TruncationsOfPresentationsByProjectiveGradedModules

Truncations of graded module presentations (for CAP) to affine semigroups

2019.10.05

5 October 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

Chapter 1

Wrapper for generators of semigroups and hyperplane constraints of cones

1.1 GAP Categories

1.1.1 IsSemigroupForPresentationsByProjectiveGradedModules (for IsObject)

▷ `IsSemigroupForPresentationsByProjectiveGradedModules(`*object*`)` (filter)

Returns: `true` or `false`

The GAP category of lists of integer-valued lists, which encode the generators of subsemigroups of Z^n.

1.1.2 IsAffineSemigroupForPresentationsByProjectiveGradedModules (for IsObject)

▷ `IsAffineSemigroupForPresentationsByProjectiveGradedModules(`*object*`)` (filter)

Returns: `true` or `false`

The GAP category of affine semigroups H in $\mathbb{Z}^n$. That means that there is a semigroup $G \subseteq \mathbb{Z}^n$ and $p \in \mathbb{Z}^n$ such that $H = p + G$.

1.2 Constructors

1.2.1 SemigroupForPresentationsByProjectiveGradedModules (for IsList, IsInt)

▷ `SemigroupForPresentationsByProjectiveGradedModules(`*L*`)` (operation)

Returns: a SemigroupGeneratorList

The argument is a list L and a non-negative integer d. We then check if this list could be the list of generators of a subsemigroup of Z^d. If so, we create the corresponding SemigroupGeneratorList.

1.2.2 SemigroupForPresentationsByProjectiveGradedModules (for IsList)

▷ `SemigroupForPresentationsByProjectiveGradedModules(`*arg*`)` (operation)

1.2.3 AffineSemigroupForPresentationsByProjectiveGradedModules (for IsSemigroupForPresentationsByProjectiveGradedModules, IsList)

▷ `AffineSemigroupForPresentationsByProjectiveGradedModules(L, p)` (operation)

Returns: an AffineSemigroup

The argument is a SemigroupForPresentationsByProjectiveGradedModules S and a point $p \in \mathbb{Z}^n$ encoded as list of integers. We then compute the affine semigroup $p+S$. Alternatively to S we allow the use of either a list of generators (or a list of generators together with the embedding dimension).

1.2.4 AffineSemigroupForPresentationsByProjectiveGradedModules (for IsList, IsList)

▷ `AffineSemigroupForPresentationsByProjectiveGradedModules(arg1, arg2)` (operation)

1.2.5 AffineSemigroupForPresentationsByProjectiveGradedModules (for IsList, IsInt, IsList)

▷ `AffineSemigroupForPresentationsByProjectiveGradedModules(arg1, arg2, arg3)` (operation)

1.3 Attributes

1.3.1 GeneratorList (for IsSemigroupForPresentationsByProjectiveGradedModules)

▷ `GeneratorList(L)` (attribute)

Returns: a list

The argument is a SemigroupForPresentationsByProjectiveGradedModules L. We then return the list of its generators.

1.3.2 EmbeddingDimension (for IsSemigroupForPresentationsByProjectiveGradedModules)

▷ `EmbeddingDimension(L)` (attribute)

Returns: a non-negative integer

The argument is a SemigroupForPresentationsByProjectiveGradedModules L. We then return the embedding dimension of this semigroup.

1.3.3 ConeHPresentationList (for IsSemigroupForPresentationsByProjectiveGradedModules)

▷ `ConeHPresentationList(L)` (attribute)

Returns: a list or fail

The argument is a SemigroupForPresentationsByProjectiveGradedModules L. If the associated semigroup is a cone semigroup, then (during construction) an H-presentation of that cone was computed. We return the list of the corresponding H-constraints. This conversion uses Normaliz and can fail if the cone if not full-dimensional. In case that such a conversion error occured, the attribute is set to the value 'fail'.

1.3.4 Offset (for IsAffineSemigroupForPresentationsByProjectiveGradedModules)

▷ `Offset(S)` (attribute)

Returns: a list of integers

The argument is an AffineSemigroupForPresentationsByProjectiveGradedModules S. This one is given as $S = p + H$ for a point $p \in \mathbb{Z}^n$ and a semigroup $H \subseteq \mathbb{Z}^n$. We then return the offset p.

