CAPCategoryOf-ProjectiveGraded-Modules

Category of projective graded modules over a graded ring

2019.01.10

10 January 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

Category of projective graded left modules

1.1 Constructors

1.1.1 CAPCategoryOfProjectiveGradedLeftModules (for IsHomalgGradedRing)

▷ `CAPCategoryOfProjectiveGradedLeftModules(`*R*`)` (attribute)

Returns: a category

The argument is a homalg graded ring *R*. The output is the category of projective graded left modules over *R*.

Chapter 2

Category of projective graded right modules

2.1 Constructors

2.1.1 CAPCategoryOfProjectiveGradedRightModules (for IsHomalgGradedRing)

▷ `CAPCategoryOfProjectiveGradedRightModules(`*R*`)` (attribute)

Returns: a category

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

Chapter 3

Objects

3.1 GAP Categories

3.1.1 IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject (for IsCapCategoryObject)

▷ `IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject(`*object*`)` (filter)

Returns: `true` or `false`

The GAP category of objects in the CAP category of projective graded left or right modules over a graded ring R.

3.1.2 IsCAPCategoryOfProjectiveGradedLeftModulesObject (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject)

▷ `IsCAPCategoryOfProjectiveGradedLeftModulesObject(`*object*`)` (filter)

Returns: `true` or `false`

The GAP category of objects in the CAP category of projective graded left modules over a graded ring R.

3.1.3 IsCAPCategoryOfProjectiveGradedRightModulesObject (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject)

▷ `IsCAPCategoryOfProjectiveGradedRightModulesObject(`*object*`)` (filter)

Returns: `true` or `false`

The GAP category of objects in the CAP category of projective graded right modules over a graded ring R.

3.2 Constructors

3.2.1 CAPCategoryOfProjectiveGradedLeftModulesObject (for IsList, IsHomalgGradedRing)

▷ `CAPCategoryOfProjectiveGradedLeftModulesObject(`*degree_list, R*`)` (operation)

Returns: an object

The arguments are a list of degrees and a homalg graded ring R. The list of degrees must be of the form [[d_1, n_1], [d_2, n_2], ...] where d_i are degrees, i.e. elements in the degree group of R and the n_i are non-negative integers. Currently there are two formats that are supported to enter the degrees. Either one can enter them as lists of integers, say $d_1 = [1,1,0,2]$, or they can be entered as Homalg_Module_Elements of the degree group of R. In either case, the result is the projective and graded left module associated to the degrees d_i and their multiplicities n_i.

3.2.2 CAPCategoryOfProjectiveGradedLeftModulesObject (for IsList, IsHomalgGradedRing, IsBool)

▷ `CAPCategoryOfProjectiveGradedLeftModulesObject(`*`degree_list, R`*`)` (operation)

Returns: an object

As 'CAPCategoryOfProjectiveGradedLeftModulesObject', but the boolean (= third argument) allows to switch off checks on the input data. If this boolean is set to true, then the input checks are performed and otherwise they are not. Calling this constructor with 'false' is therefore suited for high performance applications.

3.2.3 CAPCategoryOfProjectiveGradedRightModulesObject (for IsList, IsHomalgGradedRing)

▷ `CAPCategoryOfProjectiveGradedRightModulesObject(`*`degree_list, R`*`)` (operation)

Returns: an object

The arguments are a list of degrees and a homalg graded ring R. The list of degrees must be of the form [[d_1, n_1], [d_2, n_2], ...] where d_i are degrees, i.e. elements in the degree group of R and the n_i are non-negative integers. Currently there are two formats that are supported to enter the degrees. Either one can enter them as lists of integers, say $d_1 = [1,1,0,2]$, or they can be entered as Homalg_Module_Elements of the degree group of R. In either case, the result is the projective and graded right module associated to the degrees d_i and their multiplicities n_i.

3.2.4 CAPCategoryOfProjectiveGradedRightModulesObject (for IsList, IsHomalgGradedRing, IsBool)

▷ `CAPCategoryOfProjectiveGradedRightModulesObject(`*`degree_list, R`*`)` (operation)

Returns: an object

As 'CAPCategoryOfProjectiveGradedLeftModulesObject', but the boolean (= third argument) allows to switch off checks on the input data. If this boolean is set to true, then the input checks are performed and otherwise they are not. Calling this constructor with 'false' is therefore suited for high performance applications.

