

# MatricesForHomalg

**Matrices for the homalg project**

Version 2017.11.03

June 2015

**Mohamed Barakat**

**Markus Lange-Hegermann**

**Vinay Wagh**

*(this manual is still under construction)*

This manual is best viewed as an HTML document. The latest version is available ONLINE at:

<http://homalg.math.rwth-aachen.de/~barakat/homalg-project/MatricesForHomalg/chap0.1>

An OFFLINE version should be included in the documentation subfolder of the package. This package is part of the homalg-project:

<http://homalg.math.rwth-aachen.de/index.php/core-packages/matricesforhomalg>

**Mohamed Barakat**

Email: [barakat@mathematik.uni-kl.de](mailto:barakat@mathematik.uni-kl.de)

Homepage: <http://www.mathematik.uni-kl.de/~barakat/>

Address: Department of Mathematics,  
University of Kaiserslautern,  
67653 Kaiserslautern,  
Germany

**Markus Lange-Hegermann**

Email: [markus.lange.hegermann@rwth-aachen.de](mailto:markus.lange.hegermann@rwth-aachen.de)

Homepage: <http://wwwb.math.rwth-aachen.de/~markus/>

Address: Lehrstuhl B für Mathematik, RWTH Aachen, Templer-  
graben 64, 52056 Aachen, Germany

**Vinay Wagh**

Email: [waghoba@gmail.com](mailto: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.

## **Copyright**

© 2007-2015 by Mohamed Barakat and Markus Lange-Hegermann

This package may be distributed under the terms and conditions of the GNU Public License Version 2.

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	What is the role of the MatricesForHomalg package in the homalg project? . . . .	5
1.2	This manual . . . . .	6
<b>2</b>	<b>Installation of the MatricesForHomalg Package</b>	<b>7</b>
<b>3</b>	<b>Rings</b>	<b>8</b>
3.1	Rings: Category and Representations . . . . .	8
3.2	Rings: Constructors . . . . .	9
3.3	Rings: Properties . . . . .	11
3.4	Rings: Attributes . . . . .	19
3.5	Rings: Operations and Functions . . . . .	22
<b>4</b>	<b>Ring Maps</b>	<b>23</b>
4.1	Ring Maps: Category and Representations . . . . .	23
4.2	Ring Maps: Constructors . . . . .	23
4.3	Ring Maps: Properties . . . . .	24
4.4	Ring Maps: Attributes . . . . .	25
4.5	Ring Maps: Operations and Functions . . . . .	25
<b>5</b>	<b>Matrices</b>	<b>26</b>
5.1	Matrices: Category and Representations . . . . .	26
5.2	Matrices: Constructors . . . . .	26
5.3	Matrices: Properties . . . . .	31
5.4	Matrices: Attributes . . . . .	35
5.5	Matrices: Operations and Functions . . . . .	38
<b>6</b>	<b>Ring Relations</b>	<b>52</b>
6.1	Ring Relations: Categories and Representations . . . . .	52
6.2	Ring Relations: Constructors . . . . .	53
6.3	Ring Relations: Properties . . . . .	53
6.4	Ring Relations: Attributes . . . . .	53
6.5	Ring Relations: Operations and Functions . . . . .	53
<b>A</b>	<b>The Basic Matrix Operations</b>	<b>54</b>
A.1	Main . . . . .	54
A.2	Effective . . . . .	54

A.3	Relative . . . . .	54
A.4	Reduced . . . . .	55
<b>B</b>	<b>The Matrix Tool Operations</b>	<b>56</b>
B.1	The Tool Operations <i>without</i> a Fallback Method . . . . .	56
B.2	The Tool Operations with a Fallback Method . . . . .	62
<b>C</b>	<b>Logic Subpackages</b>	<b>72</b>
C.1	LIRNG: Logical Implications for Rings . . . . .	72
C.2	LIMAP: Logical Implications for Ring Maps . . . . .	72
C.3	LIMAT: Logical Implications for Matrices . . . . .	72
C.4	COLEM: Clever Operations for Lazy Evaluated Matrices . . . . .	72
<b>D</b>	<b>The subpackage ResidueClassRingForHomalg as a sample ring package</b>	<b>87</b>
D.1	The Mandatory Basic Operations . . . . .	87
D.2	The Mandatory Tool Operations . . . . .	93
D.3	Some of the Recommended Tool Operations . . . . .	99
<b>E</b>	<b>Debugging MatricesForHomalg</b>	<b>101</b>
E.1	Increase the assertion level . . . . .	101
E.2	Using homalgMode . . . . .	103
<b>F</b>	<b>Overview of the MatricesForHomalg Package Source Code</b>	<b>105</b>
F.1	Rings, Ring Maps, Matrices, Ring Relations . . . . .	105
F.2	The Low Level Algorithms . . . . .	105
F.3	Logical Implications for MatricesForHomalg Objects . . . . .	106
F.4	The subpackage ResidueClassRingForHomalg . . . . .	106
F.5	The homalgTable for GAP4 built-in rings . . . . .	107
	<b>References</b>	<b>108</b>
	<b>Index</b>	<b>109</b>

# Chapter 1

## Introduction

### 1.1 What is the role of the `MatricesForHomalg` package in the `homalg` project?

#### 1.1.1 `MatricesForHomalg` provides ...

The package `MatricesForHomalg` provides:

- rings
- ring elements
- ring maps
- matrices

#### 1.1.2 `homalg` delegates ...

The package `homalg` *delegates all* matrix operations as it treats matrices and their rings as *black boxes*. `homalg` comes with a single predefined class of rings and a single predefined class of matrices over these rings – the so-called internal matrices (→ 5.1.2) over so-called internal rings (→ 3.1.4). An internal matrix (resp. ring) is simply a wrapper containing a GAP-builtin matrix (resp. ring). `homalg` allows other packages to define further classes or extend existing classes of rings and matrices *together* with their operations. For example:

- The `homalg` subpackage `ResidueClassRingForHomalg` (→ Appendix D) defines the classes of residue class rings, residue class ring elements, and matrices over residue class rings. Such a matrix is defined by a matrix over the ambient ring which is nevertheless interpreted modulo the ring relations, i.e. modulo the generators of the defining ideal.
- The package `GaussForHomalg` extends the class of internal matrices enabling it to wrap sparse matrices provided by the package `Gauss`. `GaussForHomalg` delegates the essential part of the matrix creation and all matrix operations to `Gauss`.
- The package `HomalgToCAS` defines the classes of so-called external rings and matrices and the package `RingsForHomalg` delegates the essential part of the matrix creation and all matrix operations to external computer algebra systems like `Singular`, `Macaulay2`, `Sage`, `Macaulay2`,

MAGMA, Maple, ... . The package `homalg` accesses external matrices via pointers. The pointer of an external matrix is simply its name in the external system. `HomalgToCAS` chooses these names.

- The package `LocalizeRingForHomalg` defines the classes of local(ized) rings, local ring elements, and local matrices. A `homalg` local matrix contains a `homalg` matrix as a numerator and an element of the global ring as a denominator.

The matrix operations are divided into two classes called “Tools” and “Basic”. The “Tools” operations include addition, subtraction, multiplication, extracting certain rows or columns, stacking, and augmenting matrices ( $\rightarrow$  Appendix B). The “Basic” operations include the two basic operations in linear algebra needed to solve an inhomogeneous linear system  $XA = B$  with coefficients in a not necessarily commutative ring  $R$  ( $\rightarrow$  Appendix A):

- Effectively reducing  $B$  modulo  $A$ , i.e. effectively deciding if a row (or a set of rows)  $B$  lies in the  $R$ -span of the rows of the matrix  $A$ .
- Computing an  $R$ -generating set of row syzygies ( $=R$ -relations among the rows) of  $A$ , i.e. computing an  $R$ -generating set of the left kernel of  $A$ . This generating set is then given as the rows of a matrix  $Y$  and  $YA = 0$ .

The first operation is nothing but deciding the solvability of the inhomogeneous system  $XA = B$  and if solvable to compute a particular solution  $X$ , while the second is to compute an  $R$ -generating set for the homogeneous solution space, i.e. the solution space of the homogeneous system  $YA = 0$ . The above is of course also valid for the column convention.

### 1.1.3 The black box concept

Now we address the following concerns: Wouldn't the idea of using algorithms like the Gröbnerbasis algorithm(s) as a black box ( $\rightarrow$  1.1.2) contradict the following facts?

- It is known that an efficient Gröbnerbasis algorithm depends on the ring  $R$  under consideration. For example the implementation of the algorithm depends on the ground ring (or field)  $k$ .
- Often enough highly specialized implementations are used to address specific types of linear systems of equations (occurring in specific homological problems) in order to increase the speed or reduce the space needed for the computations.

The following should clarify the above concerns.

- Since each ring comes with its own black box, the first point is automatically resolved.
- Allow the black box coming with each ring to contain the different available implementations and make them accessible to `homalg` via standardized names, independent of the computer algebra system used to perform computations.

## 1.2 This manual

Chapter 2 describes the installation of this package. The remaining chapters are each devoted to one of the `MatricesForHomalg` objects ( $\rightarrow$  1.1.1) with its constructors, properties, attributes, and operations.

## Chapter 2

# Installation of the **MatricesForHomalg** Package

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

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

```
LoadPackage( "MatricesForHomalg" );
```

before its functions become available.

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

Mohamed Barakat



## Chapter 3

# Rings

### 3.1 Rings: Category and Representations

#### 3.1.1 IsHomalgRing

▷ IsHomalgRing( $R$ ) (Category)

**Returns:** true or false

The GAP category of homalg rings.

(It is a subcategory of the GAP categories IsStructureObject and IsHomalgRingOrModule.)

```
Code
DeclareCategory( "IsHomalgRing",
  IsStructureObject and
  IsRingWithOne and
  IsHomalgRingOrModule );
```

#### 3.1.2 IsPreHomalgRing

▷ IsPreHomalgRing( $R$ ) (Category)

**Returns:** true or false

The GAP category of pre homalg rings.

(It is a subcategory of the GAP category IsHomalgRing.)

These are rings with an incomplete homalgTable. They provide flexibility for developers to support a wider class of rings, as was necessary for the development of the LocalizeRingForHomalg package. They are not suited for direct usage.

```
Code
DeclareCategory( "IsPreHomalgRing",
  IsHomalgRing );
```

#### 3.1.3 IsHomalgRingElement

▷ IsHomalgRingElement( $r$ ) (Category)

**Returns:** true or false

The GAP category of elements of homalg rings which are not GAP4 built-in.

```

Code
-----
DeclareCategory( "IsHomalgRingElement",
  IsExtAElement and
  IsExtLElement and
  IsExtRElement and
  IsAdditiveElementWithInverse and
  IsMultiplicativeElementWithInverse and
  IsAssociativeElement and
  IsAdditivelyCommutativeElement and
  ## all the above guarantees IsHomalgRingElement => IsRingElement (in GAP4)
  IsAttributeStoringRep );

```

### 3.1.4 IsHomalgInternalRingRep

- ▷ `IsHomalgInternalRingRep( $R$ )` (Representation)  
**Returns:** true or false  
 The internal representation of homalg rings.  
 (It is a representation of the GAP category `IsHomalgRing`.)

## 3.2 Rings: Constructors

This section describes how to construct rings for use with `MatricesForHomalg`, which exploit the GAP4-built-in abilities to perform the necessary ring operations. By this we also mean necessary matrix operations over such rings. For the purposes of `MatricesForHomalg` only the ring of integers is properly supported in GAP4. The GAP4 extension packages `Gauss` and `GaussForHomalg` extend these built-in abilities to operations with sparse matrices over the ring  $\mathbb{Z}/p^n$  for  $p$  prime and  $n$  positive.

If a ring  $R$  is supported in `MatricesForHomalg` any of its residue class rings  $R/I$  is supported as well, provided the ideal  $I$  of relations admits a finite set of generators as a left resp. right ideal ( $\rightarrow \setminus/$  (3.2.3)). This is immediate for commutative noetherian rings.

### 3.2.1 HomalgRingOfIntegers (constructor for the integers)

- ▷ `HomalgRingOfIntegers()` (function)  
**Returns:** a homalg ring
- ▷ `HomalgRingOfIntegers( $c$ )` (function)  
**Returns:** a homalg ring  
 The no-argument form returns the ring of integers  $\mathbb{Z}$  for homalg.  
 The one-argument form accepts an integer  $c$  and returns the ring  $\mathbb{Z}/c$  for homalg:
- $c = 0$  defaults to  $\mathbb{Z}$
  - if  $c$  is a prime power then the package `GaussForHomalg` is loaded (if it fails to load an error is issued)
  - otherwise, the residue class ring constructor  $/$  ( $\rightarrow \setminus/$  (3.2.3)) is invoked

The operation `SetRingProperties` is automatically invoked to set the ring properties.

If for some reason you don't want to use the `GaussForHomalg` package (maybe because you didn't install it), then use

```
HomalgRingOfIntegers() / c;
```

but note that the computations will then be considerably slower.

### 3.2.2 HomalgFieldOfRationals (constructor for the field of rationals)

▷ `HomalgFieldOfRationals()` (function)

**Returns:** a homalg ring

The package `GaussForHomalg` is loaded and the field of rationals  $\mathbb{Q}$  is returned. If `GaussForHomalg` fails to load an error is issued.

The operation `SetRingProperties` is automatically invoked to set the ring properties.

### 3.2.3 $\setminus$ (constructor for residue class rings)

▷  $\setminus(R, ring\_rel)$  (operation)

**Returns:** a homalg ring

This is the homalg constructor for residue class rings  $R/I$ , where  $R$  is a homalg ring and  $I = ring\_rel$  is the ideal of relations generated by `ring_rel`. `ring_rel` might be:

- a set of ring relations of a left resp. right ideal
- a list of ring elements of  $R$
- a ring element of  $R$

For noncommutative rings: In the first case the set of ring relations should generate the ideal of relations  $I$  as left resp. right ideal, and their involutions should generate  $I$  as right resp. left ideal. If `ring_rel` is not a set of relations, a *left* set of relations is constructed.

The operation `SetRingProperties` is automatically invoked to set the ring properties.

Example

```
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> Display( ZZ );
<An internal ring>
gap> Z256 := ZZ / 2^8;
Z/( 256 )
gap> Display( Z256 );
<A residue class ring>
gap> Z2 := Z256 / 6;
Z/( 256, 6 )
gap> BasisOfRows( MatrixOfRelations( Z2 ) );
<An unevaluated non-zero 1 x 1 matrix over an internal ring>
gap> Z2;
Z/( 2 )
gap> Display( Z2 );
<A residue class ring>
```

### 3.3 Rings: Properties

The following properties are declared for `homalg` rings. Note that (apart from so-called true and immediate methods ( $\rightarrow$  C.1)) there are no methods installed for ring properties. This means that if the value of the ring property `Prop` is not set for a `homalg` ring  $R$ , then

```
Prop( R );
```

will cause an error. One can use the usual `GAP4` mechanism to check if the value of the property is set or not

```
HasProp( R );
```

If you discover that a specific property `Prop` is missing for a certain `homalg` ring  $R$  you can add using the usual `GAP4` mechanism

```
SetProp( R, true );
```

or

```
SetProp( R, false );
```

Be very cautious with setting "missing" properties to `homalg` objects: If the value you set is mathematically wrong `homalg` will probably draw wrong conclusions and might return wrong results.

#### 3.3.1 IsZero (for rings)

▷ `IsZero( R )` (property)

**Returns:** true or false

Check if the ring  $R$  is a zero, i.e., if  $\text{One}(R) = \text{Zero}(R)$ .

#### 3.3.2 ContainsAField

▷ `ContainsAField( R )` (property)

**Returns:** true or false

$R$  is a ring for `homalg`.

#### 3.3.3 IsRationalsForHomalg

▷ `IsRationalsForHomalg( R )` (property)

**Returns:** true or false

$R$  is a ring for `homalg`.

#### 3.3.4 IsFieldForHomalg

▷ `IsFieldForHomalg( R )` (property)

**Returns:** true or false

$R$  is a ring for `homalg`.

#### 3.3.5 IsDivisionRingForHomalg

▷ `IsDivisionRingForHomalg( R )` (property)

**Returns:** true or false

$R$  is a ring for `homalg`.

### 3.3.6 IsIntegersForHomalg

- ▷ `IsIntegersForHomalg( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.7 IsResidueClassRingOfTheIntegers

- ▷ `IsResidueClassRingOfTheIntegers( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.8 IsBezoutRing

- ▷ `IsBezoutRing( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.9 IsIntegrallyClosedDomain

- ▷ `IsIntegrallyClosedDomain( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.10 IsUniqueFactorizationDomain

- ▷ `IsUniqueFactorizationDomain( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.11 IsKaplanskyHermite

- ▷ `IsKaplanskyHermite( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.12 IsDedekindDomain

- ▷ `IsDedekindDomain( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.13 IsDiscreteValuationRing

- ▷ `IsDiscreteValuationRing( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

**3.3.14 IsFreePolynomialRing**

- ▷ IsFreePolynomialRing( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

**3.3.15 IsWeylRing**

- ▷ IsWeylRing( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

**3.3.16 IsLocalizedWeylRing**

- ▷ IsLocalizedWeylRing( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

**3.3.17 IsGlobalDimensionFinite**

- ▷ IsGlobalDimensionFinite( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

**3.3.18 IsLeftGlobalDimensionFinite**

- ▷ IsLeftGlobalDimensionFinite( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

**3.3.19 IsRightGlobalDimensionFinite**

- ▷ IsRightGlobalDimensionFinite( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

**3.3.20 HasInvariantBasisProperty**

- ▷ HasInvariantBasisProperty( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

**3.3.21 HasLeftInvariantBasisProperty**

- ▷ HasLeftInvariantBasisProperty( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.22 HasRightInvariantBasisProperty

- ▷ HasRightInvariantBasisProperty( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.23 IsLocal

- ▷ IsLocal( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.24 IsSemiLocalRing

- ▷ IsSemiLocalRing( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.25 IsIntegralDomain

- ▷ IsIntegralDomain( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.26 IsHereditary

- ▷ IsHereditary( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.27 IsLeftHereditary

- ▷ IsLeftHereditary( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.28 IsRightHereditary

- ▷ IsRightHereditary( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.29 IsHermite

- ▷ IsHermite( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.30 IsLeftHermite

- ▷ IsLeftHermite( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.31 IsRightHermite

- ▷ IsRightHermite( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.32 IsNoetherian

- ▷ IsNoetherian( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.33 IsLeftNoetherian

- ▷ IsLeftNoetherian( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.34 IsRightNoetherian

- ▷ IsRightNoetherian( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.35 IsCohenMacaulay

- ▷ IsCohenMacaulay( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.36 IsGorenstein

- ▷ IsGorenstein( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.37 IsKoszul

- ▷ IsKoszul( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.



### 3.3.38 IsArtinian (for rings)

- ▷ `IsArtinian( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.39 IsLeftArtinian

- ▷ `IsLeftArtinian( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.40 IsRightArtinian

- ▷ `IsRightArtinian( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.41 IsOreDomain

- ▷ `IsOreDomain( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.42 IsLeftOreDomain

- ▷ `IsLeftOreDomain( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.43 IsRightOreDomain

- ▷ `IsRightOreDomain( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.44 IsPrincipalIdealRing

- ▷ `IsPrincipalIdealRing( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.45 IsLeftPrincipalIdealRing

- ▷ `IsLeftPrincipalIdealRing( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.46 IsRightPrincipalIdealRing

- ▷ IsRightPrincipalIdealRing( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.47 IsRegular

- ▷ IsRegular( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.48 IsFiniteFreePresentationRing

- ▷ IsFiniteFreePresentationRing( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.49 IsLeftFiniteFreePresentationRing

- ▷ IsLeftFiniteFreePresentationRing( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.50 IsRightFiniteFreePresentationRing

- ▷ IsRightFiniteFreePresentationRing( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.51 IsSimpleRing

- ▷ IsSimpleRing( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.52 IsSemiSimpleRing

- ▷ IsSemiSimpleRing( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

### 3.3.53 IsSuperCommutative

- ▷ IsSuperCommutative( $R$ ) (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

**3.3.54 BasisAlgorithmRespectsPrincipalIdeals**

- ▷ `BasisAlgorithmRespectsPrincipalIdeals( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

**3.3.55 AreUnitsCentral**

- ▷ `AreUnitsCentral( $R$ )` (property)  
**Returns:** true or false  
 $R$  is a ring for homalg.