1.3.5 UnderlyingSemigroup (for IsAffineSemigroupForPresentationsByProjectiveGradedModules)

▷ `UnderlyingSemigroup(S)` (attribute)

Returns: a SemigroupGeneratorList

The argument is an IsAffineSemigroupForPresentationsByProjectiveGradedModules S. This one is given as $S = p + H$ for a point $p \in \mathbb{Z}^n$ and a semigroup $H \subseteq \mathbb{Z}^n$. We then return the SemigroupGeneratorList of H.

1.3.6 EmbeddingDimension (for IsAffineSemigroupForPresentationsByProjectiveGradedModules)

▷ `EmbeddingDimension(S)` (attribute)

Returns: a non-negative integer

The argument is an IsAffineSemigroupForPresentationsByProjectiveGradedModules S. We then return the embedding dimension of this affine semigroup.

1.4 Property

1.4.1 IsTrivial (for IsSemigroupForPresentationsByProjectiveGradedModules)

▷ `IsTrivial(L)` (property)

Returns: true or false

The argument is a SemigroupForPresentationsByProjectiveGradedModules L. This property returns 'true' if this semigroup is trivial and 'false' otherwise.

1.4.2 IsSemigroupOfCone (for IsSemigroupForPresentationsByProjectiveGradedModules)

▷ `IsSemigroupOfCone(L)` (property)

Returns: true, false

The argument is a SemigroupForPresentationsByProjectiveGradedModules L. We return if this is the semigroup of a cone.

1.4.3 IsTrivial (for IsAffineSemigroupForPresentationsByProjectiveGradedModules)

▷ `IsTrivial(L)` (property)

Returns: true or false

The argument is an AffineSemigroupForPresentationsByProjectiveGradedModules. This property returns 'true' if the underlying semigroup is trivial and otherwise 'false'.

1.4.4 IsAffineSemigroupOfCone (for IsAffineSemigroupForPresentationsByProjectiveGradedModules)

▷ `IsAffineSemigroupOfCone(H)` (property)

Returns: true, false or fail

The argument is an IsAffineSemigroupForPresentationsByProjectiveGradedModules H. We return if this is an AffineConeSemigroup. If Normaliz cannot decide this 'fail' is returned.

1.5 Operations

1.5.1 DecideIfIsConeSemigroupGeneratorList (for IsList)

▷ `DecideIfIsConeSemigroupGeneratorList(L)` (operation)

Returns: true, false or fail

The argument is a list L of generators of a semigroup in $\mathbb{Z}^n$. We then check if this is the semigroup of a cone. In this case we return 'true', otherwise 'false'. If the operation fails due to shortcommings in Normaliz we return 'fail'.

1.6 Check if point is contained in (affine) cone or (affine) semigroup

1.6.1 PointContainedInSemigroup (for IsSemigroupForPresentationsByProjectiveGradedModules, IsList)

▷ `PointContainedInSemigroup(S, p)` (operation)

Returns: true or false

The argument is a SemigroupForPresentationsByProjectiveGradedModules S of $\mathbb{Z}^n$, and an integral point p in this lattice. This operation then verifies if the point p is contained in S or not.

1.6.2 PointContainedInAffineSemigroup (for IsAffineSemigroupForPresentationsByProjectiveGradedModules, IsList)

▷ `PointContainedInAffineSemigroup(H, p)` (operation)

Returns: true or false

The argument is an IsAffineSemigroupForPresentationsByProjectiveGradedModules H and a point p. The second argument This method then checks if p lies in H.

Chapter 2

Functors for the category of projective graded left modules

2.1 Basic functionality for truncations

2.1.1 TruncationOfProjectiveGradedModule (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject, IsSemigroupForPresentationsByProjectiveGradedModules)

▷ `TruncationOfProjectiveGradedModule(M, H)` (operation)

Returns: an object

Consider a graded ring R such that its degree group is identical to $\mathbb{Z}^n$ for suitable $n \in \mathbb{N}_{\geq 0}$. Then consider a projective graded left module M over R and a subsemigroup H in the degree group of R. We expect that H is given to the method as a SemigroupForPresentationsByProjectiveGradedModules. Under these circumstances we truncate M to the subsemigroup H.

