:
-x_4,x_3, 0,   0,
0,   0,   -x_4,x_3,
-x_2,0,   x_1, 0,
0,   -x_2,0,   x_1
(over a graded ring)

A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 4 and degrees:
[ [ ( 1, 1 ), 4 ] ]

--------------------------------------------------------------------------------

Mapping matrix:
---------------
A morphism in the category of projective graded left modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
x_1*x_3,
x_1*x_4,
x_2*x_3,
x_2*x_4
(over a graded ring)

--------------------------------------------------------------------------------

Range:
------
A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 0 and degrees:
[  ]

A morphism in the category of projective graded left modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
(an empty 0 x 1 matrix)

A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 1 and degrees:
[ [ 0, 1 ] ]

================================================================================

gap> IdealLeftToPower2 := IdealLeft * IdealLeft;
<A graded left ideal of Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> Display( IdealLeftToPower2 );
A graded left ideal of Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) generated by
[ [ x_1^2*x_3^2 ], [ x_1^2*x_3*x_4 ], [ x_1*x_2*x_3^2 ], [ x_1*x_2*x_3*x_4 ],
[ x_1^2*x_3*x_4 ], [ x_1^2*x_4^2 ], [ x_1*x_2*x_3*x_4 ], [ x_1*x_2*x_4^2 ],
[ x_1*x_2*x_3^2 ], [ x_1*x_2*x_3*x_4 ], [ x_2^2*x_3^2 ], [ x_2^2*x_3*x_4 ],
```

```
[ x_1*x_2*x_3*x_4 ], [ x_1*x_2*x_4^2 ], [ x_2^2*x_3*x_4 ], [ x_2^2*x_4^2 ] ]
gap> 2ndFrobPowerIdealLeft := FrobeniusPower( IdealLeft, 2 );
<A graded left ideal of Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> Display( 2ndFrobPowerIdealLeft );
A graded left ideal of Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) generated by
[ [ x_1^2*x_3^2 ], [ x_1^2*x_4^2 ], [ x_2^2*x_3^2 ], [ x_2^2*x_4^2 ] ]
```

6.4 Graded right ideals

Example

```
gap> IdealRight := GradedRightSubmoduleForCAP( [ [ "x_1*x_3",
>              "x_1*x_4", "x_2*x_3", "x_2*x_4" ] ], S );
<A graded right ideal of Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> Generators( IdealRight );
[ [ "x_1*x_3", "x_1*x_4", "x_2*x_3", "x_2*x_4" ] ]
gap> HomalgGradedRing( IdealRight );
Q[x_1,x_2,x_3,x_4]
(weights: [ ( 1, 0 ), ( 1, 0 ), ( 0, 1 ), ( 0, 1 ) ])
gap> FullInformation( SuperObjectForCAP( IdealRight ) );

================================================================================

A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 0 and degrees:
[  ]

A morphism in the category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
(an empty 1 x 0 matrix)

A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 1 and degrees:
[ [ 0, 1 ] ]

================================================================================

gap> FullInformation( EmbeddingInSuperObjectForCAP( IdealRight ) );

================================================================================

Source:
-------
A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 4 and degrees:
[ [ ( 1, 2 ), 2 ], [ ( 2, 1 ), 2 ] ]

A morphism in the category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
```

```
with matrix:
-x_4,0,   -x_2,0,
x_3, 0,   0,   -x_2,
0,   -x_4,x_1, 0,
0,   x_3, 0,   x_1
(over a graded ring)

A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 4 and degrees:
[ [ ( 1, 1 ), 4 ] ]

------------------------------------------------------------------------------

Mapping matrix:
---------------
A morphism in the category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
x_1*x_3, x_1*x_4, x_2*x_3, x_2*x_4
(over a graded ring)

------------------------------------------------------------------------------

Range:
------
A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 0 and degrees:
[  ]

A morphism in the category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
(an empty 1 x 0 matrix)

A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 1 and degrees:
[ [ 0, 1 ] ]

==============================================================================

gap> IdealRightToPower2 := IdealRight * IdealRight;
<A graded right ideal of Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> Display( IdealRightToPower2 );
A graded right ideal of Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) generated by
[ [ x_1^2*x_3^2, x_1^2*x_3*x_4, x_1*x_2*x_3^2, x_1*x_2*x_3*x_4, x_1^2*x_3*x_4,
x_1^2*x_4^2, x_1*x_2*x_3*x_4, x_1*x_2*x_4^2, x_1*x_2*x_3^2, x_1*x_2*x_3*x_4,
x_2^2*x_3^2, x_2^2*x_3*x_4, x_1*x_2*x_3*x_4, x_1*x_2*x_4^2, x_2^2*x_3*x_4,
x_2^2*x_4^2 ] ]
gap> 2ndFrobPowerIdealRight := FrobeniusPower( IdealRight, 2 );
<A graded right ideal of Q[x_1,x_2,x_3,x_4]
```