**3.3.56 IsMinusOne**

- ▷ `IsMinusOne( $r$ )` (property)  
**Returns:** true or false  
 Check if the ring element  $r$  is the additive inverse of one.

**3.3.57 IsMonic (for homalg ring elements)**

- ▷ `IsMonic( $r$ )` (property)  
**Returns:** true or false  
 Check if the homalg ring element  $r$  is monic.

**3.3.58 IsMonicUptoUnit (for homalg ring elements)**

- ▷ `IsMonicUptoUnit( $r$ )` (property)  
**Returns:** true or false  
 Check if leading coefficient of the homalg ring element  $r$  is a unit.

**3.3.59 IsLeftRegular (for homalg ring elements)**

- ▷ `IsLeftRegular( $r$ )` (property)  
**Returns:** true or false  
 Check if the homalg ring element  $r$  is left regular.

**3.3.60 IsRightRegular (for homalg ring elements)**

- ▷ `IsRightRegular( $r$ )` (property)  
**Returns:** true or false  
 Check if the homalg ring element  $r$  is right regular.

**3.3.61 IsRegular (for homalg ring elements)**

- ▷ `IsRegular( $r$ )` (property)  
**Returns:** true or false  
 Check if the homalg ring element  $r$  is regular, i.e. left and right regular.

## 3.4 Rings: Attributes

### 3.4.1 Inverse (for homalg ring elements)

- ▷ `Inverse(r)` (attribute)  
**Returns:** a homalg ring element or fail  
 The inverse of the homalg ring element  $r$ .

Example

```
gap> ZZ := HomalgRingOfIntegers( );;
gap> R := ZZ / 2^8;
Z/( 256 )
gap> r := (1/3*One(R)+1/5)+3/7;
|[ 157 ]|
gap> 1 / r;      ## = r^-1;
|[ 181 ]|
gap> s := (1/3*One(R)+2/5)+3/7;
|[ 106 ]|
gap> 1 / s;
fail
```

### 3.4.2 homalgTable

- ▷ `homalgTable(R)` (attribute)  
**Returns:** a homalg table  
 The homalg table of  $R$  is a ring dictionary, i.e. the translator between homalg and the (specific implementation of the) ring.  
 Every homalg ring has a homalg table.

### 3.4.3 RingElementConstructor

- ▷ `RingElementConstructor(R)` (attribute)  
**Returns:** a function  
 The constructor of ring elements in the homalg ring  $R$ .

### 3.4.4 TypeOfHomalgMatrix

- ▷ `TypeOfHomalgMatrix(R)` (attribute)  
**Returns:** a type  
 The GAP4-type of homalg matrices over the homalg ring  $R$ .

### 3.4.5 ConstructorForHomalgMatrices

- ▷ `ConstructorForHomalgMatrices(R)` (attribute)  
**Returns:** a type  
 The constructor for homalg matrices over the homalg ring  $R$ .

### 3.4.6 Zero (for homalg rings)

- ▷ `Zero(R)` (attribute)  
**Returns:** a homalg ring element  
 The zero of the homalg ring  $R$ .

### 3.4.7 One (for homalg rings)

- ▷ `One(R)` (attribute)  
**Returns:** a homalg ring element  
 The one of the homalg ring  $R$ .

### 3.4.8 MinusOne

- ▷ `MinusOne(R)` (attribute)  
**Returns:** a homalg ring element  
 The minus one of the homalg ring  $R$ .

### 3.4.9 ProductOfIndeterminates

- ▷ `ProductOfIndeterminates(R)` (attribute)  
**Returns:** a homalg ring element  
 The product of indeterminates of the homalg ring  $R$ .

### 3.4.10 RationalParameters

- ▷ `RationalParameters(R)` (attribute)  
**Returns:** a list of homalg ring elements  
 The list of rational parameters of the homalg ring  $R$ .

### 3.4.11 IndeterminatesOfPolynomialRing

- ▷ `IndeterminatesOfPolynomialRing(R)` (attribute)  
**Returns:** a list of homalg ring elements  
 The list of indeterminates of the homalg polynomial ring  $R$ .

### 3.4.12 RelativeIndeterminatesOfPolynomialRing

- ▷ `RelativeIndeterminatesOfPolynomialRing(R)` (attribute)  
**Returns:** a list of homalg ring elements  
 The list of relative indeterminates of the homalg polynomial ring  $R$ .

### 3.4.13 IndeterminateCoordinatesOfRingOfDerivations

- ▷ `IndeterminateCoordinatesOfRingOfDerivations(R)` (attribute)  
**Returns:** a list of homalg ring elements  
 The list of indeterminate coordinates of the homalg Weyl ring  $R$ .

### 3.4.14 RelativeIndeterminateCoordinatesOfRingOfDerivations

- ▷ `RelativeIndeterminateCoordinatesOfRingOfDerivations( $R$ )` (attribute)  
**Returns:** a list of homalg ring elements  
 The list of relative indeterminate coordinates of the homalg Weyl ring  $R$ .

### 3.4.15 IndeterminateDerivationsOfRingOfDerivations

- ▷ `IndeterminateDerivationsOfRingOfDerivations( $R$ )` (attribute)  
**Returns:** a list of homalg ring elements  
 The list of indeterminate derivations of the homalg Weyl ring  $R$ .

### 3.4.16 RelativeIndeterminateDerivationsOfRingOfDerivations

- ▷ `RelativeIndeterminateDerivationsOfRingOfDerivations( $R$ )` (attribute)  
**Returns:** a list of homalg ring elements  
 The list of relative indeterminate derivations of the homalg Weyl ring  $R$ .

### 3.4.17 IndeterminateAntiCommutingVariablesOfExteriorRing

- ▷ `IndeterminateAntiCommutingVariablesOfExteriorRing( $R$ )` (attribute)  
**Returns:** a list of homalg ring elements  
 The list of anti-commuting indeterminates of the homalg exterior ring  $R$ .

### 3.4.18 RelativeIndeterminateAntiCommutingVariablesOfExteriorRing

- ▷ `RelativeIndeterminateAntiCommutingVariablesOfExteriorRing( $R$ )` (attribute)  
**Returns:** a list of homalg ring elements  
 The list of anti-commuting relative indeterminates of the homalg exterior ring  $R$ .

### 3.4.19 IndeterminatesOfExteriorRing

- ▷ `IndeterminatesOfExteriorRing( $R$ )` (attribute)  
**Returns:** a list of homalg ring elements  
 The list of all indeterminates (commuting and anti-commuting) of the homalg exterior ring  $R$ .

### 3.4.20 CoefficientsRing

- ▷ `CoefficientsRing( $R$ )` (attribute)  
**Returns:** a homalg ring  
 The ring of coefficients of the homalg ring  $R$ .

### 3.4.21 KrullDimension

- ▷ `KrullDimension( $R$ )` (attribute)  
**Returns:** a non-negative integer  
 The Krull dimension of the commutative homalg ring  $R$ .

### 3.4.22 LeftGlobalDimension

- ▷ `LeftGlobalDimension( $R$ )` (attribute)  
**Returns:** a non-negative integer  
 The left global dimension of the homalg ring  $R$ .

### 3.4.23 RightGlobalDimension

- ▷ `RightGlobalDimension( $R$ )` (attribute)  
**Returns:** a non-negative integer  
 The right global dimension of the homalg ring  $R$ .

### 3.4.24 GlobalDimension

- ▷ `GlobalDimension( $R$ )` (attribute)  
**Returns:** a non-negative integer  
 The global dimension of the homalg ring  $R$ . The global dimension is defined, only if the left and right global dimensions coincide.

### 3.4.25 GeneralLinearRank

- ▷ `GeneralLinearRank( $R$ )` (attribute)  
**Returns:** a non-negative integer  
 The general linear rank of the homalg ring  $R$  ([MR01], 11.1.14).

### 3.4.26 ElementaryRank

- ▷ `ElementaryRank( $R$ )` (attribute)  
**Returns:** a non-negative integer  
 The elementary rank of the homalg ring  $R$  ([MR01], 11.3.10).

### 3.4.27 StableRank

- ▷ `StableRank( $R$ )` (attribute)  
**Returns:** a non-negative integer  
 The stable rank of the homalg ring  $R$  ([MR01], 11.3.4).

### 3.4.28 AssociatedGradedRing

- ▷ `AssociatedGradedRing( $R$ )` (attribute)  
**Returns:** a homalg ring  
 The graded ring associated to the filtered ring  $R$ .

## 3.5 Rings: Operations and Functions

## Chapter 4

# Ring Maps

A homalg ring map is a data structure for maps between finitely generated rings. homalg more or less provides the basic declarations and installs the generic methods for ring maps, but it is up to other high level packages to install methods applicable to specific rings. For example, the package Sheaves provides methods for ring maps of (finitely generated) affine rings.

### 4.1 Ring Maps: Category and Representations

#### 4.1.1 IsHomalgRingMap

- ▷ IsHomalgRingMap( $\phi$ ) (Category)  
**Returns:** true or false  
The GAP category of ring maps.

#### 4.1.2 IsHomalgRingSelfMap

- ▷ IsHomalgRingSelfMap( $\phi$ ) (Category)  
**Returns:** true or false  
The GAP category of ring self-maps.  
(It is a subcategory of the GAP category IsHomalgRingMap.)

#### 4.1.3 IsHomalgRingMapRep

- ▷ IsHomalgRingMapRep( $\phi$ ) (Representation)  
**Returns:** true or false  
The GAP representation of homalg ring maps.  
(It is a representation of the GAP category IsHomalgRingMap (4.1.1).)

### 4.2 Ring Maps: Constructors

#### 4.2.1 RingMap (constructor for ring maps)

- ▷ RingMap( $\text{images}$ ,  $S$ ,  $T$ ) (operation)  
**Returns:** a homalg ring map



This constructor returns a ring map (homomorphism) of finitely generated rings/algebras. It is represented by the images *images* of the set of generators of the source homalg ring  $S$  in terms of the generators of the target ring  $T$  ( $\rightarrow$  3.2). Unless the source ring is free *and* given on free ring/algebra generators the returned map will cautiously be indicated using parenthesis: “homomorphism”. To verify if the result is indeed a well defined map use `IsMorphism` (4.3.1). If source and target are identical objects, and only then, the ring map is created as a selfmap.

## 4.3 Ring Maps: Properties

### 4.3.1 IsMorphism (for ring maps)

- ▷ `IsMorphism(phi)` (property)  
**Returns:** true or false  
 Check if *phi* is a well-defined map, i.e. independent of all involved presentations.

### 4.3.2 IsIdentityMorphism (for ring maps)

- ▷ `IsIdentityMorphism(phi)` (property)  
**Returns:** true or false  
 Check if the homalg ring map *phi* is the identity morphism.

### 4.3.3 IsMonomorphism (for ring maps)

- ▷ `IsMonomorphism(phi)` (property)  
**Returns:** true or false  
 Check if the homalg ring map *phi* is a monomorphism.

### 4.3.4 IsEpimorphism (for ring maps)

- ▷ `IsEpimorphism(phi)` (property)  
**Returns:** true or false  
 Check if the homalg ring map *phi* is an epimorphism.

### 4.3.5 IsIsomorphism (for ring maps)

- ▷ `IsIsomorphism(phi)` (property)  
**Returns:** true or false  
 Check if the homalg ring map *phi* is an isomorphism.

### 4.3.6 IsAutomorphism (for ring maps)

- ▷ `IsAutomorphism(phi)` (property)  
**Returns:** true or false  
 Check if the homalg ring map *phi* is an automorphism.

## 4.4 Ring Maps: Attributes

### 4.4.1 Source (for ring maps)

- ▷ `Source(phi)` (attribute)  
**Returns:** a homalg ring  
The source of the homalg ring map  $phi$ .

### 4.4.2 Range (for ring maps)

- ▷ `Range(phi)` (attribute)  
**Returns:** a homalg ring  
The target (range) of the homalg ring map  $phi$ .

### 4.4.3 DegreeOfMorphism (for ring maps)

- ▷ `DegreeOfMorphism(phi)` (attribute)  
**Returns:** an integer  
The degree of the morphism  $phi$  of graded rings.  
(no method installed)

### 4.4.4 CoordinateRingOfGraph (for ring maps)

- ▷ `CoordinateRingOfGraph(phi)` (attribute)  
**Returns:** a homalg ring  
The coordinate ring of the graph of the ring map  $phi$ .

## 4.5 Ring Maps: Operations and Functions

## Chapter 5

# Matrices

### 5.1 Matrices: Category and Representations

#### 5.1.1 IsHomalgMatrix

- ▷ `IsHomalgMatrix( $A$ )` (Category)  
**Returns:** true or false  
The GAP category of homalg matrices.

```
Code
DeclareCategory( "IsHomalgMatrix",
  IsMatrixObj and
  IsAttributeStoringRep );
```

#### 5.1.2 IsHomalgInternalMatrixRep

- ▷ `IsHomalgInternalMatrixRep( $A$ )` (Representation)  
**Returns:** true or false  
The internal representation of homalg matrices.  
(It is a representation of the GAP category `IsHomalgMatrix` (5.1.1).)

### 5.2 Matrices: Constructors

#### 5.2.1 HomalgInitialMatrix (constructor for initial matrices filled with zeros)

- ▷ `HomalgInitialMatrix( $m$ ,  $n$ ,  $R$ )` (function)  
**Returns:** a homalg matrix  
A mutable unevaluated initial  $m \times n$  homalg matrix filled with zeros over the homalg ring  $R$ . This construction is useful in case one wants to define a matrix by assigning its nonzero entries. The property `IsInitialMatrix` (5.3.26) is reset as soon as the matrix is evaluated. New computed properties or attributes of the matrix won't be cached, until the matrix is explicitly made immutable using `( $\rightarrow$  MakeImmutable (Reference: MakeImmutable))`.

```
Example
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> z := HomalgInitialMatrix( 2, 3, ZZ );
<An initial 2 x 3 matrix over an internal ring>
```

```
gap> HasIsZero( z );
false
gap> IsZero( z );
true
gap> z;
<A 2 x 3 mutable matrix over an internal ring>
gap> HasIsZero( z );
false
```

Example

```
gap> n := HomalgInitialMatrix( 2, 3, ZZ );
<An initial 2 x 3 matrix over an internal ring>
gap> SetMatElm( n, 1, 1, "1" );
gap> SetMatElm( n, 2, 3, "1" );
gap> MakeImmutable( n );
<A 2 x 3 matrix over an internal ring>
gap> Display( n );
[ [ 1, 0, 0 ],
  [ 0, 0, 1 ] ]
gap> IsZero( n );
false
gap> n;
<A non-zero 2 x 3 matrix over an internal ring>
```

## 5.2.2 HomalgInitialIdentityMatrix (constructor for initial quadratic matrices with ones on the diagonal)

▷ HomalgInitialIdentityMatrix( $m$ ,  $R$ ) (function)

**Returns:** a homalg matrix

A mutable unevaluated initial  $m \times m$  homalg quadratic matrix with ones on the diagonal over the homalg ring  $R$ . This construction is useful in case one wants to define an elementary matrix by assigning its off-diagonal nonzero entries. The property `IsInitialIdentityMatrix` (5.3.27) is reset as soon as the matrix is evaluated. New computed properties or attributes of the matrix won't be cached, until the matrix is explicitly made immutable using ( $\rightarrow$  `MakeImmutable` (**Reference: MakeImmutable**)).

Example

```
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> id := HomalgInitialIdentityMatrix( 3, ZZ );
<An initial identity 3 x 3 matrix over an internal ring>
gap> HasIsOne( id );
false
gap> IsOne( id );
true
gap> id;
<A 3 x 3 mutable matrix over an internal ring>
gap> HasIsOne( id );
false
```

Example

```
gap> e := HomalgInitialIdentityMatrix( 3, ZZ );
<An initial identity 3 x 3 matrix over an internal ring>
```

```

gap> SetMatElm( e, 1, 2, "1" );
gap> SetMatElm( e, 2, 1, "-1" );
gap> MakeImmutable( e );
<A 3 x 3 matrix over an internal ring>
gap> Display( e );
[ [ 1, 1, 0 ],
  [ -1, 1, 0 ],
  [ 0, 0, 1 ] ]
gap> IsOne( e );
false
gap> e;
<A 3 x 3 matrix over an internal ring>

```

### 5.2.3 HomalgZeroMatrix (constructor for zero matrices)

▷ HomalgZeroMatrix( $m, n, R$ )

(function)

**Returns:** a homalg matrix

An immutable unevaluated  $m \times n$  homalg zero matrix over the homalg ring  $R$ .

Example

```

gap> ZZ := HomalgRingOfIntegers( );
Z
gap> z := HomalgZeroMatrix( 2, 3, ZZ );
<An unevaluated 2 x 3 zero matrix over an internal ring>
gap> Display( z );
[ [ 0, 0, 0 ],
  [ 0, 0, 0 ] ]
gap> z;
<A 2 x 3 zero matrix over an internal ring>

```

### 5.2.4 HomalgIdentityMatrix (constructor for identity matrices)

▷ HomalgIdentityMatrix( $m, R$ )

(function)

**Returns:** a homalg matrix

An immutable unevaluated  $m \times m$  homalg identity matrix over the homalg ring  $R$ .

Example

```

gap> ZZ := HomalgRingOfIntegers( );
Z
gap> id := HomalgIdentityMatrix( 3, ZZ );
<An unevaluated 3 x 3 identity matrix over an internal ring>
gap> Display( id );
[ [ 1, 0, 0 ],
  [ 0, 1, 0 ],
  [ 0, 0, 1 ] ]
gap> id;
<A 3 x 3 identity matrix over an internal ring>

```

### 5.2.5 HomalgVoidMatrix (constructor for void matrices)

▷ HomalgVoidMatrix( $[m, ] [n, ]R$ )

(function)

**Returns:** a homalg matrix

A void  $m \times n$  homalg matrix.

### 5.2.6 HomalgMatrix (constructor for matrices using a listlist)

- ▷ HomalgMatrix(*l*list, *R*) (function)
- ▷ HomalgMatrix(*list*, *m*, *n*, *R*) (function)
- ▷ HomalgMatrix(*str\_l*list, *R*) (function)
- ▷ HomalgMatrix(*str\_list*, *m*, *n*, *R*) (function)

**Returns:** a homalg matrix

An immutable evaluated  $m \times n$  homalg matrix over the homalg ring *R*.