2.1.2 EmbeddingOfTruncationOfProjectiveGradedModule (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject, IsSemigroupForPresentationsByProjectiveGradedModules)

▷ `EmbeddingOfTruncationOfProjectiveGradedModule(M, H)` (operation)

Returns: a morphism

Consider a graded ring R such that its degree group is identical to $\mathbb{Z}^n$ for suitable $n \in \mathbb{N}_{\geq 0}$. Then consider a projective graded left module M over R and a subsemigroup H in the degree group of R. We expect that H is given to the method as a SemigroupForPresentationsByProjectiveGradedModules. Under these circumstances we compute the embedding of the truncation of M onto the subsemigroup H into M.

2.1.3 EmbeddingOfTruncationOfProjectiveGradedModuleWithGivenTruncationObject (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject,IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject)

▷ `EmbeddingOfTruncationOfProjectiveGradedModuleWithGivenTruncationObject(M, N)` (operation)

Returns: a morphism

Consider a graded ring R such that its degree group is identical to $\mathbb{Z}^n$ for suitable $n \in \mathbb{N}_{\geq 0}$. Then consider a projective graded left module M over R and a semigroup H given as a SemigroupForPresentationsByProjectiveGradedModules. The method accepts M and its truncation $M|_H$ as arguments and then computes the embedding $M|_H \hookrightarrow M$.

2.1.4 ProjectionOntoTruncationOfProjectiveGradedModule (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject, IsSemigroupForPresentationsByProjectiveGradedModules)

▷ `ProjectionOntoTruncationOfProjectiveGradedModule(M, H)` (operation)

Returns: a morphism

Consider a graded ring R such that its degree group is identical to $\mathbb{Z}^n$ for suitable $n \in \mathbb{N}_{\geq 0}$. Then consider a projective graded left module M over R and a subsemigroup H in the degree group of R. We expect that H is given to the method as a SemigroupForPresentationsByProjectiveGradedModules. Under these circumstances we compute the projection morphism of M onto its truncation to the subsemigroup H

2.1.5 ProjectionOntoTruncationOfProjectiveGradedModuleWithGivenTruncationObject (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject,IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject)

▷ `ProjectionOntoTruncationOfProjectiveGradedModuleWithGivenTruncationObject(M, N)` (operation)

Returns: a morphism

Consider a graded ring R such that its degree group is identical to $\mathbb{Z}^n$ for suitable $n \in \mathbb{N}_{\geq 0}$. Then consider a projective graded left module M over R and the semigroup H given as SemigroupForPresentationsByProjectiveGradedModules. The method accepts M and its truncation $M|_H$ and then computes the projection $M \twoheadrightarrow M|_H$.

2.2 The truncation functor

2.2.1 TruncationFunctorForProjectiveGradedLeftModules (for IsHomalgGradedRing, IsSemigroupForPresentationsByProjectiveGradedModules)

▷ `TruncationFunctorForProjectiveGradedLeftModules(R, H)` (operation)

Returns: a functor

The argument is a homalg graded ring R and a subsemigroup H (given as SemigroupForPresentationsByProjectiveGradedModules) in the degree group of the ring R. The output is the functor which truncates projective graded left-modules and left-module-morphisms to the subsemigroup H.

2.2.2 TruncationFunctorForProjectiveGradedRightModules (for IsHomalgGradedRing, IsSemigroupForPresentationsByProjectiveGradedModules)

▷ `TruncationFunctorForProjectiveGradedRightModules(R, H)` (operation)

Returns: a functor

The argument is a homalg graded ring R and a subsemigroup H (given as SemigroupForPresentationsByProjectiveGradedModules) in the degree group of the ring R. The output is the functor which truncates projective graded right-modules and right-module-morphisms to the subsemigroup H.

Chapter 3

Natural transformations

3.1 Natural transformations for projective graded modules

3.1.1 NaturalTransformationFromTruncationToIdentityForProjectiveGradedLeftModules (for IsHomalgGradedRing, IsSemigroupForPresentationsByProjectiveGradedModules)

▷ `NaturalTransformationFromTruncationToIdentityForProjectiveGradedLeftModules(S)` (operation)

Returns: a natural transformation $\cdot|_H \Rightarrow \mathrm{id}$

The argument is a homalg graded ring S and a semigroup H in the degree group of S. The output is the natural transformation from the left truncation functor (to H) to the identity functor.

3.1.2 NaturalTransformationFromTruncationToIdentityForProjectiveGradedRightModules (for IsHomalgGradedRing, IsSemigroupForPresentationsByProjectiveGradedModules)

▷ `NaturalTransformationFromTruncationToIdentityForProjectiveGradedRightModules(S)` (operation)

Returns: a natural transformation $\cdot|_H \Rightarrow \mathrm{id}$

The argument is a homalg graded ring S and a semigroup H in the degree group of S. The output is the natural transformation from the right truncation functor (to H) to the identity functor.

