TruncationsOfFP-GradedModules

A package to compute truncations of FPGradedModules

2021.11.17

17 November 2021

Martin Bies

Martin Bies
Email: martin.bies@alumni.uni-heidelberg.de
Homepage: https://martinbies.github.io/
Address: Department of Mathematics
University of Pennsylvania
David Rittenhouse Laboratory
209 S 33rd St
Philadelphia
PA 19104

Copyright

Contents

Chapter 1

Introduction

1.1 What is the goal of the TruncationsOfFPGradedModules package?

TruncationsOfFPGradedModules provides methods to compute truncations of FPGradedModules.

Chapter 2

DegreeXLayerVectorSpaceMorphisms

2.1 GAP category of DegreeXLayerVectorSpaces

2.1.1 IsDegreeXLayerVectorSpace (for IsObject)

▷ `IsDegreeXLayerVectorSpace(`*object*`)` (filter)

Returns: true or false

The GAP category for vector spaces that represent a degree layer of a f.p. graded module

2.1.2 IsDegreeXLayerVectorSpaceMorphism (for IsObject)

▷ `IsDegreeXLayerVectorSpaceMorphism(`*object*`)` (filter)

Returns: true or false

The GAP category for morphisms between vector spaces that represent a degree layer of a f.p. graded module

2.1.3 IsDegreeXLayerVectorSpacePresentation (for IsObject)

▷ `IsDegreeXLayerVectorSpacePresentation(`*object*`)` (filter)

Returns: true or false

The GAP category for (left) presentations of vector spaces that represent a degree layer of a f.p. graded module

2.1.4 IsDegreeXLayerVectorSpacePresentationMorphism (for IsObject)

▷ `IsDegreeXLayerVectorSpacePresentationMorphism(`*object*`)` (filter)

Returns: true or false

The GAP category for (left) presentation morphisms of vector spaces that represent a degree layer of a f.p. graded module

2.2 Constructors for DegreeXLayerVectorSpaces

2.2.1 DegreeXLayerVectorSpace (for IsList, IsHomalgGradedRing, IsVectorSpaceObject, IsInt)

▷ `DegreeXLayerVectorSpace(L, S, V, n)` (operation)

Returns: a CAPCategoryObject

The arguments are a list of monomials L, a homalg graded ring S (the Coxring of the variety in question), a vector space V and a non-negative integer n. V is to be given as a vector space defined in the package 'LinearAlgebraForCAP'. This method then returns the corresponding DegreeXLayerVectorSpace.

2.2.2 DegreeXLayerVectorSpaceMorphism (for IsDegreeXLayerVectorSpace, IsVectorSpaceMorphism, IsDegreeXLayerVectorSpace)

▷ `DegreeXLayerVectorSpaceMorphism(L, S, V)` (operation)

Returns: a DegreeXLayerVectorSpaceMorphism

The arguments are a DegreeXLayerVectorSpace *source*, a vector space morphism φ (as defined in 'LinearAlgebraForCAP') and a DegreeXLayerVectorSpace *range*. If φ is a vector space morphism between the underlying vector spaces of *source* and *range* this method returns the corresponding DegreeXLayerVectorSpaceMorphism.

2.2.3 DegreeXLayerVectorSpacePresentation (for IsDegreeXLayerVectorSpaceMorphism)

▷ `DegreeXLayerVectorSpacePresentation(a)` (operation)

Returns: a DegreeXLayerVectorSpaceMorphism

The arguments is a DegreeXLayerVectorSpaceMorphism `a`. This method treats this morphism as a presentation, i.e. we are interested in the cokernel of the underlying morphism of vector spaces. The corresponding DegreeXLayerVectorSpacePresentation is returned.

2.2.4 DegreeXLayerVectorSpacePresentationMorphism (for IsDegreeXLayerVectorSpacePresentation, IsVectorSpaceMorphism, IsDegreeXLayerVectorSpacePresentation)

▷ `DegreeXLayerVectorSpacePresentationMorphism(source, \varphi, range)` (operation)

Returns: a DegreeXLayerVectorSpacePresentationMorphism

The arguments is a DegreeXLayerVectorSpacePresentation *source*, a vector space morphism φ and a DegreeXLayerVectorSpacePresentation *range*. The corresponding DegreeXLayerVectorSpacePresentationMorphism is returned.

2.3 Attributes for DegreeXLayerVectorSpaces

2.3.1 UnderlyingHomalgGradedRing (for IsDegreeXLayerVectorSpace)

▷ `UnderlyingHomalgGradedRing(V)` (attribute)

Returns: a homalg graded ring

The argument is a DegreeXLayerVectorSpace V. The output is the Coxring, in which this vector space is embedded via the generators (specified when installing V).

2.3.2 Generators (for IsDegreeXLayerVectorSpace)

▷ `Generators(V)` (attribute)

Returns: a list

The argument is a DegreeXLayerVectorSpace V. The output is the list of generators, that embed V into the Coxring in question.

2.3.3 UnderlyingVectorSpaceObject (for IsDegreeXLayerVectorSpace)

▷ `UnderlyingVectorSpaceObject(V)` (attribute)

Returns: a VectorSpaceObject

The argument is a DegreeXLayerVectorSpace V. The output is the underlying vectorspace object (as defined in 'LinearAlgebraForCAP').

2.3.4 EmbeddingDimension (for IsDegreeXLayerVectorSpace)

▷ `EmbeddingDimension(V)` (attribute)

Returns: a VectorSpaceObject

The argument is a DegreeXLayerVectorSpace V. For S its 'UnderlyingHomalgGradedRing' this vector space is embedded (via its generators) into S^n. The integer n is the embedding dimension.

2.4 Attributes for DegreeXLayerVectorSpaceMorphisms

2.4.1 Source (for IsDegreeXLayerVectorSpaceMorphism)

▷ `Source(a)` (attribute)

Returns: a DegreeXLayerVectorSpace

The argument is a DegreeXLayerVectorSpaceMorphism a. The output is its source.

2.4.2 Range (for IsDegreeXLayerVectorSpaceMorphism)

▷ `Range(a)` (attribute)

Returns: a DegreeXLayerVectorSpace

The argument is a DegreeXLayerVectorSpaceMorphism a. The output is its range.

2.4.3 UnderlyingVectorSpaceMorphism (for IsDegreeXLayerVectorSpaceMorphism)

▷ `UnderlyingVectorSpaceMorphism(a)` (attribute)

Returns: a DegreeXLayerVectorSpace

The argument is a DegreeXLayerVectorSpaceMorphism a. The output is its range.

2.4.4 UnderlyingHomalgGradedRing (for IsDegreeXLayerVectorSpaceMorphism)

▷ `UnderlyingHomalgGradedRing(a)` (attribute)

Returns: a homalg graded ring

The argument is a DegreeXLayerVectorSpaceMorphism a. The output is the Coxring, in which the source and range of this is morphism are embedded.

2.5 Attributes for DegreeXLayerVectorSpacePresentations

2.5.1 UnderlyingDegreeXLayerVectorSpaceMorphism (for IsDegreeXLayerVectorSpacePresentation)

▷ `UnderlyingDegreeXLayerVectorSpaceMorphism(a)` (attribute)

Returns: a DegreeXLayerVectorSpaceMorphism

The argument is a DegreeXLayerVectorSpacePresentation a. The output is the underlying DegreeXLayerVectorSpaceMorphism

2.5.2 UnderlyingVectorSpaceObject (for IsDegreeXLayerVectorSpacePresentation)

▷ `UnderlyingVectorSpaceObject(a)` (attribute)

Returns: a VectorSpaceObject

The argument is a DegreeXLayerVectorSpacePresentation a. The output is the vector space object which is the cokernel of the underlying vector space morphism.

2.5.3 UnderlyingVectorSpaceMorphism (for IsDegreeXLayerVectorSpacePresentation)

▷ `UnderlyingVectorSpaceMorphism(a)` (attribute)

Returns: a VectorSpaceMorphism

The argument is a DegreeXLayerVectorSpacePresentation a. The output is the vector space morphism which defines the underlying morphism of DegreeXLayerVectorSpaces.

2.5.4 UnderlyingHomalgGradedRing (for IsDegreeXLayerVectorSpacePresentation)

▷ `UnderlyingHomalgGradedRing(a)` (attribute)

Returns: a homalg graded ring

The argument is a DegreeXLayerVectorSpacePresentation a. The output is the Coxring, in which the source and range of the underlying morphism of DegreeXLayerVectorSpaces are embedded.

2.5.5 UnderlyingVectorSpacePresentation (for IsDegreeXLayerVectorSpacePresentation)

▷ `UnderlyingVectorSpacePresentation(a)` (attribute)

Returns: a CAP presentation category object

The argument is a DegreeXLayerVectorSpacePresentation a. The output is the underlying vector space presentation.

2.6 Attributes for DegreeXLayerVectorSpacePresentationMorphisms

2.6.1 Source (for IsDegreeXLayerVectorSpacePresentationMorphism)

▷ `Source(a)` (attribute)

Returns: a DegreeXLayerVectorSpacePresentation

The argument is a DegreeXLayerVectorSpacePresentationMorphism *a*. The output is its source.

2.6.2 Range (for IsDegreeXLayerVectorSpacePresentationMorphism)

▷ `Range(a)` (attribute)

Returns: a DegreeXLayerVectorSpacePresentation

The argument is a DegreeXLayerVectorSpacePresentationMorphism *a*. The output is its range.