Example

```
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> m := HomalgMatrix( [ [ 1, 2, 3 ], [ 4, 5, 6 ] ], ZZ );
<A 2 x 3 matrix over an internal ring>
gap> Display( m );
[ [ 1, 2, 3 ],
  [ 4, 5, 6 ] ]
```

Example

```
gap> m := HomalgMatrix( [ [ 1, 2, 3 ], [ 4, 5, 6 ] ], 2, 3, ZZ );
<A 2 x 3 matrix over an internal ring>
gap> Display( m );
[ [ 1, 2, 3 ],
  [ 4, 5, 6 ] ]
```

Example

```
gap> m := HomalgMatrix( [ 1, 2, 3, 4, 5, 6 ], 2, 3, ZZ );
<A 2 x 3 matrix over an internal ring>
gap> Display( m );
[ [ 1, 2, 3 ],
  [ 4, 5, 6 ] ]
```

Example

```
gap> m := HomalgMatrix( "[ [ 1, 2, 3 ], [ 4, 5, 6 ] ]", ZZ );
<A 2 x 3 matrix over an internal ring>
gap> Display( m );
[ [ 1, 2, 3 ],
  [ 4, 5, 6 ] ]
```

Example

```
gap> m := HomalgMatrix( "[ [ 1, 2, 3 ], [ 4, 5, 6 ] ]", 2, 3, ZZ );
<A 2 x 3 matrix over an internal ring>
gap> Display( m );
[ [ 1, 2, 3 ],
  [ 4, 5, 6 ] ]
```

It is nevertheless recommended to use the following form to create homalg matrices. This form can also be used to define external matrices. Since whitespaces (→ **Reference: Whitespaces**) are ignored, they can be used as optical delimiters:

Example

```
gap> m := HomalgMatrix( "[ 1, 2, 3, 4, 5, 6 ]", 2, 3, ZZ );
<A 2 x 3 matrix over an internal ring>
gap> Display( m );
[ [ 1, 2, 3 ],
  [ 4, 5, 6 ] ]
```

One can split the input string over several lines using the backslash character `'\'` to end each line

Example

```
gap> m := HomalgMatrix( "[ \
> 1, 2, 3, \
> 4, 5, 6 \
> ]", 2, 3, ZZ );
<A 2 x 3 matrix over an internal ring>
gap> Display( m );
[ [ 1, 2, 3 ],
  [ 4, 5, 6 ] ]
```

### 5.2.7 HomalgDiagonalMatrix (constructor for diagonal matrices)

▷ `HomalgDiagonalMatrix(diag, R)` (function)

**Returns:** a homalg matrix

An immutable unevaluated diagonal homalg matrix over the homalg ring  $R$ . The diagonal consists of the entries of the list *diag*.

Example

```
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> d := HomalgDiagonalMatrix( [ 1, 2, 3 ], ZZ );
<An unevaluated diagonal 3 x 3 matrix over an internal ring>
gap> Display( d );
[ [ 1, 0, 0 ],
  [ 0, 2, 0 ],
  [ 0, 0, 3 ] ]
gap> d;
<A diagonal 3 x 3 matrix over an internal ring>
```

### 5.2.8 `\*` (copy a matrix over a different ring)

▷ `\*(R, mat)` (operation)

▷ `\*(mat, R)` (operation)

**Returns:** a homalg matrix

An immutable evaluated homalg matrix over the homalg ring  $R$  having the same entries as the matrix *mat*. Syntax:  $R * mat$  or  $mat * R$

Example

```
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> Z4 := ZZ / 4;
Z/( 4 )
gap> Display( Z4 );
<A residue class ring>
gap> d := HomalgDiagonalMatrix( [ 2 .. 4 ], ZZ );
<An unevaluated diagonal 3 x 3 matrix over an internal ring>
gap> d2 := Z4 * d; ## or d2 := d * Z4;
<A 3 x 3 matrix over a residue class ring>
gap> Display( d2 );
[ [ 2, 0, 0 ],
  [ 0, 3, 0 ],
  [ 0, 0, 4 ] ]
```

```

modulo [ 4 ]
gap> d;
<A diagonal 3 x 3 matrix over an internal ring>
gap> ZeroRows( d );
[ ]
gap> ZeroRows( d2 );
[ 3 ]
gap> d;
<A non-zero diagonal 3 x 3 matrix over an internal ring>
gap> d2;
<A non-zero 3 x 3 matrix over a residue class ring>

```

## 5.3 Matrices: Properties

### 5.3.1 IsZero (for matrices)

▷ IsZero( $A$ ) (property)

**Returns:** true or false

Check if the homalg matrix  $A$  is a zero matrix, taking possible ring relations into account.

(for the installed standard method see IsZeroMatrix (B.1.16))

Example

```

gap> ZZ := HomalgRingOfIntegers( );
Z
gap> A := HomalgMatrix( "[ 2 ]", ZZ );
<A 1 x 1 matrix over an internal ring>
gap> Z2 := ZZ / 2;
Z/( 2 )
gap> A := Z2 * A;
<A 1 x 1 matrix over a residue class ring>
gap> Display( A );
[ [ 2 ] ]

modulo [ 2 ]
gap> IsZero( A );
true

```

### 5.3.2 IsOne

▷ IsOne( $A$ ) (property)

**Returns:** true or false

Check if the homalg matrix  $A$  is an identity matrix, taking possible ring relations into account.

(for the installed standard method see IsIdentityMatrix (B.2.2))

### 5.3.3 IsUnitFree

▷ IsUnitFree( $A$ ) (property)

**Returns:** true or false

$A$  is a homalg matrix.



### 5.3.4 IsPermutationMatrix

- ▷ IsPermutationMatrix( $A$ ) (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.5 IsSpecialSubidentityMatrix

- ▷ IsSpecialSubidentityMatrix( $A$ ) (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.6 IsSubidentityMatrix

- ▷ IsSubidentityMatrix( $A$ ) (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.7 IsLeftRegular

- ▷ IsLeftRegular( $A$ ) (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.8 IsRightRegular

- ▷ IsRightRegular( $A$ ) (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.9 IsInvertibleMatrix

- ▷ IsInvertibleMatrix( $A$ ) (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.10 IsLeftInvertibleMatrix

- ▷ IsLeftInvertibleMatrix( $A$ ) (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.11 IsRightInvertibleMatrix

- ▷ IsRightInvertibleMatrix( $A$ ) (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.12 IsEmptyMatrix

- ▷ `IsEmptyMatrix(A)` (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.13 IsDiagonalMatrix

- ▷ `IsDiagonalMatrix(A)` (property)  
**Returns:** true or false  
 Check if the homalg matrix  $A$  is an identity matrix, taking possible ring relations into account.  
 (for the installed standard method see `IsDiagonalMatrix` (B.2.3))

### 5.3.14 IsScalarMatrix

- ▷ `IsScalarMatrix(A)` (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.15 IsUpperTriangularMatrix

- ▷ `IsUpperTriangularMatrix(A)` (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.16 IsLowerTriangularMatrix

- ▷ `IsLowerTriangularMatrix(A)` (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.17 IsStrictUpperTriangularMatrix

- ▷ `IsStrictUpperTriangularMatrix(A)` (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.18 IsStrictLowerTriangularMatrix

- ▷ `IsStrictLowerTriangularMatrix(A)` (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.19 IsUpperStairCaseMatrix

- ▷ `IsUpperStairCaseMatrix(A)` (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.20 IsLowerStairCaseMatrix

- ▷ IsLowerStairCaseMatrix( $A$ ) (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.21 IsTriangularMatrix

- ▷ IsTriangularMatrix( $A$ ) (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.22 IsBasisOfRowsMatrix

- ▷ IsBasisOfRowsMatrix( $A$ ) (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.23 IsBasisOfColumnsMatrix

- ▷ IsBasisOfColumnsMatrix( $A$ ) (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.24 IsReducedBasisOfRowsMatrix

- ▷ IsReducedBasisOfRowsMatrix( $A$ ) (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.25 IsReducedBasisOfColumnsMatrix

- ▷ IsReducedBasisOfColumnsMatrix( $A$ ) (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.26 IsInitialMatrix

- ▷ IsInitialMatrix( $A$ ) (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.27 IsInitialIdentityMatrix

- ▷ IsInitialIdentityMatrix( $A$ ) (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

### 5.3.28 IsVoidMatrix

- ▷ `IsVoidMatrix(A)` (property)  
**Returns:** true or false  
 $A$  is a homalg matrix.

## 5.4 Matrices: Attributes

### 5.4.1 NrRows

- ▷ `NrRows(A)` (attribute)  
**Returns:** a nonnegative integer  
 The number of rows of the matrix  $A$ .  
 (for the installed standard method see `NrRows` (B.1.17))

### 5.4.2 NrColumns

- ▷ `NrColumns(A)` (attribute)  
**Returns:** a nonnegative integer  
 The number of columns of the matrix  $A$ .  
 (for the installed standard method see `NrColumns` (B.1.18))

### 5.4.3 DeterminantMat

- ▷ `DeterminantMat(A)` (attribute)  
**Returns:** a ring element  
 The determinant of the quadratic matrix  $A$ .  
 You can invoke it with `Determinant(A)`.  
 (for the installed standard method see `Determinant` (B.1.19))

### 5.4.4 ZeroRows

- ▷ `ZeroRows(A)` (attribute)  
**Returns:** a (possibly empty) list of positive integers  
 The list of zero rows of the matrix  $A$ .  
 (for the installed standard method see `ZeroRows` (B.2.4))

### 5.4.5 ZeroColumns

- ▷ `ZeroColumns(A)` (attribute)  
**Returns:** a (possibly empty) list of positive integers  
 The list of zero columns of the matrix  $A$ .  
 (for the installed standard method see `ZeroColumns` (B.2.5))

### 5.4.6 NonZeroRows

- ▷ `NonZeroRows(A)` (attribute)  
**Returns:** a (possibly empty) list of positive integers  
 The list of nonzero rows of the matrix  $A$ .

### 5.4.7 NonZeroColumns

- ▷ `NonZeroColumns(A)` (attribute)  
**Returns:** a (possibly empty) list of positive integers  
 The list of nonzero columns of the matrix  $A$ .

### 5.4.8 PositionOfFirstNonZeroEntryPerRow

- ▷ `PositionOfFirstNonZeroEntryPerRow(A)` (attribute)  
**Returns:** a list of nonnegative integers  
 The list of positions of the first nonzero entry per row of the matrix  $A$ , else zero.

### 5.4.9 PositionOfFirstNonZeroEntryPerColumn

- ▷ `PositionOfFirstNonZeroEntryPerColumn(A)` (attribute)  
**Returns:** a list of nonnegative integers  
 The list of positions of the first nonzero entry per column of the matrix  $A$ , else zero.

### 5.4.10 RowRankOfMatrix

- ▷ `RowRankOfMatrix(A)` (attribute)  
**Returns:** a nonnegative integer  
 The row rank of the matrix  $A$ .

### 5.4.11 ColumnRankOfMatrix

- ▷ `ColumnRankOfMatrix(A)` (attribute)  
**Returns:** a nonnegative integer  
 The column rank of the matrix  $A$ .

### 5.4.12 LeftInverse

- ▷ `LeftInverse(M)` (attribute)  
**Returns:** a homalg matrix  
 A left inverse  $C$  of the matrix  $M$ . If no left inverse exists then `false` is returned. ( $\rightarrow$  `RightDivide` (5.5.45))  
 (for the installed standard method see `LeftInverse` (5.5.2))

### 5.4.13 RightInverse

- ▷ `RightInverse(M)` (attribute)  
**Returns:** a homalg matrix  
 A right inverse  $C$  of the matrix  $M$ . If no right inverse exists then `false` is returned. ( $\rightarrow$  `LeftDivide` (5.5.46))  
 (for the installed standard method see `RightInverse` (5.5.3))

#### 5.4.14 CoefficientsOfUnreducedNumeratorOfHilbertPoincareSeries

- ▷ `CoefficientsOfUnreducedNumeratorOfHilbertPoincareSeries(A)` (attribute)  
**Returns:** a list of integers  
 A is a `homalg` matrix (row convention).

#### 5.4.15 CoefficientsOfNumeratorOfHilbertPoincareSeries

- ▷ `CoefficientsOfNumeratorOfHilbertPoincareSeries(A)` (attribute)  
**Returns:** a list of integers  
 A is a `homalg` matrix (row convention).

#### 5.4.16 UnreducedNumeratorOfHilbertPoincareSeries

- ▷ `UnreducedNumeratorOfHilbertPoincareSeries(A)` (attribute)  
**Returns:** a univariate polynomial with rational coefficients  
 A is a `homalg` matrix (row convention).

#### 5.4.17 NumeratorOfHilbertPoincareSeries

- ▷ `NumeratorOfHilbertPoincareSeries(A)` (attribute)  
**Returns:** a univariate polynomial with rational coefficients  
 A is a `homalg` matrix (row convention).

#### 5.4.18 HilbertPoincareSeries

- ▷ `HilbertPoincareSeries(A)` (attribute)  
**Returns:** a univariate rational function with rational coefficients  
 A is a `homalg` matrix (row convention).

#### 5.4.19 HilbertPolynomial

- ▷ `HilbertPolynomial(A)` (attribute)  
**Returns:** a univariate polynomial with rational coefficients  
 A is a `homalg` matrix (row convention).

#### 5.4.20 AffineDimension

- ▷ `AffineDimension(A)` (attribute)  
**Returns:** a nonnegative integer  
 A is a `homalg` matrix (row convention).

#### 5.4.21 AffineDegree

- ▷ `AffineDegree(A)` (attribute)  
**Returns:** a nonnegative integer  
 A is a `homalg` matrix (row convention).

### 5.4.22 ProjectiveDegree

- ▷ `ProjectiveDegree(A)` (attribute)  
**Returns:** a nonnegative integer  
 $A$  is a homalg matrix (row convention).

### 5.4.23 ConstantTermOfHilbertPolynomialn

- ▷ `ConstantTermOfHilbertPolynomialn(A)` (attribute)  
**Returns:** an integer  
 $A$  is a homalg matrix (row convention).

### 5.4.24 MatrixOfSymbols

- ▷ `MatrixOfSymbols(A)` (attribute)  
**Returns:** an integer  
 $A$  is a homalg matrix.

## 5.5 Matrices: Operations and Functions

### 5.5.1 HomalgRing (for matrices)

- ▷ `HomalgRing(mat)` (operation)  
**Returns:** a homalg ring  
 The homalg ring of the homalg matrix  $mat$ .

Example

```
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> d := HomalgDiagonalMatrix( [ 2 .. 4 ], ZZ );
<An unevaluated diagonal 3 x 3 matrix over an internal ring>
gap> R := HomalgRing( d );
Z
gap> IsIdenticalObj( R, ZZ );
true
```

### 5.5.2 LeftInverse (for matrices)

- ▷ `LeftInverse(RI)` (method)  
**Returns:** a homalg matrix or fail  
 The left inverse of the matrix  $RI$ . The lazy version of this operation is `LeftInverseLazy` (5.5.4).  
 (→ `RightDivide` (5.5.45))

Code

```
InstallMethod( LeftInverse,
  "for homalg matrices",
  [ IsHomalgMatrix ],

  function( RI )
    local Id, LI;

    Id := HomalgIdentityMatrix( NrColumns( RI ), HomalgRing( RI ) );
```

```

LI := RightDivide( Id, RI );          ## ( cf. [BR08, Subsection 3.1.3] )

## CAUTION: for the following SetXXX RightDivide is assumed
## NOT to be lazy evaluated!!!

SetIsLeftInvertibleMatrix( RI, IsHomalgMatrix( LI ) );

if IsBool( LI ) then
  return fail;
fi;

if HasIsInvertibleMatrix( RI ) and IsInvertibleMatrix( RI ) then
  SetIsInvertibleMatrix( LI, true );
else
  SetIsRightInvertibleMatrix( LI, true );
fi;

SetRightInverse( LI, RI );

SetNrColumns( LI, NrRows( RI ) );

if NrRows( RI ) = NrColumns( RI ) then
  ## a left inverse of a ring element is unique
  ## and coincides with the right inverse
  SetRightInverse( RI, LI );
  SetLeftInverse( LI, RI );
fi;

return LI;

end );

```

### 5.5.3 RightInverse (for matrices)

▷ RightInverse(LI) (method)

**Returns:** a homalg matrix or fail

The right inverse of the matrix *LI*. The lazy version of this operation is RightInverseLazy (5.5.5). (→ LeftDivide (5.5.46))

```

Code
-----
InstallMethod( RightInverse,
  "for homalg matrices",
  [ IsHomalgMatrix ],

function( LI )
  local Id, RI;

  Id := HomalgIdentityMatrix( NrRows( LI ), HomalgRing( LI ) );

  RI := LeftDivide( LI, Id );          ## ( cf. [BR08, Subsection 3.1.3] )

  ## CAUTION: for the following SetXXX LeftDivide is assumed

```



```

## NOT to be lazy evaluated!!!

SetIsRightInvertibleMatrix( LI, IsHomalgMatrix( RI ) );

if IsBool( RI ) then
  return fail;
fi;

if HasIsInvertibleMatrix( LI ) and IsInvertibleMatrix( LI ) then
  SetIsInvertibleMatrix( RI, true );
else
  SetIsLeftInvertibleMatrix( RI, true );
fi;

SetLeftInverse( RI, LI );

SetNrRows( RI, NrColumns( LI ) );

if NrRows( LI ) = NrColumns( LI ) then
  ## a right inverse of a ring element is unique
  ## and coincides with the left inverse
  SetLeftInverse( LI, RI );
  SetRightInverse( RI, LI );
fi;

return RI;

end );

```

### 5.5.4 LeftInverseLazy (for matrices)

- ▷ `LeftInverseLazy( $M$ )` (operation)  
**Returns:** a homalg matrix  
 A lazy evaluated left inverse  $C$  of the matrix  $M$ . If no left inverse exists then `Eval( $C$ )` will issue an error.  
 (for the installed standard method see `Eval` (C.4.5))

### 5.5.5 RightInverseLazy (for matrices)

- ▷ `RightInverseLazy( $M$ )` (operation)  
**Returns:** a homalg matrix  
 A lazy evaluated right inverse  $C$  of the matrix  $M$ . If no right inverse exists then `Eval( $C$ )` will issue an error.  
 (for the installed standard method see `Eval` (C.4.6))

### 5.5.6 Involution (for matrices)

- ▷ `Involution( $M$ )` (method)  
**Returns:** a homalg matrix  
 The twisted transpose of the homalg matrix  $M$ .

