

Sheaf Cohomology- On Toric Varieties

**A package to compute sheaf cohomology
on 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

This package may be distributed under the terms and conditions of the GNU Public License Version 2 or (at your option) any later version.

Contents

1	Introduction	4
1.1	What is the goal of the SheafCohomologyOnToricVarieties package?	4
2	Central functions and constants	5
2.1	Input check for cohomology computations	5
3	Cohomology of coherent sheaves from resolution	6
3.1	Deductions On Sheaf Cohomology From Cohomology Of projective modules in a minimal free resolution	6
3.2	Example: Pullback line bundle	6
4	Sheaf cohomology by use of https://arxiv.org/abs/1802.08860	9
4.1	Preliminaries	9
4.2	Computation of global sections	10
4.3	Computation of the i-th sheaf cohomologies	10
4.4	Computation of all sheaf cohomologies	10
4.5	Examples	11
5	Tools for cohomology computations	14
5.1	Approximation Of Sheaf Cohomologies	14
5.2	Examples	15
6	Sheaf cohomology with Spasm	17
6.1	Cohomology from Spasm and checks over the integers	17
6.2	Truncation to sparse matrices	17
6.3	Examples	18
7	Work in progress	19
7.1	Sheaf cohomology computations on (direct products of) projective spaces	19
7.2	Maps In Resolution	21
	Index	22

Chapter 1

Introduction

1.1 What is the goal of the `SheafCohomologyOnToricVarieties` package?

SheafCohomologyOnToricVarieties provides data structures to compute sheaf cohomology on such varieties. The ultimate goal is to provide high-performance-algorithms for its computation. To this, our main theorem for the computation of sheaf cohomology is based on an idea of Gregory G. Smith (see [math/0305214](#) and DOI: [10.4171/OWR/2013/25](#)), which we combine with the powerful `cohomCalg` algorithm. Information on the latter can be found at <https://arxiv.org/abs/1003.5217v3> and references therein.

Chapter 2

Central functions and constants

2.1 Input check for cohomology computations

2.1.1 IsValidInputForCohomologyComputations (for IsToricVariety)

▷ `IsValidInputForCohomologyComputations(V)` (property)

Returns: true or false

Returns if the given variety V is a valid input for cohomology computations. If the variable `SHEAF_COHOMOLOGY_ON_TORIC_VARIETIES_INTERNAL_LAZY` is set to false (default), then we just check if the variety is smooth, complete. In case of success we return true and false otherwise. If however `SHEAF_COHOMOLOGY_ON_TORIC_VARIETIES_INTERNAL_LAZY` is set to true, then we will check if the variety is smooth, complete or simplicial, projective. In case of success we return true and false other.

Chapter 3

Cohomology of coherent sheaves from resolution

3.1 Deductions On Sheaf Cohomology From Cohomology Of projective modules in a minimal free resolution

3.1.1 CohomologiesList (for IsToricVariety, IsFpGradedLeftOrRightModulesObject)

▷ CohomologiesList(*vari*, *M*) (operation)

Returns: a list of lists of integers

Given a smooth and projective toric variety *vari* with Coxring *S* and a f. p. graded *S*-modules *M*, this method computes a minimal free resolution of *M* and then the dimension of the cohomology classes of the projective modules in this minimal free resolution.

3.1.2 DeductionOfSheafCohomologyFromResolution (for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsBool)

▷ DeductionOfSheafCohomologyFromResolution(*vari*, *M*) (operation)

Returns: a list

Given a smooth and projective toric variety *vari* with Coxring *S* and a f. p. graded *S*-modules *M*, this method computes a minimal free resolution of *M* and then the dimension of the cohomology classes of the projective modules in this minimal free resolution. From this information we draw conclusions on the sheaf cohomologies of the sheaf \tilde{M} .

