ToolsForFPGradedModules

A package to provide additional structures for toric varieties

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 ToolsForFPGradedModules package?

ToolsForFPGradedModules provides additional tools to perform computations or manipulate FPGradedModules, which are for structural reasons not part of the underlying package for *FreydCategories*.

Chapter 2

Tools for FPGradedModules

2.1 Ideals for CAP

2.1.1 LeftIdealForCAP (for IsList, IsHomalgGradedRing)

▷ `LeftIdealForCAP(L, R)` (operation)

Returns: a f.p. module presentation

The argument is a list L of generators of an ideal and a homalg graded ring R. This method then construct the left ideal in this ring generated by these generators.

2.1.2 RightIdealForCAP (for IsList, IsHomalgGradedRing)

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

Returns: a f.p. module presentation

The argument is a list L of generators of an ideal and a homalg graded ring R. This method then construct the right ideal in this ring generated by these generators.

2.2 Minimal free resolutions

2.2.1 MinimalFreeResolutionForCAP (for IsFpGradedLeftOrRightModulesObject)

▷ `MinimalFreeResolutionForCAP(M)` (attribute)

Returns: a complex of projective graded module morphisms

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

2.3 Betti tables

2.3.1 BettiTableForCAP (for IsFpGradedLeftOrRightModulesObject)

▷ `BettiTableForCAP(M)` (attribute)

Returns: a list of lists

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

2.4 Example: Ideal, minimal free resolution and Betti table

Example

```
gap> HOMALG_IO.show_banners := false;;
gap> HOMALG_IO.suppress_PID := true;;
gap> Q := HomalgFieldOfRationalsInSingular();
Q
gap> S := GradedRing( Q * "x_1, x_2, x_3" );
Q[x_1,x_2,x_3]
(weights: yet unset)
gap> SetWeightsOfIndeterminates( S, [[1],[1],[1]] );

gap> vars := IndeterminatesOfPolynomialRing( S );;
gap> IR := LeftIdealForCAP( [ vars[ 1 ], vars[ 2 ], vars[ 3 ] ], S );;
gap> IsWellDefined( IR );
true
gap> resolution := MinimalFreeResolutionForCAP( IR );
<An object in Complex category of Category of graded
rows over Q[x_1,x_2,x_3] (with weights [ 1, 1, 1 ])>
gap> FullInformation( resolution );
[ [ -1, 3 ] ]
 ^
 |
0,   -x_3,x_2,
-x_3,0,   x_1,
-x_2,x_1, 0
(over a graded ring)
 |
[ [ -2, 3 ] ]
 ^
 |
x_1,-x_2,x_3
(over a graded ring)
 |
[ [ -3, 1 ] ]

gap> IR_right := TurnIntoFpGradedRightModule( IR );;
gap> resolution_right := MinimalFreeResolutionForCAP( IR_right );
<An object in Complex category of Category of graded
columns over Q[x_1,x_2,x_3] (with weights [ 1, 1, 1 ])>
gap> differential_function :=
>                     UnderlyingZFunctorCell( resolution )!.differential_func;
function( i ) ... end
gap> IsWellDefined( differential_function( -1 ) );
true
gap> IsWellDefined( differential_function( -2 ) );
true
gap> IsWellDefined( differential_function( -3 ) );
true
gap> BT := BettiTableForCAP( IR );
[ [ -1, -1, -1 ], [ -2, -2, -2 ], [ -3 ] ]
```

Chapter 3

Conversion among f.p. graded modules

3.1 Turn CAP Graded Modules into old graded modules and vice versa

3.1.1 TurnIntoOldGradedModule (for IsFpGradedLeftOrRightModulesObject)

▷ `TurnIntoOldGradedModule(`*M*`)` (operation)

Returns: the corresponding graded modules in terms of the 'old' packages GradedModules

The argument is a graded left or right module presentation M for CAP

3.2 Save CAP f.p. graded module to file

3.2.1 SaveToFileAsOldGradedModule (for IsString, IsFpGradedLeftOrRightModulesObject)

▷ `SaveToFileAsOldGradedModule(`*M*`)` (operation)

Returns: true (in case of success) or raises error in case the file could not be written

The argument is a graded left or right module presentation M for CAP and saves this module to file as 'old' graded module presentation. By default, the files are saved in the main directory of the package 'SheafCohomologyOnToricVarieties'.

3.2.2 SaveToFileAsCAPGradedModule (for IsString, IsFpGradedLeftOrRightModulesObject)

▷ `SaveToFileAsCAPGradedModule(`*M*`)` (operation)

Returns: true (in case of success) or raises error in case the file could not be written

The argument is a graded left or right module presentation M for CAP and saves this module to file as CAP graded module presentation. By default, the files are saved in the main directory of the package 'SheafCohomologyOnToricVarieties'.

3.3 Turn left into right modules and vice versa

3.3.1 TurnIntoGradedColumn (for IsGradedRow)

▷ `TurnIntoGradedColumn(`*R*`)` (operation)

Returns: graded column

The argument is a graded row R. This method turns it into the corresponding graded column.

3.3.2 TurnIntoGradedRow (for IsGradedColumn)

▷ `TurnIntoGradedRow(C)` (operation)

Returns: graded row

The argument is a graded column C. This method turns it into the corresponding graded row.