3.3 Attributes

3.3.1 UnderlyingHomalgGradedRing (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject)

▷ `UnderlyingHomalgGradedRing(`*`A`*`)` (attribute)

Returns: a homalg graded ring

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

3.3.2 DegreeList (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject)

▷ `DegreeList(A)` (attribute)

Returns: a list

The argument is an object A in the category of projective graded left or right modules over a homalg graded ring R. The output is the degree_list of this object. To handle degree_lists most easily, degree_lists are redcued whenever an object is added to the category. E.g. the input degree_list [[d_1, 1], [d_1, 1]] will be turned into [[d_1, 2]].

3.3.3 RankOfObject (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject)

▷ `RankOfObject(A)` (attribute)

Returns: an integer

The argument is an object A in the category of projective graded left or right modules over a homalg graded ring R. The output is the rank of this module.

Chapter 4

Morphisms

4.1 GAP Categories

4.1.1 IsCAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism (for IsCapCategoryMorphism)

▷ `IsCAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism(`*`object`*`)` (filter)

Returns: `true` or `false`

The GAP category of morphisms of projective graded left or right modules over a graded ring *R*.

4.1.2 IsCAPCategoryOfProjectiveGradedLeftModulesMorphism (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism)

▷ `IsCAPCategoryOfProjectiveGradedLeftModulesMorphism(`*`object`*`)` (filter)

Returns: `true` or `false`

The GAP category of morphisms of projective graded left modules over a graded ring *R*.

4.1.3 IsCAPCategoryOfProjectiveGradedRightModulesMorphism (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism)

▷ `IsCAPCategoryOfProjectiveGradedRightModulesMorphism(`*`object`*`)` (filter)

Returns: `true` or `false`

The GAP category of morphisms of projective graded right modules over a graded ring *R*.

4.2 Constructors

4.2.1 CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject, IsHomalgMatrix,IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject)

▷ `CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism(`*`S, M, T`*`)` (operation)

Returns: a morphism in $\mathrm{Hom}(S,T)$

The arguments are an object *S* in the category of projective graded left or right modules over a homalg graded ring *R*,a homalg matrix *M* over *R*, and another object *T* in the category of projective graded left or right modules over R. The output is the morphism $S \to T$ in the category of projective graded left or right modules over R, whose underlying matrix is given by M.

4.2.2 CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject, IsHomalgMatrix,IsCAPCategoryOfProjectiveGradedLeftOrRightModulesObject, IsBool)

▷ `CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism(S, M, T)` (operation)

Returns: a morphism in $\mathrm{Hom}(S,T)$

As 'CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism' but does not perform checks on the input. Therefore this constructor is better suited for high performance applications.

4.3 Attributes

4.3.1 UnderlyingHomalgGradedRing (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism)

▷ `UnderlyingHomalgGradedRing(alpha)` (attribute)

Returns: a homalg graded ring

The argument is a morphism α in the category of projective graded left or right modules over a homalg graded ring R. The output is the homalg graded ring R.

4.3.2 UnderlyingHomalgMatrix (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism)

▷ `UnderlyingHomalgMatrix(alpha)` (attribute)

Returns: a matrix over a homalg graded ring

The argument is a morphism α in the category of projective graded left or right modules over a homalg graded ring R. The output is the underlying homalg matrix over R.

4.4 Printing all information about a morphism

4.4.1 FullInformation (for IsCAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism)

▷ `FullInformation(m)` (operation)

Returns: detailed information about the morphism

The argument is a morphism m in the category of projective graded modules. For such a morphisms it will take three command to print source, range and the mapping matrix. This method performs this task immediately. and prints all this information.

Chapter 5

Tools

5.1 Tools to simplify code

5.1.1 DeduceMapFromMatrixAndRangeLeft (for IsHomalgMatrix, IsCAPCategoryOfProjectiveGradedLeftModulesObject)

▷ `DeduceMapFromMatrixAndRangeLeft(m, R)` (operation)

Returns: a morphism

The argument is a homalg_matrix `m` and a left_module `R`. We then consider the module map induced from `m` with range `R`. This operation then deduces the source of this map and returns the map in the category of projective graded left-modules.