```
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> Display( 2ndFrobPowerIdealRight );
A graded right ideal of Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) generated by
[ [ x_1^2*x_3^2, x_1^2*x_4^2, x_2^2*x_3^2, x_2^2*x_4^2 ] ]
```

6.5 Graded left submodules

Example

```
gap> SubmoduleLeft := GradedLeftSubmoduleForCAP( [ [ "x_1*x_3" ],
>              [ "x_1*x_4" ], [ "x_2*x_3" ], [ "x_2*x_4" ] ], S );
<A graded left ideal of Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> Generators( SubmoduleLeft );
[ [ "x_1*x_3" ], [ "x_1*x_4" ], [ "x_2*x_3" ], [ "x_2*x_4"]  ]
gap> HomalgGradedRing( SubmoduleLeft );
Q[x_1,x_2,x_3,x_4]
(weights: [ ( 1, 0 ), ( 1, 0 ), ( 0, 1 ), ( 0, 1 ) ])
gap> SubmoduleLeft2 := GradedLeftSubmoduleForCAP( [ [ "x_1*x_3", 1 ],
>              [ "x_1*x_4", 1 ], [ "x_2*x_3", 1 ], [ "x_2*x_4", 1 ] ],
>              CAPCategoryOfProjectiveGradedLeftModulesObject( [[[0,0],1], [[1,1],1]], S ) );
<A graded left submodule over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> FullInformation( EmbeddingInSuperObjectForCAP( SubmoduleLeft2 ) );

==============================================================================

Source:
-------
A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 3 and degrees:
[ [ ( 1, 2 ), 1 ], [ ( 2, 1 ), 1 ], [ ( 2, 2 ), 1 ] ]

A morphism in the category of projective graded left modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
x_4,             -x_3,            -x_4,           x_3,
x_2,             -x_2,            -x_1,           x_1,
-x_2*x_3+x_1*x_4,-x_1*x_3+x_2*x_3,x_1*x_3-x_1*x_4,0
(over a graded ring)
A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 4 and degrees:
[ [ ( 1, 1 ), 4 ] ]

------------------------------------------------------------------------------

Mapping matrix:
---------------
A morphism in the category of projective graded left modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
x_1*x_3,1,
```

```
x_1*x_4,1,
x_2*x_3,1,
x_2*x_4,1
(over a graded ring)

-------------------------------------------------------------------------------

Range:
------
A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 0 and degrees:
[  ]

A morphism in the category of projective graded left modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
(an empty 0 x 2 matrix)

A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 2 and degrees:
[ [ 0, 1 ], [ ( 1, 1 ), 1 ] ]

===============================================================================

gap> IsGradedLeftSubmoduleForCAP( SubmoduleLeft2 );
true
gap> SubmoduleLeft3 := GradedLeftSubmoduleForCAP( [[ "x_1", 1 ], [ "x_2", 1 ]],
>                   CAPCategoryOfProjectiveGradedLeftModulesObject(
>                                                       [[[-1,0],1],[[0,0],1]], S )
>                                                                               );
<A graded left submodule over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> FullInformation( EmbeddingInSuperObjectForCAP( SubmoduleLeft3 ) );

===============================================================================

Source:
-------
A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 0 and degrees:
[  ]

A morphism in the category of projective graded left modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
(an empty 0 x 2 matrix)

A projective graded left module over Q[x_1,x_2,x_3,x_4] (with weights
[ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 2 and degrees:
[ [ 0, 2 ] ]

-------------------------------------------------------------------------------
```

```
Mapping matrix:
---------------
A morphism in the category of projective graded left modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
x_1,1,
x_2,1
(over a graded ring)

------------------------------------------------------------------------------

Range:
------
A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 0 and degrees:
[  ]

A morphism in the category of projective graded left modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
(an empty 0 x 2 matrix)

A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 2 and degrees:
[ [ ( -1, 0 ), 1 ], [ 0, 1 ] ]

==============================================================================
```

6.6 Graded right submodules

Example