(for the installed standard method see Eval (C.4.7))

### 5.5.7 CertainRows (for matrices)

- ▷ `CertainRows( $M$ ,  $plist$ )` (method)  
**Returns:** a homalg matrix  
 The matrix of which the  $i$ -th row is the  $k$ -th row of the homalg matrix  $M$ , where  $k = plist[i]$ .  
 (for the installed standard method see Eval (C.4.8))

### 5.5.8 CertainColumns (for matrices)

- ▷ `CertainColumns( $M$ ,  $plist$ )` (method)  
**Returns:** a homalg matrix  
 The matrix of which the  $j$ -th column is the  $l$ -th column of the homalg matrix  $M$ , where  $l = plist[j]$ .  
 (for the installed standard method see Eval (C.4.9))

### 5.5.9 UnionOfRows (for matrices)

- ▷ `UnionOfRows( $A$ ,  $B$ )` (method)  
**Returns:** a homalg matrix  
 Stack the two homalg matrices  $A$  and  $B$ .  
 (for the installed standard method see Eval (C.4.10))

### 5.5.10 UnionOfColumns (for matrices)

- ▷ `UnionOfColumns( $A$ ,  $B$ )` (method)  
**Returns:** a homalg matrix  
 Augment the two homalg matrices  $A$  and  $B$ .  
 (for the installed standard method see Eval (C.4.11))

### 5.5.11 DiagMat (for matrices)

- ▷ `DiagMat( $list$ )` (method)  
**Returns:** a homalg matrix  
 Build the block diagonal matrix out of the homalg matrices listed in  $list$ . An error is issued if  $list$  is empty or if one of the arguments is not a homalg matrix.  
 (for the installed standard method see Eval (C.4.12))

### 5.5.12 KroneckerMat (for matrices)

- ▷ `KroneckerMat( $A$ ,  $B$ )` (method)  
**Returns:** a homalg matrix  
 The Kronecker (or tensor) product of the two homalg matrices  $A$  and  $B$ .  
 (for the installed standard method see Eval (C.4.13))

**5.5.13  $\backslash*$  (for ring elements and matrices)**

▷  $\backslash*(a, A)$  (method)

**Returns:** a homalg matrix

The product of the ring element  $a$  with the homalg matrix  $A$  (enter:  $a * A$ );  
(for the installed standard method see Eval (C.4.14))

**5.5.14  $\backslash+$  (for matrices)**

▷  $\backslash+(A, B)$  (method)

**Returns:** a homalg matrix

The sum of the two homalg matrices  $A$  and  $B$  (enter:  $A + B$ );  
(for the installed standard method see Eval (C.4.15))

**5.5.15  $\backslash-$  (for matrices)**

▷  $\backslash-(A, B)$  (method)

**Returns:** a homalg matrix

The difference of the two homalg matrices  $A$  and  $B$  (enter:  $A - B$ );  
(for the installed standard method see Eval (C.4.16))

**5.5.16  $\backslash*$  (for composable matrices)**

▷  $\backslash*(A, B)$  (method)

**Returns:** a homalg matrix

The matrix product of the two homalg matrices  $A$  and  $B$  (enter:  $A * B$ );  
(for the installed standard method see Eval (C.4.17))

**5.5.17  $\backslash=$  (for matrices)**

▷  $\backslash=(A, B)$  (operation)

**Returns:** true or false

Check if the homalg matrices  $A$  and  $B$  are equal (enter:  $A = B$ ); taking possible ring relations into account.

(for the installed standard method see AreEqualMatrices (B.2.1))

Example

```
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> A := HomalgMatrix( "[ 1 ]", ZZ );
<A 1 x 1 matrix over an internal ring>
gap> B := HomalgMatrix( "[ 3 ]", ZZ );
<A 1 x 1 matrix over an internal ring>
gap> Z2 := ZZ / 2;
Z/( 2 )
gap> A := Z2 * A;
<A 1 x 1 matrix over a residue class ring>
gap> B := Z2 * B;
<A 1 x 1 matrix over a residue class ring>
gap> Display( A );
[[ 1 ]]
```

```

modulo [ 2 ]
gap> Display( B );
[ [ 3 ] ]

modulo [ 2 ]
gap> A = B;
true

```

### 5.5.18 GetColumnIndependentUnitPositions (for matrices)

▷ `GetColumnIndependentUnitPositions(A, poslist)` (operation)

**Returns:** a (possibly empty) list of pairs of positive integers

The list of column independent unit position of the matrix  $A$ . We say that a unit  $A[i, k]$  is column independent from the unit  $A[l, j]$  if  $i > l$  and  $A[l, k] = 0$ . The rows are scanned from top to bottom and within each row the columns are scanned from right to left searching for new units, column independent from the preceding ones. If  $A[i, k]$  is a new column independent unit then  $[i, k]$  is added to the output list. If  $A$  has no units the empty list is returned.

(for the installed standard method see `GetColumnIndependentUnitPositions` (B.2.6))

### 5.5.19 GetRowIndependentUnitPositions (for matrices)

▷ `GetRowIndependentUnitPositions(A, poslist)` (operation)

**Returns:** a (possibly empty) list of pairs of positive integers

The list of row independent unit position of the matrix  $A$ . We say that a unit  $A[k, j]$  is row independent from the unit  $A[i, l]$  if  $j > l$  and  $A[k, l] = 0$ . The columns are scanned from left to right and within each column the rows are scanned from bottom to top searching for new units, row independent from the preceding ones. If  $A[k, j]$  is a new row independent unit then  $[j, k]$  (yes  $[j, k]$ ) is added to the output list. If  $A$  has no units the empty list is returned.

(for the installed standard method see `GetRowIndependentUnitPositions` (B.2.7))

### 5.5.20 GetUnitPosition (for matrices)

▷ `GetUnitPosition(A, poslist)` (operation)

**Returns:** a (possibly empty) list of pairs of positive integers

The position  $[i, j]$  of the first unit  $A[i, j]$  in the matrix  $A$ , where the rows are scanned from top to bottom and within each row the columns are scanned from left to right. If  $A[i, j]$  is the first occurrence of a unit then the position pair  $[i, j]$  is returned. Otherwise `fail` is returned.

(for the installed standard method see `GetUnitPosition` (B.2.8))

### 5.5.21 Eliminate

▷ `Eliminate(rel, indets)` (operation)

**Returns:** a homalg matrix

Eliminate the independents  $indets$  from the matrix (or list of ring elements)  $rel$ , i.e. compute a generating set of the ideal defined as the intersection of the ideal generated by the entries of the list  $rel$  with the subring generated by all indeterminates except those in  $indets$ . by the list of indeterminates  $indets$ .

### 5.5.22 BasisOfRowModule (for matrices)

▷ `BasisOfRowModule( $M$ )` (operation)

**Returns:** a homalg matrix

Let  $R$  be the ring over which  $M$  is defined ( $R := \text{HomalgRing}(M)$ ) and  $S$  be the row span of  $M$ , i.e. the  $R$ -submodule of the free module  $R^{(1 \times \text{NrColumns}(M))}$  spanned by the rows of  $M$ . A solution to the “submodule membership problem” is an algorithm which can decide if an element  $m$  in  $R^{(1 \times \text{NrColumns}(M))}$  is contained in  $S$  or not. And exactly like the Gaussian (resp. Hermite) normal form when  $R$  is a field (resp. principal ideal ring), the row span of the resulting matrix  $B$  coincides with the row span  $S$  of  $M$ , and computing  $B$  is typically the first step of such an algorithm. (→ Appendix A)

### 5.5.23 BasisOfColumnModule (for matrices)

▷ `BasisOfColumnModule( $M$ )` (operation)

**Returns:** a homalg matrix

Let  $R$  be the ring over which  $M$  is defined ( $R := \text{HomalgRing}(M)$ ) and  $S$  be the column span of  $M$ , i.e. the  $R$ -submodule of the free module  $R^{(\text{NrRows}(M) \times 1)}$  spanned by the columns of  $M$ . A solution to the “submodule membership problem” is an algorithm which can decide if an element  $m$  in  $R^{(\text{NrRows}(M) \times 1)}$  is contained in  $S$  or not. And exactly like the Gaussian (resp. Hermite) normal form when  $R$  is a field (resp. principal ideal ring), the column span of the resulting matrix  $B$  coincides with the column span  $S$  of  $M$ , and computing  $B$  is typically the first step of such an algorithm. (→ Appendix A)

### 5.5.24 DecideZeroRows (for pairs of matrices)

▷ `DecideZeroRows( $A, B$ )` (operation)

**Returns:** a homalg matrix

Let  $A$  and  $B$  be matrices having the same number of columns and defined over the same ring  $R$  ( $R := \text{HomalgRing}(A)$ ) and  $S$  be the row span of  $B$ , i.e. the  $R$ -submodule of the free module  $R^{(1 \times \text{NrColumns}(B))}$  spanned by the rows of  $B$ . The result is a matrix  $C$  having the same shape as  $A$ , for which the  $i$ -th row  $C^i$  is equivalent to the  $i$ -th row  $A^i$  of  $A$  modulo  $S$ , i.e.  $C^i - A^i$  is an element of the row span  $S$  of  $B$ . Moreover, the row  $C^i$  is zero, if and only if the row  $A^i$  is an element of  $S$ . So `DecideZeroRows` decides which rows of  $A$  are zero modulo the rows of  $B$ . (→ Appendix A)

### 5.5.25 DecideZeroColumns (for pairs of matrices)

▷ `DecideZeroColumns( $A, B$ )` (operation)

**Returns:** a homalg matrix

Let  $A$  and  $B$  be matrices having the same number of rows and defined over the same ring  $R$  ( $R := \text{HomalgRing}(A)$ ) and  $S$  be the column span of  $B$ , i.e. the  $R$ -submodule of the free module  $R^{(\text{NrRows}(B) \times 1)}$  spanned by the columns of  $B$ . The result is a matrix  $C$  having the same shape as  $A$ , for which the  $i$ -th column  $C_i$  is equivalent to the  $i$ -th column  $A_i$  of  $A$  modulo  $S$ , i.e.  $C_i - A_i$  is an element of the column span  $S$  of  $B$ . Moreover, the column  $C_i$  is zero, if and only if the column  $A_i$  is an element of  $S$ . So `DecideZeroColumns` decides which columns of  $A$  are zero modulo the columns of  $B$ . (→ Appendix A)

### 5.5.26 SyzygiesGeneratorsOfRows (for matrices)

▷ SyzygiesGeneratorsOfRows( $M$ ) (operation)

**Returns:** a homalg matrix

Let  $R$  be the ring over which  $M$  is defined ( $R := \text{HomalgRing}(M)$ ). The matrix of row syzygies  $\text{SyzygiesGeneratorsOfRows}(M)$  is a matrix whose rows span the left kernel of  $M$ , i.e. the  $R$ -submodule of the free module  $R^{(1 \times \text{NrRows}(M))}$  consisting of all rows  $X$  satisfying  $XM = 0$ . (→ Appendix A)

### 5.5.27 SyzygiesGeneratorsOfColumns (for matrices)

▷ SyzygiesGeneratorsOfColumns( $M$ ) (operation)

**Returns:** a homalg matrix

Let  $R$  be the ring over which  $M$  is defined ( $R := \text{HomalgRing}(M)$ ). The matrix of column syzygies  $\text{SyzygiesGeneratorsOfColumns}(M)$  is a matrix whose columns span the right kernel of  $M$ , i.e. the  $R$ -submodule of the free module  $R^{(\text{NrColumns}(M) \times 1)}$  consisting of all columns  $X$  satisfying  $MX = 0$ . (→ Appendix A)

### 5.5.28 SyzygiesGeneratorsOfRows (for pairs of matrices)

▷ SyzygiesGeneratorsOfRows( $M, M2$ ) (operation)

**Returns:** a homalg matrix

Let  $R$  be the ring over which  $M$  is defined ( $R := \text{HomalgRing}(M)$ ). The matrix of *relative* row syzygies  $\text{SyzygiesGeneratorsOfRows}(M, M2)$  is a matrix whose rows span the left kernel of  $M$  modulo  $M2$ , i.e. the  $R$ -submodule of the free module  $R^{(1 \times \text{NrRows}(M))}$  consisting of all rows  $X$  satisfying  $XM + YM2 = 0$  for some row  $Y \in R^{(1 \times \text{NrRows}(M2))}$ . (→ Appendix A)

### 5.5.29 SyzygiesGeneratorsOfColumns (for pairs of matrices)

▷ SyzygiesGeneratorsOfColumns( $M, M2$ ) (operation)

**Returns:** a homalg matrix

Let  $R$  be the ring over which  $M$  is defined ( $R := \text{HomalgRing}(M)$ ). The matrix of *relative* column syzygies  $\text{SyzygiesGeneratorsOfColumns}(M, M2)$  is a matrix whose columns span the right kernel of  $M$  modulo  $M2$ , i.e. the  $R$ -submodule of the free module  $R^{(\text{NrColumns}(M) \times 1)}$  consisting of all columns  $X$  satisfying  $MX + M2Y = 0$  for some column  $Y \in R^{(\text{NrColumns}(M2) \times 1)}$ . (→ Appendix A)

### 5.5.30 ReducedBasisOfRowModule (for matrices)

▷ ReducedBasisOfRowModule( $M$ ) (operation)

**Returns:** a homalg matrix

Like  $\text{BasisOfRowModule}(M)$  but where the matrix  $\text{SyzygiesGeneratorsOfRows}(\text{ReducedBasisOfRowModule}(M))$  contains no units. This can easily be achieved starting from  $B := \text{BasisOfRowModule}(M)$  (and using  $\text{GetColumnIndependentUnitPositions}$  (5.5.18) applied to the matrix of row syzygies of  $B$ , etc). (→ Appendix A)

### 5.5.31 ReducedBasisOfColumnModule (for matrices)

▷ ReducedBasisOfColumnModule( $M$ ) (operation)

**Returns:** a homalg matrix

Like `BasisOfColumnModule( M )` but where the matrix `SyzygiesGeneratorsOfColumns( ReducedBasisOfColumnModule( M ) )` contains no units. This can easily be achieved starting from  $B := \text{BasisOfColumnModule}( M )$  (and using `GetRowIndependentUnitPositions` (5.5.19) applied to the matrix of column syzygies of  $B$ , etc.). (→ Appendix A)

### 5.5.32 ReducedSyzygiesGeneratorsOfRows (for matrices)

▷ `ReducedSyzygiesGeneratorsOfRows( M )` (operation)

**Returns:** a homalg matrix

Like `SyzygiesGeneratorsOfRows( M )` but where the matrix `SyzygiesGeneratorsOfRows( ReducedSyzygiesGeneratorsOfRows( M ) )` contains no units. This can easily be achieved starting from  $C := \text{SyzygiesGeneratorsOfRows}( M )$  (and using `GetColumnIndependentUnitPositions` (5.5.18) applied to the matrix of row syzygies of  $C$ , etc.). (→ Appendix A)

### 5.5.33 ReducedSyzygiesGeneratorsOfColumns (for matrices)

▷ `ReducedSyzygiesGeneratorsOfColumns( M )` (operation)

**Returns:** a homalg matrix

Like `SyzygiesGeneratorsOfColumns( M )` but where the matrix `SyzygiesGeneratorsOfColumns( ReducedSyzygiesGeneratorsOfColumns( M ) )` contains no units. This can easily be achieved starting from  $C := \text{SyzygiesGeneratorsOfColumns}( M )$  (and using `GetRowIndependentUnitPositions` (5.5.19) applied to the matrix of column syzygies of  $C$ , etc.). (→ Appendix A)

### 5.5.34 BasisOfRowsCoeff (for matrices)

▷ `BasisOfRowsCoeff( M, T )` (operation)

**Returns:** a homalg matrix

Returns  $B := \text{BasisOfRowModule}( M )$  and assigns the *void* matrix  $T$  (→ `HomalgVoidMatrix` (5.2.5)) such that  $B = TM$ . (→ Appendix A)

### 5.5.35 BasisOfColumnsCoeff (for matrices)

▷ `BasisOfColumnsCoeff( M, T )` (operation)

**Returns:** a homalg matrix

Returns  $B := \text{BasisOfRowModule}( M )$  and assigns the *void* matrix  $T$  (→ `HomalgVoidMatrix` (5.2.5)) such that  $B = MT$ . (→ Appendix A)

### 5.5.36 DecideZeroRowsEffectively (for pairs of matrices)

▷ `DecideZeroRowsEffectively( A, B, T )` (operation)

**Returns:** a homalg matrix

Returns  $M := \text{DecideZeroRows}( A, B )$  and assigns the *void* matrix  $T$  (→ `HomalgVoidMatrix` (5.2.5)) such that  $M = A + TB$ . (→ Appendix A)

### 5.5.37 DecideZeroColumnsEffectively (for pairs of matrices)

▷ `DecideZeroColumnsEffectively(A, B, T)` (operation)

**Returns:** a homalg matrix

Returns  $M := \text{DecideZeroColumns}(A, B)$  and assigns the *void* matrix  $T$  ( $\rightarrow$  `HomalgVoidMatrix` (5.2.5)) such that  $M = A + BT$ . ( $\rightarrow$  Appendix A)

### 5.5.38 BasisOfRows (for matrices)

▷ `BasisOfRows(M)` (operation)

▷ `BasisOfRows(M, T)` (operation)

**Returns:** a homalg matrix

With one argument it is a synonym of `BasisOfRowModule` (5.5.22). with two arguments it is a synonym of `BasisOfRowsCoeff` (5.5.34).

### 5.5.39 BasisOfColumns (for matrices)

▷ `BasisOfColumns(M)` (operation)

▷ `BasisOfColumns(M, T)` (operation)

**Returns:** a homalg matrix

With one argument it is a synonym of `BasisOfColumnModule` (5.5.23). with two arguments it is a synonym of `BasisOfColumnsCoeff` (5.5.35).

### 5.5.40 DecideZero (for matrices and relations)

▷ `DecideZero(mat, rel)` (operation)

**Returns:** a homalg matrix

Code

```
InstallMethod( DecideZero,
  "for sets of ring relations",
  [ IsHomalgMatrix, IsHomalgRingRelations ],

  function( mat, rel )

    return DecideZero( mat, MatrixOfRelations( rel ) );

  end );
```

### 5.5.41 SyzygiesOfRows (for matrices)

▷ `SyzygiesOfRows(M)` (operation)

▷ `SyzygiesOfRows(M, M2)` (operation)

**Returns:** a homalg matrix

With one argument it is a synonym of `SyzygiesGeneratorsOfRows` (5.5.26). with two arguments it is a synonym of `SyzygiesGeneratorsOfRows` (5.5.28).



### 5.5.42 SyzygiesOfColumns (for matrices)

- ▷ `SyzygiesOfColumns( $M$ )` (operation)
- ▷ `SyzygiesOfColumns( $M, M2$ )` (operation)

**Returns:** a homalg matrix

With one argument it is a synonym of `SyzygiesGeneratorsOfColumns` (5.5.27). with two arguments it is a synonym of `SyzygiesGeneratorsOfColumns` (5.5.29).