5.1.2 DeduceMapFromMatrixAndSourceLeft (for IsHomalgMatrix, IsCAPCategoryOfProjectiveGradedLeftModulesObject)

▷ `DeduceMapFromMatrixAndSourceLeft(m, S)` (operation)

Returns: a morphism

The argument is a homalg_matrix `m` and a left_module `S`. We then consider the module map induced from `m` with source `S`. This operation then deduces the range of this map and returns the map in the category of projective graded left-modules.

5.1.3 DeduceMapFromMatrixAndRangeRight (for IsHomalgMatrix, IsCAPCategoryOfProjectiveGradedRightModulesObject)

▷ `DeduceMapFromMatrixAndRangeRight(m, R)` (operation)

Returns: a morphism

The argument is a homalg_matrix `m` and a right_module `R`. We then consider the module map induced from `m` with range `R`. This operation then deduces the source of this map and returns the map in the category of projective graded right-modules.

5.1.4 DeduceMapFromMatrixAndSourceRight (for IsHomalgMatrix, IsCAPCategoryOfProjectiveGradedRightModulesObject)

▷ `DeduceMapFromMatrixAndSourceRight(m, S)` (operation)

Returns: a morphism

The argument is a homalg_matrix `m` and a left_module `S`. We then consider the module map induced from `m` with source `S`. This operation then deduces the range of this map and returns the map in the category of projective graded right-modules.

Chapter 6

Examples and Tests

6.1 Constructors of objects and reduction of degree lists

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> ObjectL := CAPCategoryOfProjectiveGradedLeftModulesObject( [ [[1,0],2] ], S );
<A projective graded left module of rank 2>
gap> DegreeList( ObjectL );
[ [ ( 1, 0 ), 2 ] ]
gap> Object2L := CAPCategoryOfProjectiveGradedLeftModulesObject( [ [[1,0],2],
>             [[1,0],3],[[0,1],2],[[1,0],1] ], S );
<A projective graded left module of rank 8>
gap> DegreeList( Object2L );
[ [ ( 1, 0 ), 5 ], [ ( 0, 1 ), 2 ], [ ( 1, 0 ), 1 ] ]
gap> UnzipDegreeList( Object2L );
[ ( 1, 0 ), ( 1, 0 ), ( 1, 0 ), ( 1, 0 ), ( 1, 0 ), ( 0, 1 ), ( 0, 1 ), ( 1, 0 ) ]
gap> ObjectR := CAPCategoryOfProjectiveGradedRightModulesObject( [ [[1,0],2] ], S );
<A projective graded right module of rank 2>
gap> DegreeList( ObjectR );
[ [ ( 1, 0 ), 2 ] ]
gap> Object2R := CAPCategoryOfProjectiveGradedRightModulesObject( [ [[1,0],2],
>             [[1,0],3],[[0,1],2],[[1,0],1] ], S );
<A projective graded right module of rank 8>
gap> DegreeList( Object2R );
[ [ ( 1, 0 ), 5 ], [ ( 0, 1 ), 2 ], [ ( 1, 0 ), 1 ] ]
gap> UnzipDegreeList( Object2R );
[ ( 1, 0 ), ( 1, 0 ), ( 1, 0 ), ( 1, 0 ), ( 1, 0 ), ( 0, 1 ), ( 0, 1 ), ( 1, 0 ) ]
```

Whenever the object constructor is called, it tried to simplify the given degree list. To this end it checks if subsequent degree group elements match. If so, their multiplicities are added. So, as in the example above we have:

$$[[[1,0],2],[[1,0],3],[[0,1],2],[[1,0],1]] \mapsto [[(1,0),5],[(0,1),2],[(1,0),1]]$$

Note that, even though there are two occurances of $(1,0)$ in the final degree list, which we do not simplify. The reason for this is as follows. Assume that we have a map of graded modules

$$\varphi : A \to B$$

given by a homomgeneous matrix M and that we want to compute the kernel embedding of this mapping. To this end we first compute the syzygies (of rows or columns, depending on whether we are dealing with right or left-modules) of M. Let us call the corresponding matrix N. Then we deduce the degree list of the kernel object from N and from the projective graded module A. Once this degree list is known, we would call the object constructor. If this object constructor summarised all (and not only subsequent) occurances of one degree element in the degree list, then in order to make sure that the kernel embedding is a mapping of graded modules, rows/columns of the matrix N would have to be shuffled. The latter we do not wish to perform. Nevertheless, the 'IsEqualForObjects' methods returns true whenever two projective modules are isomorphic. So in particular it returns true, if the degree lists are mere permutations of one another. Here is an example.

Example
```
gap> Object2LShuffle := CAPCategoryOfProjectiveGradedLeftModulesObject( [ [[0,1],1],
>            [[1,0],2],[[0,1],1],[[1,0],4] ], S );
<A projective graded left module of rank 8>
gap> IsEqualForObjects( Object2L, Object2LShuffle );
true
gap> Object2RShuffle := CAPCategoryOfProjectiveGradedRightModulesObject( [ [[0,1],1],
>            [[1,0],2],[[0,1],1],[[1,0],4] ], S );
<A projective graded right module of rank 8>
gap> IsEqualForObjects( Object2R, Object2RShuffle );
true
```

6.2 Constructors of morphisms

Example
```
gap> Q1L := CAPCategoryOfProjectiveGradedLeftModulesObject( [ [[1,0],1] ], S );
<A projective graded left module of rank 1>
gap> IsWellDefined( Q1L );
true
gap> Q2L := CAPCategoryOfProjectiveGradedLeftModulesObject( [ [[0,0],2] ], S );
<A projective graded left module of rank 2>
gap> m1L := CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism(
>       Q1L, HomalgMatrix( [["x_1","x_2"]], S ) ,Q2L );
<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> IsWellDefined( m1L );
true
gap> Display( Source( m1L ) );
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> Display( Range( m1L ) );
A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 2 and degrees:
[ [ 0, 2 ] ]
gap> Display( UnderlyingHomalgMatrix( m1L ) );
```