3.3.3 TurnIntoGradedColumnMorphism (for IsGradedRowMorphism)

▷ `TurnIntoGradedColumnMorphism(C)` (operation)

Returns: graded columns morphism

The argument is a graded row morphism m. This method turns it into the corresponding morphism of graded columns.

3.3.4 TurnIntoGradedRowMorphism (for IsGradedColumnMorphism)

▷ `TurnIntoGradedRowMorphism(C)` (operation)

Returns: graded row morphism

The argument is a graded column morphism m. This method turns it into the corresponding morphism of graded rows.

3.3.5 TurnIntoFpGradedRightModule (for IsFpGradedLeftModulesObject)

▷ `TurnIntoFpGradedRightModule(M)` (operation)

Returns: f.p. graded right module

The argument is an f.p. graded left module M. This method turns it into the corresponding right module.

3.3.6 TurnIntoFpGradedLeftModule (for IsFpGradedRightModulesObject)

▷ `TurnIntoFpGradedLeftModule(M)` (operation)

Returns: f.p. graded left module

The argument is an f.p. graded right module M. This method turns it into the corresponding left module.

3.3.7 TurnIntoFpGradedRightModuleMorphism (for IsFpGradedLeftModulesMorphism)

▷ `TurnIntoFpGradedRightModuleMorphism(M)` (operation)

Returns: f.p. graded right module morphism

The argument is an f.p. graded left module morphism M. This method turns it into the corresponding right module morphism.

3.3.8 TurnIntoFpGradedLeftModuleMorphism (for IsFpGradedRightModulesMorphism)

▷ TurnIntoFpGradedLeftModuleMorphism(*M*) (operation)

Returns: f.p. graded left module morphism

The argument is an f.p. graded right module morphism M. This method turns it into the corresponding left module morphism.

3.4 Examples

3.4.1 Conversion of modules

We can turn the modules provided by the legendary GradedModules package into the ones provided by FreydCategories:

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> irP1xP1 := LeftIdealForCAP( [ vars[ 1 ] * vars[ 3 ], vars[ 1 ] * vars[ 4 ],
>                               vars[ 2 ] * vars[ 3 ], vars[ 2 ] * vars[ 4 ] ], S );;
gap> IsWellDefined( irP1xP1 );
true
gap> module2 := TurnIntoOldGradedModule( irP1xP1 );
<A graded left module presented by 4 relations for 4 generators>
gap> module3 := TurnIntoCAPGradedModule( module2 );
<An object in Category of f.p. graded left
modules over Q[x_1,x_2,x_3,x_4] (with weights
[ [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> module3 = irP1xP1;
true
```

We can also turn left into right modules:

Example

```
gap> graded_row := GradedRow( [ [[1,1],2],[[-1,0],1] ], S );;
gap> graded_col := TurnIntoGradedColumn( graded_row );;
gap> graded_row2 := TurnIntoGradedRow( graded_col );;
gap> IsEqualForObjects( graded_row, graded_row2 );
true
gap> irP1xP1_right := TurnIntoFpGradedRightModule( irP1xP1 );;
gap> TurnIntoOldGradedModule( irP1xP1_right );;
gap> irP1xP1_2 := TurnIntoFpGradedLeftModule( irP1xP1_right );;
gap> IsEqualForObjects( irP1xP1, irP1xP1_2 );
true
```

After long computations, we can also save modules to files.

Example

```
gap> SaveToFileAsOldGradedModule( "old_Ideal", irP1xP1 );;
gap> SaveToFileAsCAPGradedModule( "new_Ideal", irP1xP1 );;
```

These files are located in the package folder of "ToolsForFPGradedModules":

```
gap> name := Filename( DirectoriesPackageLibrary( "ToolsForFPGradedModules", "" )[ 1 ], "old_Idea
gap> IsExistingFile( name );
true
gap> RemoveFile( name );;
gap> name := Filename( DirectoriesPackageLibrary( "ToolsForFPGradedModules", "" )[ 1 ], "new_Idea
gap> IsExistingFile( name );
true
gap> RemoveFile( name );;
```

Likewise, we can turn morphisms of left modules into morphisms of right modules and vice versa:

```
gap> mor := RelationMorphism( irP1xP1 );;
gap> mor_right := TurnIntoGradedColumnMorphism( mor );;
gap> mor2 := TurnIntoGradedRowMorphism( mor_right );;
gap> IsEqualForMorphisms( mor, mor2 );
true
gap> k := WeakCokernelProjection( RelationMorphism( irP1xP1 ) );;
gap> range := AsFreydCategoryObject( Range( k ) );;
gap> fp_mor := FreydCategoryMorphism( irP1xP1, k, range );;
gap> fp_mor_right := TurnIntoFpGradedRightModuleMorphism( fp_mor );;
gap> fp_mor2 := TurnIntoFpGradedLeftModuleMorphism( fp_mor_right );;
gap> IsEqualForMorphisms( fp_mor, fp_mor2 );
true
```

Chapter 4

Overloaded functions

4.1 A simpler presentation for an f.p. graded module

4.1.1 ByASmallerPresentation (for IsFpGradedLeftOrRightModulesObject)

▷ `ByASmallerPresentation(M)` (operation)

The argument is an FPGradedMOdule. We then compute an equivalent yet simpler presentation for this module.

Index