Chapter 4

Functors for graded module presentations for CAP

4.1 The truncation functor to semigroups

4.1.1 Truncation (for IsGradedLeftOrRightModulePresentationForCAP, IsSemigroupForPresentationsByProjectiveGradedModules)

▷ `Truncation(M, H)` (operation)

Returns: a graded left or right module presentation for CAP

The argument is a graded left or right module presentation M for CAP and and a semigroup H given as SemigroupForPresentationsByProjectiveGradedModules. We then return the truncation of M onto H.

4.1.2 Truncation (for IsGradedLeftOrRightModulePresentationMorphismForCAP, IsSemigroupForPresentationsByProjectiveGradedModules)

▷ `Truncation(a, H)` (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 and a semigroup H given as IsSemigroupForPresentationsByProjectiveGradedModules. We then return the truncation of a to H.

4.1.3 TruncationFunctorLeft (for IsHomalgGradedRing, IsSemigroupForPresentationsByProjectiveGradedModules)

▷ `TruncationFunctorLeft(R, C)` (operation)

Returns: a functor

The argument is a homalg graded ring R and a semigroup H (given as SemigroupForPresentationsByProjectiveGradedModules) in the degree group of the ring R. The output is the functor which truncates left-presentations over R to this subsemigroup.

4.1.4 TruncationFunctorRight (for IsHomalgGradedRing, IsSemigroupForPresentationsByProjectiveGradedModules)

▷ `TruncationFunctorRight(R, C)` (operation)

Returns: a functor

The argument is a homalg graded ring *R* and a semigroup *H* (given as SemigroupForPresentationsByProjectiveGradedModules) in the degree group of the ring *R*. The output is the functor which truncates right-presentations over *R* to this subsemigroup.

Chapter 5

Examples and Tests

5.1 Cone and semigroup wrappers

The following commands are used to handle generators of semigroups in $\mathbb{Z}^n$, generators of cones in $\mathbb{Z}^n$ as well as hyperplane constraints that define cones in $\mathbb{Z}^n$. Here are some examples:

Example
```
gap> semigroup1 := SemigroupForPresentationsByProjectiveGradedModules( [[ 1,0 ], [ 1,1 ]] );
<A cone-semigroup in Z^2 formed as the span of 2 generators>
gap> IsSemigroupForPresentationsByProjectiveGradedModules( semigroup1 );
true
gap> GeneratorList( semigroup1 );
[ [ 1, 0 ], [ 1, 1 ] ]
gap> semigroup2 := SemigroupForPresentationsByProjectiveGradedModules( [[ 2,0 ], [ 1,1 ]] );
<A non-cone semigroup in Z^2 formed as the span of 2 generators>
gap> IsSemigroupForPresentationsByProjectiveGradedModules( semigroup2 );
true
gap> GeneratorList( semigroup2 );
[ [ 2, 0 ], [ 1, 1 ] ]
```

We can check if a semigroup in $\mathbb{Z}^n$ is the semigroup of a cone. In case we can look at an H-presentation of this cone.

Example
```
gap> IsSemigroupOfCone( semigroup1 );
true
gap> ConeHPresentationList( semigroup1 );
[ [ 0, 1 ], [ 1, -1 ] ]
gap> Display( ConeHPresentationList( semigroup1 ) );
[ [   0,   1 ],
  [   1,  -1 ] ]
gap> IsSemigroupOfCone( semigroup2 );
false
gap> HasConeHPresentationList( semigroup2 );
false
```

We can check membership of points in semigroups.

Example
```
gap> PointContainedInSemigroup( semigroup2, [ 1,0 ] );
false
```

```
gap> PointContainedInSemigroup( semigroup2, [ 2,0 ] );
true
```

Given a semigroup $S \subseteq \mathbb{Z}^n$ and a point $p \in \mathbb{Z}^n$ we can consider

$$H := p + S = \{p + x \, , \, x \in S\} \, .$$

We term this an affine semigroup. Given that $S = C \cap \mathbb{Z}^n$ for a cone $C \subseteq \mathbb{Z}^n$, we use the term affine cone_semigroup. The constructors are as follows:

Example

```
gap> affine_semigroup1 := AffineSemigroupForPresentationsByProjectiveGradedModules( semigroup1, [
<A non-trivial affine cone-semigroup in Z^2>
gap> affine_semigroup2 := AffineSemigroupForPresentationsByProjectiveGradedModules( semigroup2, [
<A non-trivial affine non-cone semigroup in Z^2>
```

We can access the properties of these affine semigroups as follows.