```
x_1,x_2
(over a graded ring)
gap> Q1R := CAPCategoryOfProjectiveGradedRightModulesObject( [ [[1,0],1] ], S );
<A projective graded right module of rank 1>
gap> IsWellDefined( Q1R );
true
gap> Q2R := CAPCategoryOfProjectiveGradedRightModulesObject( [ [[0,0],2] ], S );
<A projective graded right module of rank 2>
gap> m1R := CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism(
>       Q1R, HomalgMatrix( [["x_1"],["x_2"]], S ) ,Q2R );
<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> IsWellDefined( m1R );
true
gap> Display( Source( m1R ) );
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> Display( Range( m1R ) );
A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 2 and degrees:
[ [ 0, 2 ] ]
gap> Display( UnderlyingHomalgMatrix( m1R ) );
x_1,
x_2
(over a graded ring)
```

6.3 The GAP categories

Example

```
gap> categoryL := CapCategory( Q1L );
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> categoryR := CapCategory( Q1R );
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 ] ])
```

6.4 A few categorical constructions for projective left modules

Example

```
gap> ZeroObject( categoryL );
<A projective graded left module of rank 0>
gap> O1L := CAPCategoryOfProjectiveGradedLeftModulesObject( [ [[1,0],2] ], S );
<A projective graded left module of rank 2>
gap> Display( ZeroMorphism( ZeroObject( categoryL ), O1L ) );
A morphism in the category of projective graded left modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
(an empty 0 x 2 matrix)
gap> O2L := CAPCategoryOfProjectiveGradedLeftModulesObject( [ [[0,0],1] ], S );
<A projective graded left module of rank 1>
```