3.2 Example: Pullback line bundle

Example

```
gap> var := ProjectiveSpace( 2 );
<A projective toric variety of dimension 2>
gap> cox_ring := CoxRing( var );
Q[x_1,x_2,x_3]
(weights: [ 1, 1, 1 ])
gap> vars := IndeterminatesOfPolynomialRing( cox_ring );
[ x_1, x_2, x_3 ]
gap> range := GradedRow( [[2],1], cox_ring );
<A graded row of rank 1>
```

```

gap> source := GradedRow( [[1],1], cox_ring );
<A graded row of rank 1>
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> pullback_line_bundle := FreydCategoryObject( mor );
<An object in Category of f.p. graded left modules over
Q[x_1,x_2,x_3] (with weights
[ 1, 1, 1 ])>
gap> coh := DeductionOfSheafCohomologyFromResolution( var, pullback_line_bundle );
[ 3, 0, 0 ]

```

Example

```

gap> P1 := ProjectiveSpace( 1 );
<A projective toric variety of dimension 1>
gap> P2 := ProjectiveSpace( 2 );
<A projective toric variety of dimension 2>
gap> var2 := P1 * P1 * P2;
<A projective toric variety of dimension 4
which is a product of 3 toric varieties>
gap> cox_ring2 := CoxRing( var2 );
Q[x_1,x_2,x_3,x_4,x_5,x_6,x_7]
(weights: [ ( 0, 0, 1 ), ( 0, 1, 0 ), ( 1, 0, 0 ),
( 1, 0, 0 ), ( 1, 0, 0 ), ( 0, 1, 0 ), ( 0, 0, 1 ) ])
gap> vars2 := IndeterminatesOfPolynomialRing( cox_ring2 );
[ x_1, x_2, x_3, x_4, x_5, x_6, x_7 ]
gap> range2 := GradedRow( [[[1,1,2],1]], cox_ring2 );
<A graded row of rank 1>
gap> source2 := GradedRow( [[0,1,2],2]], cox_ring2 );
<A graded row of rank 2>
gap> matrix2 := HomalgMatrix( [[vars2[3]], [vars2[4]]], cox_ring2 );
<A 2 x 1 matrix over a graded ring>
gap> mor2 := GradedRowOrColumnMorphism( source2, matrix2, range2 );
<A morphism in Category of graded rows over
Q[x_1,x_2,x_3,x_4,x_5,x_6,x_7]
(with weights [ [ 0, 0, 1 ], [ 0, 1, 0 ], [ 1, 0, 0 ],
[ 1, 0, 0 ], [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ])>
gap> IsWellDefined( mor2 );
true
gap> pullback_line_bundle2 := FreydCategoryObject( mor2 );
<An object in Category of f.p. graded left modules over
Q[x_1,x_2,x_3,x_4,x_5,x_6,x_7] (with weights
[ [ 0, 0, 1 ], [ 0, 1, 0 ], [ 1, 0, 0 ], [ 1, 0, 0 ],
[ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ])>
gap> coh2 := DeductionOfSheafCohomologyFromResolution( var2, pullback_line_bundle2 );
[ 6, 0, 0, 0, 0 ]

```

Example

```

gap> P2 := ProjectiveSpace( 2 );

```

```

<A projective toric variety of dimension 2>
gap> var3 := P2 * P2;
<A projective toric variety of dimension 4
which is a product of 2 toric varieties>
gap> cox_ring3 := CoxRing( var3 );
Q[x_1,x_2,x_3,x_4,x_5,x_6]
(weights: [ ( 0, 1 ), ( 1, 0 ), ( 1, 0 ),
( 1, 0 ), ( 0, 1 ), ( 0, 1 ) ])
gap> range3 := GradedRow( [[1,1],4], cox_ring3 );
<A graded row of rank 4>
gap> source3 := ZeroObject( CapCategory( range3 ) );
<A graded row of rank 0>
gap> matrix3 := HomalgZeroMatrix( 0, 4, cox_ring3 );
<An unevaluated 0 x 4 zero matrix over a graded ring>
gap> mor3 := GradedRowOrColumnMorphism( source3, matrix3, range3 );
<A morphism in Category of graded rows over
Q[x_1,x_2,x_3,x_4,x_5,x_6] (with weights
[ [ 0, 1 ], [ 1, 0 ], [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> line_bundle3 := FreydCategoryObject( mor3 );
<An object in Category of f.p. graded left modules over
Q[x_1,x_2,x_3,x_4,x_5,x_6] (with weights
[ [ 0, 1 ], [ 1, 0 ], [ 1, 0 ], [ 1, 0 ], [ 0, 1 ], [ 0, 1 ] ])>
gap> IsWellDefined( line_bundle3 );
true
gap> coh3 := DeductionOfSheafCohomologyFromResolution( var3, line_bundle3 );
[ 36, 0, 0, 0, 0 ]