### 5.5.43 ReducedSyzygiesOfRows (for matrices)

- ▷ `ReducedSyzygiesOfRows( $M$ )` (operation)
- ▷ `ReducedSyzygiesOfRows( $M, M2$ )` (operation)

**Returns:** a homalg matrix

With one argument it is a synonym of `ReducedSyzygiesGeneratorsOfRows` (5.5.32). With two arguments it calls `ReducedBasisOfRowModule( SyzygiesGeneratorsOfRows(  $M, M2$  ) )`. ( $\rightarrow$  `ReducedBasisOfRowModule` (5.5.30) and `SyzygiesGeneratorsOfRows` (5.5.28))

### 5.5.44 ReducedSyzygiesOfColumns (for matrices)

- ▷ `ReducedSyzygiesOfColumns( $M$ )` (operation)
- ▷ `ReducedSyzygiesOfColumns( $M, M2$ )` (operation)

**Returns:** a homalg matrix

With one argument it is a synonym of `ReducedSyzygiesGeneratorsOfColumns` (5.5.33). With two arguments it calls `ReducedBasisOfColumnModule( SyzygiesGeneratorsOfColumns(  $M, M2$  ) )`. ( $\rightarrow$  `ReducedBasisOfColumnModule` (5.5.31) and `SyzygiesGeneratorsOfColumns` (5.5.29))

### 5.5.45 RightDivide (for pairs of matrices)

- ▷ `RightDivide( $B, A$ )` (operation)

**Returns:** a homalg matrix or fail

Let  $B$  and  $A$  be matrices having the same number of columns and defined over the same ring. The matrix `RightDivide( $B, A$ )` is a particular solution of the inhomogeneous (one sided) linear system of equations  $XA = B$  in case it is solvable. Otherwise fail is returned. The name `RightDivide` suggests “ $X = BA^{-1}$ ”. This generalizes `LeftInverse` (5.5.2) for which  $B$  becomes the identity matrix. ( $\rightarrow$  `SyzygiesGeneratorsOfRows` (5.5.26))

### 5.5.46 LeftDivide (for pairs of matrices)

- ▷ `LeftDivide( $A, B$ )` (operation)

**Returns:** a homalg matrix or fail

Let  $A$  and  $B$  be matrices having the same number of rows and defined over the same ring. The matrix `LeftDivide( $A, B$ )` is a particular solution of the inhomogeneous (one sided) linear system of equations  $AX = B$  in case it is solvable. Otherwise fail is returned. The name `LeftDivide` suggests “ $X = A^{-1}B$ ”. This generalizes `RightInverse` (5.5.3) for which  $B$  becomes the identity matrix. ( $\rightarrow$  `SyzygiesGeneratorsOfColumns` (5.5.27))

### 5.5.47 RightDivide (for triples of matrices)

▷ `RightDivide(B, A, L)` (operation)

**Returns:** a homalg matrix or fail

Let  $B$ ,  $A$  and  $L$  be matrices having the same number of columns and defined over the same ring. The matrix `RightDivide(B, A, L)` is a particular solution of the inhomogeneous (one sided) linear system of equations  $XA + YL = B$  in case it is solvable (for some  $Y$  which is forgotten). Otherwise fail is returned. The name `RightDivide` suggests “ $X = BA^{-1}$  modulo  $L$ ”. (Cf. [BR08, Subsection 3.1.1])

Code

```

InstallMethod( RightDivide,
  "for homalg matrices",
  [ IsHomalgMatrix, IsHomalgMatrix, IsHomalgMatrix ],

function( B, A, L )      ## CAUTION: Do not use lazy evaluation here!!!
  local R, BL, ZA, AL, CA, IAL, ZB, CB, NF, X;

  R := HomalgRing( B );

  BL := BasisOfRows( L );

  ## first reduce A modulo L
  ZA := DecideZeroRows( A, BL );

  AL := UnionOfRows( ZA, BL );

  ## CA * AL = IAL
  CA := HomalgVoidMatrix( R );
  IAL := BasisOfRows( AL, CA );

  ## also reduce B modulo L
  ZB := DecideZeroRows( B, BL );

  ## knowing this will avoid computations
  IsOne( IAL );

  ## IsSpecialSubidentityMatrix( IAL );      ## does not increase performance

  ## NF = ZB + CB * IAL
  CB := HomalgVoidMatrix( R );
  NF := DecideZeroRowsEffectively( ZB, IAL, CB );

  ## NF <> 0
  if not IsZero( NF ) then
    return fail;
  fi;

  ## CD = -CB * CA => CD * A = B
  X := -CB * CertainColumns( CA, [ 1 .. NrRows( A ) ] );

  ## check assertion
  Assert( 5, IsZero( DecideZeroRows( X * A - B, BL ) ) );

```

```

return X;

## technical: -CB * CA := (-CB) * CA and COLEM should take over
## since CB := -matrix

end );

```

### 5.5.48 LeftDivide (for triples of matrices)

▷ LeftDivide( $A, B, L$ ) (operation)

**Returns:** a homalg matrix or fail

Let  $A, B$  and  $L$  be matrices having the same number of columns and defined over the same ring. The matrix LeftDivide( $A, B, L$ ) is a particular solution of the inhomogeneous (one sided) linear system of equations  $AX + LY = B$  in case it is solvable (for some  $Y$  which is forgotten). Otherwise fail is returned. The name LeftDivide suggests “ $X = A^{-1}B$  modulo  $L$ ”. (Cf. [BR08, Subsection 3.1.1])

```

Code
-----
InstallMethod( LeftDivide,
  "for homalg matrices",
  [ IsHomalgMatrix, IsHomalgMatrix, IsHomalgMatrix ],

function( A, B, L )      ## CAUTION: Do not use lazy evaluation here!!!
  local R, BL, ZA, AL, CA, IAL, ZB, CB, NF, X;

  R := HomalgRing( B );

  BL := BasisOfColumns( L );

  ## first reduce A modulo L
  ZA := DecideZeroColumns( A, BL );

  AL := UnionOfColumns( ZA, BL );

  ## AL * CA = IAL
  CA := HomalgVoidMatrix( R );
  IAL := BasisOfColumns( AL, CA );

  ## also reduce B modulo L
  ZB := DecideZeroColumns( B, BL );

  ## knowing this will avoid computations
  IsOne( IAL );

  ## IsSpecialSubidentityMatrix( IAL );      ## does not increase performance

  ## NF = ZB + IAL * CB
  CB := HomalgVoidMatrix( R );
  NF := DecideZeroColumnsEffectively( ZB, IAL, CB );

  ## NF <> 0
  if not IsZero( NF ) then

```

```

        return fail;
    fi;

    ## CD = CA * -CB => A * CD = B
    X := CertainRows( CA, [ 1 .. NrColumns( A ) ] ) * -CB;

    ## check assertion
    Assert( 5, IsZero( DecideZeroColumns( A * X - B, BL ) ) );

    return X;

    ## technical: CA * -CB := CA * (-CB) and COLEM should take over since
    ## CB := -matrix

end );

```

#### 5.5.49 GenerateSameRowModule (for pairs of matrices)

- ▷ `GenerateSameRowModule( $M$ ,  $N$ )` (operation)  
**Returns:** true or false  
 Check if the row span of  $M$  and of  $N$  are identical or not ( $\rightarrow$  `RightDivide` (5.5.45)).

#### 5.5.50 GenerateSameColumnModule (for pairs of matrices)

- ▷ `GenerateSameColumnModule( $M$ ,  $N$ )` (operation)  
**Returns:** true or false  
 Check if the column span of  $M$  and of  $N$  are identical or not ( $\rightarrow$  `LeftDivide` (5.5.46)).

## Chapter 6

# Ring Relations

### 6.1 Ring Relations: Categories and Representations

#### 6.1.1 IsHomalgRingRelations

- ▷ `IsHomalgRingRelations(rel)` (Category)  
**Returns:** true or false  
The GAP category of homalg ring relations.

#### 6.1.2 IsHomalgRingRelationsAsGeneratorsOfLeftIdeal

- ▷ `IsHomalgRingRelationsAsGeneratorsOfLeftIdeal(rel)` (Category)  
**Returns:** true or false  
The GAP category of homalg ring relations as generators of a left ideal.  
(It is a subcategory of the GAP category `IsHomalgRingRelations`.)

#### 6.1.3 IsHomalgRingRelationsAsGeneratorsOfRightIdeal

- ▷ `IsHomalgRingRelationsAsGeneratorsOfRightIdeal(rel)` (Category)  
**Returns:** true or false  
The GAP category of homalg ring relations as generators of a right ideal.  
(It is a subcategory of the GAP category `IsHomalgRingRelations`.)

#### 6.1.4 IsRingRelationsRep

- ▷ `IsRingRelationsRep(rel)` (Representation)  
**Returns:** true or false  
The GAP representation of a finite set of relations of a homalg ring.  
(It is a representation of the GAP category `IsHomalgRingRelations` (6.1.1))

## 6.2 Ring Relations: Constructors

## 6.3 Ring Relations: Properties

### 6.3.1 CanBeUsedToDecideZero

- ▷ `CanBeUsedToDecideZero(rel)` (property)  
**Returns:** true or false  
Check if the homalg set of relations *rel* can be used for normal form reductions.  
(no method installed)

### 6.3.2 IsInjectivePresentation

- ▷ `IsInjectivePresentation(rel)` (property)  
**Returns:** true or false  
Check if the homalg set of relations *rel* has zero syzygies.

## 6.4 Ring Relations: Attributes

## 6.5 Ring Relations: Operations and Functions

# Appendix A

## The Basic Matrix Operations

These are the operations used to solve one-sided (in)homogeneous linear systems  $XA = B$  resp.  $AX = B$ .

### A.1 Main

- [BasisOfRowModule \(5.5.22\)](#)
- [BasisOfColumnModule \(5.5.23\)](#)
- [DecideZeroRows \(5.5.24\)](#)
- [DecideZeroColumns \(5.5.25\)](#)
- [SyzygiesGeneratorsOfRows \(5.5.26\)](#)
- [SyzygiesGeneratorsOfColumns \(5.5.27\)](#)

### A.2 Effective

- [BasisOfRowsCoeff \(5.5.34\)](#)
- [BasisOfColumnsCoeff \(5.5.35\)](#)
- [DecideZeroRowsEffectively \(5.5.36\)](#)
- [DecideZeroColumnsEffectively \(5.5.37\)](#)

### A.3 Relative

- [SyzygiesGeneratorsOfRows \(5.5.28\)](#)
- [SyzygiesGeneratorsOfColumns \(5.5.29\)](#)

## A.4 Reduced

- `ReducedBasisOfRowModule` (5.5.30)
- `ReducedBasisOfColumnModule` (5.5.31)
- `ReducedSyzygiesGeneratorsOfRows` (5.5.32)
- `ReducedSyzygiesGeneratorsOfColumns` (5.5.33)



## Appendix B

# The Matrix Tool Operations

The functions listed below are components of the `homalgTable` object stored in the ring. They are only indirectly accessible through standard methods that invoke them.

### B.1 The Tool Operations *without* a Fallback Method

There are matrix methods for which `homalg` needs a `homalgTable` entry for non-internal rings, as it cannot provide a suitable fallback. Below is the list of these `homalgTable` entries.

#### B.1.1 `InitialMatrix` (`homalgTable` entry for initial matrices)

▷ `InitialMatrix(C)` (function)  
**Returns:** the `Eval` value of a `homalg` matrix  $C$   
Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the `homalgTable` component  $RP!.InitialMatrix$  is bound then the method `Eval` (C.4.1) resets the filter `IsInitialMatrix` and returns  $RP!.InitialMatrix(C)$ .

#### B.1.2 `InitialIdentityMatrix` (`homalgTable` entry for initial identity matrices)

▷ `InitialIdentityMatrix(C)` (function)  
**Returns:** the `Eval` value of a `homalg` matrix  $C$   
Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the `homalgTable` component  $RP!.InitialIdentityMatrix$  is bound then the method `Eval` (C.4.2) resets the filter `IsInitialIdentityMatrix` and returns  $RP!.InitialIdentityMatrix(C)$ .

#### B.1.3 `ZeroMatrix` (`homalgTable` entry)

▷ `ZeroMatrix(C)` (function)  
**Returns:** the `Eval` value of a `homalg` matrix  $C$   
Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the `homalgTable` component  $RP!.ZeroMatrix$  is bound then the method `Eval` (C.4.3) returns  $RP!.ZeroMatrix(C)$ .

### B.1.4 IdentityMatrix (homalgTable entry)

▷ IdentityMatrix( $C$ ) (function)

**Returns:** the Eval value of a homalg matrix  $C$

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the homalgTable component  $RP!.IdentityMatrix$  is bound then the method Eval (C.4.4) returns  $RP!.IdentityMatrix(C)$ .

### B.1.5 Involution (homalgTable entry)

▷ Involution( $M$ ) (function)

**Returns:** the Eval value of a homalg matrix  $C$

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the homalgTable component  $RP!.Involution$  is bound then the method Eval (C.4.7) returns  $RP!.Involution$  applied to the content of the attribute  $\text{EvalInvolution}(C) = M$ .

### B.1.6 CertainRows (homalgTable entry)

▷ CertainRows( $M, plist$ ) (function)

**Returns:** the Eval value of a homalg matrix  $C$

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the homalgTable component  $RP!.CertainRows$  is bound then the method Eval (C.4.8) returns  $RP!.CertainRows$  applied to the content of the attribute  $\text{EvalCertainRows}(C) = [M, plist]$ .

### B.1.7 CertainColumns (homalgTable entry)

▷ CertainColumns( $M, plist$ ) (function)

**Returns:** the Eval value of a homalg matrix  $C$

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the homalgTable component  $RP!.CertainColumns$  is bound then the method Eval (C.4.9) returns  $RP!.CertainColumns$  applied to the content of the attribute  $\text{EvalCertainColumns}(C) = [M, plist]$ .

### B.1.8 UnionOfRows (homalgTable entry)

▷ UnionOfRows( $A, B$ ) (function)

**Returns:** the Eval value of a homalg matrix  $C$

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the homalgTable component  $RP!.UnionOfRows$  is bound then the method Eval (C.4.10) returns  $RP!.UnionOfRows$  applied to the content of the attribute  $\text{EvalUnionOfRows}(C) = [A, B]$ .

### B.1.9 UnionOfColumns (homalgTable entry)

▷ UnionOfColumns( $A, B$ ) (function)

**Returns:** the Eval value of a homalg matrix  $C$

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the homalgTable component  $RP!.UnionOfColumns$  is bound then the method Eval (C.4.11) returns  $RP!.UnionOfColumns$  applied to the content of the attribute  $\text{EvalUnionOfColumns}(C) = [A, B]$ .

**B.1.10 DiagMat (homalgTable entry)**

▷ `DiagMat(e)` (function)

**Returns:** the Eval value of a homalg matrix  $C$

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the homalgTable component  $RP!.DiagMat$  is bound then the method Eval (C.4.12) returns  $RP!.DiagMat$  applied to the content of the attribute  $\text{EvalDiagMat}(C) = e$ .

**B.1.11 KroneckerMat (homalgTable entry)**

▷ `KroneckerMat(A, B)` (function)

**Returns:** the Eval value of a homalg matrix  $C$

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the homalgTable component  $RP!.KroneckerMat$  is bound then the method Eval (C.4.13) returns  $RP!.KroneckerMat$  applied to the content of the attribute  $\text{EvalKroneckerMat}(C) = [A, B]$ .

**B.1.12 MulMat (homalgTable entry)**

▷ `MulMat(a, A)` (function)

**Returns:** the Eval value of a homalg matrix  $C$

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the homalgTable component  $RP!.MulMat$  is bound then the method Eval (C.4.14) returns  $RP!.MulMat$  applied to the content of the attribute  $\text{EvalMulMat}(C) = [a, A]$ .

**B.1.13 AddMat (homalgTable entry)**

▷ `AddMat(A, B)` (function)

**Returns:** the Eval value of a homalg matrix  $C$

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the homalgTable component  $RP!.AddMat$  is bound then the method Eval (C.4.15) returns  $RP!.AddMat$  applied to the content of the attribute  $\text{EvalAddMat}(C) = [A, B]$ .

**B.1.14 SubMat (homalgTable entry)**

▷ `SubMat(A, B)` (function)

**Returns:** the Eval value of a homalg matrix  $C$

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the homalgTable component  $RP!.SubMat$  is bound then the method Eval (C.4.16) returns  $RP!.SubMat$  applied to the content of the attribute  $\text{EvalSubMat}(C) = [A, B]$ .

**B.1.15 Compose (homalgTable entry)**

▷ `Compose(A, B)` (function)

**Returns:** the Eval value of a homalg matrix  $C$

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the homalgTable component  $RP!.Compose$  is bound then the method Eval (C.4.17) returns  $RP!.Compose$  applied to the content of the attribute  $\text{EvalCompose}(C) = [A, B]$ .

**B.1.16 IsZeroMatrix (homalgTable entry)**

▷ IsZeroMatrix( $M$ ) (function)

**Returns:** true or false

Let  $R := \text{HomalgRing}(M)$  and  $RP := \text{homalgTable}(R)$ . If the homalgTable component  $RP!.IsZeroMatrix$  is bound then the standard method for the property IsZero (5.3.1) shown below returns  $RP!.IsZeroMatrix(M)$ .

Code

```

InstallMethod( IsZero,
               "for homalg matrices",
               [ IsHomalgMatrix ],

function( M )
  local R, RP;

  R := HomalgRing( M );

  RP := homalgTable( R );

  if IsBound(RP!.IsZeroMatrix) then
    ## CAUTION: the external system must be able
    ## to check zero modulo possible ring relations!

    return RP!.IsZeroMatrix( M ); ## with this, \= can fall back to IsZero
  fi;

  #=====# the fallback method #=====#

  ## from the GAP4 documentation: ?Zero
  ## 'ZeroSameMutability( <obj> )' is equivalent to '0 * <obj>'.

  return M = 0 * M; ## hence, by default, IsZero falls back to \= (see below)

end );

```

**B.1.17 NrRows (homalgTable entry)**

▷ NrRows( $C$ ) (function)

**Returns:** a nonnegative integer

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the homalgTable component  $RP!.NrRows$  is bound then the standard method for the attribute NrRows (5.4.1) shown below returns  $RP!.NrRows(C)$ .

Code

```

InstallMethod( NrRows,
               "for homalg matrices",
               [ IsHomalgMatrix ],

function( C )
  local R, RP;

  R := HomalgRing( C );

```

```

RP := homalgTable( R );

if IsBound(RP!.NrRows) then
  return RP!.NrRows( C );
fi;

if not IsHomalgInternalMatrixRep( C ) then
  Error( "could not find a procedure called NrRows ",
        "in the homalgTable of the non-internal ring\n" );
fi;

##### can only work for homalg internal matrices #####

return Length( Eval( C )!.matrix );

end );

```

### B.1.18 NrColumns (homalgTable entry)

▷ NrColumns( $C$ )

(function)

**Returns:** a nonnegative integer

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the `homalgTable` component  $RP!.NrColumns$  is bound then the standard method for the attribute `NrColumns` (5.4.2) shown below returns  $RP!.NrColumns(C)$ .

Code

```

InstallMethod( NrColumns,
  "for homalg matrices",
  [ IsHomalgMatrix ],

function( C )
  local R, RP;

  R := HomalgRing( C );

  RP := homalgTable( R );

  if IsBound(RP!.NrColumns) then
    return RP!.NrColumns( C );
  fi;

  if not IsHomalgInternalMatrixRep( C ) then
    Error( "could not find a procedure called NrColumns ",
          "in the homalgTable of the non-internal ring\n" );
  fi;

  ##### can only work for homalg internal matrices #####

  return Length( Eval( C )!.matrix[ 1 ] );

end );

```

**B.1.19 Determinant (homalgTable entry)**

▷ Determinant( $C$ ) (function)

**Returns:** a ring element

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the `homalgTable` component  $RP!.Determinant$  is bound then the standard method for the attribute `DeterminantMat` (5.4.3) shown below returns  $RP!.Determinant(C)$ .

Code

```

InstallMethod( DeterminantMat,
  "for homalg matrices",
  [ IsHomalgMatrix ],

function( C )
  local R, RP;

  R := HomalgRing( C );

  RP := homalgTable( R );

  if NrRows( C ) <> NrColumns( C ) then
    Error( "the matrix is not a square matrix\n" );
  fi;

  if IsEmptyMatrix( C ) then
    return One( R );
  elif IsZero( C ) then
    return Zero( R );
  fi;

  if IsBound(RP!.Determinant) then
    return RingElementConstructor( R )( RP!.Determinant( C ), R );
  fi;

  if not IsHomalgInternalMatrixRep( C ) then
    Error( "could not find a procedure called Determinant ",
          "in the homalgTable of the non-internal ring\n" );
  fi;

  #=====# can only work for homalg internal matrices #=====#

  return Determinant( Eval( C )!.matrix );

end );

InstallMethod( Determinant,
  "for homalg matrices",
  [ IsHomalgMatrix ],

function( C )

  return DeterminantMat( C );

```

```
end );
```

## B.2 The Tool Operations with a Fallback Method

These are the methods for which it is recommended for performance reasons to have a `homalgTable` entry for non-internal rings. `homalg` only provides a generic fallback method.