Example

```
gap> IsAffineSemigroupOfCone( affine_semigroup2 );
false
gap> UnderlyingSemigroup( affine_semigroup2 );
<A non-cone semigroup in Z^2 formed as the span of 2 generators>
gap> Display( UnderlyingSemigroup( affine_semigroup2 ) );
A non-cone semigroup in Z^2 formed as the span of 2 generators - generators are as follows:
[ [  2,  0 ],
  [  1,  1 ] ]
gap> IsAffineSemigroupOfCone( affine_semigroup1 );
true
gap> Offset( affine_semigroup2 );
[ 2, 2 ]
gap> ConeHPresentationList( UnderlyingSemigroup( affine_semigroup1 ) );
[ [ 0, 1 ], [ 1, -1 ] ]
```

Of course we can also decide membership in affine (cone_)semigroups.

Example

```
gap> Display( affine_semigroup1 );
A non-trivial affine cone-semigroup in Z^2
Offset: [ -1, -1 ]
Hilbert basis: [ [ 1, 0 ], [ 1, 1 ] ]
gap> PointContainedInAffineSemigroup( affine_semigroup1, [ -2,-2 ] );
false
gap> PointContainedInAffineSemigroup( affine_semigroup1, [ 3,1 ] );
true
gap> Display( affine_semigroup2 );
A non-trivial affine non-cone semigroup in Z^2
Offset: [ 2, 2 ]
Semigroup generators: [ [ 2, 0 ], [ 1, 1 ] ]
gap> PointContainedInAffineSemigroup( affine_semigroup2, [ 3,2 ] );
false
gap> PointContainedInAffineSemigroup( affine_semigroup2, [ 3,3 ] );
true
```

5.2 Truncations of projective graded left modules

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> D := DegreeGroup( S );
<A free left module of rank 2 on free generators>
gap> IsFree( D );
true
gap> NewObjectL := CAPCategoryOfProjectiveGradedLeftModulesObject(
>                  [ [[1,0],1], [[-1,-1],2] ], S );
<A projective graded left module of rank 3>
gap> tL := TruncationOfProjectiveGradedModule( NewObjectL,
>        SemigroupForPresentationsByProjectiveGradedModules( [[1,0],[0,1]] ) );
<A projective graded left module of rank 1>
gap> Display( tL );
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: [ [ ( 1, 0 ), 1 ] ]
gap> tL2 := TruncationOfProjectiveGradedModule( NewObjectL,
>         SemigroupForPresentationsByProjectiveGradedModules( [[ 1,0 ], [ 0,2 ]] ) );
<A projective graded left module of rank 1>
gap> Display( tL2 );
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: [ [ ( 1, 0 ), 1 ] ]
gap> embL := EmbeddingOfTruncationOfProjectiveGradedModule( NewObjectL,
>          SemigroupForPresentationsByProjectiveGradedModules( [[1,0],[0,1]] ) );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( embL ) );
1, 0, 0
(over a graded ring)
gap> embL2 := EmbeddingOfTruncationOfProjectiveGradedModule( NewObjectL,
>          SemigroupForPresentationsByProjectiveGradedModules( [[1,0],[0,2]] ) );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( embL2 ) );
1, 0, 0
(over a graded ring)
gap> embL3 := EmbeddingOfTruncationOfProjectiveGradedModuleWithGivenTruncationObject(
>           NewObjectL, tL );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( embL3 ) );
1, 0, 0
(over a graded ring)
gap> projL := ProjectionOntoTruncationOfProjectiveGradedModule( NewObjectL,
```