2.6.3 UnderlyingHomalgGradedRing (for IsDegreeXLayerVectorSpacePresentationMorphism)

▷ `UnderlyingHomalgGradedRing(a)` (attribute)

Returns: a homalg graded ring

The argument is a DegreeXLayerVectorSpacePresentationMorphism *a*. The output is the underlying graded ring of its source.

2.6.4 UnderlyingVectorSpacePresentationMorphism (for IsDegreeXLayerVectorSpacePresentationMorphism)

▷ `UnderlyingVectorSpacePresentationMorphism(a)` (attribute)

Returns: a CAP presentation category morphism

The argument is a DegreeXLayerVectorSpacePresentationMorphism *a*. The output is the underlying vector space presentation morphism.

2.7 Convenience methods

2.7.1 FullInformation (for IsDegreeXLayerVectorSpacePresentation)

▷ `FullInformation(p)` (operation)

Returns: detailed information about p

The argument is a DegreeXLayerVectorSpacePresentation *p*. This method displays *p* in great detail.

2.7.2 FullInformation (for IsDegreeXLayerVectorSpacePresentationMorphism)

▷ `FullInformation(p)` (operation)

Returns: detailed information about p

The argument is a DegreeXLayerVectorSpacePresentationMorphism *p*. This method displays *p* in great detail.

2.8 Examples

2.8.1 DegreeXLayerVectorSpaces

Example

```
gap> HOMALG_IO.show_banners := false;;
gap> HOMALG_IO.suppress_PID := true;;
gap> mQ := HomalgFieldOfRationals();
Q
gap> P1 := ProjectiveSpace( 1 );
<A projective toric variety of dimension 1>
gap> cox_ring := CoxRing( P1 );
Q[x_1,x_2]
(weights: [ 1, 1 ])
gap> mons := MonomsOfCoxRingOfDegreeByNormalizAsColumnMatrices
>          ( P1, [1], 1, 1 );;
gap> vector_space := VectorSpaceObject( Length( mons ), mQ );
<A vector space object over Q of dimension 2>
gap> DXVS := DegreeXLayerVectorSpace( mons, cox_ring, vector_space, 1 );
<A vector space embedded into (Q[x_1,x_2] (with weights [ 1, 1 ]))^1>
gap> EmbeddingDimension( DXVS );
1
gap> Generators( DXVS );
[ <A 1 x 1 matrix over a graded ring>, <A 1 x 1 matrix over a graded ring> ]
```

2.8.2 Morphisms of DegreeXLayerVectorSpaces

Example

```
gap> mons2 := Concatenation(
>           MonomsOfCoxRingOfDegreeByNormalizAsColumnMatrices
>           ( P1, [1], 1, 2 ),
>           MonomsOfCoxRingOfDegreeByNormalizAsColumnMatrices
>           ( P1, [1], 2, 2 ) );;
gap> vector_space2 := VectorSpaceObject( Length( mons2 ), mQ );
<A vector space object over Q of dimension 4>
gap> DXVS2 := DegreeXLayerVectorSpace( mons2, cox_ring, vector_space2, 2 );
<A vector space embedded into (Q[x_1,x_2] (with weights [ 1, 1 ]))^2>
gap> matrix := HomalgMatrix( [ [ 1, 0, 0, 0 ],
>                              [ 0, 1, 0, 0 ] ], mQ );
<A matrix over an internal ring>
gap> vector_space_morphism := VectorSpaceMorphism( vector_space,
>                                                  matrix,
>                                                  vector_space2 );;
gap> IsWellDefined( vector_space_morphism );
true
gap> morDXVS := DegreeXLayerVectorSpaceMorphism(
>             DXVS, vector_space_morphism, DXVS2 );
<A morphism of two vector spaces embedded into
(suitable powers of) Q[x_1,x_2] (with weights [ 1, 1 ])>
gap> UnderlyingVectorSpaceMorphism( morDXVS );
<A morphism in Category of matrices over Q>
gap> UnderlyingHomalgGradedRing( morDXVS );
Q[x_1,x_2]
```

```
(weights: [ 1, 1 ])
```

2.8.3 DegreeXLayerVectorSpacePresentations

Example

```
gap> DXVSPresentation := DegreeXLayerVectorSpacePresentation( morDXVS );
<A vector space embedded into (a suitable power of)
Q[x_1,x_2] (with weights [ 1, 1 ]) given as the
cokernel of a vector space morphism>
gap> UnderlyingVectorSpaceObject( DXVSPresentation );
<A vector space object over Q of dimension 2>
gap> relation := RelationMorphism(
>              UnderlyingVectorSpacePresentation( DXVSPresentation ) );
<A morphism in Category of matrices over Q>
gap> m := UnderlyingMatrix( relation );
<A 2 x 4 matrix over an internal ring>
gap> m = matrix;
true
```

2.8.4 Morphisms of DegreeXLayerVectorSpacePresentations

Example

```
gap> zero_space := ZeroObject( CapCategory( vector_space ) );;
gap> source := DegreeXLayerVectorSpace( [], cox_ring, zero_space, 1 );;
gap> vector_space_morphism := ZeroMorphism( zero_space, vector_space );;
gap> morDXVS2 := DegreeXLayerVectorSpaceMorphism(
>              source, vector_space_morphism, DXVS );;
gap> DXVSPresentation2 := DegreeXLayerVectorSpacePresentation( morDXVS2 );
<A vector space embedded into (a suitable power of)
Q[x_1,x_2] (with weights [ 1, 1 ]) given as the
cokernel of a vector space morphism>
gap> matrix := HomalgMatrix( [ [ 0, 0, 1, 0 ],
>                            [ 0, 0, 0, 1 ] ], mQ );
<A matrix over an internal ring>
gap> source := Range( UnderlyingVectorSpaceMorphism( DXVSPresentation2 ) );;
gap> range := Range( UnderlyingVectorSpaceMorphism( DXVSPresentation ) );;
gap> vector_space_morphism := VectorSpaceMorphism( source, matrix, range );;
gap> IsWellDefined( vector_space_morphism );
true
gap> DXVSPresentationMorphism := DegreeXLayerVectorSpacePresentationMorphism(
>                                        DXVSPresentation2,
>                                        vector_space_morphism,
>                                        DXVSPresentation );
<A vector space presentation morphism of vector spaces embedded into
(a suitable power of) Q[x_1,x_2] (with weights [ 1, 1 ]) and given as
cokernels>
gap> uVSMor := UnderlyingVectorSpacePresentationMorphism
>                                        ( DXVSPresentationMorphism );
<A morphism in Freyd( Category of matrices over Q )>
gap> IsWellDefined( uVSMor );
true
```

Chapter 3

Truncations of graded rows and columns

3.1 Truncations of graded rows and columns

3.1.1 TruncateGradedRowOrColumn (for IsToricVariety, IsGradedRowOrColumn, IsList, IsFieldForHomalg)

▷ `TruncateGradedRowOrColumn(V, M, degree_list, field)` (operation)

Returns: Vector space

The arguments are a toric variety V, a graded row or column M over the Cox ring of V and a `degree_list` specifying an element of the degree group of the toric variety V. The latter can either be specified by a list of integers or as a HomalgModuleElement. Based on this input, the method computes the truncation of M to the specified degree. We return this finite dimensional vector space. Optionally, we allow for a field F as fourth input. This field is then used to construct the vector space. Otherwise, we use the coefficient field of the Cox ring of V.

3.1.2 TruncateGradedRowOrColumn (for IsToricVariety, IsGradedRowOrColumn, IsHomalgModuleElement, IsFieldForHomalg)

▷ `TruncateGradedRowOrColumn(V, M, m, field)` (operation)

Returns: Vector space

As above, but with a HomalgModuleElement m specifying the degree.

3.1.3 TruncateGradedRowOrColumn (for IsToricVariety, IsGradedRowOrColumn, IsList)

▷ `TruncateGradedRowOrColumn(V, M, degree)` (operation)

Returns: Vector space

As above, but the coefficient ring of the Cox ring will be used as field

3.1.4 TruncateGradedRowOrColumn (for IsToricVariety, IsGradedRowOrColumn, IsHomalgModuleElement)

▷ `TruncateGradedRowOrColumn(V, M, m)` (operation)

Returns: Vector space

As above, but a HomalgModuleElement m specifies the degree and we use the coefficient ring of the Cox ring as field.

3.1.5 DegreeXLayerOfGradedRowOrColumn (for IsToricVariety, IsGradedRowOrColumn, IsList, IsFieldForHomalg)

▷ `DegreeXLayerOfGradedRowOrColumn(V, M, degree_list, field)` (operation)

Returns: DegreeXLayerVectorSpace

The arguments are a toric variety V, a graded row or column M over the Cox ring of V and a *degree_list* specifying an element of the degree group of the toric variety V. The latter can either be specified by a list of integers or as a HomalgModuleElement. Based on this input, the method computes the truncation of M to the specified degree. This is a finite dimensional vector space. We return the corresponding DegreeXLayerVectorSpace. Optionally, we allow for a field F as fourth input. This field is used to construct the DegreeXLayerVectorSpace. Namely, the wrapper DegreeXLayerVectorSpace contains a representation of the obtained vector space as F^n. In case F is specified, we use this particular field. Otherwise, HomalgFieldOfRationals() will be used.

3.1.6 DegreeXLayerOfGradedRowOrColumn (for IsToricVariety, IsGradedRowOrColumn, IsHomalgModuleElement, IsFieldForHomalg)

▷ `DegreeXLayerOfGradedRowOrColumn(V, M, m, field)` (operation)

Returns: DegreeXLayerVectorSpace

As above, but with a HomalgModuleElement m specifying the degree.