### B.2.1 AreEqualMatrices (homalgTable entry)

▷ `AreEqualMatrices(M1, M2)` (function)

**Returns:** true or false

Let  $R := \text{HomalgRing}(M1)$  and  $RP := \text{homalgTable}(R)$ . If the `homalgTable` component  $RP!.AreEqualMatrices$  is bound then the standard method for the operation `\=` (5.5.17) shown below returns  $RP!.AreEqualMatrices(M1, M2)$ .

```
Code
InstallMethod( \=,
  "for homalg comparable matrices",
  [ IsHomalgMatrix, IsHomalgMatrix ],

function( M1, M2 )
  local R, RP, are_equal;

  ## do not touch mutable matrices
  if not ( IsMutable( M1 ) or IsMutable( M2 ) ) then

    if IsBound( M1!.AreEqual ) then
      are_equal := _ElmWPObj_ForHomalg( M1!.AreEqual, M2, fail );
      if are_equal <> fail then
        return are_equal;
      fi;
    else
      M1!.AreEqual :=
        ContainerForWeakPointers(
          TheTypeContainerForWeakPointersOnComputedValues,
          [ "operation", "AreEqual" ] );
    fi;

    if IsBound( M2!.AreEqual ) then
      are_equal := _ElmWPObj_ForHomalg( M2!.AreEqual, M1, fail );
      if are_equal <> fail then
        return are_equal;
      fi;
    fi;
    ## do not store things symmetrically below to "save" memory

  fi;

  R := HomalgRing( M1 );

  RP := homalgTable( R );
```

```

if IsBound(RP!.AreEqualMatrices) then
  ## CAUTION: the external system must be able to check equality
  ## modulo possible ring relations (known to the external system)!
  are_equal := RP!.AreEqualMatrices( M1, M2 );
elif IsBound(RP!.Equal) then
  ## CAUTION: the external system must be able to check equality
  ## modulo possible ring relations (known to the external system)!
  are_equal := RP!.Equal( M1, M2 );
elif IsBound(RP!.IsZeroMatrix) then ## ensuring this avoids infinite loops
  are_equal := IsZero( M1 - M2 );
fi;

if IsBound( are_equal ) then

  ## do not touch mutable matrices
  if not ( IsMutable( M1 ) or IsMutable( M2 ) ) then

    if are_equal then
      MatchPropertiesAndAttributes( M1, M2,
        LIMAT.intrinsic_properties,
        LIMAT.intrinsic_attributes,
        LIMAT.intrinsic_components
      );
    fi;

    ## do not store things symmetrically to ‘‘save’’ memory
    _AddTwoElmWPObj_ForHomalg( M1!.AreEqual, M2, are_equal );

  fi;

  return are_equal;
fi;

TryNextMethod( );

end );

```

## B.2.2 IsIdentityMatrix (homalgTable entry)

▷ IsIdentityMatrix( $M$ ) (function)

**Returns:** true or false

Let  $R := \text{HomalgRing}(M)$  and  $RP := \text{homalgTable}(R)$ . If the homalgTable component  $RP!.IsIdentityMatrix$  is bound then the standard method for the property IsOne (5.3.2) shown below returns  $RP!.IsIdentityMatrix(M)$ .

Code

```

InstallMethod( IsOne,
  "for homalg matrices",
  [ IsHomalgMatrix ],

  function( M )
    local R, RP;

```



```

if NrRows( M ) <> NrColumns( M ) then
    return false;
fi;

R := HomalgRing( M );

RP := homalgTable( R );

if IsBound(RP!.IsIdentityMatrix) then
    return RP!.IsIdentityMatrix( M );
fi;

#=====# the fallback method #=====#

return M = HomalgIdentityMatrix( NrRows( M ), HomalgRing( M ) );

end );

```

### B.2.3 IsDiagonalMatrix (homalgTable entry)

▷ IsDiagonalMatrix( $M$ )

(function)

**Returns:** true or false

Let  $R := \text{HomalgRing}(M)$  and  $RP := \text{homalgTable}(R)$ . If the `homalgTable` component  $RP!.IsDiagonalMatrix$  is bound then the standard method for the property `IsDiagonalMatrix` (5.3.13) shown below returns  $RP!.IsDiagonalMatrix(M)$ .

Code

```

InstallMethod( IsDiagonalMatrix,
    "for homalg matrices",
    [ IsHomalgMatrix ],

function( M )
    local R, RP, diag;

    R := HomalgRing( M );

    RP := homalgTable( R );

    if IsBound(RP!.IsDiagonalMatrix) then
        return RP!.IsDiagonalMatrix( M );
    fi;

    #=====# the fallback method #=====#

    diag := DiagonalEntries( M );

    return M = HomalgDiagonalMatrix( diag, NrRows( M ), NrColumns( M ), R );

end );

```

## B.2.4 ZeroRows (homalgTable entry)

▷ ZeroRows( $C$ ) (function)

**Returns:** a (possibly empty) list of positive integers

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the `homalgTable` component  $RP!.ZeroRows$  is bound then the standard method of the attribute `ZeroRows` (5.4.4) shown below returns  $RP!.ZeroRows(C)$ .

```

Code
InstallMethod( ZeroRows,
               "for homalg matrices",
               [ IsHomalgMatrix ],

               function( C )
                 local R, RP, z;

                 R := HomalgRing( C );

                 RP := homalgTable( R );

                 if IsBound(RP!.ZeroRows) then
                   return RP!.ZeroRows( C );
                 fi;

                 #=====# the fallback method #=====#

                 z := HomalgZeroMatrix( 1, NrColumns( C ), R );

                 return Filtered( [ 1 .. NrRows( C ) ], a -> CertainRows( C, [ a ] ) = z );

               end );
```

## B.2.5 ZeroColumns (homalgTable entry)

▷ ZeroColumns( $C$ ) (function)

**Returns:** a (possibly empty) list of positive integers

Let  $R := \text{HomalgRing}(C)$  and  $RP := \text{homalgTable}(R)$ . If the `homalgTable` component  $RP!.ZeroColumns$  is bound then the standard method of the attribute `ZeroColumns` (5.4.5) shown below returns  $RP!.ZeroColumns(C)$ .

```

Code
InstallMethod( ZeroColumns,
               "for homalg matrices",
               [ IsHomalgMatrix ],

               function( C )
                 local R, RP, z;

                 R := HomalgRing( C );

                 RP := homalgTable( R );

                 if IsBound(RP!.ZeroColumns) then
```

```

    return RP!.ZeroColumns( C );
fi;

#=====# the fallback method #=====#

z := HomalgZeroMatrix( NrRows( C ), 1, R );

return Filtered( [ 1 .. NrColumns( C ) ], a -> CertainColumns( C, [ a ] ) = z );

end );

```

## B.2.6 GetColumnIndependentUnitPositions (homalgTable entry)

▷ `GetColumnIndependentUnitPositions( $M$ ,  $poslist$ )` (function)

**Returns:** a (possibly empty) list of pairs of positive integers

Let  $R := \text{HomalgRing}(M)$  and  $RP := \text{homalgTable}(R)$ . If the `homalgTable` component  $RP!.GetColumnIndependentUnitPositions$  is bound then the standard method of the operation `GetColumnIndependentUnitPositions` (5.5.18) shown below returns  $RP!.GetColumnIndependentUnitPositions(M, poslist)$ .

```

Code
InstallMethod( GetColumnIndependentUnitPositions,
  "for homalg matrices",
  [ IsHomalgMatrix, IsHomogeneousList ],

function( M, poslist )
  local cache, R, RP, rest, pos, i, j, k;

  if IsBound( M!.GetColumnIndependentUnitPositions ) then
    cache := M!.GetColumnIndependentUnitPositions;
    if IsBound( cache.(String( poslist )) ) then
      return cache.(String( poslist ));
    fi;
  else
    cache := rec( );
    M!.GetColumnIndependentUnitPositions := cache;
  fi;

  R := HomalgRing( M );

  RP := homalgTable( R );

  if IsBound(RP!.GetColumnIndependentUnitPositions) then
    pos := RP!.GetColumnIndependentUnitPositions( M, poslist );
    if pos <> [ ] then
      SetIsZero( M, false );
    fi;
    cache.(String( poslist )) := pos;
    return pos;
  fi;

#=====# the fallback method #=====#

```

```

rest := [ 1 .. NrColumns( M ) ];
pos := [ ];
for i in [ 1 .. NrRows( M ) ] do
  for k in Reversed( rest ) do
    if not [ i, k ] in poslist and
      IsUnit( R, MatElm( M, i, k ) ) then
      Add( pos, [ i, k ] );
      rest := Filtered( rest,
        a -> IsZero( MatElm( M, i, a ) ) );
      break;
    fi;
  od;
od;

if pos <> [ ] then
  SetIsZero( M, false );
fi;

cache.(String( poslist )) := pos;

return pos;

end );

```

## B.2.7 GetRowIndependentUnitPositions (homalgTable entry)

▷ `GetRowIndependentUnitPositions(M, poslist)` (function)

**Returns:** a (possibly empty) list of pairs of positive integers

Let  $R := \text{HomalgRing}(M)$  and  $RP := \text{homalgTable}(R)$ . If the `homalgTable` component  $RP!.GetRowIndependentUnitPositions$  is bound then the standard method of the operation `GetRowIndependentUnitPositions` (5.5.19) shown below returns  $RP!.GetRowIndependentUnitPositions(M, poslist)$ .

```

Code
InstallMethod( GetRowIndependentUnitPositions,
  "for homalg matrices",
  [ IsHomalgMatrix, IsHomogeneousList ],

function( M, poslist )
  local cache, R, RP, rest, pos, j, i, k;

  if IsBound( M!.GetRowIndependentUnitPositions ) then
    cache := M!.GetRowIndependentUnitPositions;
    if IsBound( cache.(String( poslist )) ) then
      return cache.(String( poslist ));
    fi;
  else
    cache := rec( );
    M!.GetRowIndependentUnitPositions := cache;

```

```

fi;

R := HomalgRing( M );

RP := homalgTable( R );

if IsBound(RP!.GetRowIndependentUnitPositions) then
  pos := RP!.GetRowIndependentUnitPositions( M, poslist );
  if pos <> [ ] then
    SetIsZero( M, false );
    fi;
    cache.( String( poslist ) ) := pos;
    return pos;
fi;

#=====# the fallback method #=====#

rest := [ 1 .. NrRows( M ) ];

pos := [ ];

for j in [ 1 .. NrColumns( M ) ] do
  for k in Reversed( rest ) do
    if not [ j, k ] in poslist and
      IsUnit( R, MatElm( M, k, j ) ) then
      Add( pos, [ j, k ] );
      rest := Filtered( rest,
        a -> IsZero( MatElm( M, a, j ) ) );
      break;
    fi;
  od;
od;

if pos <> [ ] then
  SetIsZero( M, false );
fi;

cache.( String( poslist ) ) := pos;

return pos;

end );

```

## B.2.8 GetUnitPosition (homalgTable entry)

▷ GetUnitPosition( $M$ ,  $poslist$ )

(function)

**Returns:** a (possibly empty) list of pairs of positive integers

Let  $R := \text{HomalgRing}(M)$  and  $RP := \text{homalgTable}(R)$ . If the `homalgTable` component  $RP!.GetUnitPosition$  is bound then the standard method of the operation `GetUnitPosition` (5.5.20) shown below returns  $RP!.GetUnitPosition(M, poslist)$ .

Code

```
InstallMethod( GetUnitPosition,
```

```

    "for homalg matrices",
    [ IsHomalgMatrix, IsHomogeneousList ],

function( M, poslist )
  local R, RP, pos, m, n, i, j;

  R := HomalgRing( M );

  RP := homalgTable( R );

  if IsBound(RP!.GetUnitPosition) then
    pos := RP!.GetUnitPosition( M, poslist );
    if IsList( pos ) and IsPosInt( pos[1] ) and IsPosInt( pos[2] ) then
      SetIsZero( M, false );
    fi;
    return pos;
  fi;

  #=====# the fallback method #=====#

  m := NrRows( M );
  n := NrColumns( M );

  for i in [ 1 .. m ] do
    for j in [ 1 .. n ] do
      if not [ i, j ] in poslist and not j in poslist and
        IsUnit( R, MatElm( M, i, j ) ) then
        SetIsZero( M, false );
        return [ i, j ];
      fi;
    od;
  od;

  return fail;

end );

```

## B.2.9 PositionOfFirstNonZeroEntryPerRow (homalgTable entry)

▷ PositionOfFirstNonZeroEntryPerRow( $M$ ,  $poslist$ ) (function)

**Returns:** a list of nonnegative integers

Let  $R := \text{HomalgRing}(M)$  and  $RP := \text{homalgTable}(R)$ . If the `homalgTable` component  $RP!.PositionOfFirstNonZeroEntryPerRow$  is bound then the standard method of the attribute `PositionOfFirstNonZeroEntryPerRow` (5.4.8) shown below returns  $RP!.PositionOfFirstNonZeroEntryPerRow(M)$ .

```

Code
InstallMethod( PositionOfFirstNonZeroEntryPerRow,
  "for homalg matrices",
  [ IsHomalgMatrix ],

function( M )

```

```

local R, RP, pos, entries, r, c, i, k, j;

R := HomalgRing( M );

RP := homalgTable( R );

if IsBound(RP!.PositionOfFirstNonZeroEntryPerRow) then
  return RP!.PositionOfFirstNonZeroEntryPerRow( M );
elif IsBound(RP!.PositionOfFirstNonZeroEntryPerColumn) then
  return PositionOfFirstNonZeroEntryPerColumn( Involution( M ) );
fi;

#=====# the fallback method #=====#

entries := EntriesOfHomalgMatrix( M );

r := NrRows( M );
c := NrColumns( M );

pos := ListWithIdenticalEntries( r, 0 );

for i in [ 1 .. r ] do
  k := ( i - 1 ) * c;
  for j in [ 1 .. c ] do
    if not IsZero( entries[k + j] ) then
      pos[i] := j;
      break;
    fi;
  od;
od;

return pos;

end );

```

### B.2.10 PositionOfFirstNonZeroEntryPerColumn (homalgTable entry)

▷ PositionOfFirstNonZeroEntryPerColumn( $M$ ,  $poslist$ ) (function)

**Returns:** a list of nonnegative integers

Let  $R := \text{HomalgRing}(M)$  and  $RP := \text{homalgTable}(R)$ . If the `homalgTable` component  $RP!.PositionOfFirstNonZeroEntryPerColumn$  is bound then the standard method of the attribute `PositionOfFirstNonZeroEntryPerColumn` (5.4.9) shown below returns  $RP!.PositionOfFirstNonZeroEntryPerColumn(M)$ .

Code

```

InstallMethod( PositionOfFirstNonZeroEntryPerColumn,
  "for homalg matrices",
  [ IsHomalgMatrix ],

function( M )
  local R, RP, pos, entries, r, c, j, i, k;

```

```
R := HomalgRing( M );
RP := homalgTable( R );

if IsBound(RP!.PositionOfFirstNonZeroEntryPerColumn) then
  return RP!.PositionOfFirstNonZeroEntryPerColumn( M );
elif IsBound(RP!.PositionOfFirstNonZeroEntryPerRow) then
  return PositionOfFirstNonZeroEntryPerRow( Involution( M ) );
fi;

#=====# the fallback method #=====#

entries := EntriesOfHomalgMatrix( M );

r := NrRows( M );
c := NrColumns( M );

pos := ListWithIdenticalEntries( c, 0 );

for j in [ 1 .. c ] do
  for i in [ 1 .. r ] do
    k := (i - 1) * c;
    if not IsZero( entries[k + j] ) then
      pos[j] := i;
      break;
    fi;
  od;
od;

return pos;

end );
```



# Appendix C

## Logic Subpackages

### C.1 LIRNG: Logical Implications for Rings

### C.2 LIMAP: Logical Implications for Ring Maps

### C.3 LIMAT: Logical Implications for Matrices

### C.4 COLEM: Clever Operations for Lazy Evaluated Matrices

Most of the matrix tool operations listed in Appendix B.1 which return a new matrix are lazy evaluated. The value of a homalg matrix is stored in the attribute Eval. Below is the list of the installed methods for the attribute Eval.

#### C.4.1 Eval (for matrices created with HomalgInitialMatrix)

▷ Eval(*C*) (method)

**Returns:** the Eval value of a homalg matrix *C*

In case the matrix *C* was created using HomalgInitialMatrix (5.2.1) then the filter IsInitialMatrix for *C* is set to true and the homalgTable function ( $\rightarrow$  InitialMatrix (B.1.1)) will be used to set the attribute Eval and resets the filter IsInitialMatrix.

```
Code
InstallMethod( Eval,
  "for homalg matrices (IsInitialMatrix)",
  [ IsHomalgMatrix and IsInitialMatrix and
    HasNrRows and HasNrColumns ],

function( C )
  local R, RP, z, zz;

  R := HomalgRing( C );

  RP := homalgTable( R );

  if IsBound( RP!.InitialMatrix ) then
    ResetFilterObj( C, IsInitialMatrix );
  return RP!.InitialMatrix( C );
```

```

fi;

if not IsHomalgInternalMatrixRep( C ) then
  Error( "could not find a procedure called InitialMatrix in the ",
        "homalgTable to evaluate a non-internal initial matrix\n" );
fi;

#=====# can only work for homalg internal matrices #=====#

z := Zero( HomalgRing( C ) );

ResetFilterObj( C, IsInitialMatrix );

zz := ListWithIdenticalEntries( NrColumns( C ), z );

return homalgInternalMatrixHull(
  List( [ 1 .. NrRows( C ) ], i -> ShallowCopy( zz ) ) );

end );

```

## C.4.2 Eval (for matrices created with HomalgInitialIdentityMatrix)

▷ Eval(*C*)

(method)

**Returns:** the Eval value of a homalg matrix *C*

In case the matrix *C* was created using HomalgInitialIdentityMatrix (5.2.2) then the filter IsInitialIdentityMatrix for *C* is set to true and the homalgTable function (→ InitialIdentityMatrix (B.1.2)) will be used to set the attribute Eval and resets the filter IsInitialIdentityMatrix.