```
gap> SubmoduleRight := GradedRightSubmoduleForCAP( [ [ "x_1*x_3",
>                   "x_1*x_4", "x_2*x_3", "x_2*x_4" ] ], S );
<A graded right ideal of Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> Generators( SubmoduleRight );
[ [ "x_1*x_3", "x_1*x_4", "x_2*x_3", "x_2*x_4"] ]
gap> HomalgGradedRing( SubmoduleRight );
Q[x_1,x_2,x_3,x_4]
(weights: [ ( 1, 0 ), ( 1, 0 ), ( 0, 1 ), ( 0, 1 ) ])
gap> SubmoduleRight2 := GradedRightSubmoduleForCAP( [ [ "x_1*x_3",
>                    "x_1*x_4", "x_2*x_3", "x_2*x_4" ], [ 1, 1, 1, 1 ] ],
>                    CAPCategoryOfProjectiveGradedRightModulesObject( [[[0,0],1], [[1,1],1]], S )
<A graded right submodule over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> FullInformation( EmbeddingInSuperObjectForCAP( SubmoduleRight2 ) );

==============================================================================

Source:
-------
```

```
A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 3 and degrees:
[ [ ( 1, 2 ), 1 ], [ ( 2, 1 ), 1 ], [ ( 2, 2 ), 1 ] ]

A morphism in the category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
x_4, x_2, -x_2*x_3+x_1*x_4,
-x_3,-x_2,-x_1*x_3+x_2*x_3,
-x_4,-x_1,x_1*x_3-x_1*x_4,
x_3, x_1, 0
(over a graded ring)

A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 4 and degrees:
[ [ ( 1, 1 ), 4 ] ]

--------------------------------------------------------------------------------

Mapping matrix:
---------------
A morphism in the category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
x_1*x_3,x_1*x_4,x_2*x_3,x_2*x_4,
1,      1,      1,      1
(over a graded ring)

--------------------------------------------------------------------------------

Range:
------
A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 0 and degrees:
[  ]

A morphism in the category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
(an empty 2 x 0 matrix)

A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 2 and degrees:
[ [ 0, 1 ], [ ( 1, 1 ), 1 ] ]

================================================================================

gap> IsGradedRightSubmoduleForCAP( SubmoduleRight2 );
true
gap> SubmoduleRight3 := GradedRightSubmoduleForCAP( [[ "x_1", "x_2" ], [ 1, 1 ]],
>                   CAPCategoryOfProjectiveGradedRightModulesObject(
>                                                   [[[-1,0],1],[[0,0],1]], S )
```

```
>                                                                                );
<A graded right submodule over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> FullInformation( EmbeddingInSuperObjectForCAP( SubmoduleRight3 ) );

==============================================================================

Source:
-------
A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 0 and degrees:
[  ]

A morphism in the category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
(an empty 2 x 0 matrix)

A projective graded right module over Q[x_1,x_2,x_3,x_4] (with weights
[ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 2 and degrees:
[ [ 0, 2 ] ]

------------------------------------------------------------------------------

Mapping matrix:
---------------
A morphism in the category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
x_1,x_2,
1,1
(over a graded ring)

------------------------------------------------------------------------------

Range:
------
A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 0 and degrees:
[  ]

A morphism in the category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
(an empty 2 x 0 matrix)

A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 2 and degrees:
[ [ ( -1, 0 ), 1 ], [ 0, 1 ] ]

==============================================================================
```

6.7 The Frobenius functor

Example

```
gap> frob_functor_left := FrobeniusPowerFunctorLeft( S, 2 );
Frobenius functor for Category of graded left module presentations over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
to the power 2
gap> FullInformation( ApplyFunctor( frob_functor_left,
>                                   EmbeddingInSuperObjectForCAP( IdealLeft ) ) );

==============================================================================

Source:
-------
A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 4 and degrees:
[ [ ( 2, 4 ), 2 ], [ ( 4, 2 ), 2 ] ]

A morphism in the category of projective graded left modules over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ]) with matrix:
-x_4^2,x_3^2, 0,     0,
0,     0,     -x_4^2,x_3^2,
-x_2^2,0,     x_1^2, 0,
0,     -x_2^2,0,     x_1^2
(over a graded ring)

A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 4 and degrees:
[ [ ( 2, 2 ), 4 ] ]

------------------------------------------------------------------------------

Mapping matrix:
---------------
A morphism in the category of projective graded left modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
x_1^2*x_3^2,
x_1^2*x_4^2,
x_2^2*x_3^2,
x_2^2*x_4^2
(over a graded ring)

------------------------------------------------------------------------------

Range:
------
A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 0 and degrees:
[  ]

A morphism in the category of projective graded left modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
```

```
(an empty 0 x 1 matrix)

A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 1 and degrees:
[ [ 0, 1 ] ]