3.1.7 DegreeXLayerOfGradedRowOrColumn (for IsToricVariety, IsGradedRowOrColumn, IsList)

▷ `DegreeXLayerOfGradedRowOrColumn(V, M, degree)` (operation)

Returns: DegreeXLayerVectorSpace

As above, but the coefficient ring of the Cox ring will be used as field

3.1.8 DegreeXLayerOfGradedRowOrColumn (for IsToricVariety, IsGradedRowOrColumn, IsHomalgModuleElement)

▷ `DegreeXLayerOfGradedRowOrColumn(V, M, m)` (operation)

Returns: DegreeXLayerVectorSpace

As above, but a HomalgModuleElement m specifies the degree and we use the coefficient ring of the Cox ring as field.

3.2 Formats for generators of truncations of graded rows and columns

3.2.1 GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsListOfColumnMatrices (for IsToricVariety, IsGradedRowOrColumn, IsList)

▷ `GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsListOfColumnMatrices(V, M, l)` (operation)

Returns: a list

The arguments are a variety V, a graded row or column M and a list l, specifying a degree in the class group of the Cox ring of V. We then compute the truncation of M to the specified degree and return its generators as list of column matrices.

3.2.2 GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsListOfColumnMatrices (for IsToricVariety, IsGradedRowOrColumn, IsHomalgModuleElement)

▷ `GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsListOfColumnMatrices(V, M, m)` (operation)

Returns: a list

The arguments are a variety V, a graded row or column M and a HomalgModuleElement m, specifying a degree in the class group of the Cox ring of V. We then compute the truncation of M to the specified degree and return its generators as list of column matrices.

3.2.3 GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsUnionOfColumnMatrices (for IsToricVariety, IsGradedRowOrColumn, IsList)

▷ `GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsUnionOfColumnMatrices(V, M, m)` (operation)

Returns: a list

The arguments are a variety V, a graded row or column M and a list l, specifying a degree in the class group of the Cox ring of V. We then compute the truncation of M to the specified degree and its generators as column matrices. The matrix formed from the union of these column matrices is returned.

3.2.4 GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsUnionOfColumnMatrices (for IsToricVariety, IsGradedRowOrColumn, IsHomalgModuleElement)

▷ `GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsUnionOfColumnMatrices(V, M, m)` (operation)

Returns: a list

The arguments are a variety V, a graded row or column M and a HomalgModuleElement m, specifying a degree in the class group of the Cox ring of V. We then compute the truncation of M to the specified degree and its generators as column matrices. The matrix formed from the union of these column matrices is returned.

3.2.5 GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsListsOfRecords (for IsToricVariety, IsGradedRowOrColumn, IsList)

▷ `GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsListsOfRecords(V, M, l)` (operation)

Returns: a list

The arguments are a variety V, a graded row or column M and a list l, specifying a degree in the class group of the Cox ring of V. We then compute the truncation of M to the specified degree and return its generators as list [n, rec_list]. n specifies the number of generators. rec_list is a list of record. The i-th record contains the generators of the i-th direct summand of M.

The arguments are a variety V, a graded row or column M and a HomalgModuleElement m, specifying a degree in the class group of the Cox ring of V. We then compute the truncation of M to the specified degree and return its generators as list [n, rec_list]. n specifies the number of generators. rec_list is a list of record. The i-th record contains the generators of the i-th direct summand of M.

3.2.6 GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsListsOfRecords (for IsToricVariety, IsGradedRowOrColumn, IsHomalgModuleElement)

▷ `GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsListsOfRecords(V, M, m)` (operation)

Returns: a list

3.2.7 GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsListList (for IsToricVariety, IsGradedRowOrColumn, IsList)

▷ `GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsListList(V, M, l)` (operation)

Returns: a list

The arguments are a variety V, a graded row or column M and a list l, specifying a degree in the class group of the Cox ring of V. We then compute the truncation of M to the specified degree and identify its generators. We format each generator as list [n, g], where g denotes a generator of the n-th direct summand of M. We return the list of all these lists [n, g].

3.2.8 GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsListList (for IsToricVariety, IsGradedRowOrColumn, IsHomalgModuleElement)

▷ `GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsListList(V, M, m)` (operation)

Returns: a list

The arguments are a variety V, a graded row or column M and a HomalgModuleElement m, specifying a degree in the class group of the Cox ring of V. We then compute the truncation of M to the specified degree and identify its generators. We format each generator as list [n, g], where g denotes a generator of the n-th direct summand of M. We return the list of all these lists [n, g].

3.3 Truncations of graded row and column morphisms

3.3.1 TruncateGradedRowOrColumnMorphism (for IsToricVariety, IsGradedRowOrColumnMorphism, IsList, IsBool, IsFieldForHomalg)

▷ `TruncateGradedRowOrColumnMorphism(V, a, d, B, F)` (operation)

Returns: a vector space morphism

The arguments are a toric variety V, a morphism a of graded rows or columns, a list d specifying a degree in the class group of V, a field F for homalg and a boolean B. We then truncate m to the specified degree d. We express this result as morphism of vector spaces over the field F. We return this vector space morphism. If the boolean B is true, we display additional output during the computation, otherwise this output is surpressed.

3.3.2 TruncateGradedRowOrColumnMorphism (for IsToricVariety, IsGradedRowOrColumnMorphism, IsHomalgModuleElement, IsBool, IsHomalgRing)

▷ `TruncateGradedRowOrColumnMorphism(V, a, m, B, F)` (operation)

Returns: a vector space morphism

The arguments are a toric variety V, a morphism a of graded rows or columns, and a HomalgModuleElement m specifying a degree in the class group of V, a field F for homalg and a boolean B. We then truncate m to the specified degree d. We express this result as morphism of vector spaces over the field F. We return this vector space morphism. If the boolean B is true, we display additional output during the computation, otherwise this output is surpressed.

3.3.3 TruncateGradedRowOrColumnMorphism (for IsToricVariety, IsGradedRowOrColumnMorphism, IsList, IsBool)

▷ `TruncateGradedRowOrColumnMorphism(V, a, d, B)` (operation)

Returns: a vector space morphism

This method operates just as 'TruncateGradedRowOrColumnMorphism' above. However, here the field F is taken as the field of coefficients of the Cox ring of the variety V.

3.3.4 TruncateGradedRowOrColumnMorphism (for IsToricVariety, IsGradedRowOrColumnMorphism, IsHomalgModuleElement, IsBool)

▷ `TruncateGradedRowOrColumnMorphism(V, a, m, B)` (operation)

Returns: a vector space morphism

This method operates just as 'TruncateGradedRowOrColumnMorphism' above. However, here the field F is taken as the field of coefficients of the Cox ring of the variety V.

3.3.5 TruncateGradedRowOrColumnMorphism (for IsToricVariety, IsGradedRowOrColumnMorphism, IsList)

▷ `TruncateGradedRowOrColumnMorphism(V, a, d)` (operation)

Returns: a vector space morphism

This method operates just as 'TruncateGradedRowOrColumnMorphism' above. However, here the field F is taken as the field of coefficients of the Cox ring of the variety V. Also, B is set to false, i.e. no additional information is being displayed.

3.3.6 TruncateGradedRowOrColumnMorphism (for IsToricVariety, IsGradedRowOrColumnMorphism, IsHomalgModuleElement)

▷ `TruncateGradedRowOrColumnMorphism(V, a, m)` (operation)

Returns: a vector space morphism

This method operates just as 'TruncateGradedRowOrColumnMorphism' above. However, here the field F is taken as the field of coefficients of the Cox ring of the variety V. Also, B is set to false, i.e. no additional information is being displayed.

3.3.7 DegreeXLayerOfGradedRowOrColumnMorphism (for IsToricVariety, IsGradedRowOrColumnMorphism, IsList, IsFieldForHomalg, IsBool)

▷ `DegreeXLayerOfGradedRowOrColumnMorphism(V, a, d, F, B)` (operation)

Returns: a DegreeXLayerVectorSpaceMorphism

The arguments are a toric variety V, a morphism a of graded rows or columns, a list d specifying a degree in the class group of V, a field F for homalg and a boolean B. We then truncate m to the specified degree d. We express this result as morphism of vector spaces over the field F. We return the corresponding DegreeXLayerVectorSpaceMorphism. If the boolean B is true, we display additional output during the computation, otherwise this output is surpressed.

3.3.8 DegreeXLayerOfGradedRowOrColumnMorphism (for IsToricVariety, IsGradedRowOrColumnMorphism, IsHomalgModuleElement, IsHomalgRing, IsBool)

▷ `DegreeXLayerOfGradedRowOrColumnMorphism(V, a, m, F, B)` (operation)

Returns: a DegreeXLayerVectorSpaceMorphism

The arguments are a toric variety V, a morphism a of graded rows or columns, a HomalgModuleElement m specifying a degree in the class group of V, a field F for homalg and a boolean B. We then truncate m to the specified degree d. We express this result as morphism of vector spaces over the field F. We return the corresponding DegreeXLayerVectorSpaceMorphism. If the boolean B is true, we display additional output during the computation, otherwise this output is surpressed.