Code

```

InstallMethod( Eval,
  "for homalg matrices (IsInitialIdentityMatrix)",
  [ IsHomalgMatrix and IsInitialIdentityMatrix and
    HasNrRows and HasNrColumns ],

function( C )
  local R, RP, o, z, zz, id;

  R := HomalgRing( C );

  RP := homalgTable( R );

  if IsBound( RP!.InitialIdentityMatrix ) then
    ResetFilterObj( C, IsInitialIdentityMatrix );
    return RP!.InitialIdentityMatrix( C );
  fi;

  if not IsHomalgInternalMatrixRep( C ) then
    Error( "could not find a procedure called InitialIdentityMatrix in the ",
          "homalgTable to evaluate a non-internal initial identity matrix\n" );
  fi;

  #=====# can only work for homalg internal matrices #=====#

```

```

z := Zero( HomalgRing( C ) );
o := One( HomalgRing( C ) );

ResetFilterObj( C, IsInitialIdentityMatrix );

zz := ListWithIdenticalEntries( NrColumns( C ), z );

id := List( [ 1 .. NrRows( C ) ],
            function(i)
              local z;
              z := ShallowCopy( zz ); z[i] := o; return z;
            end );

return homalgInternalMatrixHull( id );

end );

```

### C.4.3 Eval (for matrices created with HomalgZeroMatrix)

▷ Eval(*C*) (method)

**Returns:** the Eval value of a homalg matrix *C*

In case the matrix *C* was created using HomalgZeroMatrix (5.2.3) then the filter IsZeroMatrix for *C* is set to true and the homalgTable function ( $\rightarrow$  ZeroMatrix (B.1.3)) will be used to set the attribute Eval.

```

Code
InstallMethod( Eval,
              "for homalg matrices (IsZero)",
              [ IsHomalgMatrix and IsZero and HasNrRows and HasNrColumns ], 20,

function( C )
  local R, RP, z;

  R := HomalgRing( C );

  RP := homalgTable( R );

  if ( NrRows( C ) = 0 or NrColumns( C ) = 0 ) and
      not ( IsBound( R!.SafeToEvaluateEmptyMatrices ) and
            R!.SafeToEvaluateEmptyMatrices = true ) then
    Info( InfoWarning, 1, "\033[01m\033[5;31;47m",
          "an empty matrix is about to get evaluated!",
          "\033[0m" );
  fi;

  if IsBound( RP!.ZeroMatrix ) then
    return RP!.ZeroMatrix( C );
  fi;

  if not IsHomalgInternalMatrixRep( C ) then
    Error( "could not find a procedure called ZeroMatrix ",
           "homalgTable to evaluate a non-internal zero matrix\n" );
  fi;
end );

```

```

fi;

##### can only work for homalg internal matrices #####

z := Zero( HomalgRing( C ) );

## copying the rows saves memory;
## we assume that the entries are never modified!!!
return homalgInternalMatrixHull(
    ListWithIdenticalEntries( NrRows( C ),
        ListWithIdenticalEntries( NrColumns( C ), z ) ) );

end );

```

#### C.4.4 Eval (for matrices created with HomalgIdentityMatrix)

▷ Eval(*C*) (method)

**Returns:** the Eval value of a homalg matrix *C*

In case the matrix *C* was created using HomalgIdentityMatrix (5.2.4) then the filter IsOne for *C* is set to true and the homalgTable function (→ IdentityMatrix (B.1.4)) will be used to set the attribute Eval.

```

Code
InstallMethod( Eval,
    "for homalg matrices (IsOne)",
    [ IsHomalgMatrix and IsOne and HasNrRows and HasNrColumns ], 10,

function( C )
    local R, id, RP, o, z, zz;

    R := HomalgRing( C );

    if IsBound( R!.IdentityMatrices ) then
        id := ElmWPObj( R!.IdentityMatrices!.weak_pointers, NrColumns( C ) );
        if id <> fail then
            R!.IdentityMatrices!.cache_hits := R!.IdentityMatrices!.cache_hits + 1;
            return id;
        fi;
        ## we do not count cache_misses as it is equivalent to counter
    fi;

    RP := homalgTable( R );

    if IsBound( RP!.IdentityMatrix ) then
        id := RP!.IdentityMatrix( C );
        SetElmWPObj( R!.IdentityMatrices!.weak_pointers, NrColumns( C ), id );
        R!.IdentityMatrices!.counter := R!.IdentityMatrices!.counter + 1;
        return id;
    fi;

    if not IsHomalgInternalMatrixRep( C ) then
        Error( "could not find a procedure called IdentityMatrix ",
            "homalgTable to evaluate a non-internal identity matrix\n" );
    fi;
end );

```

```

fi;

##### can only work for homalg internal matrices #####

z := Zero( HomalgRing( C ) );
o := One( HomalgRing( C ) );

zz := ListWithIdenticalEntries( NrColumns( C ), z );

id := List( [ 1 .. NrRows( C ) ],
            function(i)
              local z;
              z := ShallowCopy( zz ); z[i] := o; return z;
            end );

id := homalgInternalMatrixHull( id );

SetElmWPObj( R!.IdentityMatrices!.weak_pointers, NrColumns( C ), id );

return id;

end );

```

### C.4.5 Eval (for matrices created with LeftInverseLazy)

▷ Eval(*LI*)

(method)

**Returns:** see below

In case the matrix *LI* was created using `LeftInverseLazy` (5.5.4) then the filter `HasEvalLeftInverse` for *LI* is set to true and the method listed below will be used to set the attribute `Eval`. (→ `LeftInverse` (5.5.2))

```

Code
InstallMethod( Eval,
              "for homalg matrices",
              [ IsHomalgMatrix and HasEvalLeftInverse ],

              function( LI )
                local left_inv;

                left_inv := LeftInverse( EvalLeftInverse( LI ) );

                if IsBool( left_inv ) then
                  return false;
                fi;

                return Eval( left_inv );

              end );

```

### C.4.6 Eval (for matrices created with RightInverseLazy)

▷ Eval(*RI*) (method)

**Returns:** see below

In case the matrix *RI* was created using RightInverseLazy (5.5.5) then the filter HasEvalRightInverse for *RI* is set to true and the method listed below will be used to set the attribute Eval. (→ RightInverse (5.5.3))

```
Code
InstallMethod( Eval,
  "for homalg matrices",
  [ IsHomalgMatrix and HasEvalRightInverse ],

function( RI )
  local right_inv;

  right_inv := RightInverse( EvalRightInverse( RI ) );

  if IsBool( right_inv ) then
    return false;
  fi;

  return Eval( right_inv );

end );
```

### C.4.7 Eval (for matrices created with Involution)

▷ Eval(*C*) (method)

**Returns:** the Eval value of a homalg matrix *C*

In case the matrix was created using Involution (5.5.6) then the filter HasEvalInvolution for *C* is set to true and the homalgTable function Involution (B.1.5) will be used to set the attribute Eval.

```
Code
InstallMethod( Eval,
  "for homalg matrices (HasEvalInvolution)",
  [ IsHomalgMatrix and HasEvalInvolution ],

function( C )
  local R, RP, M;

  R := HomalgRing( C );

  RP := homalgTable( R );

  M := EvalInvolution( C );

  if IsBound(RP!.Involution) then
    return RP!.Involution( M );
  fi;

  if not IsHomalgInternalMatrixRep( C ) then
    Error( "could not find a procedure called Involution ",
```

```

        "in the homalgTable of the non-internal ring\n" );
fi;

#=====# can only work for homalg internal matrices #=====#

return homalgInternalMatrixHull( TransposedMat( Eval( M )!.matrix ) );

end );

```

### C.4.8 Eval (for matrices created with CertainRows)

▷ Eval(*C*) (method)

**Returns:** the Eval value of a homalg matrix *C*

In case the matrix was created using `CertainRows` (5.5.7) then the filter `HasEvalCertainRows` for *C* is set to true and the homalgTable function `CertainRows` (B.1.6) will be used to set the attribute `Eval`.

```

----- Code -----
InstallMethod( Eval,
  "for homalg matrices (HasEvalCertainRows)",
  [ IsHomalgMatrix and HasEvalCertainRows ],

function( C )
  local R, RP, e, M, plist;

  R := HomalgRing( C );

  RP := homalgTable( R );

  e := EvalCertainRows( C );

  M := e[1];
  plist := e[2];

  if IsBound(RP!.CertainRows) then
    return RP!.CertainRows( M, plist );
  fi;

  if not IsHomalgInternalMatrixRep( C ) then
    Error( "could not find a procedure called CertainRows ",
          "in the homalgTable of the non-internal ring\n" );
  fi;

  #=====# can only work for homalg internal matrices #=====#

  return homalgInternalMatrixHull( Eval( M )!.matrix{ plist } );

end );

```

### C.4.9 Eval (for matrices created with CertainColumns)

▷ Eval(*C*) (method)

**Returns:** the Eval value of a homalg matrix *C*

In case the matrix was created using CertainColumns (5.5.8) then the filter HasEvalCertainColumns for *C* is set to true and the homalgTable function CertainColumns (B.1.7) will be used to set the attribute Eval.

```

Code
InstallMethod( Eval,
  "for homalg matrices (HasEvalCertainColumns)",
  [ IsHomalgMatrix and HasEvalCertainColumns ],

function( C )
  local R, RP, e, M, plist;

  R := HomalgRing( C );

  RP := homalgTable( R );

  e := EvalCertainColumns( C );

  M := e[1];
  plist := e[2];

  if IsBound(RP!.CertainColumns) then
    return RP!.CertainColumns( M, plist );
  fi;

  if not IsHomalgInternalMatrixRep( C ) then
    Error( "could not find a procedure called CertainColumns ",
          "in the homalgTable of the non-internal ring\n" );
  fi;

  #=====# can only work for homalg internal matrices #=====#

  return homalgInternalMatrixHull(
    Eval( M )!.matrix[[ 1 .. NrRows( M ) ]]{plist} );

end );
```

### C.4.10 Eval (for matrices created with UnionOfRows)

▷ Eval(*C*) (method)

**Returns:** the Eval value of a homalg matrix *C*

In case the matrix was created using UnionOfRows (5.5.9) then the filter HasEvalUnionOfRows for *C* is set to true and the homalgTable function UnionOfRows (B.1.8) will be used to set the attribute Eval.

```

Code
InstallMethod( Eval,
  "for homalg matrices (HasEvalUnionOfRows)",
  [ IsHomalgMatrix and HasEvalUnionOfRows ],
```



```

function( C )
  local R, RP, e, A, B, U;

  R := HomalgRing( C );

  RP := homalgTable( R );

  e := EvalUnionOfRows( C );

  A := e[1];
  B := e[2];

  if IsBound(RP!.UnionOfRows) then
    return RP!.UnionOfRows( A, B );
  fi;

  if not IsHomalgInternalMatrixRep( C ) then
    Error( "could not find a procedure called UnionOfRows ",
          "in the homalgTable of the non-internal ring\n" );
  fi;

  #=====# can only work for homalg internal matrices #=====#

  U := ShallowCopy( Eval( A )!.matrix );

  U{ [ NrRows( A ) + 1 .. NrRows( A ) + NrRows( B ) ] } := Eval( B )!.matrix;

  return homalgInternalMatrixHull( U );

end );

```

#### C.4.11 Eval (for matrices created with UnionOfColumns)

▷ Eval(*C*)

(method)

**Returns:** the Eval value of a homalg matrix *C*

In case the matrix was created using UnionOfColumns (5.5.10) then the filter HasEvalUnionOfColumns for *C* is set to true and the homalgTable function UnionOfColumns (B.1.9) will be used to set the attribute Eval.

Code

```

InstallMethod( Eval,
  "for homalg matrices (HasEvalUnionOfColumns)",
  [ IsHomalgMatrix and HasEvalUnionOfColumns ],

function( C )
  local R, RP, e, A, B, U;

  R := HomalgRing( C );

  RP := homalgTable( R );

  e := EvalUnionOfColumns( C );

```

```

A := e[1];
B := e[2];

if IsBound(RP!.UnionOfColumns) then
  return RP!.UnionOfColumns( A, B );
fi;

if not IsHomalgInternalMatrixRep( C ) then
  Error( "could not find a procedure called UnionOfColumns ",
        "in the homalgTable of the non-internal ring\n" );
fi;

#=====# can only work for homalg internal matrices #=====#

U := List( Eval( A )!.matrix, ShallowCopy );

U{ [ 1 .. NrRows( A ) ] }
  { [ NrColumns( A ) + 1 .. NrColumns( A ) + NrColumns( B ) ] }
  := Eval( B )!.matrix;

return homalgInternalMatrixHull( U );

end );

```

#### C.4.12 Eval (for matrices created with DiagMat)

▷ Eval(*C*)

(method)

**Returns:** the Eval value of a homalg matrix *C*

In case the matrix was created using DiagMat (5.5.11) then the filter HasEvalDiagMat for *C* is set to true and the homalgTable function DiagMat (B.1.10) will be used to set the attribute Eval.

```

Code
InstallMethod( Eval,
  "for homalg matrices (HasEvalDiagMat)",
  [ IsHomalgMatrix and HasEvalDiagMat ],

function( C )
  local R, RP, e, z, m, n, diag, mat;

  R := HomalgRing( C );

  RP := homalgTable( R );

  e := EvalDiagMat( C );

  if IsBound(RP!.DiagMat) then
    return RP!.DiagMat( e );
  fi;

  if not IsHomalgInternalMatrixRep( C ) then
    Error( "could not find a procedure called DiagMat ",
          "in the homalgTable of the non-internal ring\n" );
  fi;

```

```

##### can only work for homalg internal matrices #####

z := Zero( R );

m := Sum( List( e, NrRows ) );
n := Sum( List( e, NrColumns ) );

diag := List( [ 1 .. m ], a -> List( [ 1 .. n ], b -> z ) );

m := 0;
n := 0;

for mat in e do
  diag{ [ m + 1 .. m + NrRows( mat ) ] }{ [ n + 1 .. n + NrColumns( mat ) ] }
    := Eval( mat )!.matrix;

  m := m + NrRows( mat );
  n := n + NrColumns( mat );
od;

return homalgInternalMatrixHull( diag );

end );

```

### C.4.13 Eval (for matrices created with KroneckerMat)

▷ Eval(*C*)

(method)

**Returns:** the Eval value of a homalg matrix *C*

In case the matrix was created using `KroneckerMat` (5.5.12) then the filter `HasEvalKroneckerMat` for *C* is set to true and the homalgTable function `KroneckerMat` (B.1.11) will be used to set the attribute Eval.

```

Code
InstallMethod( Eval,
  "for homalg matrices (HasEvalKroneckerMat)",
  [ IsHomalgMatrix and HasEvalKroneckerMat ],

function( C )
  local R, RP, A, B;

  R := HomalgRing( C );

  if ( HasIsCommutative( R ) and not IsCommutative( R ) ) and
    ( HasIsSuperCommutative( R ) and not IsSuperCommutative( R ) ) then
    Info( InfoWarning, 1, "\033[01m\033[5;31;47m",
      "the Kronecker product is only defined for (super) commutative rings!",
      "\033[0m" );
  fi;

  RP := homalgTable( R );

  A := EvalKroneckerMat( C )[1];

```

```

B := EvalKroneckerMat( C )[2];

if IsBound(RP!.KroneckerMat) then
  return RP!.KroneckerMat( A, B );
fi;

if not IsHomalgInternalMatrixRep( C ) then
  Error( "could not find a procedure called KroneckerMat ",
        "in the homalgTable of the non-internal ring\n" );
fi;

##### can only work for homalg internal matrices #####

return homalgInternalMatrixHull(
  KroneckerProduct( Eval( A )!.matrix, Eval( B )!.matrix );
## this was easy, thanks GAP :)

end );

```

#### C.4.14 Eval (for matrices created with MulMat)

▷ Eval(*C*)

(method)

**Returns:** the Eval value of a homalg matrix *C*

In case the matrix was created using `\*` (5.5.13) then the filter HasEvalMulMat for *C* is set to true and the homalgTable function MulMat (B.1.12) will be used to set the attribute Eval.

```

Code
InstallMethod( Eval,
  "for homalg matrices (HasEvalMulMat)",
  [ IsHomalgMatrix and HasEvalMulMat ],

function( C )
  local R, RP, e, a, A;

  R := HomalgRing( C );

  RP := homalgTable( R );

  e := EvalMulMat( C );

  a := e[1];
  A := e[2];

  if IsBound(RP!.MulMat) then
    return RP!.MulMat( a, A );
  fi;

  if not IsHomalgInternalMatrixRep( C ) then
    Error( "could not find a procedure called MulMat ",
          "in the homalgTable of the non-internal ring\n" );
  fi;

  ##### can only work for homalg internal matrices #####

```

```

    return a * Eval( A );
end );

InstallMethod( Eval,
    "for homalg matrices (HasEvalMulMatRight)",
    [ IsHomalgMatrix and HasEvalMulMatRight ],

function( C )
    local R, RP, e, A, a;

    R := HomalgRing( C );

    RP := homalgTable( R );

    e := EvalMulMatRight( C );

    A := e[1];
    a := e[2];

    if IsBound(RP!.MulMatRight) then
        return RP!.MulMatRight( A, a );
    fi;

    if not IsHomalgInternalMatrixRep( C ) then
        Error( "could not find a procedure called MulMatRight ",
            "in the homalgTable of the non-internal ring\n" );
    fi;

    #=====# can only work for homalg internal matrices #=====#

    return Eval( A ) * a;

end );

```

### C.4.15 Eval (for matrices created with AddMat)

▷ Eval(*C*)

(method)

**Returns:** the Eval value of a homalg matrix *C*

In case the matrix was created using `\+` (5.5.14) then the filter `HasEvalAddMat` for *C* is set to true and the `homalgTable` function `AddMat` (B.1.13) will be used to set the attribute `Eval`.

```

Code
InstallMethod( Eval,
    "for homalg matrices (HasEvalAddMat)",
    [ IsHomalgMatrix and HasEvalAddMat ],

function( C )
    local R, RP, e, A, B;

    R := HomalgRing( C );

```

```

RP := homalgTable( R );

e := EvalAddMat( C );

A := e[1];
B := e[2];

if IsBound(RP!.AddMat) then
  return RP!.AddMat( A, B );
fi;

if not IsHomalgInternalMatrixRep( C ) then
  Error( "could not find a procedure called AddMat ",
        "in the homalgTable of the non-internal ring\n" );
fi;

##### can only work for homalg internal matrices #####

return Eval( A ) + Eval( B );

end );

```

#### C.4.16 Eval (for matrices created with SubMat)

▷ Eval(*C*)

(method)

**Returns:** the Eval value of a homalg matrix *C*

In case the matrix was created using \- (5.5.15) then the filter HasEvalSubMat for *C* is set to true and the homalgTable function SubMat (B.1.14) will be used to set the attribute Eval.

Code

```

InstallMethod( Eval,
  "for homalg matrices (HasEvalSubMat)",
  [ IsHomalgMatrix and HasEvalSubMat ],

function( C )
  local R, RP, e, A, B;

  R := HomalgRing( C );

  RP := homalgTable( R );

  e := EvalSubMat( C );

  A := e[1];
  B := e[2];

  if IsBound(RP!.SubMat) then
    return RP!.SubMat( A, B );
  fi;

  if not IsHomalgInternalMatrixRep( C ) then
    Error( "could not find a procedure called SubMat ",
          "in the homalgTable of the non-internal ring\n" );
  fi;
end );

```

```

fi;

#=====# can only work for homalg internal matrices #=====#

return Eval( A ) - Eval( B );

end );

```

### C.4.17 Eval (for matrices created with Compose)

▷ Eval(*C*) (method)

**Returns:** the Eval value of a homalg matrix *C*

In case the matrix was created using \\* (5.5.16) then the filter HasEvalCompose for *C* is set to true and the homalgTable function Compose (B.1.15) will be used to set the attribute Eval.