================================================================================
gap> frob_functor_right := FrobeniusPowerFunctorRight( S, 2 );
Frobenius functor for Category of graded right module presentations over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
to the power 2
gap> FullInformation( ApplyFunctor( frob_functor_right,
>                                  EmbeddingInSuperObjectForCAP( IdealRight ) ) );

================================================================================

Source:
-------
A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 4 and degrees:
[ [ ( 2, 4 ), 2 ], [ ( 4, 2 ), 2 ] ]

A morphism in the category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
with matrix:
-x_4^2,0,     -x_2^2,0,
x_3^2, 0,     0,     -x_2^2,
0,     -x_4^2,x_1^2, 0,
0,     x_3^2, 0,     x_1^2
(over a graded ring)

A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 4 and degrees:
[ [ ( 2, 2 ), 4 ] ]

--------------------------------------------------------------------------------

Mapping matrix:
---------------
A morphism in the category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
with matrix:
x_1^2*x_3^2,x_1^2*x_4^2,x_2^2*x_3^2,x_2^2*x_4^2
(over a graded ring)

--------------------------------------------------------------------------------

Range:
------
A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 0 and degrees:
[  ]
```

```
A morphism in the category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
with matrix:
(an empty 1 x 0 matrix)

A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 1 and degrees:
[ [ 0, 1 ] ]

================================================================================
```

6.8 Minimal free resolutions and Betti tables

Example

```
gap> res1 := MinimalFreeResolutionForCAP( IdealLeft );
<An object in Complex category of CAP category of projective graded left modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> FullInformation( res1 );
[ [ [ 1, 1 ], 4 ] ]
 ^
 |
-x_4,x_3, 0,   0,
0,   0,   -x_4,x_3,
-x_2,0,   x_1, 0,
0,   -x_2,0,   x_1
(over a graded ring)
 |
[ [ [ 1, 2 ], 2 ], [ [ 2, 1 ], 2 ] ]
 ^
 |
-x_2,x_1,x_4,-x_3
(over a graded ring)
 |
[ [ [ 2, 2 ], 1 ] ]

gap> betti1 := BettiTableForCAP( IdealLeft );
[ [ ( 1, 1 ), ( 1, 1 ), ( 1, 1 ), ( 1, 1 ) ], [ ( 1, 2 ), ( 1, 2 ),
( 2, 1 ), ( 2, 1 ) ], [ ( 2, 2 ) ] ]
gap> res2 := MinimalFreeResolutionForCAP( IdealRight );
<An object in Complex category of CAP category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> FullInformation( res2 );
[ [ [ 1, 1 ], 4 ] ]
 ^
 |
-x_4,0,   -x_2,0,
x_3, 0,   0,   -x_2,
0,   -x_4,x_1, 0,
0,   x_3, 0,   x_1
(over a graded ring)
 |
```

```
[ [ [ 1, 2 ], 2 ], [ [ 2, 1 ], 2 ] ]
 ^
 |
-x_2,
x_1,
x_4,
-x_3
(over a graded ring)
 |
[ [ [ 2, 2 ], 1 ] ]