3.3.9 DegreeXLayerOfGradedRowOrColumnMorphism (for IsToricVariety, IsGradedRowOrColumnMorphism, IsList, IsBool)

▷ `DegreeXLayerOfGradedRowOrColumnMorphism(V, a, d, B)` (operation)

Returns: a vector space morphism

This method operates just as 'DegreeXLayerOfGradedRowOrColumnMorphism' above. However, here the field F is taken as the field of coefficients of the Cox ring of the variety V.

3.3.10 DegreeXLayerOfGradedRowOrColumnMorphism (for IsToricVariety, IsGradedRowOrColumnMorphism, IsHomalgModuleElement, IsBool)

▷ `DegreeXLayerOfGradedRowOrColumnMorphism(V, a, m, B)` (operation)

Returns: a vector space morphism

This method operates just as 'DegreeXLayerOfGradedRowOrColumnMorphism' above. However, here the field F is taken as the field of coefficients of the Cox ring of the variety V.

3.3.11 DegreeXLayerOfGradedRowOrColumnMorphism (for IsToricVariety, IsGradedRowOrColumnMorphism, IsList)

▷ `DegreeXLayerOfGradedRowOrColumnMorphism(V, a, d)` (operation)

Returns: a vector space morphism

This method operates just as 'DegreeXLayerOfGradedRowOrColumnMorphism' above. However, here the field F is taken as the field of coefficients of the Cox ring of the variety V. Also, B is set to false, i.e. no additional information is being displayed.

3.3.12 DegreeXLayerOfGradedRowOrColumnMorphism (for IsToricVariety, IsGradedRowOrColumnMorphism, IsHomalgModuleElement)

▷ `DegreeXLayerOfGradedRowOrColumnMorphism(V, a, m)` (operation)

Returns: a vector space morphism

This method operates just as 'DegreeXLayerOfGradedRowOrColumnMorphism' above. However, here the field F is taken as the field of coefficients of the Cox ring of the variety V. Also, B is set to false, i.e. no additional information is being displayed.

3.4 Truncations of morphisms of graded rows and columns in parallel

3.4.1 TruncateGradedRowOrColumnMorphismInParallel (for IsToricVariety, IsGradedRowOrColumnMorphism, IsList, IsPosInt, IsBool, IsFieldForHomalg)

▷ `TruncateGradedRowOrColumnMorphismInParallel(V, a, d, N, B, F)` (operation)

Returns: a vector space morphism

This method operates just as 'TruncateGradedRowOrColumnMorphism' above. However, as fourth argument an integer *N* is to be specified. The computation of the truncation will then be performed in parallel in *N* child processes.

3.4.2 TruncateGradedRowOrColumnMorphismInParallel (for IsToricVariety, IsGradedRowOrColumnMorphism, IsHomalgModuleElement, IsPosInt, IsBool, IsFieldForHomalg)

▷ `TruncateGradedRowOrColumnMorphismInParallel(V, a, m, N, B, F)` (operation)

Returns: a vector space morphism

This method operates just as 'TruncateGradedRowOrColumnMorphism' above. However, as fourth argument an integer *N* is to be specified. The computation of the truncation will then be performed in parallel in *N* child processes.

3.4.3 TruncateGradedRowOrColumnMorphismInParallel (for IsToricVariety, IsGradedRowOrColumnMorphism, IsList, IsPosInt, IsBool)

▷ `TruncateGradedRowOrColumnMorphismInParallel(V, a, d, N, B)` (operation)

Returns: a vector space morphism

This method operates just as 'TruncateGradedRowOrColumnMorphism' above. However, as fourth argument an integer *N* is to be specified. The computation of the truncation will then be performed in parallel in *N* child processes.

3.4.4 TruncateGradedRowOrColumnMorphismInParallel (for IsToricVariety, IsGradedRowOrColumnMorphism, IsHomalgModuleElement, IsPosInt, IsBool)

▷ `TruncateGradedRowOrColumnMorphismInParallel(V, a, m, N, B)` (operation)

Returns: a vector space morphism

This method operates just as 'TruncateGradedRowOrColumnMorphism' above. However, as fourth argument an integer *N* is to be specified. The computation of the truncation will then be performed in parallel in *N* child processes.

3.4.5 TruncateGradedRowOrColumnMorphismInParallel (for IsToricVariety, IsGradedRowOrColumnMorphism, IsList, IsPosInt)

▷ `TruncateGradedRowOrColumnMorphismInParallel(V, a, d, N)` (operation)

Returns: a vector space morphism

This method operates just as 'TruncateGradedRowOrColumnMorphism' above. However, as fourth argument an integer *N* is to be specified. The computation of the truncation will then be performed in parallel in *N* child processes.

3.4.6 TruncateGradedRowOrColumnMorphismInParallel (for IsToricVariety, IsGradedRowOrColumnMorphism, IsHomalgModuleElement, IsPosInt)

▷ `TruncateGradedRowOrColumnMorphismInParallel(V, a, m, N)` (operation)

Returns: a vector space morphism

This method operates just as 'TruncateGradedRowOrColumnMorphism' above. However, as fourth argument an integer N is to be specified. The computation of the truncation will then be performed in parallel in N child processes.

3.5 Examples

3.5.1 Truncations of graded rows and columns

Example
```
gap> P2 := ProjectiveSpace( 2 );
<A projective toric variety of dimension 2>
gap> cox_ring := CoxRing( P2 );
Q[x_1,x_2,x_3]
(weights: [ 1, 1, 1 ])
gap> row := GradedRow( [[[2],1]], cox_ring );
<A graded row of rank 1>
gap> trunc1 := DegreeXLayerOfGradedRowOrColumn( P2, row, [ -3 ] );
<A vector space embedded into (Q[x_1,x_2,x_3] (with weights [ 1, 1, 1 ]))^1>
gap> Length( Generators( trunc1 ) );
0
gap> trunc2 := DegreeXLayerOfGradedRowOrColumn( P2, row, [ -1 ] );
<A vector space embedded into (Q[x_1,x_2,x_3] (with weights [ 1, 1, 1 ]))^1>
gap> Length( Generators( trunc2 ) );
3
```

3.5.2 Formats for generators of truncations of graded rows and columns

Example
```
gap> row2 := GradedRow( [[[2],2]], cox_ring );
<A graded row of rank 2>
gap> gens1 := GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsListOfColumnMatrices
>             (P2, row2, [ -1 ] );;
gap> Length( gens1 );
6
gap> gens1[ 1 ];
<A 2 x 1 matrix over a graded ring>
gap> Display( gens1[ 1 ] );
x_1,
0
(over a graded ring)
gap> Display( gens1[ 4 ] );
0,
x_1
(over a graded ring)
gap> gens2 := GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsListsOfRecords
>             (P2, row2, [ -1 ] );
[ 6, [ rec( x_1 := 1, x_2 := 2, x_3 := 3 ),
```

```
       rec( x_1 := 4, x_2 := 5, x_3 := 6 ) ] ]
gap> gens3 := GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsUnionOfColumnMatrices
>              (P2, row2, [ -1 ] );
<A 2 x 6 mutable matrix over a graded ring>
gap> Display( gens3 );
x_1,x_2,x_3,0,  0,  0,
0,  0,  0,  x_1,x_2,x_3
(over a graded ring)
gap> gens4 := GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsListList
>              (P2, row2, [ -1 ] );
[ [ 1, x_1 ], [ 1, x_2 ], [ 1, x_3 ], [ 2, x_1 ], [ 2, x_2 ], [ 2, x_3 ] ]
```

3.5.3 Truncatons of morphisms of graded rows and columns

Example

```
gap> source := GradedRow( [[[-1],1]], cox_ring );
<A graded row of rank 1>
gap> range := GradedRow( [[[0],1]], cox_ring );
<A graded row of rank 1>
gap> trunc_generators := GeneratorsOfDegreeXLayerOfGradedRowOrColumnAsListsOfRecords
>                        (P2, range, [ 2 ] );
[ 6, [ rec( ("x_1*x_2") := 2, ("x_1*x_3") := 4, ("x_1^2") := 1,
            ("x_2*x_3") := 5, ("x_2^2") := 3, ("x_3^2") := 6 ) ] ]
gap> vars := IndeterminatesOfPolynomialRing( cox_ring );;
gap> matrix := HomalgMatrix( [[ vars[ 1 ] ]], cox_ring );
<A 1 x 1 matrix over a graded ring>
gap> mor := GradedRowOrColumnMorphism( source, matrix, range );
<A morphism in Category of graded rows over
Q[x_1,x_2,x_3] (with weights [ 1, 1, 1 ])>
gap> IsWellDefined( mor );
true
gap> trunc_mor := TruncateGradedRowOrColumnMorphism( P2, mor, [ 2 ] );
<A morphism in Category of matrices over Q (with weights [ 1 ])>
gap> Display( UnderlyingMatrix( trunc_mor ) );
1,0,0,0,0,0,
0,1,0,0,0,0,
0,0,0,1,0,0
(over a graded ring)
gap> matrix2 := HomalgMatrix( [[ 1/2*vars[ 1 ] ]], cox_ring );
<A 1 x 1 matrix over a graded ring>
gap> mor2 := GradedRowOrColumnMorphism( source, matrix2, range );
<A morphism in Category of graded rows over
Q[x_1,x_2,x_3] (with weights [ 1, 1, 1 ])>
gap> IsWellDefined( mor2 );
true
gap> trunc_mor2 := TruncateGradedRowOrColumnMorphism( P2, mor2, [ 2 ] );
<A morphism in Category of matrices over Q (with weights [ 1 ])>
gap> Display( UnderlyingMatrix( trunc_mor2 ) );
1/2,0,0,0,0,0,
0,1/2,0,0,0,0,
0,0,0,1/2,0,0
(over a graded ring)
```