```
gap> Display( IdentityMorphism( O2L ) );
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)
gap> directSumL := DirectSum( [ O1L, O2L ] );
<A projective graded left module of rank 3>
gap> Display( directSumL );
A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 3 and degrees:
[ [ ( 1, 0 ), 2 ], [ 0, 1 ] ]
gap> i1L := InjectionOfCofactorOfDirectSum( [ O1L, O2L ], 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( i1L ) );
1,0,0,
0,1,0
(over a graded ring)
gap> i2L := InjectionOfCofactorOfDirectSum( [ O1L, O2L ], 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( i2L ) );
0,0,1
(over a graded ring)
gap> proj1L := ProjectionInFactorOfDirectSum( [ O1L, O2L ], 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( proj1L ) );
1,0,
0,1,
0,0
(over a graded ring)
gap> proj2L := ProjectionInFactorOfDirectSum( [ O1L, O2L ], 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( proj2L ) );
0,
0,
1
(over a graded ring)
gap> kL := KernelEmbedding( proj1L );
<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( kL ) );
0,0,1
(over a graded ring)
gap> ckL := CokernelProjection( kL );
<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( ckL ) );
1,0,
0,1,
0,0
(over a graded ring)
gap> IsMonomorphism( kL );
true
gap> IsEpimorphism( kL );
false
gap> IsMonomorphism( ckL );
false
gap> IsEpimorphism( ckL );
true
gap> m1L := CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism( O1L,
>       HomalgMatrix( [[ "x_1" ], [ "x_2" ]], S ), O2L );
<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 := IdentityMorphism( O2L );
<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> liftL := Lift( m1L, m2L );
<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( liftL ) );
x_1,
x_2
(over a graded ring)
gap> O3L := CAPCategoryOfProjectiveGradedLeftModulesObject( [ [[-1,0],2] ], S );
<A projective graded left module of rank 2>
gap> morL := CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism(
>        O2L, HomalgMatrix( [[ "x_1, x_2" ]], S ), O3L );
<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> coliftL := Colift( m2L, morL );
<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( coliftL ) );
x_1,x_2
(over a graded ring)
gap> fpL := FiberProduct( [ m1L, m2L, IdentityMorphism( Range( m2L ) ) ] );
<A projective graded left module of rank 2>
gap> fp_proj1L := ProjectionInFactorOfFiberProduct( [ m1L, m2L ], 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( fp_proj1L ) );
1,0,
0,1
(over a graded ring)
gap> fp_proj2L := ProjectionInFactorOfFiberProduct( [ m1L, m2L ], 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( fp_proj2L ) );
x_1,
x_2
```

```
(over a graded ring)
gap> poL := Pushout( morL, m2L );
<A projective graded left module of rank 2>
gap> inj1L := InjectionOfCofactorOfPushout( [ morL, m2L ], 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( inj1L ) );
1,0,
0,1
(over a graded ring)
gap> inj2L := InjectionOfCofactorOfPushout( [ morL, m2L ], 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( inj2L ) );
x_1,x_2
(over a graded ring)
gap> tensorProductL := TensorProductOnObjects( O1L, O2L );
<A projective graded left module of rank 2>
gap> Display( tensorProductL );
A projective graded left module over Q[x_1,x_2,x_3,x_4] (with weights
[ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 2 and degrees:
[ [ ( 1, 0 ), 2 ] ]
gap> tensorProductMorphismL := TensorProductOnMorphisms( m2L, morL );
<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( tensorProductMorphismL );
A morphism in the category of projective graded left modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
x_1,x_2
(over a graded ring)
gap> Display( DualOnObjects( TensorProductOnObjects( ObjectL, Object2L ) ) );
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 16 and degrees:
[ [ ( -2, 0 ), 5 ], [ ( -1, -1 ), 2 ], [ ( -2, 0 ), 6 ], [ ( -1, -1 ), 2 ],
[ ( -2, 0 ), 1 ] ]
gap> Display( DualOnMorphisms( m1L ) );
A morphism in the category of projective graded left modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
x_1,x_2
(over a graded ring)
gap> Display( EvaluationForDualWithGivenTensorProduct( TensorProductOnObjects(
>           DualOnObjects( ObjectL ), ObjectL ), ObjectL, TensorUnit( categoryL ) ) );
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,
0,
0,
1
(over a graded ring)
```

```
gap> Display( InternalHomOnObjects( ObjectL, ObjectL ) );
A projective graded left module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
of rank 4 and degrees:
[ [ 0, 4 ] ]
```

6.5 A few categorical constructions for projective right modules

Example

```
gap> ZeroObject( categoryR );
<A projective graded right module of rank 0>
gap> O1R := CAPCategoryOfProjectiveGradedRightModulesObject( [ [[1,0],2] ], S );
<A projective graded right module of rank 2>
gap> Display( ZeroMorphism( ZeroObject( categoryR ), O1R ) );
A morphism in the category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
with matrix:
(an empty 2 x 0 matrix)
gap> O2R := CAPCategoryOfProjectiveGradedRightModulesObject( [ [[0,0],1] ], S );
<A projective graded right module of rank 1>
gap> Display( IdentityMorphism( O2R ) );
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)
gap> directSumR := DirectSum( [ O1R, O2R ] );
<A projective graded right module of rank 3>
gap> Display( directSumR );
A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 3 and degrees:
[ [ ( 1, 0 ), 2 ], [ 0, 1 ] ]
gap> i1R := InjectionOfCofactorOfDirectSum( [ O1R, O2R ], 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( i1R ) );
1,0,
0,1,
0,0
(over a graded ring)
gap> i2R := InjectionOfCofactorOfDirectSum( [ O1R, O2R ], 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( i2R ) );
0,
0,
1
(over a graded ring)
gap> proj1R := ProjectionInFactorOfDirectSum( [ O1R, O2R ], 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( proj1R ) );
```