gap> betti2 := BettiTableForCAP( IdealRight );
[ [ ( 1, 1 ), ( 1, 1 ), ( 1, 1 ), ( 1, 1 ) ], [ ( 1, 2 ), ( 1, 2 ),
( 2, 1 ), ( 2, 1 ) ], [ ( 2, 2 ) ] ]
```

Index

BettiTableForCAP
for IsGradedLeftOrRightModulePresentationForCAP, 16
*
for IsGradedLeftSubmoduleForCAP, IsGradedLeftSubmoduleForCAP, 9
for IsGradedRightSubmoduleForCAP, IsGradedRightSubmoduleForCAP, 9
\^
for IsGradedLeftSubmoduleForCAP, IsInt, 9
for IsGradedRightSubmoduleForCAP, IsInt, 9
ByASmallerPresentation
for IsGradedLeftOrRightModulePresentationForCAP, 11
for IsGradedLeftOrRightModulePresentationMorphismForCAP, 12

EmbeddingInProjectiveObject
for IsGradedLeftOrRightModulePresentationForCAP, 16
EmbeddingInSaturationOfGradedModulePresentation
for IsGradedLeftModulePresentationForCAP, IsGradedLeftIdealForCAP, 15
for IsGradedRightModulePresentationForCAP, IsGradedRightIdealForCAP, 15
EmbeddingInSuperObjectForCAP
for IsGradedLeftOrRightSubmoduleForCAP, 9

FrobeniusPower
for IsGradedLeftOrRightModulePresentationForCAP, IsInt, 12
for IsGradedLeftOrRightModulePresentationMorphismForCAP, IsInt, 12
FrobeniusPowerFunctorLeft
for IsHomalgGradedRing, IsInt, 13
FrobeniusPowerFunctorRight
for IsHomalgGradedRing, IsInt, 13
FrobeniusPowerWithGivenSourceAndRangePowers
for IsGradedLeftOrRightModulePresentationMorphismForCAP, IsInt,IsGradedLeftOrRightModulePresentationForCAP, IsGradedLeftOrRightModulePresentationForCAP, 13
FunctorByASmallerPresentationLeft
for IsHomalgGradedRing, 12
FunctorByASmallerPresentationRight
for IsHomalgGradedRing, 12
FunctorGradedStandardModuleLeft
for IsHomalgGradedRing, 11
FunctorGradedStandardModuleRight
for IsHomalgGradedRing, 11
FunctorLessGradedGeneratorsLeft
for IsHomalgGradedRing, 10
FunctorLessGradedGeneratorsRight
for IsHomalgGradedRing, 10

Generators
for IsGradedLeftOrRightSubmoduleForCAP, 8
GradedExtForCAP
for IsInt, IsGradedLeftOrRightModulePresentationForCAP, IsGradedLeftOrRightModulePresentationForCAP, 16
GradedLeftSubmoduleForCAP
for IsCAPCategoryOfProjectiveGradedLeftModulesMorphism, 8
for IsList, IsCAPCategoryOfProjectiveGradedLeftModulesObject, 7
for IsList, IsHomalgGradedRing, 7
GradedRightSubmoduleForCAP
for IsCAPCategoryOfProjectiveGradedRightModulesMorphism, 8

for IsList, IsCAPCategoryOfProjectiveGradedRightModulesObject, 7
for IsList, IsHomalgGradedRing, 7
GradedStandardModule
for IsGradedLeftOrRightModulePresentationForCAP, 11
for IsGradedLeftOrRightModulePresentationMorphismForCAP, 11

HomalgGradedRing
for IsGradedLeftOrRightSubmoduleForCAP, 8

IsGradedLeftIdealForCAP
for IsGradedLeftSubmoduleForCAP, 6
IsGradedLeftModulePresentationForCAP
for IsGradedLeftOrRightModulePresentationForCAP, 4
IsGradedLeftModulePresentationMorphismForCAP
for IsGradedLeftOrRightModulePresentationMorphismForCAP, 5
IsGradedLeftOrRightIdealForCAP
for IsGradedLeftOrRightSubmoduleForCAP, 7
IsGradedLeftOrRightModulePresentationForCAP
for IsCAPPresentationCategoryObject, 4
IsGradedLeftOrRightModulePresentationMorphismForCAP
for IsCAPPresentationCategoryMorphism, 5
IsGradedLeftOrRightSubmoduleForCAP
for IsGradedLeftOrRightModulePresentationForCAP, 6
IsGradedLeftSubmoduleForCAP
for IsGradedLeftModulePresentationForCAP, 6
IsGradedRightIdealForCAP
for IsGradedRightSubmoduleForCAP, 7
IsGradedRightModulePresentationForCAP
for IsGradedLeftOrRightModulePresentationForCAP, 4
IsGradedRightModulePresentationMorphismForCAP
for IsGradedLeftOrRightModulePresentationMorphismForCAP, 5
IsGradedRightSubmoduleForCAP
for IsGradedRightModulePresentationForCAP, 6

LessGradedGenerators
for IsGradedLeftOrRightModulePresentationForCAP, 10
for IsGradedLeftOrRightModulePresentationMorphismForCAP, 10

MinimalFreeResolutionForCAP
for IsGradedLeftOrRightModulePresentationForCAP, 16

NaturalIsomorphismFromIdentityToGradedStandardModuleLeft
for IsHomalgGradedRing, 14
NaturalIsomorphismFromIdentityToGradedStandardModuleRight
for IsHomalgGradedRing, 14

PresentationForCAP
for IsGradedLeftOrRightSubmoduleForCAP, 8

Saturate
for IsGradedLeftModulePresentationForCAP, IsGradedLeftIdealForCAP, 15
for IsGradedRightModulePresentationForCAP, IsGradedRightIdealForCAP, 15
SfpgrmodLeft
for IsHomalgGradedRing, 5
SfpgrmodRight
for IsHomalgGradedRing, 5
SuperObjectForCAP
for IsGradedLeftOrRightSubmoduleForCAP, 9

Twist
for IsGradedLeftOrRightModulePresentationForCAP, IsHomalgModuleElement, 17
for IsGradedLeftOrRightModulePresentationForCAP, IsList, 17