LocalizeRing-ForHomalg

A Package for Localization of Polynomial Rings

2022.11-01

23 November 2022

Mohamed Barakat

Markus Lange-Hegermann

Vinay Wagh

Mohamed Barakat
Email: mohamed.barakat@uni-siegen.de
Homepage: https://mohamed-barakat.github.io
Address: Walter-Flex-Str. 3
57072 Siegen
Germany

Markus Lange-Hegermann
Email: markus.lange-hegermann@hs-owl.de
Homepage: https://www.th-owl.de/eecs/fachbereich/team/markus-lange-hegermann/
Address: Markus Lange-Hegermann
Hochschule Ostwestfalen-Lippe
Liebigstraße 87
32657 Lemgo
Germany

Vinay Wagh
Email: waghoba@gmail.com
Homepage: http://www.iitg.ernet.in/vinay.wagh/
Address: E-102, Department of Mathematics,
Indian Institute of Technology Guwahati,
Guwahati, Assam, India.
PIN: 781 039.
India

Copyright

Contents

1 Introduction **4**
- 1.1 What is the Role of the LocalizeRingForHomalg Package in the homalg Project? . 4
- 1.2 Functionality 4
- 1.3 The Math Behind This Package 4
- 1.4 Which Ring to Use? 4

2 Installation of the LocalizeRingForHomalg Package **6**

3 Quick Start **7**
- 3.1 Localization of $\mathbb{Z}$ 7

4 Localize Rings **9**
- 4.1 Category and Representations 9
- 4.2 Rings: Attributes 10
- 4.3 Operations and Functions 10

5 Examples **14**
- 5.1 An Easy Polynomial Example 14
- 5.2 Hom(Hom(-,Z128),Z16) 15
- 5.3 ResidueClass 16
- 5.4 Testing the Intersection Formula 19

A Overview of the LocalizeRingForHomalg Package Source Code **21**
- A.1 The generic Methods 21
- A.2 The Local Decide Zero trick 23
- A.3 Tools 24

References **26**

Index **27**

Chapter 1

Introduction

1.1 What is the Role of the LocalizeRingForHomalg Package in the homalg Project?

The homalg project [hom22] aims at providing a general and abstract framework for homological computations. The package LocalizeRingForHomalg enables the homalg project to construct localizations from commutative rings in homalg at their maximal ideals.

1.2 Functionality

The package LocalizeRingForHomalg on the one hand builds on the package MatricesForHomalg and on the other hands adds functionality to MatricesForHomalg. It uses the computability (i.e. capability to solve linear systems) of a commutative ring R declared in MatricesForHomalg to construct the localization R_m of R at a maximal ideal m (given by a finite set of generators). This localized ring R_m is again computable and can thus be used by MatricesForHomalg.

Furthermore, via the package RingsForHomalg, an interface to Singular is used to compute in localized polynomial rings with the help of Mora's algorithm.

1.3 The Math Behind This Package

The math behind this package is a simple trick in allowing global computation to be done instead of local computations. This works on any commutative computable ring (in the sense of homalg [BLH20]) without need of implementing new low level algorithms. Details can be found in the paper [BLH11]. This ring can be constructed by `LocalizeAt` (4.3.14) and `LocalizeAtZero` (4.3.15).

Furthermore we use the package RingsForHomalg to communicate with Singular and use the implementation of Mora's algorithm there. This is restricted to polynomial rings and needs the package RingsForHomalg. This ring can be constructed by `LocalizePolynomialRingAtZeroWithMora` (4.3.16).

1.4 Which Ring to Use?

Since there are two kinds of rings included in this package, we want to offer a short comparison of these.

As usually one important part of such a comparison is the computation time. In our experience the general localization is much faster than Mora's algorithm for large examples.

The main advantage of using local bases with Mora's algorithm is the possibility of computing Hilbert polynomials and other combinatorical invariants. This is not possible with our localization algorithm. But it is possible to do a large computation without Mora's algorithm, which perhaps would not terminate in acceptable time, and afterwards compute a local standard basis of the - in comparison to intermediate computations usually much smaller - result to get the combinatorical information and invariants.

Furthermore we remark, that our localization algorithm works on any maximal ideal in any computable commutative ring, whereas Mora's algorithm only works for polynomial rings at the maximal ideal generated by the indeterminates. Of course by affine transformation Mora's algorithm will work on any maximal ideal in a polynomial ring where the residue class field is isomorphic to the ground field.

Chapter 2

Installation of the LocalizeRingForHomalg Package

To install this package just extract the package's archive file to the GAP pkg directory. LocalizeRingForHomalg also needs the package homalg.

By default the LocalizeRingForHomalg package is not automatically loaded by GAP when it is installed. You must load the package with

```
LoadPackage("LocalizeRingForHomalg");
```

before its functions become available.

Please, send me us e-mail if you have any questions, remarks, suggestions, etc. concerning this package. Also, we would be pleased to hear about applications of this package.