```
>          SemigroupForPresentationsByProjectiveGradedModules( [[ 1,0 ], [ 0,1 ]] ) );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( projL ) );
1,
0,
0
(over a graded ring)
gap> projL2 := ProjectionOntoTruncationOfProjectiveGradedModule( NewObjectL,
>          SemigroupForPresentationsByProjectiveGradedModules( [[ 1,0 ], [ 0,2 ]] ) );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( projL2 ) );
1,
0,
0
(over a graded ring)
gap> projL3 := ProjectionOntoTruncationOfProjectiveGradedModuleWithGivenTruncationObject(
>           NewObjectL, tL );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( projL3 ) );
1,
0,
0
(over a graded ring)
gap> truncatorL := TruncationFunctorForProjectiveGradedLeftModules(
>                         S, SemigroupForPresentationsByProjectiveGradedModules( [[ 1,0 ], [ 0,2
Truncation functor for 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 ] ])
to the semigroup generated by [ [ 1, 0 ], [ 0, 2 ] ]
gap> truncatorL2 := TruncationFunctorForProjectiveGradedLeftModules(
>                         S, SemigroupForPresentationsByProjectiveGradedModules( [[ 1,0 ], [ 0,1
Truncation functor for 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 ] ])
to the semigroup generated by [ [ 1,0 ], [ 0, 1 ] ]
gap> tL2 := ApplyFunctor( truncatorL, NewObjectL );
<A projective graded left module of rank 1>
gap> Display( tL2 );
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: [ [ ( 1, 0 ), 1 ] ]
gap> sourceL := CAPCategoryOfProjectiveGradedLeftModulesObject(
>             [ [[1,0],1], [[0,1],1] ], S );
<A projective graded left module of rank 2>
gap> rangeL := CAPCategoryOfProjectiveGradedLeftModulesObject(
>             [ [[1,0],1] ], S );
<A projective graded left module of rank 1>
gap> test_morphismL := CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism(
>       sourceL, HomalgMatrix( [ [ 1 ],[ 0 ] ], S ) ,rangeL );
```

```
<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 ] ])>
gap> tr_test_morphismL := ApplyFunctor( truncatorL, test_morphismL );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( tr_test_morphismL ) );
1
(over a graded ring)
gap> tr2_test_morphismL := ApplyFunctor( truncatorL2, test_morphismL );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( tr2_test_morphismL ) );
1,
0
(over a graded ring)
gap> nat_trans_l := NaturalTransformationFromTruncationToIdentityForProjectiveGradedLeftModules(
>                           S, SemigroupForPresentationsByProjectiveGradedModules( [[ 1,0 ], [ 0,1
Natural transformation from Truncation functor for 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 ] ])
to the semigroup generated by [ [ 1, 0 ], [ 0, 1 ] ] to id
gap> component_l := ApplyNaturalTransformation( nat_trans_l, NewObjectL );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( component_l ) );
1, 0, 0
(over a graded ring)
```

5.3 Truncations of projective graded right modules

Example

```
gap> NewObjectR := CAPCategoryOfProjectiveGradedRightModulesObject(
>                   [ [[1,0],1], [[-1,-1],2] ], S );
<A projective graded right module of rank 3>
gap> tR := TruncationOfProjectiveGradedModule( NewObjectR,
>        SemigroupForPresentationsByProjectiveGradedModules( [[1,0],[0,1]] ) );
<A projective graded right module of rank 1>
gap> Display( tR );
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: [ [ ( 1, 0 ), 1 ] ]
gap> tR2 := TruncationOfProjectiveGradedModule( NewObjectR,
>         SemigroupForPresentationsByProjectiveGradedModules( [[ 1,0 ], [ 0,2 ]] ) );
<A projective graded right module of rank 1>
gap> Display( tR2 );
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: [ [ ( 1, 0 ), 1 ] ]
gap> embR := EmbeddingOfTruncationOfProjectiveGradedModule( NewObjectR,
>          SemigroupForPresentationsByProjectiveGradedModules( [[1,0],[0,1]] ) );
<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 ] ])>
```