```
1,0,0,
0,1,0
(over a graded ring)
gap> proj2R := ProjectionInFactorOfDirectSum( [ O1R, O2R ], 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( proj2R ) );
0,0,1
(over a graded ring)
gap> kR := KernelEmbedding( proj1R );
<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( kR ) );
0,
0,
1
(over a graded ring)
gap> ckR := CokernelProjection( kR );
<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( ckR ) );
1,0,0,
0,1,0
(over a graded ring)
gap> IsMonomorphism( kR );
true
gap> IsEpimorphism( kR );
false
gap> IsMonomorphism( ckR );
false
gap> IsEpimorphism( ckR );
true
gap> m1R := CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism( O1R,
>       HomalgMatrix( [[ "x_1", "x_2" ]], S ), O2R );
<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 := IdentityMorphism( O2R );
<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> liftR := Lift( m1R, m2R );
<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( liftR ) );
x_1,x_2
(over a graded ring)
gap> O3R := CAPCategoryOfProjectiveGradedRightModulesObject( [ [[-1,0],2] ], S );
<A projective graded right module of rank 2>
gap> morR := CAPCategoryOfProjectiveGradedLeftOrRightModulesMorphism(
>        O2R, HomalgMatrix( [[ "x_1" ], [ "x_2" ]], S ), O3R );
<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> coliftR := Colift( m2R, morR );
```

```
<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( coliftR ) );
x_1,
x_2
(over a graded ring)
gap> fpR := FiberProduct( [ m1R, m2R, IdentityMorphism( Range( m2R ) ) ] );
<A projective graded right module of rank 2>
gap> fp_proj1R := ProjectionInFactorOfFiberProduct( [ m1R, m2R ], 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( fp_proj1R ) );
1,0,
0,1
(over a graded ring)
gap> fp_proj2R := ProjectionInFactorOfFiberProduct( [ m1R, m2R ], 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( fp_proj2R ) );
x_1, x_2
(over a graded ring)
gap> poR := Pushout( morR, m2R );
<A projective graded right module of rank 2>
gap> inj1R := InjectionOfCofactorOfPushout( [ morR, m2R ], 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( inj1R ) );
1,0,
0,1
(over a graded ring)
gap> inj2R := InjectionOfCofactorOfPushout( [ morR, m2R ], 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( inj2R ) );
x_1,
x_2
(over a graded ring)
gap> tensorProductR := TensorProductOnObjects( O1R, O2R );
<A projective graded right module of rank 2>
gap> Display( tensorProductR );
A projective graded right module over Q[x_1,x_2,x_3,x_4] (with weights
[ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ]) of rank 2 and degrees:
[ [ ( 1, 0 ), 2 ] ]
gap> tensorProductMorphismR := TensorProductOnMorphisms( m2R, morR );
<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( tensorProductMorphismR );
A morphism in the category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [0, 1 ] ])
with matrix:
x_1,
x_2
```

```
(over a graded ring)
gap> Display( DualOnObjects( TensorProductOnObjects( ObjectR, Object2R ) ) );
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 16 and degrees:
[ [ ( -2, 0 ), 5 ], [ ( -1, -1 ), 2 ], [ ( -2, 0 ), 6 ], [ ( -1, -1 ), 2 ],
[ ( -2, 0 ), 1 ] ]
gap> Display( DualOnMorphisms( m1R ) );
A morphism in the category of projective graded right modules over
Q[x_1,x_2,x_3,x_4] (with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
with matrix:
x_1,
x_2
(over a graded ring)
gap> Display( EvaluationForDualWithGivenTensorProduct( TensorProductOnObjects(
>           DualOnObjects( ObjectR ), ObjectR ), ObjectR, TensorUnit( categoryR ) ) );
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:
1,0,0,1
(over a graded ring)
gap> Display( InternalHomOnObjects( ObjectR, ObjectR ) );
A projective graded right module over Q[x_1,x_2,x_3,x_4]
(with weights [ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])
of rank 4 and degrees:
[ [ 0, 4 ] ]
```

Index