Mohamed Barakat and Markus Lange-Hegermann

Chapter 3

Quick Start

3.1 Localization of $\mathbb{Z}$

The following example is taken from Section 2 of [BR06].

The computation takes place over the local ring $R = \mathbb{Z}_{\langle 2 \rangle}$ (i.e. $\mathbb{Z}$ localized at the maximal ideal generated by 2).

Here we compute the (infinite) long exact homology sequence of the covariant functor $Hom(Hom(-,R/2^7R),R/2^4R)$ (and its left derived functors) applied to the short exact sequence

$0 \longrightarrow M\_ = R/2^2R \xrightarrow{\alpha_1} M = R/2^5R \xrightarrow{\alpha_2} \_M = R/2^3R \longrightarrow 0 .$

We want to lead your attention to the commands `LocalizeAt` and `HomalgLocalMatrix`. The first one creates a localized ring from a global one and generators of a maximal ideal and the second one creates a local matrix from a global matrix. The other commands used here are well known from homalg.

Example

```
gap> LoadPackage( "LocalizeRingForHomalg" );;
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> R := LocalizeAt( ZZ , [ 2 ] );
Z_< 2 >
gap> Display( R );
<A local ring>
gap> LoadPackage( "Modules" );
true
gap> M := LeftPresentation( HomalgMatrix( [ 2^5 ], R ) );
<A cyclic left module presented by 1 relation for a cyclic generator>
gap> _M := LeftPresentation( HomalgMatrix( [ 2^3 ], R ) );
<A cyclic left module presented by 1 relation for a cyclic generator>
gap> alpha2 := HomalgMap( HomalgMatrix( [ 1 ], R ), M, _M );
<A "homomorphism" of left modules>
gap> M_ := Kernel( alpha2 );
<A cyclic left module presented by yet unknown relations for a cyclic generato\
r>
gap> alpha1 := KernelEmb( alpha2 );
<A monomorphism of left modules>
```

```
gap> Display( M_ );
Z_< 2 >/< -4/1 >
gap> Display( alpha1 );
[ [  24 ] ]
/ 1

the map is currently represented by the above 1 x 1 matrix
gap> ByASmallerPresentation( M_ );
<A cyclic left module presented by 1 relation for a cyclic generator>
gap> Display( M_ );
Z_< 2 >/< 4/1 >
```

Chapter 4

Localize Rings

The package LocalizeRingForHomalg defines the classes of local(ized) rings, local ring elements and local matrices. These three objects can be used as data structures defined in MatricesForHomalg on which the homalg project can rely to do homological computations over localized rings.

A homalg local ring element contains two homalg ring elements, a numerator (→ `Numerator` (4.3.4)) and a denominator (→ `Denominator` (4.3.6)). A homalg local matrix contains a global homalg matrix as a numerator (→ `Numerator` (4.3.5)) and a ring element as a denominator (→ `Denominator` (4.3.7)). New constructors for ring elements and matrices are `HomalgLocalRingElement` (4.3.17) and `HomalgLocalMatrix` (4.3.18) in addition to the standard contructors introduced in other packages of the homalg project.

The local rings most prominently can be used with methods known from general homalg rings. The methods for doing the computations are presented in the appendix (A), since they are not for external use. The new attributes and operations are documented here.

Since the objects inplemented here are representations from objects elsewhere in the homalg project (i.e. MatricesForHomalg), we want to stress that there are many other operations in homalg, which can be used in connection with the ones presented here. A few of them can be found in the examples and the appendix of this documentation.

4.1 Category and Representations

4.1.1 IsHomalgLocalRingRep

▷ `IsHomalgLocalRingRep(R)` (Representation)

Returns: true or false

The representation of homalg local rings.

(It is a subrepresentation of the GAP representation `IsHomalgRingOrFinitelyPresentedModuleRep`.)

Code
```
DeclareRepresentation( "IsHomalgLocalRingRep",
        IsHomalgRing
        and IsHomalgRingOrFinitelyPresentedModuleRep,
        [ "ring" ] );
```

4.1.2 IsHomalgLocalRingElementRep

▷ `IsHomalgLocalRingElementRep(r)` (Representation)

Returns: true or false

The representation of elements of homalg local rings.

(It is a representation of the GAP category `IsHomalgRingElement`.)

Code
```
DeclareRepresentation( "IsHomalgLocalRingElementRep",
        IsHomalgRingElement,
        [ "pointer" ] );
```

4.1.3 IsHomalgLocalMatrixRep

▷ `IsHomalgLocalMatrixRep(A)` (Representation)

Returns: true or false

The representation of homalg matrices with entries in a homalg local ring.

(It is a representation of the GAP category `IsHomalgMatrix`.)