```
gap> Display( UnderlyingHomalgMatrix( embR ) );
1,
0,
0
(over a graded ring)
gap> embR2 := EmbeddingOfTruncationOfProjectiveGradedModule( NewObjectL,
>         SemigroupForPresentationsByProjectiveGradedModules( [[1,0],[0,2]] ) );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( embR2 ) );
1,
0,
0
(over a graded ring)
gap> embR3 := EmbeddingOfTruncationOfProjectiveGradedModuleWithGivenTruncationObject(
>          NewObjectR, tR );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( embR3 ) );
1,
0,
0
(over a graded ring)
gap> projR := ProjectionOntoTruncationOfProjectiveGradedModule( NewObjectR,
>          SemigroupForPresentationsByProjectiveGradedModules( [[ 1,0 ], [ 0,1 ]] ) );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( projR ) );
1, 0, 0
(over a graded ring)
gap> projR2 := ProjectionOntoTruncationOfProjectiveGradedModule( NewObjectR,
>          SemigroupForPresentationsByProjectiveGradedModules( [[ 1,0 ], [ 0,2 ]] ) );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( projR2 ) );
1, 0, 0
(over a graded ring)
gap> projR3 := ProjectionOntoTruncationOfProjectiveGradedModuleWithGivenTruncationObject(
>           NewObjectR, tR );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( projR3 ) );
1, 0, 0
(over a graded ring)
gap> truncatorR := TruncationFunctorForProjectiveGradedRightModules( S,
>                               SemigroupForPresentationsByProjectiveGradedModules( [[ 1,0 ], [ 0,
Truncation functor for 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 ] ])
to the semigroup generated by [ [ 1, 0 ], [ 0, 2 ] ]
gap> truncatorR2 := TruncationFunctorForProjectiveGradedRightModules(
>                          S, SemigroupForPresentationsByProjectiveGradedModules( [[ 1,0 ], [ 0,1
```

```
Truncation functor for 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 ] ])
to the semigroup generated by [ [ 1, 0 ], [ 0, 1 ] ]
gap> tR2 := ApplyFunctor( truncatorR, NewObjectR );
<A projective graded right module of rank 1>
gap> Display( tR2 );
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: [ [ ( 1, 0 ), 1 ] ]
gap> sourceR := CAPCategoryOfProjectiveGradedRightModulesObject(
>             [ [[1,0],1], [[0,1],1] ], S );
<A projective graded right module of rank 2>
gap> rangeR := CAPCategoryOfProjectiveGradedRightModulesObject(
>             [ [[1,0],1] ], S );
<A projective graded right module of rank 1>
gap> test_morphismR := CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism(
>       sourceR, HomalgMatrix( [ [ 1, 0 ] ], S ) ,rangeR );
<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 ] ])>
gap> tr_test_morphismR := ApplyFunctor( truncatorR, test_morphismR );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( tr_test_morphismR ) );
1
(over a graded ring)
gap> tr2_test_morphismR := ApplyFunctor( truncatorR2, test_morphismR );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( tr2_test_morphismR ) );
1, 0
(over a graded ring)
gap> nat_trans_r := NaturalTransformationFromTruncationToIdentityForProjectiveGradedRightModules
>                              ( S, SemigroupForPresentationsByProjectiveGradedModules( [[1,0],
Natural transformation from Truncation functor for 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 ] ])
to the semigroup generated by [ [ 1,0 ], [ 0, 1 ] ] to id
gap> component_r := ApplyNaturalTransformation( nat_trans_r, NewObjectR );
<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 ] ])>
gap> Display( UnderlyingHomalgMatrix( component_r ) );
1, 0, 0
(over a graded ring)
```

5.4 Truncations in SfpgrmodLeft

Example

```
gap> Q1 := CAPCategoryOfProjectiveGradedLeftModulesObject( [ [[2,0],1] ], S );
<A projective graded left module of rank 1>
gap> Q2 := CAPCategoryOfProjectiveGradedLeftModulesObject( [ [[1,0],1], [[-1,0],1] ], S );
<A projective graded left module of rank 2>
```

```
gap> Q3 := CAPCategoryOfProjectiveGradedLeftModulesObject( [ [[1,0],1] ], S );
<A projective graded left module of rank 1>
gap> Q4 := CAPCategoryOfProjectiveGradedLeftModulesObject( [ [[1,0],1] ], S );
<A projective graded left module of rank 1>
gap> m1l := CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism(
>       Q1, HomalgMatrix( [["x_1","x_2^3"]], S ) ,Q2 );
<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 ] ])>
gap> m2l := CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism(
>       Q2, HomalgMatrix( [[1],[0]], S ) ,Q3 );
<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 ] ])>
gap> m3l := CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism(
>       Q4, HomalgMatrix( [[1]], S ) ,Q3 );
<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 ] ])>
gap> left_category := CapCategory( Q1 );
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> left_presentation1 := CAPPresentationCategoryObject( m1l );
<A graded left module presentation over the ring Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> left_presentation2 := CAPPresentationCategoryObject( m2l );
<A graded left module presentation over the ring Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> left_presentation3 := CAPPresentationCategoryObject( m3l );
<A graded left module presentation over the ring Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> truncation_functor_left := TruncationFunctorLeft(
>                             S, SemigroupForPresentationsByProjectiveGradedModules( [[1,0],[0,1]]
Truncation 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 semigroup generated by [ [ 1, 0 ], [ 0, 1 ] ]
gap> truncation1l := ApplyFunctor( truncation_functor_left, left_presentation1 );
<A graded left module presentation over the ring Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> FullInformation( truncation1l );
================================================================================