```


Chapter 4

Sheaf cohomology by use of <https://arxiv.org/abs/1802.08860>

4.1 Preliminaries

4.1.1 ParameterCheck (for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsFpGradedLeftOrRightModulesObject, IsInt)

▷ ParameterCheck(V, M_1, M_2, i) (operation)

Returns: true or false

Given a toric variety V , we eventually wish to compute the i -th sheaf cohomology of the sheafification of the f.p. graded S -module M_2 (S being the Cox ring of vari). To this end we use modules M_1 which sheafify to the structure sheaf of vari . This method tests if the truncation to degree zero of $\text{Ext}_S^i(M_1, M_2)$ is isomorphic to $H^i(V, \widetilde{M}_2)$.

4.1.2 FindLeftIdeal (for IsToricVariety, IsFpGradedLeftModulesObject, IsInt)

▷ FindLeftIdeal(V, M, i) (operation)

Returns: a list

Given a toric variety V and an f.p. graded S -module M (S being the Cox ring of vari), we wish to compute the i -th sheaf cohomology of \widetilde{M} . To this end, this method identifies an ideal I of S such that \widetilde{I} is the structure sheaf of V and such that $\text{Ext}_S^i(I, M)$ is isomorphic to $H^i(V, \widetilde{M})$. We identify I by determining an ample degree $d \in \text{Cl}(V)$. Then, for a suitable non-negative integer e , the generators of I are the e -th power of all monomials of degree d in the Cox ring of S . We return the list $[e, d, I]$.

4.1.3 FindRightIdeal (for IsToricVariety, IsFpGradedRightModulesObject, IsInt)

▷ FindRightIdeal($\text{arg1}, \text{arg2}, \text{arg3}$) (operation)

4.2 Computation of global sections

4.2.1 H^0 (for IsToricVariety, IsFpGradedLeftOrRightModulesObject)

▷ $H^0(V, M)$ (operation)

Returns: a vector space

Given a variety V and an f.p. graded S -module M (S being the Cox ring of V), this method computes $H^0(V, \tilde{M})$.

4.2.2 H^0 Parallel (for IsToricVariety, IsFpGradedLeftOrRightModulesObject)

▷ $H^0\text{Parallel}(V, M)$ (operation)

Returns: a vector space

Given a variety V and an f.p. graded S -module M (S being the Cox ring of V), this method computes $H^0(V, \tilde{M})$. This method is parallelized and is thus best suited for long and complicated computations.

4.3 Computation of the i -th sheaf cohomologies

4.3.1 H^i (for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsInt)

▷ $H^i(V, M, i)$ (operation)

Returns: a vector space

Given a variety V and an f.p. graded S -module M (S being the Cox ring of V), this method computes $H^i(V, \tilde{M})$.

4.3.2 H^i Parallel (for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsInt)

▷ $H^i\text{Parallel}(V, M, i)$ (operation)

Returns: a vector space

Given a variety V and an f.p. graded S -module M (S being the Cox ring of V), this method computes $H^i(V, \tilde{M})$. This method is parallelized and is thus best suited for long and complicated computations.

4.4 Computation of all sheaf cohomologies

4.4.1 AllHi (for IsToricVariety, IsFpGradedLeftOrRightModulesObject)

▷ $\text{AllHi}(V, M)$ (operation)

Returns: a list of vector spaces

Given a variety V and an f.p. graded S -module M (S being the Cox ring of V), this method computes all sheaf cohomologies $H^*(V, \tilde{M})$.

4.4.2 AllHiParallel (for IsToricVariety, IsFpGradedLeftOrRightModulesObject)

▷ $\text{AllHiParallel}(V, M)$ (operation)

Returns: a list of vector spaces

Given a variety V and an f.p. graded S -module M (S being the Cox ring of V), this method computes all sheaf cohomologies $H^*(V, \tilde{M})$. This method is parallelized and is thus best suited for long and complicated computations.