3.5.4 Truncatons of morphisms of graded rows and columns in parallel

Example

```
gap> trunc_mor_parallel := TruncateGradedRowOrColumnMorphismInParallel
>                                            ( P2, mor, [ 2 ], 2 );
<A morphism in Category of matrices over Q (with weights [ 1 ])>
gap> Display( UnderlyingMatrix( trunc_mor_parallel ) );
1,0,0,0,0,0,
0,1,0,0,0,0,
0,0,0,1,0,0
(over a graded ring)
gap> trunc_mor2_parallel := TruncateGradedRowOrColumnMorphismInParallel
>                                            ( P2, mor2, [ 2 ], 2 );
<A morphism in Category of matrices over Q (with weights [ 1 ])>
gap> Display( UnderlyingMatrix( trunc_mor2_parallel ) );
1/2,0,0,0,0,0,
0,1/2,0,0,0,0,
0,0,0,1/2,0,0
(over a graded ring)
gap> trunc_mor2_parallel2 := TruncateGradedRowOrColumnMorphismInParallel
>                                            ( P2, mor2, [ 10 ], 3 );;
gap> IsWellDefined( trunc_mor2_parallel2 );
true
gap> NrRows( UnderlyingMatrix( trunc_mor2_parallel2 ) );
55
gap> NrColumns( UnderlyingMatrix( trunc_mor2_parallel2 ) );
66
```

Chapter 4

Truncations of f.p. graded modules

4.1 Truncations of fp graded modules

4.1.1 TruncateFPGradedModule (for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsList, IsBool, IsFieldForHomalg)

▷ `TruncateFPGradedModule(V, M, d, B, F)` (operation)

Returns: a FreydCategoryObject

The arguments are a toric variety V, an f.p. graded module M, a list d (specifying a element of the class group of V) a boolean B and a field F. We then compute the truncation of M to the degree d and return the corresponding vector space presentation as a FreydCategoryObject. If B is true, we display additional information during the computation. The latter may be useful for longer computations.

Example

```
gap> P2 := ProjectiveSpace( 2 );
<A projective toric variety of dimension 2>
gap> cox_ring := CoxRing( P2 );
Q[x_1,x_2,x_3]
(weights: [ 1, 1, 1 ])
gap> source := GradedRow( [[[-1],1]], cox_ring );
<A graded row of rank 1>
gap> range := GradedRow( [[[0],1]], cox_ring );
<A graded row of rank 1>
gap> vars := IndeterminatesOfPolynomialRing( cox_ring );;
gap> matrix := HomalgMatrix( [[ vars[ 1 ] ]], cox_ring );
<A 1 x 1 matrix over a graded ring>
gap> obj1 := FreydCategoryObject(
>           GradedRowOrColumnMorphism( source, matrix, range ) );
<An object in Category of f.p. graded
left modules over Q[x_1,x_2,x_3]
(with weights [ 1, 1, 1 ])>
gap> IsWellDefined( obj1 );
true
gap> trunc_obj1 := TruncateFPGradedModule( P2, obj1, [ 2 ] );
<An object in Freyd( Category of matrices
over Q (with weights [ 1 ]) )>
gap> IsWellDefined( trunc_obj1 );
true
gap> Display( UnderlyingMatrix( RelationMorphism( trunc_obj1 ) ) );
```

```
1,0,0,0,0,0,
0,1,0,0,0,0,
0,0,0,1,0,0
(over a graded ring)
gap> trunc_obj2 := TruncateFPGradedModuleInParallel( P2, obj1, [ 2 ], 2 );
<An object in Freyd( Category of matrices
over Q (with weights [ 1 ]) )>
gap> IsWellDefined( trunc_obj2 );
true
gap> Display( UnderlyingMatrix( RelationMorphism( trunc_obj2 ) ) );
1,0,0,0,0,0,
0,1,0,0,0,0,
0,0,0,1,0,0
(over a graded ring)
```

4.2 Truncations of fp graded modules in parallel

4.2.1 TruncateFPGradedModuleInParallel (for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsList, IsPosInt, IsBool, IsFieldForHomalg)

▷ `TruncateFPGradedModuleInParallel(V, M, d, N, B., F)` (operation)

Returns: a FreydCategoryObject

The arguments are a toric variety V, an f.p. graded module M, a list d (specifying a element of the class group of V), an integer N, a boolean B and a field F. We then compute the truncation of M to the degree d and return the corresponding vector space presentation encoded as a FreydCategoryObject. This is performed in N child processes in parallel. If B is true, we display additional information during the computation. The latter may be useful for longer computations.

4.3 Truncations of fp graded modules morphisms

4.3.1 TruncateFPGradedModuleMorphism (for IsToricVariety, IsFpGradedLeftOrRightModulesMorphism, IsList, IsBool, IsFieldForHomalg)

▷ `TruncateFPGradedModuleMorphism(V, M, d, B, F)` (operation)

Returns: a FreydCategoryMorphism

The arguments are a toric variety V, an f.p. graded module morphism M, a list d (specifying a element of the class group of V), a boolean B and a field F. We then compute the truncation of M to the degree d and return the corresponding morphism of vector space presentations encoded as a FreydCategoryMorphism. If B is true, we display additional information during the computation. The latter may be useful for longer computations.

4.4 Truncations of fp graded modules morphisms in parallel

4.4.1 TruncateFPGradedModuleMorphismInParallel (for IsToricVariety, IsFpGradedLeftOrRightModulesMorphism, IsList, IsList, IsBool, IsFieldForHomalg)

▷ `TruncateFPGradedModuleMorphismInParallel(V, M, d[, N1, N2, N3], B, F)` (operation)

Returns: a FreydCategoryMorphism

The arguments are a toric variety V, an f.p. graded module morphism M, a list d (specifying a element of the class group of V), a list of 3 non-negative integers [N_1, N_2, N_3], a boolean B and a field F. We then compute the truncation of M to the degree d and return the corresponding morphism of vector space presentations encoded as a FreydCategoryMorphism. This is done in parallel: the truncation of the source is done by N_1 child processes in parallel, the truncation of the morphism datum is done by N_2 child processes and the truncation of the range of M by N_3 processes. If the boolean B is set to true, we display additional information during the computation. The latter may be useful for longer computations.

4.5 Truncations of f.p. graded module morphisms

Example

```
gap> source := GradedRow( [[[-1],1]], cox_ring );
<A graded row of rank 1>
gap> range := GradedRow( [[[1],2]], cox_ring );
<A graded row of rank 2>
gap> matrix := HomalgMatrix( [[ vars[ 1 ] * vars[ 2 ],
>                               vars[ 1 ] * vars[ 3 ] ]], cox_ring );
<A 1 x 2 matrix over a graded ring>
gap> obj2 := FreydCategoryObject(
>           GradedRowOrColumnMorphism( source, matrix, range ) );
<An object in Category of f.p. graded
left modules over Q[x_1,x_2,x_3]
(with weights [ 1, 1, 1 ])>
gap> source := GradedRow( [[[0],1]], cox_ring );
<A graded row of rank 1>
gap> range := GradedRow( [[[1],2]], cox_ring );
<A graded row of rank 2>
gap> matrix := HomalgMatrix( [[ vars[ 2 ], vars[ 3 ] ]], cox_ring );
<A 1 x 2 matrix over a graded ring>
gap> mor := GradedRowOrColumnMorphism( source, matrix, range );
<A morphism in Category of graded rows
over Q[x_1,x_2,x_3] (with weights [ 1, 1, 1 ])>
gap> pres_mor := FreydCategoryMorphism( obj1, mor, obj2 );
<A morphism in Category of f.p. graded
left modules over Q[x_1,x_2,x_3]
(with weights [ 1, 1, 1 ])>
gap> IsWellDefined( pres_mor );
true
gap> trunc_pres_mor1 := TruncateFPGradedModuleMorphism( P2, pres_mor, [ 2 ] );
<A morphism in Freyd( Category of
matrices over Q (with weights [ 1 ]) )>
gap> IsWellDefined( trunc_pres_mor1 );
true
```

```
gap> trunc_pres_mor2 := TruncateFPGradedModuleMorphismInParallel
>                              ( P2, pres_mor, [ 2 ], [ 2, 2, 2 ] );
<A morphism in Freyd( Category of
matrices over Q (with weights [ 1 ]))>
gap> IsWellDefined( trunc_pres_mor2 );
true
```

Chapter 5

Truncation functors for f.p. graded modules

5.1 Truncation functor for graded rows and columns

5.1.1 TruncationFunctorForGradedRows (for IsToricVariety, IsList)

▷ `TruncationFunctorForGradedRows(V, d)` (operation)

Returns: a functor

The arguments are a toric variety V and degree_list d specifying an element of the degree group of the toric variety V. The latter can either be a list of integers or a HomalgModuleElement. Based on this input, this method returns the functor for the truncation of graded rows over the Cox ring of V to degree d.

5.1.2 TruncationFunctorForGradedColumns (for IsToricVariety, IsList)

▷ `TruncationFunctorForGradedColumns(V, d)` (operation)

Returns: a functor

The arguments are a toric variety V and degree_list d specifying an element of the degree group of the toric variety V. The latter can either be a list of integers or a HomalgModuleElement. Based on this input, this method returns the functor for the truncation of graded columns over the Cox ring of V to degree d.