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:
[ [ ( 2, 0 ), 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_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 1 and degrees:
[ [ ( 1, 0 ), 1 ] ]
```

```
================================================================================
gap> truncation2l := ApplyFunctor( truncation_functor_left, left_presentation2 );
<A graded left module presentation over the ring Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> FullInformation( truncation2l );
================================================================================

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:
[ [ ( 1, 0 ), 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:
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 1 and degrees:
[ [ ( 1, 0 ), 1 ] ]

================================================================================
gap> morl := CAPPresentationCategoryMorphism( left_presentation1, m2l, left_presentation3 );
<A morphism 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> trmorl := ApplyFunctor( truncation_functor_left, morl );
<A morphism 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> FullInformation( trmorl );

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

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 1 and degrees:
[ [ ( 2, 0 ), 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_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 1 and degrees:
[ [ ( 1, 0 ), 1 ] ]

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

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:
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 1 and degrees:
[ [ ( 1, 0 ), 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:
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 1 and degrees:
[ [ ( 1, 0 ), 1 ] ]

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

5.5 Truncations for graded module presentations (for CAP)

Example

```
gap> P1 := CAPCategoryOfProjectiveGradedRightModulesObject( [ [[2,0],1] ], S );
<A projective graded right module of rank 1>
gap> P2 := CAPCategoryOfProjectiveGradedRightModulesObject( [ [[1,0],1], [[-1,0],1] ], S );
<A projective graded right module of rank 2>
gap> P3 := CAPCategoryOfProjectiveGradedRightModulesObject( [ [[1,0],1] ], S );
<A projective graded right module of rank 1>
gap> P4 := CAPCategoryOfProjectiveGradedRightModulesObject( [ [[1,0],1] ], S );
<A projective graded right module of rank 1>
gap> m1r := CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism(
>       P1, HomalgMatrix( [["x_1"],["x_2^3"]], S ) ,P2 );
<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 ] ])>
gap> m2r := CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism(
>       P2, HomalgMatrix( [[1,0]], S ) ,P3 );
<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 ] ])>
gap> m3r := CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism(
>       P4, HomalgMatrix( [[1]], S ) ,P3 );
<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 ] ])>
gap> right_category := CapCategory( P1 );
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> right_presentation1 := CAPPresentationCategoryObject( m1r );
<A graded right module presentation over the ring Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> right_presentation2 := CAPPresentationCategoryObject( m2r );
<A graded right module presentation over the ring Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> right_presentation3 := CAPPresentationCategoryObject( m3r );
<A graded right module presentation over the ring Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> truncation_functor_right := TruncationFunctorRight(
>                               S, SemigroupForPresentationsByProjectiveGradedModules( [[1,0],[0,1]
Truncation 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 semigroup generated by [ [ 1, 0 ], [ 0, 1 ] ]
gap> truncation1r := ApplyFunctor( truncation_functor_right, right_presentation1 );
<A graded right module presentation over the ring Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> FullInformation( truncation1r );
================================================================================

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:
[ [ ( 2, 0 ), 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_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 1 and degrees:
[ [ ( 1, 0 ), 1 ] ]

================================================================================
gap> truncation2r := ApplyFunctor( truncation_functor_right, right_presentation2 );
<A graded right module presentation over the ring Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> FullInformation( truncation2r );
================================================================================

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:
[ [ ( 1, 0 ), 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:
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 1 and degrees:
[ [ ( 1, 0 ), 1 ] ]

================================================================================
gap> morr := CAPPresentationCategoryMorphism( right_presentation1, m2r, right_presentation3 );
<A morphism 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> trmorr := ApplyFunctor( truncation_functor_right, morr );
<A morphism 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> FullInformation( trmorr );

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

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 1 and degrees:
[ [ ( 2, 0 ), 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_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 1 and degrees:
[ [ ( 1, 0 ), 1 ] ]

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

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:
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 1 and degrees:
[ [ ( 1, 0 ), 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:
```

```
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 1 and degrees:
[ [ ( 1, 0 ), 1 ] ]

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

Index