Code
```
DeclareRepresentation( "IsHomalgLocalMatrixRep",
        IsHomalgMatrix,
        [ ] );
```

4.2 Rings: Attributes

4.2.1 GeneratorsOfMaximalLeftIdeal

▷ `GeneratorsOfMaximalLeftIdeal(R)` (attribute)

Returns: a homalg matrix

Returns the generators of the maximal ideal, at which R was created. The generators are given as a column over the associated global ring.

4.2.2 GeneratorsOfMaximalRightIdeal

▷ `GeneratorsOfMaximalRightIdeal(R)` (attribute)

Returns: a homalg matrix

Returns the generators of the maximal ideal, at which R was created. The generators are given as a row over the associated global ring.

4.3 Operations and Functions

4.3.1 AssociatedGlobalRing (for homalg local rings)

▷ `AssociatedGlobalRing(R)` (operation)

Returns: a (global) homalg ring

The global homalg ring, from which the local ring *R* was created.

4.3.2 AssociatedGlobalRing (for homalg local ring elements)

▷ `AssociatedGlobalRing(r)` (operation)

Returns: a (global) homalg ring

The global homalg ring, from which the local ring element *r* was created.

4.3.3 AssociatedGlobalRing (for homalg local matrices)

▷ `AssociatedGlobalRing(mat)` (operation)

Returns: a (global) homalg ring

The global homalg ring, from which the local matrix *mat* was created.

4.3.4 Numerator (for homalg local ring elements)

▷ `Numerator(r)` (operation)

Returns: a (global) homalg ring element

The numerator from a local ring element *r*, which is a homalg ring element from the computation ring.

4.3.5 Numerator (for homalg local matrices)

▷ `Numerator(mat)` (operation)

Returns: a (global) homalg matrix

The numerator from a local matrix *mat*, which is a homalg matrix from the computation ring.

4.3.6 Denominator (for homalg local ring elements)

▷ `Denominator(r)` (operation)

Returns: a (global) homalg ring element

The denominator from a local ring element *r*, which is a homalg ring element from the computation ring.

4.3.7 Denominator (for homalg local matrices)

▷ `Denominator(mat)` (operation)

Returns: a (global) homalg ring element

The denominator from a local matrix *mat*, which is a homalg matrix from the computation ring.

4.3.8 Name (for homalg local ring elements)

▷ `Name(r)` (operation)

Returns: a string

The name of the local ring element *r*.

4.3.9 SetMatElm (for homalg local matrices)

▷ `SetMatElm(mat, i, j, r, R)` (operation)

Changes the entry (i,j) of the local matrix `mat` to the value `r`. Here `R` is the (local) homalg ring involved in these computations.

4.3.10 AddToMatElm (for homalg local matrices)

▷ `AddToMatElm(mat, i, j, r, R)` (operation)

Changes the entry (i,j) of the local matrix `mat` by adding the value `r` to it. Here `R` is the (local) homalg ring involved in these computations.

4.3.11 MatElmAsString (for homalg local matrices)

▷ `MatElmAsString(mat, i, j, R)` (operation)

Returns: a string

Returns the entry (i,j) of the local matrix `mat` as a string. Here `R` is the (local) homalg ring involved in these computations.

4.3.12 MatElm (for homalg local matrices)

▷ `MatElm(mat, i, j, R)` (operation)

Returns: a local ring element

Returns the entry (i,j) of the local matrix `mat`. Here `R` is the (local) homalg ring involved in these computations.

4.3.13 Cancel (for pairs of ring elements)

▷ `Cancel(a, b)` (operation)

Returns: two ring elements

For $a = a' * c$ and $b = b' * c$ return a' and b'. The exact form of c depends on whether a procedure for gcd computation is included in the ring package.

4.3.14 LocalizeAt (for a commutative ring and a maximal ideal)

▷ `LocalizeAt(R, l)` (operation)

Returns: a local ring

If `l` is a list of elements of the global ring `R` generating a maximal ideal, the method creates the corresponding localization of `R` at the complement of the maximal ideal.

4.3.15 LocalizeAtZero (for a free polynomial ring)

▷ `LocalizeAtZero(R)` (operation)

Returns: a local ring

This method creates the corresponding localization of `R` at the complement of the maximal ideal generated by the indeterminates ("at zero").

4.3.16 LocalizePolynomialRingAtZeroWithMora (constructor for homalg localized rings using Mora's algorithm)

▷ `LocalizePolynomialRingAtZeroWithMora(R)` (operation)

Returns: a local ring

This method localizes the ring *R* at zero and this localized ring is returned. The ring table uses Mora's algorithm as implemented Singular for low level computations.

4.3.17 HomalgLocalRingElement (constructor for local ring elements using numerator and denominator)

▷ `HomalgLocalRingElement(numer, denom, R)` (function)

▷ `HomalgLocalRingElement(numer, R)` (function)

Returns: a local ring element

Creates the local ring element *numer*/*denom* or in the second case *numer*/1 for the local ring *R*. Both *numer* and *denom* may either be a string describing a valid global ring element or from the global ring or computation ring.