4.5 Examples

4.5.1 Sheaf cohomology of toric vector bundles

Example

```
gap> F1 := Fan( [[1],[-1]], [[1],[2]] );
<A fan in |R^1>
gap> P1 := ToricVariety( F1 );
<A toric variety of dimension 1>
gap> P1xP1 := P1 * P1;
<A toric variety of dimension 2 which is a product of 2 toric varieties>
gap> VForCAP := AsFreydCategoryObject( GradedRow( [[1,1],1],[[-2,0],1]],
>                                         CoxRing( P1xP1 ) ) );
<A projective object in Category of f.p. graded
left modules over Q[x_1,x_2,x_3,x_4] (with weights
[ [ 0, 1 ], [ 1, 0 ], [ 1, 0 ], [ 0, 1 ] ])>
gap> V2ForCAP := AsFreydCategoryObject( GradedRow( [[-2,0],1]],
>                                         CoxRing( P1xP1 ) ) );
<A projective object in Category of f.p. graded
left modules over Q[x_1,x_2,x_3,x_4] (with weights
[ [ 0, 1 ], [ 1, 0 ], [ 1, 0 ], [ 0, 1 ] ])>
gap> AllHi( P1xP1, VForCAP, false, false );
Computing h^0
-----

Computing h^1
-----

Computing h^2
-----

[ [ 0, <A vector space object over Q of dimension 4> ],
  [ 1, <A vector space object over Q of dimension 1> ],
  [ 1, <A vector space object over Q of dimension 0> ] ]
gap> AllHiParallel( P1xP1, VForCAP, false, false );
Computing h^0
-----

Computing h^1
-----

Computing h^2
-----

[ [ 0, <A vector space object over Q of dimension 4> ],
  [ 1, <A vector space object over Q of dimension 1> ],
  [ 1, <A vector space object over Q of dimension 0> ] ]
gap> AllHi( P1xP1, V2ForCAP, false, false );
```

```

Computing h^0
-----

Computing h^1
-----

Computing h^2
-----

[ [ 0, <A vector space object over Q of dimension 0> ],
  [ 1, <A vector space object over Q of dimension 1> ],
  [ 1, <A vector space object over Q of dimension 0> ] ]
gap> AllHiParallel( P1xP1, V2ForCAP, false, false );
Computing h^0
-----

Computing h^1
-----

Computing h^2
-----

[ [ 0, <A vector space object over Q of dimension 0> ],
  [ 1, <A vector space object over Q of dimension 1> ],
  [ 1, <A vector space object over Q of dimension 0> ] ]

```

4.5.2 Sheaf cohomologies of the irrelevant ideal of $P1xP1$

Example

```

gap> irP1xP1 := IrrelevantLeftIdealForCAP( P1xP1 );
<An object in Category of f.p. graded left
modules over Q[x_1,x_2,x_3,x_4] (with weights
[ [ 0, 1 ], [ 1, 0 ], [ 1, 0 ], [ 0, 1 ] ])>
gap> AllHi( P1xP1, irP1xP1, false, false );
Computing h^0
-----

Computing h^1
-----

Computing h^2
-----

[ [ 1, <A vector space object over Q of dimension 1> ],
  [ 1, <A vector space object over Q of dimension 0> ],
  [ 0, <A vector space object over Q of dimension 0> ] ]
gap> AllHiParallel( P1xP1, irP1xP1, false, false );
Computing h^0
-----

Computing h^1
-----

```

Computing h^2

```
[ [ 1, <A vector space object over Q of dimension 1> ],  
  [ 1, <A vector space object over Q of dimension 0> ],  
  [ 0, <A vector space object over Q of dimension 0> ] ]
```

Chapter 5

Tools for cohomology computations

5.1 Approximation Of Sheaf Cohomologies

5.1.1 BPowerLeft (for IsToricVariety, IsInt)

▷ `BPowerLeft(V, e)` (operation)

Returns: a CAP graded left module

The argument is a toric variety V and a non-negative integer e . The method computes the e -th Frobenius power of the irrelevant left ideal of V .

5.1.2 BPowerRight (for IsToricVariety, IsInt)

▷ `BPowerRight(V, e)` (operation)

Returns: a CAP graded right module

The argument is a toric variety V and a non-negative integer e . The method computes the e -th Frobenius power of the irrelevant right ideal of V .

5.1.3 ApproxH0 (for IsToricVariety, IsInt, IsFpGradedLeftOrRightModulesObject)

▷ `ApproxH0(V, e, M)` (operation)

Returns: a non-negative integer

The argument is a toric variety V , a non-negative integer e and a graded CAP module M . The method computes the degree zero layer of $\text{Hom}(B(e), M)$ and returns its vector space dimension.