5.2 Truncation functor for f.p. graded modules

5.2.1 TruncationFunctorForFpGradedLeftModules (for IsToricVariety, IsList)

▷ `TruncationFunctorForFpGradedLeftModules(V, d)` (operation)

Returns: a functor

The arguments are a toric variety V and degree list d, which specifies an element of the degree group of the toric variety V. d can either be a list of integers or a HomalgModuleElement. Based on this input, this method returns the functor for the truncation of f.p. graded right modules to degree d.

5.2.2 TruncationFunctorForFpGradedRightModules (for IsToricVariety, IsList)

▷ `TruncationFunctorForFpGradedRightModules(V, d)` (operation)

Returns: a functor

The arguments are a toric variety V and degree list d, which specifies an element of the degree group of the toric variety V. d can either be a list of integers or a HomalgModuleElement. Based on this input, this method returns the functor for the truncation of f.p. graded right modules to degree d.

5.3 Examples

Example

```
gap> P2 := ProjectiveSpace( 2 );
<A projective toric variety of dimension 2>
gap> P1 := ProjectiveSpace( 1 );
<A projective toric variety of dimension 1>
gap> tor := P2 * P1;
<A projective toric variety of dimension 3
which is a product of 2 toric varieties>
gap> TruncationFunctorForGradedRows( tor, [ 2, 3 ] );
Trunction functor for Category of graded rows
over Q[x_1,x_2,x_3,x_4,x_5] (with weights
[ [ 0, 1 ], [ 1, 0 ], [ 1, 0 ],
[ 0, 1 ], [ 0, 1 ] ] ) to the degree [ 2, 3 ]
gap> TruncationFunctorForFpGradedLeftModules( tor, [ 4, 5 ] );
Truncation functor for Category of f.p.
graded left modules over Q[x_1,x_2,x_3,x_4,x_5]
(with weights [ [ 0, 1 ], [ 1, 0 ], [ 1, 0 ],
[ 0, 1 ], [ 0, 1 ] ]) to the degree [ 4, 5 ]
```

Chapter 6

Truncations of GradedExt for f.p. graded modules

6.1 Truncations of InternalHoms of FpGradedModules

6.1.1 TruncateInternalHom (for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsFpGradedLeftOrRightModulesObject, IsList, IsBool, IsFieldForHomalg)

▷ `TruncateInternalHom(`*arg1, arg2, arg3, arg4, arg5, arg6*`)` (operation)

6.1.2 TruncateInternalHomEmbedding (for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsFpGradedLeftOrRightModulesObject, IsList, IsBool, IsFieldForHomalg)

▷ `TruncateInternalHomEmbedding(`*arg1, arg2, arg3, arg4, arg5, arg6*`)` (operation)

6.1.3 TruncateInternalHom (for IsToricVariety, IsFpGradedLeftOrRightModulesMorphism, IsFpGradedLeftOrRightModulesMorphism, IsList, IsBool, IsFieldForHomalg)

▷ `TruncateInternalHom(`*arg1, arg2, arg3, arg4, arg5, arg6*`)` (operation)

6.2 Truncations of InternalHoms of FpGradedModules to degree zero

6.2.1 TruncateInternalHomToZero (for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsFpGradedLeftOrRightModulesObject, IsBool, IsFieldForHomalg)

▷ `TruncateInternalHomToZero(`*arg1, arg2, arg3, arg4, arg5*`)` (operation)

6.2.2 TruncateInternalHomEmbeddingToZero (for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsFpGradedLeftOrRightModulesObject, IsBool, IsFieldForHomalg)

▷ `TruncateInternalHomEmbeddingToZero(arg1, arg2, arg3, arg4, arg5)` (operation)

6.2.3 TruncateInternalHomToZero (for IsToricVariety, IsFpGradedLeftOrRightModulesMorphism, IsFpGradedLeftOrRightModulesMorphism, IsBool, IsFieldForHomalg)

▷ `TruncateInternalHomToZero(arg1, arg2, arg3, arg4, arg5)` (operation)

6.3 Truncations of InternalHoms of FpGradedModules in parallel

6.3.1 TruncateInternalHomInParallel (for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsFpGradedLeftOrRightModulesObject, IsList, IsBool, IsFieldForHomalg)

▷ `TruncateInternalHomInParallel(arg1, arg2, arg3, arg4, arg5, arg6)` (operation)

6.3.2 TruncateInternalHomEmbeddingInParallel (for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsFpGradedLeftOrRightModulesObject, IsList, IsBool, IsFieldForHomalg)

▷ `TruncateInternalHomEmbeddingInParallel(arg1, arg2, arg3, arg4, arg5, arg6)` (operation)

6.3.3 TruncateInternalHomInParallel (for IsToricVariety, IsFpGradedLeftOrRightModulesMorphism, IsFpGradedLeftOrRightModulesMorphism, IsList, IsBool, IsFieldForHomalg)

▷ `TruncateInternalHomInParallel(arg1, arg2, arg3, arg4, arg5, arg6)` (operation)

6.4 Truncations of InternalHoms of FpGradedModules to degree zero in parallel

6.4.1 TruncateInternalHomToZeroInParallel (for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsFpGradedLeftOrRightModulesObject, IsBool, IsFieldForHomalg)

▷ `TruncateInternalHomToZeroInParallel(arg1, arg2, arg3, arg4, arg5)` (operation)

6.4.2 TruncateInternalHomEmbeddingToZeroInParallel (for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsFpGradedLeftOrRightModulesObject, IsBool, IsFieldForHomalg)

▷ `TruncateInternalHomEmbeddingToZeroInParallel(`*arg1, arg2, arg3, arg4, arg5*`)` (operation)

6.4.3 TruncateInternalHomToZeroInParallel (for IsToricVariety, IsFpGradedLeftOrRightModulesMorphism, IsFpGradedLeftOrRightModulesMorphism, IsBool, IsFieldForHomalg)

▷ `TruncateInternalHomToZeroInParallel(`*arg1, arg2, arg3, arg4, arg5*`)` (operation)

6.4.4 TruncateGradedExt (for IsInt, IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsFpGradedLeftOrRightModulesObject, IsList, IsList)

▷ `TruncateGradedExt(`*arg1, arg2, arg3, arg4, arg5, arg6*`)` (operation)

6.4.5 TruncateGradedExtToZero (for IsInt, IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsFpGradedLeftOrRightModulesObject, IsBool, IsFieldForHomalg)

▷ `TruncateGradedExtToZero(`*arg1, arg2, arg3, arg4, arg5, arg6*`)` (operation)

6.4.6 TruncateGradedExtInParallel (for IsInt, IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsFpGradedLeftOrRightModulesObject, IsList, IsList)

▷ `TruncateGradedExtInParallel(`*arg1, arg2, arg3, arg4, arg5, arg6*`)` (operation)

6.4.7 TruncateGradedExtToZeroInParallel (for IsInt, IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsFpGradedLeftOrRightModulesObject, IsBool, IsFieldForHomalg)

▷ `TruncateGradedExtToZeroInParallel(`*arg1, arg2, arg3, arg4, arg5, arg6*`)` (operation)

6.5 Examples

6.5.1 Truncation of IntHom

Example
```
gap> P2 := ProjectiveSpace( 2 );
<A projective toric variety of dimension 2>
gap> cox_ring := CoxRing( P2 );
```

```
Q[x_1,x_2,x_3]
(weights: [ 1, 1, 1 ])
gap> source := GradedRow( [[[-1],1]], cox_ring );
<A graded row of rank 1>
gap> range := GradedRow( [[[0],1]], cox_ring );
<A graded row of rank 1>
gap> vars := IndeterminatesOfPolynomialRing( cox_ring );;
gap> matrix := HomalgMatrix( [[ vars[ 1 ] ]], cox_ring );
<A 1 x 1 matrix over a graded ring>
gap> obj1 := FreydCategoryObject(
>           GradedRowOrColumnMorphism( source, matrix, range ) );
<An object in Category of f.p. graded
left modules over Q[x_1,x_2,x_3]
(with weights [ 1, 1, 1 ])>
gap> IsWellDefined( obj1 );
true
gap> source := GradedRow( [[[-1],1]], cox_ring );
<A graded row of rank 1>
gap> range := GradedRow( [[[1],2]], cox_ring );
<A graded row of rank 2>
gap> matrix := HomalgMatrix( [[ vars[ 1 ] * vars[ 2 ],
>                              vars[ 1 ] * vars[ 3 ] ]], cox_ring );
<A 1 x 2 matrix over a graded ring>
gap> obj2 := FreydCategoryObject(
>           GradedRowOrColumnMorphism( source, matrix, range ) );
<An object in Category of f.p. graded
left modules over Q[x_1,x_2,x_3]
(with weights [ 1, 1, 1 ])>
gap> IsWellDefined( obj2 );
true
gap> source := GradedRow( [[[0],1]], cox_ring );
<A graded row of rank 1>
gap> range := GradedRow( [[[1],2]], cox_ring );
<A graded row of rank 2>
gap> matrix := HomalgMatrix( [[ vars[ 2 ], vars[ 3 ] ]], cox_ring );
<A 1 x 2 matrix over a graded ring>
gap> mor := GradedRowOrColumnMorphism( source, matrix, range );
<A morphism in Category of graded rows
over Q[x_1,x_2,x_3] (with weights [ 1, 1, 1 ])>
gap> pres_mor := FreydCategoryMorphism( obj1, mor, obj2 );
<A morphism in Category of f.p. graded
left modules over Q[x_1,x_2,x_3]
(with weights [ 1, 1, 1 ])>
gap> IsWellDefined( pres_mor );
true
gap> Q := HomalgFieldOfRationalsInSingular();
Q
gap> m1 := TruncateInternalHom( P2, obj1, obj2, [ 4 ], false, Q );
<An object in Freyd( Category of matrices over Q )>
gap> IsWellDefined( m1 );
true
gap> m2 := TruncateInternalHomEmbedding( P2, obj1, obj2, [ 4 ], false, Q );
```

```
<A monomorphism in Freyd( Category of matrices over Q )>
gap> IsWellDefined( m2 );
true
gap> m3 := TruncateInternalHom( P2, pres_mor, IdentityMorphism( obj2 ), [ 4 ], false, Q );
<A morphism in Freyd( Category of matrices over Q )>
gap> IsWellDefined( m3 );
true
```