4.3.18 HomalgLocalMatrix (constructor for local matrices using numerator and denominator)

▷ `HomalgLocalMatrix(numer, denom, R)` (function)

▷ `HomalgLocalMatrix(numer, R)` (function)

Returns: a local matrix

Creates the local matrix *numer*/*denom* or in the second case *numer*/1 for the local ring *R*. Both *numer* and *denom* may either be from the global ring or the computation ring.

Chapter 5

Examples

5.1 An Easy Polynomial Example

The ground ring used in this example is $F_3[x,y]$. We want to see, how the different rings in this package can be used to localize at different points and how the results differ.

Example
```
gap> LoadPackage("RingsForHomalg");;
gap> F3xy := HomalgRingOfIntegersInSingular(3) * "x,y";;
gap> x1 := HomalgRingElement( "x+2", F3xy );;
gap> y0 := HomalgRingElement( "y", F3xy );;
gap> LoadPackage("LocalizeRingForHomalg");;
gap> R00 := LocalizeAtZero( F3xy );;
gap> R10 := LocalizeAt( F3xy, [ x1, y0 ] );;
gap> RMora := LocalizePolynomialRingAtZeroWithMora( F3xy );;
gap> M := HomalgMatrix( "[\
>        y^3+2*y^2+x+x^2+2*x*y+y^4+x*y^2, \
>        x*y^3+2*x^2*y+y^3+y^2+x+2*y+x^2, \
>        x^2*y^2+2*x^3+x^2*y+y^3+2*x^2+2*x*y+y^2+2*y\
>      ]", 1, 3, F3xy );;
gap> LoadPackage( "Modules" );;
gap> I := RightPresentation( M );;
gap> M00 := HomalgLocalMatrix( M, R00 );;
gap> M10 := HomalgLocalMatrix( M, R10 );;
gap> MMora := HomalgLocalMatrix( M, RMora );;
gap> I00 := RightPresentation( M00 );;
gap> I10 := RightPresentation( M10 );;
gap> IMora := RightPresentation( MMora );;
```

This ring is able to compute a standard basis of the module.

Example
```
gap> Display( IMora );
GF(3)[x,y]_< x, y >/< (x+x^2-x*y-y^2+x*y^2+y^3+y^4)/1, (x-y+x^2+y^2-x^2*y+y^3+\
x*y^3)/1, (-y-x^2-x*y+y^2-x^3+x^2*y+y^3+x^2*y^2)/1 >
gap> ByASmallerPresentation( IMora );
<A cyclic torsion right module on a cyclic generator satisfying 2 relations>
gap> Display( IMora );
GF(3)[x,y]_< x, y >/< x/1, y/1 >
```

This ring recognizes, that the module is not zero, but is not able to find better generators.

Example
```
gap> Display( I00 );
GF(3)[x,y]_< x, y >/< (y^4+x*y^2+y^3+x^2-x*y-y^2+x)/1, (x*y^3-x^2*y+y^3+x^2+y^\
2+x-y)/1, (x^2*y^2-x^3+x^2*y+y^3-x^2-x*y+y^2-y)/1 >
gap> ByASmallerPresentation( I00 );
<A cyclic right module on a cyclic generator satisfying 3 relations>
gap> Display( I00 );
GF(3)[x,y]_< x, y >/< (y^4+x*y^2+y^3+x^2-x*y-y^2+x)/1, (x*y^3-x^2*y+y^3+x^2+y^\
2+x-y)/1, (x^2*y^2-x^3+x^2*y+y^3-x^2-x*y+y^2-y)/1 >
```

We are able to change the ring, to compute a nicer basis.

Example
```
gap> I00ToMora := RMora * I00;
<A cyclic right module on a cyclic generator satisfying 3 relations>
gap> Display( I00ToMora );
GF(3)[x,y]_< x, y >/< (x+x^2-x*y-y^2+x*y^2+y^3+y^4)/1, (x-y+x^2+y^2-x^2*y+y^3+\
x*y^3)/1, (-y-x^2-x*y+y^2-x^3+x^2*y+y^3+x^2*y^2)/1 >
gap> ByASmallerPresentation( I00ToMora );
<A cyclic torsion right module on a cyclic generator satisfying 2 relations>
gap> Display( I00ToMora );
GF(3)[x,y]_< x, y >/< x/1, y/1 >
```

We are able to find out, that this module is actually zero.

Example
```
gap> Display( I10 );
GF(3)[x,y]_< x-1, y >/< (y^4+x*y^2+y^3+x^2-x*y-y^2+x)/1, (x*y^3-x^2*y+y^3+x^2+\
y^2+x-y)/1, (x^2*y^2-x^3+x^2*y+y^3-x^2-x*y+y^2-y)/1 >
gap> ByASmallerPresentation( I10 );
<A zero right module>
gap> Display( I10 );
0
```

5.2 Hom(Hom(-,Z128),Z16)

The following example is taken from Section 2 of [BR06].