5.1.4 ApproxH0Parallel (for IsToricVariety, IsInt, IsFpGradedLeftOrRightModulesObject)

▷ `ApproxH0Parallel(V, e, M)` (operation)

Returns: a non-negative integer

The argument is a toric variety V , a non-negative integer e and a graded CAP module M . The method computes the degree zero layer of $\text{Hom}(B(e), M)$ by use of parallelisation and returns its vector space dimension.

5.1.5 ApproxHi (for IsToricVariety, IsInt, IsInt, IsFpGradedLeftOrRightModulesObject)

▷ `ApproxHi(V, i, e, M)` (operation)

Returns: a non-negative integer

The argument is a toric variety V , non-negative integers i, e and a graded CAP module M . The method computes the degree zero layer of $\text{Ext}^i(B(e), M)$ and returns its vector space dimension.

5.1.6 ApproxHiParallel (for IsToricVariety, IsInt, IsInt, IsFpGradedLeftOrRightModulesObject)

▷ `ApproxHiParallel(V, i, e, M)` (operation)

Returns: a non-negative integer

The argument is a toric variety V , non-negative integer i, e and a graded CAP module M . The method computes the degree zero layer of $\text{Ext}^i(B(e), M)$ by use of parallelisation and returns its vector space dimension.

5.2 Examples

5.2.1 Approximation of 0-th sheaf cohomology

Example

```
gap> ApproxH0( P1xP1, 0, irP1xP1 );
<A vector space object over Q of dimension 0>
gap> ApproxH0( P1xP1, 1, irP1xP1 );
<A vector space object over Q of dimension 1>
gap> ApproxH0( P1xP1, 2, irP1xP1 );
<A vector space object over Q of dimension 1>
gap> ApproxH0Parallel( P1xP1, 0, irP1xP1 );
<A vector space object over Q of dimension 0>
gap> ApproxH0Parallel( P1xP1, 1, irP1xP1 );
<A vector space object over Q of dimension 1>
gap> ApproxH0Parallel( P1xP1, 2, irP1xP1 );
<A vector space object over Q of dimension 1>
```

5.2.2 Approximation of 1-st sheaf cohomology

Example

```
gap> F1 := Fan( [[1],[-1]], [[1],[2]] );
<A fan in |R^1>
gap> P1 := ToricVariety( F1 );
<A toric variety of dimension 1>
gap> P1xP1 := P1 * P1;
<A toric variety of dimension 2 which is a product of 2 toric varieties>
gap> VForCAP := AsFreydCategoryObject( GradedRow( [[[1,1],1],[[-2,0],1]],
> CoxRing( P1xP1 ) ) );
<A projective object in Category of f.p. graded
left modules over Q[x_1,x_2,x_3,x_4] (with weights
[ [ 0, 1 ], [ 1, 0 ], [ 1, 0 ], [ 0, 1 ] ])>
gap> ApproxHi( P1xP1, 1, 0, VForCAP );
<A vector space object over Q of dimension 0>
gap> ApproxHi( P1xP1, 1, 1, VForCAP );
```

```
<A vector space object over Q of dimension 1>
gap> ApproxHi( P1xP1, 1, 2, VForCAP );
<A vector space object over Q of dimension 1>
gap> ApproxHiParallel( P1xP1, 1, 0, VForCAP );
<A vector space object over Q of dimension 0>
gap> ApproxHiParallel( P1xP1, 1, 1, VForCAP );
<A vector space object over Q of dimension 1>
gap> ApproxHiParallel( P1xP1, 1, 2, VForCAP );
<A vector space object over Q of dimension 1>
```


Chapter 6

Sheaf cohomology with Spasm

6.1 Cohomology from Spasm and checks over the integers

6.1.1 H0ParallelBySpasm (for IsToricVariety, IsFpGradedLeftOrRightModulesObject)

▷ H0ParallelBySpasm($V, M, \text{prime}, p, \text{boolean}, b$) (operation)

Returns: a vector space

Given a variety V and an f.p. graded S -module M (S being the Cox ring of V), this method computes $H^0(V, \tilde{M})$. It uses a combination of Singular and Spasm to perform this task. The latter operates in a finite field. By default we use the field modulo 42013. However, a prime can be specified as third argument to overwrite this choice. In addition, the boolean 'false' can be specified as fourth argument to suppress output during the computation.