6.5.2 Truncation of IntHom to degree zero

Example

```
gap> m4 := TruncateInternalHomToZero( P2, obj1, obj2, false, Q );
<An object in Freyd( Category of matrices over Q )>
gap> IsWellDefined( m4 );
true
gap> m5 := TruncateInternalHomEmbeddingToZero( P2, obj1, obj2, false, Q );
<A monomorphism in Freyd( Category of matrices over Q )>
gap> IsWellDefined( m5 );
true
gap> m6 := TruncateInternalHomToZero( P2, pres_mor, IdentityMorphism( obj2 ), false, Q );
<A morphism in Freyd( Category of matrices over Q )>
gap> IsWellDefined( m6 );
true
```

6.5.3 Truncation of IntHom in parallel

Example

```
gap> m7 := TruncateInternalHomInParallel( P2, obj1, obj2, [ 4 ], false, Q );
<An object in Freyd( Category of matrices over Q )>
gap> m1 = m7;
true
gap> m8 := TruncateInternalHomEmbeddingInParallel( P2, obj1, obj2, [ 4 ], false, Q );
<A monomorphism in Freyd( Category of matrices over Q )>
gap> m8 = m2;
true
gap> m9 := TruncateInternalHomInParallel( P2, pres_mor, IdentityMorphism( obj2 ), [ 4 ], false, Q
<A morphism in Freyd( Category of matrices over Q )>
gap> m9 = m3;
true
```

6.5.4 Truncation of IntHom to degree zero in parallel

Example

```
gap> m10 := TruncateInternalHomToZeroInParallel( P2, obj1, obj2, false, Q );
<An object in Freyd( Category of matrices over Q )>
gap> m10 = m4;
true
gap> m11 := TruncateInternalHomEmbeddingToZeroInParallel( P2, obj1, obj2, false, Q );
<A monomorphism in Freyd( Category of matrices over Q )>
gap> m11 = m5;
true
gap> m12 := TruncateInternalHomToZeroInParallel( P2, pres_mor, IdentityMorphism( obj2 ), false, Q
<A morphism in Freyd( Category of matrices over Q )>
```

```
gap> m12 = m6;
true
```

6.5.5 Truncation of GradedExt

Example

```
gap> v1 := TruncateGradedExt( 1, P2, obj1, obj2, [ 4 ], [ false, Q ] );
<An object in Freyd( Category of matrices over Q )>
gap> IsWellDefined( v1 );
true
gap> v2 := TruncateGradedExt( 1, P2, obj1, obj2, [ 0 ], [ false, Q ] );
<An object in Freyd( Category of matrices over Q )>
gap> IsWellDefined( v2 );
true
gap> v3 := TruncateGradedExtToZero( 1, P2, obj1, obj2, false, Q );
<An object in Freyd( Category of matrices over Q )>
gap> v3 = v2;
true
gap> v4 := TruncateGradedExtInParallel( 1, P2, obj1, obj2, [ 4 ], [ false, Q ] );
<An object in Freyd( Category of matrices over Q )>
gap> IsWellDefined( v4 );
true
gap> v5 := TruncateGradedExtInParallel( 1, P2, obj1, obj2, [ 0 ], [ false, Q ] );
<An object in Freyd( Category of matrices over Q )>
gap> IsWellDefined( v5 );
true
gap> v6 := TruncateGradedExtToZeroInParallel( 1, P2, obj1, obj2, false, Q );
<An object in Freyd( Category of matrices over Q )>
gap> v6 = v5;
true
```

Chapter 7

Localized degree-0 rings

7.1 Localized degree-0-layer of graded rings

7.1.1 Localized_degree_zero_monomials (for IsHomalgGradedRing, IsList)

▷ `Localized_degree_zero_monomials(R, L)` (operation)

Returns: a list

This method computes the generators of vanishing degree of of a graded ring R localized at a list L of variables.

7.1.2 Localized_degree_zero_ring (for IsHomalgGradedRing, IsList)

▷ `Localized_degree_zero_ring(R, L)` (operation)

Returns: a ring

This method accepts a homalg graded ring R and a list L of variables on which this ring is to be localized. We then compute the degree-0-layer of this localization and express it as a quotient ring. This method then returns this quotient ring.

7.1.3 Localized_degree_zero_ring_and_generators (for IsHomalgGradedRing, IsList)

▷ `Localized_degree_zero_ring_and_generators(R, L)` (operation)

Returns: a list

This method accepts a homalg graded ring R and a list L of variables on which this ring is to be localized. We then compute the generators of the degree-0-layer of this localization and the corresponding quotient ring. Finally, we return the list formed from the generators and this quotient ring.

7.2 Examples

We can localize a graded ring and then truncate it to a given degree. Here is an example:

Example

```
gap> Q := HomalgFieldOfRationalsInSingular();;
gap> S := GradedRing( Q * "x_1, x_2, x_3, x_4" );;
gap> SetWeightsOfIndeterminates( S, [[1,0],[1,0],[0,1],[0,1]] );;
gap> Length( Localized_degree_zero_monomials( S, [ 1,3 ] ) );
2
```

```
gap> Localized_degree_zero_ring( S, [ 1,3 ] );
Q[t1,t2]
```

Chapter 8

Localized truncations of graded rows or columns

8.1 Technical tools

8.1.1 Degree_basis (for IsHomalgGradedRing, IsList, IsList)

▷ `Degree_basis(R, L)` (operation)

Returns: a list

This function accepts a graded ring R and a list of variables L as well as a twist T. We can then consider the ring R localized at L and twisted by T. We can view this as a R_L module, and this function computes a basis of this module (over R_L).

8.1.2 Degree_part_relations (for IsList, IsList, IsHomalgRing)

▷ `Degree_part_relations(R, L)` (operation)

Returns: a list

This function computes relations among generators.

8.2 Localized degree-0-layer of graded rows and columns

8.2.1 LocalizedDegreeZero (for IsGradedRow, IsList)

▷ `LocalizedDegreeZero(R, L)` (operation)

Returns: a fp graded module

First localize a graded row R at a list L of variables and subsequently truncate this localization to degree 0.

8.2.2 LocalizedDegreeZero (for IsGradedRow, IsList, IsHomalgGradedRing, IsList, IsCapCategory)

▷ `LocalizedDegreeZero(arg1, arg2, arg3, arg4, arg5)` (operation)

8.2.3 LocalizedDegreeZero (for IsGradedColumn, IsList)

▷ LocalizedDegreeZero(*C*, *L*) (operation)

Returns: a fp graded module

First localize a graded column C at a list L of variables and subsequently truncate this localization to degree 0.

8.2.4 LocalizedDegreeZero (for IsGradedColumn, IsList, IsHomalgGradedRing, IsList, IsCapCategory)

▷ LocalizedDegreeZero(*arg1*, *arg2*, *arg3*, *arg4*, *arg5*) (operation)

8.2.5 LocalizedDegreeZero (for IsGradedRowOrColumnMorphism, IsList)

▷ LocalizedDegreeZero(*m*, *L*) (operation)

Returns: an fp graded module morphism

Localize a graded row morphism m at a list L of variables and subsequently truncate this localization to degree 0.

8.2.6 LocalizedDegreeZero (for IsGradedRowOrColumnMorphism, IsList, IsHomalgGradedRing, IsList, IsCapCategory)

▷ LocalizedDegreeZero(*arg1*, *arg2*, *arg3*, *arg4*, *arg5*) (operation)

8.3 Examples

We can perform localized truncations of graded rows:

Example
```
gap> Q := HomalgFieldOfRationalsInSingular();;
gap> S := GradedRing( Q * "x_1, x_2, x_3, x_4" );;
gap> SetWeightsOfIndeterminates( S, [[1,0],[1,0],[0,1],[0,1]] );;
gap> vars := IndeterminatesOfPolynomialRing( S );;
gap> row := GradedRow( [ [[1,1],2] ], S );;
gap> new_row := LocalizedDegreeZero( row, [ 1,3 ] );;
gap> IsWellDefined( new_row );
true
```

Similarly, we can compute localized truncations of graded row morphisms:

Example
```
gap> ideal := LeftIdealForCAP( [ vars[ 1 ] * vars[ 3 ], vars[ 1 ] * vars[ 4 ],
>                              vars[ 2 ] * vars[ 3 ], vars[ 2 ] * vars[ 4 ] ], S );;
gap> IsWellDefined( ideal );
true
gap> mor := RelationMorphism( ideal );;
gap> new_mor := LocalizedDegreeZero( mor, [ 1,3 ] );;
gap> IsWellDefined( new_mor );
true
```

Here is another example, where we compute the localized truncation of a morphism of graded rows:

Example
```
gap> Q := HomalgFieldOfRationalsInSingular();;
gap> S := GradedRing( Q * "x_1, x_2, x_3, x_4" );;
gap> SetWeightsOfIndeterminates( S, [[1,-7],[0,1],[1,0],[0,1]] );;
gap> S2 := Localized_degree_zero_ring_and_generators( S, [ 1,2 ] );;
gap> M := HomalgMatrix( "[ x_1*x_2^7, x_3, x_1*x_4^8, 0 ]", 2,2, S );;
gap> range := GradedRow( [ [[0,0],2] ], S );;
gap> mor := DeduceSomeMapFromMatrixAndRangeForGradedRows( M, range );;
gap> new_mor := LocalizedDegreeZero( mor, [ 1, 2 ] );;
gap> IsWellDefined( new_mor );
true
```

Here is another example which should be placed in the graded rows and columns

Example
```
gap> S := HomalgFieldOfRationalsInSingular() * "x1..3";;
gap> S := GradedRing( S );;
gap> SetWeightsOfIndeterminates( S, [1,1,2] );;
gap> vars := IndeterminatesOfPolynomialRing( S );;
gap> mons := Localized_degree_zero_monomials( S, [3] );;
gap> Length( mons );
3
gap> source := GradedRow( [ [[ 0 ], 2 ] ], S );;
gap> IsWellDefined( LocalizedDegreeZero( source, [ 3 ] ) );
true
gap> range := GradedRow( [ [[ 1 ], 1 ] ], S );;
gap> IsWellDefined( LocalizedDegreeZero( range, [ 3 ] ) );
true
gap> matrix := HomalgMatrix( [ [ vars[ 1 ] ], [ vars[ 2 ] ] ], S );;
gap> mor := GradedRowOrColumnMorphism( source, matrix, range );;
gap> IsWellDefined( mor );
true
gap> mor2 := LocalizedDegreeZero( mor, [ 3 ] );;
gap> IsWellDefined( mor2 );
true
```

Chapter 9

Localized truncations of FPGradedModules

9.1 Localized degree-0-layer of f.p. graded modules

9.1.1 LocalizedDegreeZero (for IsFpGradedLeftOrRightModulesObject, IsList)

▷ `LocalizedDegreeZero(M, L)` (operation)

Returns: an fp graded module

This method accepts an fp graded module M and a list L of variables. It then localizes M at these variables and computes the degree-0-layer.

9.1.2 LocalizedDegreeZero (for IsFpGradedLeftOrRightModulesObject, IsList, IsHomalgGradedRing, IsList, IsCapCategory)

▷ `LocalizedDegreeZero(arg1, arg2, arg3, arg4, arg5)` (operation)

9.1.3 LocalizedDegreeZero (for IsFpGradedLeftOrRightModulesMorphism, IsList)

▷ `LocalizedDegreeZero(M, L)` (operation)

Returns: a morphism of fp graded modules

This method accepts an fp graded module morphism M and a list L of variables. It then localizes M at these variables and computes the degree-0-layer.

9.2 Examples

We can perform localized truncations of fp graded modules:

Example
```
gap> Q := HomalgFieldOfRationalsInSingular();;
gap> S := GradedRing( Q * "x_1, x_2, x_3, x_4" );;
gap> SetWeightsOfIndeterminates( S, [[1,0],[1,0],[0,1],[0,1]] );;
gap> vars := IndeterminatesOfPolynomialRing( S );;
gap> ideal := LeftIdealForCAP( [ vars[ 1 ] * vars[ 3 ], vars[ 1 ] * vars[ 4 ],
>                               vars[ 2 ] * vars[ 3 ], vars[ 2 ] * vars[ 4 ] ], S );;
```

```
gap> IsWellDefined( ideal );
true
gap> new_ideal := LocalizedDegreeZero( ideal, [ 1,3 ] );;
gap> IsWellDefined( new_ideal );
true
```

We can also compute localized truncations of fp graded module morphisms:

Example

```
gap> pr := WeakCokernelProjection( RelationMorphism( ideal ) );;
gap> range := AsFreydCategoryObject( Range( pr ) );;
gap> mor := FreydCategoryMorphism( ideal, pr, range );;
gap> new_mor := LocalizedDegreeZero( mor, [ 1,3 ] );;
gap> IsWellDefined( new_mor );
true
```

Chapter 10

Functors for localized truncations to degree 0

10.1 Localized truncation functors for graded rows and columns

10.1.1 LocalizedTruncationFunctorForGradedRows (for IsHomalgGradedRing, IsList)

▷ `LocalizedTruncationFunctorForGradedRows(S, L)` (operation)

Returns: a functor

The arguments are a graded ring S and a list L of variables. This function then computes the localized truncation functor at the variables L to degree 0 for graded rows.

10.1.2 LocalizedTruncationFunctorForGradedColumns (for IsHomalgGradedRing, IsList)

▷ `LocalizedTruncationFunctorForGradedColumns(S, L)` (operation)

Returns: a functor

The arguments are a graded ring S and a list L of variables. This function then computes the localized truncation functor at the variables L to degree 0 for graded columns.

10.2 Localized truncation functors for f.p. graded modules

10.2.1 LocalizedTruncationFunctorForFPGradedLeftModules (for IsHomalgGradedRing, IsList)

▷ `LocalizedTruncationFunctorForFPGradedLeftModules(S, L)` (operation)

Returns: a functor

The arguments are a graded ring S and a list L of variables. This function then computes the localized truncation functor at the variables L to degree 0 for fp graded left modules.

10.2.2 LocalizedTruncationFunctorForFPGradedRightModules (for IsHomalgGradedRing, IsList)

▷ `LocalizedTruncationFunctorForFPGradedRightModules(S, L)` (operation)

Returns: a functor

The arguments are a graded ring S and a list L of variables. This function then computes the localized truncation functor at the variables L to degree 0 for fp graded right modules.

10.3 Examples

We can compute the truncation functors for graded rows, graded columns and f.p. graded modules:

Example
```
gap> Q := HomalgFieldOfRationalsInSingular();;
gap> S := GradedRing( Q * "x_1, x_2, x_3, x_4" );;
gap> SetWeightsOfIndeterminates( S, [[1,0],[1,0],[0,1],[0,1]] );;
gap> f1 := LocalizedTruncationFunctorForGradedRows( S, [ 1 ] );;
gap> f2 := LocalizedTruncationFunctorForGradedColumns( S, [ 1 ] );;
gap> f3 := LocalizedTruncationFunctorForFPGradedLeftModules( S, [ 1 ] );;
gap> f4 := LocalizedTruncationFunctorForFPGradedRightModules( S, [ 1 ] );;
```

Chapter 11

Technical functions

11.1 Functions to facilitate localized truncations

11.1.1 Get_image_of_generator (for IsList, IsHomalgRingElement)

▷ `Get_image_of_generator(`*arg1*`,` *arg2*`)` (operation)

11.1.2 Result_of_generator (for IsList, IsHomalgRingElement, IsList, IsList)

▷ `Result_of_generator(`*arg1*`,` *arg2*`,` *arg3*`,` *arg4*`)` (operation)

11.1.3 Block_matrix_to_matrix (for IsList)

▷ `Block_matrix_to_matrix(`*arg*`)` (operation)

11.1.4 New_matrix_mapping_by_generator_lists (for IsList, IsList, IsList, IsList, IsHomalgRing)

▷ `New_matrix_mapping_by_generator_lists(`*arg1*`,` *arg2*`,` *arg3*`,` *arg4*`,` *arg5*`)` (operation)

11.2 Functions to convert rows and columns (and presentations thereof)

11.2.1 TurnIntoColumn (for IsCategoryOfRowsObject)

▷ `TurnIntoColumn(`*R*`)` (operation)

Returns: a column

Turn a row R into the corresponding column.

11.2.2 TurnIntoRow (for IsCategoryOfColumnsObject)

▷ `TurnIntoRow(`*C*`)` (operation)

Returns: a row

Turn a column C into the corresponding row.

11.2.3 TurnIntoColumnMorphism (for IsCategoryOfRowsMorphism)

▷ `TurnIntoColumnMorphism(m)` (operation)

Returns: a morphism of columns

Turn a morphism m of rows into the corresponding morphism of columns.

11.2.4 TurnIntoRowMorphism (for IsCategoryOfColumnsMorphism)

▷ `TurnIntoRowMorphism(m)` (operation)

Returns: a morphism of rows

Turn a morphism m of columns into the corresponding morphism of row.

11.2.5 TurnIntoColumnPresentation (for IsFreydCategoryObject)

▷ `TurnIntoColumnPresentation(P)` (operation)

Returns: a column presentation

Turn a row presentation P into the corresponding column presentation.

11.2.6 TurnIntoRowPresentation (for IsFreydCategoryObject)

▷ `TurnIntoRowPresentation(P)` (operation)

Returns: a row presentation

Turn a column presentation P into the corresponding row presentation.

11.2.7 TurnIntoColumnPresentationMorphism (for IsFreydCategoryMorphism)

▷ `TurnIntoColumnPresentationMorphism(m)` (operation)

Returns: a column presentation morphism

Turn a row presentation morphism m into the corresponding column presentation morphism.

11.2.8 TurnIntoRowPresentationMorphism (for IsFreydCategoryMorphism)

▷ `TurnIntoRowPresentationMorphism(m)` (operation)

Returns: a row presentation morphism

Turn a column presentation morphism m into the corresponding row presentation morphism.

Index