The computation takes place over the local ring $R = \mathbb{Z}_{\langle 2\rangle}$ (i.e. $\mathbb{Z}$ localized at the maximal ideal generated by 2).

Here we compute the (infinite) long exact homology sequence of the covariant functor $Hom(Hom(-,R/2^7R),R/2^4R)$ (and its left derived functors) applied to the short exact sequence

$$0 \longrightarrow M\_ = R/2^2R \xrightarrow{\alpha_1} M = R/2^5R \xrightarrow{\alpha_2} \_M = R/2^3R \longrightarrow 0\,.$$

Example
```
gap> LoadPackage( "LocalizeRingForHomalg" );;
gap> GlobalR := HomalgRingOfIntegersInExternalGAP( );
Z
gap> Display( GlobalR );
<An external ring residing in the CAS GAP>
gap> LoadPackage( "RingsForHomalg" );;
```

```
gap> R := LocalizeAt( GlobalR , [ 2 ] );
Z_< 2 >
gap> Display( R );
<A local ring>
gap> M := LeftPresentation( HomalgMatrix( [ 2^5 ], R ) );
<A cyclic left module presented by 1 relation for a cyclic generator>
gap> _M := LeftPresentation( HomalgMatrix( [ 2^3 ], R ) );
<A cyclic left module presented by 1 relation for a cyclic generator>
gap> alpha2 := HomalgMap( HomalgMatrix( [ 1 ], R ), M, _M );
<A "homomorphism" of left modules>
gap> M_ := Kernel( alpha2 );
<A cyclic left module presented by yet unknown relations for a cyclic generato\
r>
gap> alpha1 := KernelEmb( alpha2 );
<A monomorphism of left modules>
gap> seq := HomalgComplex( alpha2 );
<A "complex" containing a single morphism of left modules at degrees
[ 0 .. 1 ]>
gap> Add( seq, alpha1 );
gap> IsShortExactSequence( seq );
true
gap> K := LeftPresentation( HomalgMatrix( [ 2^7 ], R ) );
<A cyclic left module presented by 1 relation for a cyclic generator>
gap> L := RightPresentation( HomalgMatrix( [ 2^4 ], R ) );
<A cyclic right module on a cyclic generator satisfying 1 relation>
gap> triangle := LHomHom( 4, seq, K, L, "t" );
<An exact triangle containing 3 morphisms of left complexes at degrees
[ 1, 2, 3, 1 ]>
gap> lehs := LongSequence( triangle );
<A sequence containing 14 morphisms of left modules at degrees [ 0 .. 14 ]>
gap> ByASmallerPresentation( lehs );
<A non-zero sequence containing 14 morphisms of left modules at degrees
[ 0 .. 14 ]>
gap> IsExactSequence( lehs );
true
```

5.3 ResidueClass

We want to show, how localization can work together with residue class rings.

Example

```
gap> LoadPackage( "RingsForHomalg", ">= 2020.04.17" );;
gap> Qxy := HomalgFieldOfRationalsInDefaultCAS( ) * "x,y";
Q[x,y]
gap> wmat := HomalgMatrix(
>            "[ y^3-y^2 , x^3-x^2 , y^3+y^2 , x^3+x^2 ]",
>            2, 2, Qxy );
<A 2 x 2 matrix over an external ring>
gap> ec := HomalgRingElement( "-x^3-x^2+2*y^2", Qxy );
-x^3-x^2+2*y^2
```

Compute globally:

Example
```
gap> LoadPackage( "Modules" );;
gap> W := LeftPresentation( wmat );
<A left module presented by 2 relations for 2 generators>
gap> Res := Resolution( 2 , W );
<A right acyclic complex containing 2 morphisms of left modules at degrees
[ 0 .. 2 ]>
gap> Display( Res );
-------------------------
at homology degree: 2
0
-------------------------
(an empty 0 x 2 matrix)

the map is currently represented by the above 0 x 2 matrix
------------v------------
at homology degree: 1
Q[x,y]^(1 x 2)
-------------------------
y^2,      x^2,
x*y^2-y^3,0

the map is currently represented by the above 2 x 2 matrix
------------v------------
at homology degree: 0
Q[x,y]^(1 x 2)
-------------------------
```

Try a localization of a residue class ring:

Example
```
gap> R1 := Qxy / ec;
Q[x,y]/( -x^3-x^2+2*y^2 )
gap> Display( R1 );
<A residue class ring>
gap> wmat1 := R1 * wmat;
<A 2 x 2 matrix over a residue class ring>
gap> LoadPackage( "LocalizeRingForHomalg" );;
gap> R10 := LocalizeAt( R1 ,
>           [ HomalgRingElement( "x", R1 ),
>             HomalgRingElement( "y", R1 ) ]
>         );
Q[x,y]/( x^3+x^2-2*y^2 )_< |[ x ]|, |[ y ]| >
gap> Display( R10 );
<A local ring>
gap> wmat10 := HomalgLocalMatrix( wmat, R10 );
<A 2 x 2 matrix over a local ring>
gap> W10 := LeftPresentation( wmat10 );
<A left module presented by 2 relations for 2 generators>
gap> Res10 := Resolution( 2 , W10 );
<A right acyclic complex containing 2 morphisms of left modules at degrees
[ 0 .. 2 ]>
gap> Display( Res10 );
-------------------------
```

```
at homology degree: 2
0
-------------------------
(an empty 0 x 2 matrix)

the map is currently represented by the above 0 x 2 matrix
------------v------------
at homology degree: 1
Q[x,y]/( x^3+x^2-2*y^2 )_< |[ x ]|, |[ y ]| >^(1 x 2)
-------------------------
x*y^2+y^2,2*y^2,
y^2,      y^4-2*y^3+2*y^2

modulo [ x^3+x^2-2*y^2 ]
/ |[ 1 ]|

the map is currently represented by the above 2 x 2 matrix
------------v------------
at homology degree: 0
Q[x,y]/( x^3+x^2-2*y^2 )_< |[ x ]|, |[ y ]| >^(1 x 2)
-------------------------
```

Try a residue class ring of a localization:

Example

```
gap> R0 := LocalizeAtZero( Qxy );
Q[x,y]_< x, y >
gap> Display( R0 );
<A local ring>
gap> wmat0 := R0 * wmat;
<A 2 x 2 matrix over a local ring>
gap> R01 := R0 / ( ec / R0 );
Q[x,y]_< x, y >/( (-x^3-x^2+2*y^2)/1 )
gap> Display( R01 );
<A residue class ring>
gap> wmat01 := R01 * wmat0;
<A 2 x 2 matrix over a residue class ring>
gap> W01 := LeftPresentation( wmat01 );
<A left module presented by 2 relations for 2 generators>
gap> Res01 := Resolution( 2 , W01 );
<A right acyclic complex containing 2 morphisms of left modules at degrees
[ 0 .. 2 ]>
gap> Display( Res01 );
-------------------------
at homology degree: 2
0
-------------------------
(an empty 0 x 2 matrix)

the map is currently represented by the above 0 x 2 matrix
------------v------------
at homology degree: 1
Q[x,y]_< x, y >/( (x^3+x^2-2*y^2)/1 )^(1 x 2)
-------------------------
```

```
y^3+y^2,2*y^2,
0,      x*y^2-y^3
/ 1

modulo [ (x^3+x^2-2*y^2)/1 ]

the map is currently represented by the above 2 x 2 matrix
------------v------------
at homology degree: 0
Q[x,y]_< x, y >/( (x^3+x^2-2*y^2)/1 )^(1 x 2)
-------------------------
```

5.4 Testing the Intersection Formula

We want to check Serre's intersection formula $i(I_1,I_2;0) = \sum_i (-1)^i length(Tor_i^{R_0}(R_0/I_1,R_0/I_2))$ on an easy affine example.

Example

```
gap> LoadPackage( "RingsForHomalg" );;
gap> R := HomalgFieldOfRationalsInSingular() * "w,x,y,z";;
gap> LoadPackage( "LocalizeRingForHomalg" );;
gap> R0 := LocalizePolynomialRingAtZeroWithMora( R );;
gap> M1 := HomalgMatrix( "[\
>        (w-x^2)*y, \
>        (w-x^2)*z, \
>        (x-w^2)*y, \
>        (x-w^2)*z  \
>      ]", 4, 1, R );;
gap> M2 := HomalgMatrix( "[\
>        (w-x^2)-y, \
>        (x-w^2)-z  \
>      ]", 2, 1, R );;
gap> LoadPackage( "Modules" );;
gap> RmodI1 := LeftPresentation( M1 );;
gap> RmodI2 := LeftPresentation( M2 );;
gap> T:=Tor( RmodI1, RmodI2 );
<A graded homology object consisting of 4 left modules at degrees [ 0 .. 3 ]>
gap> List( ObjectsOfComplex( T ), AffineDegree );
[ 12, 4, 0, 0 ]
```

We read, that the intersection multiplicity is 12-4=8 globally.

Example

```
gap> M10 := R0 * M1;
<A 4 x 1 matrix over a local (Mora) ring>
gap> M20 := R0 * M2;
<A 2 x 1 matrix over a local (Mora) ring>
gap> R0modI10 := LeftPresentation( M10 );;
gap> R0modI20 := LeftPresentation( M20 );;
gap> T0 := Tor( R0modI10, R0modI20 );
<A graded homology object consisting of 4 left modules at degrees [ 0 .. 3 ]>
```

```
gap> List( ObjectsOfComplex( T0 ), AffineDegree );
[ 3, 1, 0, 0 ]
```

The intersection multiplicity at zero is 3-1=2.