6.1.2 H0ParallelBySpasmAndCheckOverIntegers (for IsToricVariety, IsFpGradedLeftOrRightModulesObject)

▷ H0ParallelBySpasmAndCheckOverIntegers($V, M, \text{prime}, p, \text{boolean}, b$) (operation)

Returns: a vector space

Just as the previous method. However, for every kernel computation by Spasm, it triggers a rank computation with Singular to cross-check these results over the integers.

6.2 Truncation to sparse matrices

6.2.1 TruncateFPGradedModuleMorphismToZeroInParallelToSparseMatrices (for IsToricVariety, IsFpGradedLeftOrRightModulesMorphism, IsBool)

▷ TruncateFPGradedModuleMorphismToZeroInParallelToSparseMatrices(V, α, b) (operation)

Returns: a list of 3 sparse matrices

Truncates a morphism α of f.p. graded left or right modules over the Cox ring of a toric variety V and returns a list of 3 sparse matrices. The boolean b at third position specifies whether information is printed whilst running this computation.

6.3 Examples

Example

```
gap> F1 := Fan( [[1],[-1]],[[1],[2]] );
<A fan in |R^1>
gap> P1 := ToricVariety( F1 );
<A toric variety of dimension 1>
gap> P1xP1 := P1 * P1;
<A toric variety of dimension 2 which is a product of 2 toric varieties>
gap> irP1xP1 := IrrelevantLeftIdealForCAP( P1xP1 );
<An object in Category of f.p. graded left
modules over Q[x_1,x_2,x_3,x_4] (with weights
[ [ 0, 1 ], [ 1, 0 ], [ 1, 0 ], [ 0, 1 ] ])>
gap> h0 := H0ParallelBySpasm( P1xP1, irP1xP1, 42013, false, true )[ 3 ];
1
gap> irP1xP1 := IrrelevantRightIdealForCAP( P1xP1 );
<An object in Category of f.p. graded right
modules over Q[x_1,x_2,x_3,x_4] (with weights
[ [ 0, 1 ], [ 1, 0 ], [ 1, 0 ], [ 0, 1 ] ])>
gap> h0 := H0ParallelBySpasm( P1xP1, irP1xP1, 42013, false, true )[ 3 ];
1
```

We provide more convenient methods calls. Namely, `H0ParallelBySpasmAndCheckOverIntegers(P1xP1, irP1xP1);` is short for `H0ParallelBySpasm(P1xP1, irP1xP1, 42013, true, true)`. The fourth argument being true, this will always display information on the status of the computation. Since the fifth argument is true, this method will perform checks on the rank of the involved matrices over the integers. Recall that `Spasm` only operates over a finite field. We also provide the method `H0ParallelBySpasm(P1xP1, irP1xP1)`. It calls `H0ParallelBySpasm(P1xP1, irP1xP1, 42013, true, true)`. Consequently, this method operates similar to the previous one. However, in contrast, it will not perform checks over the integers.

Chapter 7

Work in progress

The following functionality is not yet supported, but work in progress.

7.1 Sheaf cohomology computations on (direct products of) projective spaces

7.1.1 H0OnProjectiveSpaceViaLinearRegularity (for IsToricVariety, IsGradedModuleOrGradedSubmoduleRep)

▷ `H0OnProjectiveSpaceViaLinearRegularity(vari, M)` (operation)

Returns: an integer

Given that variety is a complex projective space with Coxring S and M a f. p. graded S -module, this method computes the dimension of the vector space $H^0(X_\Sigma, \tilde{M})$ and returns this integer. To achieve this an integer e is computed such that $H^0(\tilde{M}) \cong H_S^0(\mathfrak{m}^e, \tilde{M})_0$. This integer e in turn is computed by use of linear regularity of the module M .

7.1.2 H0OnProjectiveSpaceForAllPositiveTwistsViaLinearRegularity (for IsToricVariety, IsGradedModuleOrGradedSubmoduleRep)

▷ `H0OnProjectiveSpaceForAllPositiveTwistsViaLinearRegularity(vari, M)` (operation)

Returns: a function

Given that variety is a complex projective space with Coxring S and M a f. p. graded S -module, this method computes a function $f: \mathbb{N}_{\geq 0} \rightarrow \mathbb{N}_{\geq 0}$ such that $f(n) = \dim(H^0(X_\Sigma, \tilde{M}(n)))$. This function is returned.

