SheafCohomology-OnToricVarieties

A package to compute sheaf cohomology on toric varieties

2020.01.20

20 January 2020

Martin Bies

Martin Bies
Email: martin.bies@alumni.uni-heidelberg.de
Homepage: https://martinbies.github.io/
Address: Mathematical Institute
University of Oxford
Andrew Wiles Building
Radcliffe Observatory Quarter
Woodstock Road
Oxford OX2 6GG
United Kingdom

Contents

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, M1, M2, 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 $Ext^i_S(M_1, M_2)$ 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 $\tilde{M}$. To this end, this method identifies an ideal I of S such that $\tilde{I}$ is the structure sheaf of V and such that $Ext^i_S(I, M)$ is isomorphic to $H^i(V, \tilde{M})$. We identify I by determining an ample degree $d \in \mathrm{Cl}(V)$. 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(arg1, arg2, arg3)` (operation)

4.2 Computation of global sections

4.2.1 H0 (for IsToricVariety, IsFpGradedLeftOrRightModulesObject)

▷ `H0(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 H0Parallel (for IsToricVariety, IsFpGradedLeftOrRightModulesObject)

▷ `H0Parallel(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 Hi (for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsInt)

▷ `Hi(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 HiParallel (for IsToricVariety, IsFpGradedLeftOrRightModulesObject, IsInt)

▷ `HiParallel(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)

▷ `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)

▷ `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 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 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 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 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 computations (https://arxiv.org/abs/1802.08860) with Spasm

6.1 Cohomology from Spasm and Singular

6.1.1 H0ParallelBySpasm (for IsToricVariety, IsFpGradedLeftOrRightModulesObject)

▷ H0ParallelBySpasm(*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 $H^0(V, \tilde{M})$. 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.

6.2 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> H0ParallelBySpasm( P1xP1, irP1xP1 );;
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> H0ParallelBySpasm( P1xP1, irP1xP1 );;
```

Index