Appendix A

Overview of the LocalizeRingForHomalg Package Source Code

This appendix is included in the documentation to shine some light on the mathematical backgrounds of this Package. Neither is it needed to work with this package nor should the methods presented here be called directly. The functions documented here are entries of the so called ring table and not to be called directly. There are higher level methods in declared and installed in MatricesForHomalg, which call this functions (→ `?MatricesForHomalg:The Basic Matrix Operations`).

We only present the simpler procedures, where no transformation matrices are computed, since the computation of transformation matrices carries no further mathematical ideas.

A.1 The generic Methods

There are some methods in localized rings, where homalg is able to fall back on procedures of the corresponding global ring. Furthermore these methods work quite good together with Mora's algorithm as implemented in Singular, since we can treat it like a global ring. We will present some methods as an example, to show the idea:

A.1.1 BasisOfRowModule (for local rings)

▷ `BasisOfRowModule(M)` (function)

Returns: a "basis" of the module generated by M

This procedure computes a basis by using the Funcod of the underlying computation ring. If the computation ring is given by Mora's Algorithm, we will indeed compute a local basis. If we just use the global ring for computations, this will be a global basis and is just computed for some simplifications and not for the use of reducing by it. Of course we can just forget about the denominator of `M`.

Code

```
BasisOfRowModule :=
  function( M )

    Info(
      InfoLocalizeRingForHomalg,
```

```
      2,
      "Start BasisOfRowModule with ",
      NrRows( M ), "x", NrColumns( M )
    );

    return HomalgLocalMatrix( BasisOfRowModule( Numerator( M ) ), HomalgRing( M ) );

  end,
```

A.1.2 DecideZeroRows (for local rings with Mora's algorithm)

▷ `DecideZeroRows(A, B)` (function)

Returns: a "reduced" form of *A* with respect to *B*

This procedure just calls the DecideZeroRows of the computation ring for the numerator of *A*.

If we use Mora's algorithm this procedure will just call it. The result is divided by the denominator of *A* afterwards. Again we do not need to care about the denominator of B.

If we use the reduction implemented in this package, this Funcod is overwritten and will not be called.

Code

```
DecideZeroRows :=
  function( A, B )
    local R, ComputationRing, hook, result;

    Info(
      InfoLocalizeRingForHomalg,
      2,
      "Start DecideZeroRows with ",
      NrRows( A ), "x", NrColumns( A ),
      " and ",
      NrRows( B ), "x", NrColumns( B )
    );

    R := HomalgRing( A );
    ComputationRing := AssociatedComputationRing( R );

    result := DecideZeroRows( Numerator( A ) , Numerator( B ) );
    result := HomalgLocalMatrix( result, Denominator( A ) , R );
    Info( InfoLocalizeRingForHomalgShowUnits, 1, "DecideZeroRows: produces denominator: ", Name(
    return result;

  end,
```

A.1.3 SyzygiesGeneratorsOfRows (for local rings)

▷ `SyzygiesGeneratorsOfRows(M)` (function)

Returns: a "basis" of the syzygies of the arguments (for details consult the homalg help)

It is easy to see, that a global syzygy is also a local syzygy and vice versa when clearing the local Syzygy of its denominators. So this procedure just calls the syzygy Funcod of the underlying computation ring.

Code
```
SyzygiesGeneratorsOfRows :=
  function( M )

    Info(
      InfoLocalizeRingForHomalg,
      2,
      "Start SyzygiesGeneratorsOfRows with ",
      NrRows( M ), "x", NrColumns( M )
    );

    return HomalgLocalMatrix(\
             SyzygiesGeneratorsOfRows( Numerator( M ) ), HomalgRing( M )\
           );

  end,
```

A.2 The Local Decide Zero trick

A.2.1 DecideZeroRows (for local rings)

▷ DecideZeroRows(*B*, *A*) (function)

Returns: a "reduced" form of *B* with respect to *A*

This procedure is the mathematical core procedure of this package. We use a trick to decide locally, whether *B* can be reduced to zero by *A* with a global computation. First a heuristic is used by just checking, whether the element lies inside the global module, generated by the generators of the local module. This of course implies this for the local module having the advantage of a short computation time and leaving a normal form free of denominators. If this check fails, we use our trick to check for each row of *B* independently, whether it lies in the module generated by *B*.