7.1.3 H0OnProjectiveSpaceInRangeViaLinearRegularity (for IsToricVariety, IsGradedModuleOrGradedSubmoduleRep, IsList)

▷ `H0OnProjectiveSpaceInRangeViaLinearRegularity(vari, M, range)` (operation)

Returns: a list of non-negative integers

Given that variety is a complex projective space with Coxring S and M a f. p. graded S -module, this method uses the function $f: \mathbb{N}_{\geq 0} \rightarrow \mathbb{N}_{\geq 0}$ computed by `H0OnCPNForAllTwistsViaLinReg` and evaluates it for all integers in range. The resulting list of non-negative integers is returned.

7.1.4 H0OnDirectProductsOfProjectiveSpaces (for IsToricVariety, IsGradedModule-OrGradedSubmoduleRep, IsBool)

▷ `H0OnDirectProductsOfProjectiveSpaces(vari, M, b)` (operation)

Returns: a vector space

Given that variety `vari` is a direct product of complex projective spaces with Coxring S and that M a f. p. graded S -module, this is an experimental method to compute sheaf cohomology. Note that all coherent sheaves on direct products of projective spaces have finite dimensional vector spaces as their cohomology groups. Thus this method returns such a vector space. A boolean b can be specified as third argument. If $b = \text{true}$ we saturate the module before the core procedure is applied. The default value is 'false', i.e. do not saturate the module.

7.1.5 Examples for graded modules on P1 and P1xP1

Example

```
gap> F1 := Fan( [[1],[-1]], [[1],[2]] );
<A fan in |R^1>
gap> P1 := ToricVariety( F1 );
<A toric variety of dimension 1>
gap> ByASmallerPresentation( ClassGroup( P1 ) );
<A free left module of rank 1 on a free generator>
gap> CoxRing( P1 );
Q[x_1,x_2]
(weights: [ 1, 1 ])
gap> C := GradedLeftSubmodule( [ "x_1", "x_2" ], CoxRing( P1 ) );
<A graded torsion-free (left) ideal given by 2 generators>
gap> P1xP1 := P1 * P1;
<A toric variety of dimension 2 which is a product of 2 toric varieties>
gap> ByASmallerPresentation( ClassGroup( P1xP1 ) );
<A free left module of rank 2 on free generators>
gap> C2 := GradedLeftSubmoduleForCAP( [ "x_2*x_4" ], [ "x_2*x_3" ], [ "x_1 * x_4" ],
> [ "x_1 * x_3" ], CoxRing( P1xP1 ) );
<A graded left ideal of Q[x_1,x_2,x_3,x_4]
(with weights [ [ 0, 1 ], [ 1, 0 ], [ 1, 0 ], [ 0, 1 ] ])>
gap> V := CAPCategoryOfProjectiveGradedLeftModulesObject(
> [[[-1,-1],1],[[2,0],1]], CoxRing( P1xP1 ) );
<A projective graded left module of rank 2>
gap> V := ApplyFunctor( EmbeddingOfProjCategory( CapCategory( V ) ), V );
<A graded left module presentation over the ring Q[x_1,x_2,x_3,x_4]
(with weights [ [ 0, 1 ], [ 1, 0 ], [ 1, 0 ], [ 0, 1 ] ])>
gap> VPrime := CAPCategoryOfProjectiveGradedLeftModulesObject( [[[-1,-1],1]], CoxRing( P1xP1 ) );
<A projective graded left module of rank 1>
gap> VPrime := ApplyFunctor( EmbeddingOfProjCategory( CapCategory( VPrime ) ), VPrime );
<A graded left module presentation over the ring Q[x_1,x_2,x_3,x_4]
(with weights [ [ 0, 1 ], [ 1, 0 ], [ 1, 0 ], [ 0, 1 ] ])>
```

7.1.6 Examples for cohomology computations on P1

Note that these computations still rely on the GradedModules package and thus do not yet use CAP. This will hopefully change soon.

Example

```
gap> H0OnProjectiveSpaceViaLinearRegularity( P1, C );
1
gap> H0OnProjectiveSpaceInRangeViaLinearRegularity( P1, C, [ 0 .. 5 ] );
[ [ 0, 1 ], [ 1, 2 ], [ 2, 3 ], [ 3, 4 ], [ 4, 5 ], [ 5, 6 ] ]
```

7.2 Maps In Resolution

7.2.1 MapsInResolution (for IsToricVariety, IsGradedLeftOrRightModulePresentationForCAP)

▷ `MapsInResolution(V , M)` (operation)

Returns: a list of vector space morphisms

The arguments are a toric variety V and a f.p. graded module presentation M over the Cox ring of this variety.

7.2.2 InducedCohomologyMaps (for IsToricVariety, IsCAPCategoryOfProjectiveGradedLeftModulesMorphism)

▷ `InducedCohomologyMaps(V , m)` (operation)

Returns: the list of induced vector space morphisms between the sheaf cohomologies of the associated vector bundles

The arguments are a toric variety V and a map of projective graded module presentation m over the Cox ring of this variety.

Index

- AllHi
 - for IsToricVariety, IsFpGradedLeftOrRightModulesObject, [10](#)
- AllHiParallel
 - for IsToricVariety, IsFpGradedLeftOrRightModulesObject, [10](#)
- ApproxH0
 - for IsToricVariety, IsInt, IsFpGradedLeftOrRightModulesObject, [14](#)
- ApproxH0Parallel
 - for IsToricVariety, IsInt, IsFpGradedLeftOrRightModulesObject, [14](#)
- ApproxHi
 - for IsToricVariety, IsInt, IsInt, IsFpGradedLeftOrRightModulesObject, [15](#)
- ApproxHiParallel
 - for IsToricVariety, IsInt, IsInt, IsFpGradedLeftOrRightModulesObject, [15](#)
- BPowerLeft
 - for IsToricVariety, IsInt, [14](#)
- BPowerRight
 - for IsToricVariety, IsInt, [14](#)
- CohomologiesList
 - for IsToricVariety, IsFpGradedLeftOrRightModulesObject, [6](#)
- DeductionOfSheafCohomologyFromResolution
 - for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsBool, [6](#)
- FindLeftIdeal
 - for IsToricVariety, IsFpGradedLeftModuleObject, IsInt, [9](#)
- FindRightIdeal
 - for IsToricVariety, IsFpGradedRightModuleObject, IsInt, [9](#)
- H0
 - for IsToricVariety, IsFpGradedLeftOrRightModulesObject, [10](#)
- H0OnDirectProductsOfProjectiveSpaces
 - for IsToricVariety, IsGradedModuleOrGradedSubmoduleRep, IsBool, [20](#)
- H0OnProjectiveSpaceForAllPositiveTwistsViaLinearRegularity
 - for IsToricVariety, IsGradedModuleOrGradedSubmoduleRep, [19](#)
- H0OnProjectiveSpaceInRangeViaLinearRegularity
 - for IsToricVariety, IsGradedModuleOrGradedSubmoduleRep, IsList, [19](#)
- H0OnProjectiveSpaceViaLinearRegularity
 - for IsToricVariety, IsGradedModuleOrGradedSubmoduleRep, [19](#)
- H0Parallel
 - for IsToricVariety, IsFpGradedLeftOrRightModulesObject, [10](#)
- H0ParallelBySpasm
 - for IsToricVariety, IsFpGradedLeftOrRightModulesObject, [17](#)
- H0ParallelBySpasmAndCheckOverIntegers
 - for IsToricVariety, IsFpGradedLeftOrRightModulesObject, [17](#)
- Hi
 - for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsInt, [10](#)
- HiParallel
 - for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsInt, [10](#)
- InducedCohomologyMaps
 - for IsToricVariety, IsCAPCategoryOfProjectiveGradedLeftModulesMorphism, [21](#)
- IsValidInputForCohomologyComputations
 - for IsToricVariety, [5](#)

MapsInResolution

for IsToricVariety, IsGradedLeftOrRight-
ModulePresentationForCAP, [21](#)

ParameterCheck

for IsToricVariety, IsFpGradedLeftOrRight-
ModulesObject, IsFpGradedLeftOr-
RightModulesObject, IsInt, [9](#)

TruncateFPGradedModuleMorphismToZero-
InParallelToSparseMatrices

for IsToricVariety, IsFpGradedLeftOrRight-
ModulesMorphism, IsBool, [17](#)