Code
```
DecideZeroRows :=
  function( B, A )
    local R, T, m, gens, n, GlobalR, one, N, b, numB, denB, i, B1, A1, B2, A2, B3;

    Info(
       InfoLocalizeRingForHomalg,
       2,
       "Start DecideZeroRows with ",
       NrRows( B ), "x", NrColumns( B ),
       " and ",
       NrRows( A ), "x", NrColumns( A )
    );

    R := HomalgRing( B );
    GlobalR := AssociatedComputationRing( R );
    T := HomalgVoidMatrix( R );
    gens := GeneratorsOfMaximalLeftIdeal( R );
    n := NrRows( gens );
    one := One( GlobalR );

    m := NrRows( A );
```

```
    A1 := Numerator( A );

    N := HomalgZeroMatrix( 0, NrColumns( B ), R );
    b := Eval( B );
    numB := b[1];
    denB := b[2];

    for i in [ 1 .. NrRows( B ) ] do

        #use global reduction as heuristic
        B1 := CertainRows( numB, [ i ] );
        B2 := HomalgLocalMatrix( DecideZeroRows( B1, A1 ), R );

        #if it is nonzero, check whether local reduction makes it zero
        if not IsZero( B2 ) then
          A2 := UnionOfRows( A1, gens * B1 );
          A2 := BasisOfRows( A2 );
          B3 := HomalgLocalMatrix( DecideZeroRows( B1, A2 ), R );
          if IsZero( B3 ) then
            B2 := B3;
          fi;
        fi;

        N := UnionOfRows( N, B2 );

    od;

    N := HomalgRingElement( one, denB, R ) * N;

    Info( InfoLocalizeRingForHomalgShowUnits, 1, "DecideZeroRows: produces denominator: ", Name(

    return N;

  end,
```

A.3 Tools

The package LocalizeRingForHomalg also implements tool functions. These are referred to from MatricesForHomalg automatically. We list the implemented methods here are and refer to the MatricesForHomalg documentation (→ `?MatricesForHomalg: The Matrix Tool Operations` and `?MatricesForHomalg:RingElement`) for details. All tools functions from MatricesForHomalg not listed here are also supported by fallback tools.

- IsZero
- IsOne
- Minus
- DivideByUnit
- IsUnit

- Sum
- Product
- ShallowCopy
- ZeroMatrix
- IdentityMatrix
- AreEqualMatrices
- Involution
- CertainRows
- CertainColumns
- UnionOfRows
- UnionOfColumns
- DiagMat
- KroneckerMat
- DualKroneckerMat
- MulMat
- AddMat
- SubMat
- Compose
- NrRows
- NrColumns
- IsZeroMatrix
- IsDiagonalMatrix
- ZeroRows
- ZeroColumns

References

[BLH11] Mohamed Barakat and Markus Lange-Hegermann. An axiomatic setup for algorithmic homological algebra and an alternative approach to localization. *J. Algebra Appl.*, 10(2):269–293, 2011. 4

[BLH20] Mohamed Barakat and Markus Lange-Hegermann. *The* `homalg` *package – A homological algebra* GAP4 *meta-package for computable Abelian categories*, 2007–2020. (`https://homalg-project.github.io/pkg/homalg`). 4

[BR06] Mohamed Barakat and Daniel Robertz. `homalg`: First steps to an abstract package for homological algebra. In *Proceedings of the X meeting on computational algebra and its applications - EACA 2006*, pages 29–32, Sevilla, Spain, September 2006. 7, 15

[hom22] homalg project authors. The `homalg` project – Algorithmic Homological Algebra. (`https://homalg-project.github.io/prj/homalg_project`), 2003–2022. 4

Index

LocalizeRingForHomalg, 4

AddToMatElm
  for homalg local matrices, 12
AssociatedGlobalRing
  for homalg local matrices, 11
  for homalg local ring elements, 11
  for homalg local rings, 10

BasisOfRowModule
  for local rings, 21

Cancel
  for pairs of ring elements, 12

DecideZeroRows
  for local rings, 23
  for local rings with Mora's algorithm, 22
Denominator
  for homalg local matrices, 11
  for homalg local ring elements, 11

GeneratorsOfMaximalLeftIdeal, 10
GeneratorsOfMaximalRightIdeal, 10

HomalgLocalMatrix
  constructor for local matrices using a given numerator and one as denominator, 13
  constructor for local matrices using numerator and denominator, 13
HomalgLocalRingElement
  constructor for local ring elements using a given numerator and one as denominator, 13
  constructor for local ring elements using numerator and denominator, 13

IsHomalgLocalMatrixRep, 10
IsHomalgLocalRingElementRep, 10
IsHomalgLocalRingRep, 9

LocalizeAt
  for a commutative ring and a maximal ideal, 12
LocalizeAtZero
  for a free polynomial ring, 12
LocalizePolynomialRingAtZeroWithMora
  constructor for homalg localized rings using Mora's algorithm, 13

MatElm
  for homalg local matrices, 12
MatElmAsString
  for homalg local matrices, 12

Name
  for homalg local ring elements, 11
Numerator
  for homalg local matrices, 11
  for homalg local ring elements, 11

SetMatElm
  for homalg local matrices, 12
SyzygiesGeneratorsOfRows
  for local rings, 22