Code

```

InstallMethod( Eval,
  "for homalg matrices (HasEvalCompose)",
  [ IsHomalgMatrix and HasEvalCompose ],

function( C )
  local R, RP, e, A, B;

  R := HomalgRing( C );

  RP := homalgTable( R );

  e := EvalCompose( C );

  A := e[1];
  B := e[2];

  if IsBound(RP!.Compose) then
    return RP!.Compose( A, B );
  fi;

  if not IsHomalgInternalMatrixRep( C ) then
    Error( "could not find a procedure called Compose ",
          "in the homalgTable of the non-internal ring\n" );
  fi;

  #=====# can only work for homalg internal matrices #=====#

  return Eval( A ) * Eval( B );

end );

```

## Appendix D

# The subpackage ResidueClassRingForHomalg as a sample ring package

### D.1 The Mandatory Basic Operations

#### D.1.1 BasisOfRowModule (ResidueClassRing)

▷ `BasisOfRowModule( $M$ )` (function)  
**Returns:** a `homalg` matrix over the ambient ring

```
Code
BasisOfRowModule :=
function( M )
  local Mrel;

  Mrel := UnionOfRows( M );

  Mrel := HomalgResidueClassMatrix(
    BasisOfRowModule( Mrel ), HomalgRing( M ) );

  return GetRidOfObsoleteRows( Mrel );

end,
```

#### D.1.2 BasisOfColumnModule (ResidueClassRing)

▷ `BasisOfColumnModule( $M$ )` (function)  
**Returns:** a `homalg` matrix over the ambient ring

```
Code
BasisOfColumnModule :=
function( M )
  local Mrel;

  Mrel := UnionOfColumns( M );
```



```

Mrel := HomalgResidueClassMatrix(
    BasisOfColumnModule( Mrel ), HomalgRing( M ) );

return GetRidOfObsoleteColumns( Mrel );

end,

```

### D.1.3 DecideZeroRows (ResidueClassRing)

▷ DecideZeroRows( $A$ ,  $B$ ) (function)

**Returns:** a homalg matrix over the ambient ring

```

Code
DecideZeroRows :=
function( A, B )
    local Brel;

    Brel := UnionOfRows( B );

    Brel := BasisOfRowModule( Brel );

    return HomalgResidueClassMatrix(
        DecideZeroRows( Eval( A ), Brel ), HomalgRing( A ) );

end,

```

### D.1.4 DecideZeroColumns (ResidueClassRing)

▷ DecideZeroColumns( $A$ ,  $B$ ) (function)

**Returns:** a homalg matrix over the ambient ring

```

Code
DecideZeroColumns :=
function( A, B )
    local Brel;

    Brel := UnionOfColumns( B );

    Brel := BasisOfColumnModule( Brel );

    return HomalgResidueClassMatrix(
        DecideZeroColumns( Eval( A ), Brel ), HomalgRing( A ) );

end,

```

### D.1.5 SyzygiesGeneratorsOfRows (ResidueClassRing)

▷ SyzygiesGeneratorsOfRows( $M$ ) (function)

**Returns:** a homalg matrix over the ambient ring

```

Code
SyzygiesGeneratorsOfRows :=
function( M )

```

```

local R, ring_rel, rel, S;

R := HomalgRing( M );

ring_rel := RingRelations( R );

rel := MatrixOfRelations( ring_rel );

if IsHomalgRingRelationsAsGeneratorsOfRightIdeal( ring_rel ) then
    rel := Involution( rel );
fi;

rel := DiagMat( ListWithIdenticalEntries( NrColumns( M ), rel ) );

S := SyzygiesGeneratorsOfRows( Eval( M ), rel );

S := HomalgResidueClassMatrix( S, R );

S := GetRidOfObsoleteRows( S );

if IsZero( S ) then
    SetIsLeftRegular( M, true );
fi;

return S;

end,

```

### D.1.6 SyzygiesGeneratorsOfColumns (ResidueClassRing)

▷ SyzygiesGeneratorsOfColumns( $M$ )

(function)

**Returns:** a homalg matrix over the ambient ring

Code

```

SyzygiesGeneratorsOfColumns :=
function( M )
    local R, ring_rel, rel, S;

    R := HomalgRing( M );

    ring_rel := RingRelations( R );

    rel := MatrixOfRelations( ring_rel );

    if IsHomalgRingRelationsAsGeneratorsOfLeftIdeal( ring_rel ) then
        rel := Involution( rel );
    fi;

    rel := DiagMat( ListWithIdenticalEntries( NrRows( M ), rel ) );

    S := SyzygiesGeneratorsOfColumns( Eval( M ), rel );

```

```

S := HomalgResidueClassMatrix( S, R );
S := GetRidOfObsoleteColumns( S );
if IsZero( S ) then
    SetIsRightRegular( M, true );
fi;
return S;
end,

```

### D.1.7 BasisOfRowsCoeff (ResidueClassRing)

▷ BasisOfRowsCoeff( $M$ ,  $T$ )

(function)

**Returns:** a homalg matrix over the ambient ring

Code

```

BasisOfRowsCoeff :=
function( M, T )
    local Mrel, TT, bas, nz;

    Mrel := UnionOfRows( M );

    TT := HomalgVoidMatrix( HomalgRing( Mrel ) );

    bas := BasisOfRowsCoeff( Mrel, TT );

    bas := HomalgResidueClassMatrix( bas, HomalgRing( M ) );

    nz := NonZeroRows( bas );

    SetEval( T, CertainRows( CertainColumns( TT, [ 1 .. NrRows( M ) ] ), nz ) );

    ResetFilterObj( T, IsVoidMatrix );

    ## the generic BasisOfRowsCoeff will assume that
    ## ( NrRows( B ) = 0 ) = IsZero( B )
    return CertainRows( bas, nz );

end,

```

### D.1.8 BasisOfColumnsCoeff (ResidueClassRing)

▷ BasisOfColumnsCoeff( $M$ ,  $T$ )

(function)

**Returns:** a homalg matrix over the ambient ring

Code

```

BasisOfColumnsCoeff :=
function( M, T )

```

```

local Mrel, TT, bas, nz;

Mrel := UnionOfColumns( M );

TT := HomalgVoidMatrix( HomalgRing( Mrel ) );

bas := BasisOfColumnsCoeff( Mrel, TT );

bas := HomalgResidueClassMatrix( bas, HomalgRing( M ) );

nz := NonZeroColumns( bas );

SetEval( T, CertainColumns( CertainRows( TT, [ 1 .. NrColumns( M ) ] ), nz ) );

ResetFilterObj( T, IsVoidMatrix );

## the generic BasisOfColumnsCoeff will assume that
## ( NrColumns( B ) = 0 ) = IsZero( B )
return CertainColumns( bas, nz );

end,

```

### D.1.9 DecideZeroRowsEffectively (ResidueClassRing)

▷ DecideZeroRowsEffectively( $A, B, T$ ) (function)  
**Returns:** a homalg matrix over the ambient ring

Code

```

DecideZeroRowsEffectively :=
function( A, B, T )
  local Brel, TT, red;

  Brel := UnionOfRows( B );

  TT := HomalgVoidMatrix( HomalgRing( Brel ) );

  red := DecideZeroRowsEffectively( Eval( A ), Brel, TT );

  SetEval( T, CertainColumns( TT, [ 1 .. NrRows( B ) ] ) );

  ResetFilterObj( T, IsVoidMatrix );

  return HomalgResidueClassMatrix( red, HomalgRing( A ) );

end,

```

### D.1.10 DecideZeroColumnsEffectively (ResidueClassRing)

▷ DecideZeroColumnsEffectively( $A, B, T$ ) (function)  
**Returns:** a homalg matrix over the ambient ring

Code

```

DecideZeroColumnsEffectively :=
function( A, B, T )
  local Brel, TT, red;

  Brel := UnionOfColumns( B );

  TT := HomalgVoidMatrix( HomalgRing( Brel ) );

  red := DecideZeroColumnsEffectively( Eval( A ), Brel, TT );

  SetEval( T, CertainRows( TT, [ 1 .. NrColumns( B ) ] ) );

  ResetFilterObj( T, IsVoidMatrix );

  return HomalgResidueClassMatrix( red, HomalgRing( A ) );

end,

```

### D.1.11 RelativeSyzygiesGeneratorsOfRows (ResidueClassRing)

▷ RelativeSyzygiesGeneratorsOfRows( $M$ ,  $M2$ ) (function)

**Returns:** a homalg matrix over the ambient ring

Code

```

RelativeSyzygiesGeneratorsOfRows :=
function( M, M2 )
  local M2rel, S;

  M2rel := UnionOfRows( M2 );

  S := SyzygiesGeneratorsOfRows( Eval( M ), M2rel );

  S := HomalgResidueClassMatrix( S, HomalgRing( M ) );

  S := GetRidOfObsoleteRows( S );

  if IsZero( S ) then
    SetIsLeftRegular( M, true );
  fi;

  return S;

end,

```

### D.1.12 RelativeSyzygiesGeneratorsOfColumns (ResidueClassRing)

▷ RelativeSyzygiesGeneratorsOfColumns( $M$ ,  $M2$ ) (function)

**Returns:** a homalg matrix over the ambient ring

```

Code
-----
RelativeSyzygiesGeneratorsOfColumns :=
function( M, M2 )
  local M2rel, S;

  M2rel := UnionOfColumns( M2 );

  S := SyzygiesGeneratorsOfColumns( Eval( M ), M2rel );

  S := HomalgResidueClassMatrix( S, HomalgRing( M ) );

  S := GetRidOfObsoleteColumns( S );

  if IsZero( S ) then
    SetIsRightRegular( M, true );
  fi;

  return S;

end,

```

## D.2 The Mandatory Tool Operations

Here we list those matrix operations for which homalg provides no fallback method.

### D.2.1 InitialMatrix (ResidueClassRing)

▷ InitialMatrix() (function)  
**Returns:** a homalg matrix over the ambient ring  
 (→ InitialMatrix (B.1.1))

```

Code
-----
InitialMatrix := C -> HomalgInitialMatrix(
  NrRows( C ), NrColumns( C ), AmbientRing( HomalgRing( C ) ) ),

```

### D.2.2 InitialIdentityMatrix (ResidueClassRing)

▷ InitialIdentityMatrix() (function)  
**Returns:** a homalg matrix over the ambient ring  
 (→ InitialIdentityMatrix (B.1.2))

```

Code
-----
InitialIdentityMatrix := C -> HomalgInitialIdentityMatrix(
  NrRows( C ), AmbientRing( HomalgRing( C ) ) ),

```

### D.2.3 ZeroMatrix (ResidueClassRing)

▷ ZeroMatrix() (function)  
**Returns:** a homalg matrix over the ambient ring

(→ ZeroMatrix (B.1.3))

Code

```
ZeroMatrix := C -> HomalgZeroMatrix(
    NrRows( C ), NrColumns( C ), AmbientRing( HomalgRing( C ) ) ),
```

#### D.2.4 IdentityMatrix (ResidueClassRing)

▷ IdentityMatrix()

(function)

**Returns:** a homalg matrix over the ambient ring

(→ IdentityMatrix (B.1.4))

Code

```
IdentityMatrix := C -> HomalgIdentityMatrix(
    NrRows( C ), AmbientRing( HomalgRing( C ) ) ),
```

#### D.2.5 Involution (ResidueClassRing)

▷ Involution()

(function)

**Returns:** a homalg matrix over the ambient ring

(→ Involution (B.1.5))

Code

```
Involution :=
function( M )
    local N, R;

    N := Involution( Eval( M ) );

    R := HomalgRing( N );

    if not ( HasIsCommutative( R ) and IsCommutative( R ) and
            HasIsReducedModuloRingRelations( M ) and
            IsReducedModuloRingRelations( M ) ) then

        ## reduce the matrix N w.r.t. the ring relations
        N := DecideZero( N, HomalgRing( M ) );
    fi;

    return N;

end,
```

#### D.2.6 CertainRows (ResidueClassRing)

▷ CertainRows()

(function)

**Returns:** a homalg matrix over the ambient ring

(→ CertainRows (B.1.6))

Code

```
CertainRows :=
function( M, plist )
    local N;
```

```

N := CertainRows( Eval( M ), plist );

if not ( HasIsReducedModuloRingRelations( M ) and
         IsReducedModuloRingRelations( M ) ) then

    ## reduce the matrix N w.r.t. the ring relations
    N := DecideZero( N, HomalgRing( M ) );
fi;

return N;

end,

```

### D.2.7 CertainColumns (ResidueClassRing)

▷ CertainColumns()

(function)

**Returns:** a homalg matrix over the ambient ring  
(→ CertainColumns (B.1.7))

Code

```

CertainColumns :=
function( M, plist )
  local N;

  N := CertainColumns( Eval( M ), plist );

  if not ( HasIsReducedModuloRingRelations( M ) and
           IsReducedModuloRingRelations( M ) ) then

      ## reduce the matrix N w.r.t. the ring relations
      N := DecideZero( N, HomalgRing( M ) );
  fi;

  return N;

end,

```

### D.2.8 UnionOfRows (ResidueClassRing)

▷ UnionOfRows()

(function)

**Returns:** a homalg matrix over the ambient ring  
(→ UnionOfRows (B.1.8))

Code

```

UnionOfRows :=
function( A, B )
  local N;

  N := UnionOfRows( Eval( A ), Eval( B ) );

  if not ForAll( [ A, B ], HasIsReducedModuloRingRelations and
                 IsReducedModuloRingRelations ) then

```



```

    ## reduce the matrix N w.r.t. the ring relations
    N := DecideZero( N, HomalgRing( A ) );
fi;

return N;

end,

```

## D.2.9 UnionOfColumns (ResidueClassRing)

▷ UnionOfColumns() (function)  
**Returns:** a homalg matrix over the ambient ring  
(→ UnionOfColumns (B.1.9))

Code

```

UnionOfColumns :=
function( A, B )
  local N;

  N := UnionOfColumns( Eval( A ), Eval( B ) );

  if not ForAll( [ A, B ], HasIsReducedModuloRingRelations and
    IsReducedModuloRingRelations ) then

    ## reduce the matrix N w.r.t. the ring relations
    N := DecideZero( N, HomalgRing( A ) );
  fi;

  return N;

end,

```

## D.2.10 DiagMat (ResidueClassRing)

▷ DiagMat() (function)  
**Returns:** a homalg matrix over the ambient ring  
(→ DiagMat (B.1.10))

Code

```

DiagMat :=
function( e )
  local N;

  N := DiagMat( List( e, Eval ) );

  if not ForAll( e, HasIsReducedModuloRingRelations and
    IsReducedModuloRingRelations ) then

    ## reduce the matrix N w.r.t. the ring relations
    N := DecideZero( N, HomalgRing( e[1] ) );
  fi;

  return N;

```

```
end,
```

### D.2.11 KroneckerMat (ResidueClassRing)

▷ KroneckerMat() (function)

**Returns:** a homalg matrix over the ambient ring  
(→ KroneckerMat (B.1.11))

Code

```
KroneckerMat :=
function( A, B )
  local N;

  N := KroneckerMat( Eval( A ), Eval( B ) );

  if not ForAll( [ A, B ], HasIsReducedModuloRingRelations and
    IsReducedModuloRingRelations ) then

    ## reduce the matrix N w.r.t. the ring relations
    N := DecideZero( N, HomalgRing( A ) );
  fi;

  return N;

end,
```

### D.2.12 MulMat (ResidueClassRing)

▷ MulMat() (function)

**Returns:** a homalg matrix over the ambient ring  
(→ MulMat (B.1.12))

Code

```
MulMat :=
function( a, A )

  return DecideZero( EvalRingElement( a ) * Eval( A ), HomalgRing( A ) );

end,
MulMatRight :=
function( A, a )

  return DecideZero( Eval( A ) * EvalRingElement( a ), HomalgRing( A ) );

end,
```

### D.2.13 AddMat (ResidueClassRing)

▷ AddMat() (function)

**Returns:** a homalg matrix over the ambient ring  
(→ AddMat (B.1.13))

Code

```
AddMat :=
  function( A, B )

    return DecideZero( Eval( A ) + Eval( B ), HomalgRing( A ) );

  end,
```

### D.2.14 SubMat (ResidueClassRing)

▷ SubMat() (function)  
**Returns:** a homalg matrix over the ambient ring  
 (→ SubMat (B.1.14))

Code

```
SubMat :=
  function( A, B )

    return DecideZero( Eval( A ) - Eval( B ), HomalgRing( A ) );

  end,
```

### D.2.15 Compose (ResidueClassRing)

▷ Compose() (function)  
**Returns:** a homalg matrix over the ambient ring  
 (→ Compose (B.1.15))

Code

```
Compose :=
  function( A, B )

    return DecideZero( Eval( A ) * Eval( B ), HomalgRing( A ) );

  end,
```

### D.2.16 IsZeroMatrix (ResidueClassRing)

▷ IsZeroMatrix( $M$ ) (function)  
**Returns:** true or false  
 (→ IsZeroMatrix (B.1.16))

Code

```
IsZeroMatrix := M -> IsZero( DecideZero( Eval( M ), HomalgRing( M ) ) ),
```

### D.2.17 NrRows (ResidueClassRing)

▷ NrRows( $C$ ) (function)  
**Returns:** a nonnegative integer  
 (→ NrRows (B.1.17))

Code

```
NrRows := C -> NrRows( Eval( C ) ),
```

**D.2.18 NrColumns (ResidueClassRing)**

▷ `NrColumns(C)` (function)

**Returns:** a nonnegative integer  
(→ `NrColumns (B.1.18)`)

```
Code
NrColumns := C -> NrColumns( Eval( C ) ),
```

**D.2.19 Determinant (ResidueClassRing)**

▷ `Determinant(C)` (function)

**Returns:** an element of ambient homalg ring  
(→ `Determinant (B.1.19)`)

```
Code
Determinant := C -> DecideZero( Determinant( Eval( C ) ), HomalgRing( C ) ),
```

**D.3 Some of the Recommended Tool Operations**

Here we list those matrix operations for which `homalg` does provide a fallback method. But specifying the below `homalgTable` functions increases the performance by replacing the fallback method.

**D.3.1 AreEqualMatrices (ResidueClassRing)**

▷ `AreEqualMatrices(A, B)` (function)

**Returns:** true or false  
(→ `AreEqualMatrices (B.2.1)`)

```
Code
AreEqualMatrices :=
function( A, B )

    return IsZero( DecideZero( Eval( A ) - Eval( B ), HomalgRing( A ) ) );

end,
```

**D.3.2 IsOne (ResidueClassRing)**

▷ `IsOne(M)` (function)

**Returns:** true or false  
(→ `IsIdentityMatrix (B.2.2)`)

```
Code
IsIdentityMatrix := M ->
    IsOne( DecideZero( Eval( M ), HomalgRing( M ) ) ),
```

**D.3.3 IsDiagonalMatrix (ResidueClassRing)**

▷ `IsDiagonalMatrix(M)` (function)

**Returns:** true or false  
(→ `IsDiagonalMatrix (B.2.3)`)

Code

```
IsDiagonalMatrix := M ->
  IsDiagonalMatrix( DecideZero( Eval( M ), HomalgRing( M ) ) ),
```

### D.3.4 ZeroRows (ResidueClassRing)

▷ ZeroRows( $C$ ) (function)  
**Returns:** a homalg matrix over the ambient ring  
 (→ ZeroRows (B.2.4))

Code

```
ZeroRows := C -> ZeroRows( DecideZero( Eval( C ), HomalgRing( C ) ) ),
```

### D.3.5 ZeroColumns (ResidueClassRing)

▷ ZeroColumns( $C$ ) (function)  
**Returns:** a homalg matrix over the ambient ring  
 (→ ZeroColumns (B.2.5))

Code

```
ZeroColumns := C -> ZeroColumns( DecideZero( Eval( C ), HomalgRing( C ) ) ),
```

## Appendix E

# Debugging MatricesForHomalg

Beside the GAP builtin debugging facilities ( $\rightarrow$  **(Reference: Debugging and Profiling Facilities)**) MatricesForHomalg provides two ways to debug the computations.

### E.1 Increase the assertion level

MatricesForHomalg comes with numerous builtin assertion checks. They are activated if the user increases the assertion level using

```
SetAssertionLevel( level );
```

( $\rightarrow$  **(Reference: SetAssertionLevel)**), where *level* is one of the values below:

<i>level</i>	description
0	no assertion checks whatsoever
4	assertions about basic matrix operations are checked ( $\rightarrow$ <a href="#">Appendix A</a> ) (these are among the operations often delegated to external systems)

In particular, if MatricesForHomalg delegates matrix operations to an external system then `SetAssertionLevel( 4 );` can be used to let MatricesForHomalg debug the external system.

Below you can find the record of the available level-4 assertions, which is a GAP-component of every homalg ring. Each assertion can thus be overwritten by package developers or even ordinary users.

Code

```
asserts :=
  rec(
    BasisOfRowModule :=
      function( B ) return ( NrRows( B ) = 0 ) = IsZero( B ); end,

    BasisOfColumnModule :=
      function( B ) return ( NrColumns( B ) = 0 ) = IsZero( B ); end,
```

```

BasisOfRowsCoeff :=
  function( B, T, M ) return B = T * M; end,

BasisOfColumnsCoeff :=
  function( B, M, T ) return B = M * T; end,

DecideZeroRows_Effectively :=
  function( M, A, B ) return M = DecideZeroRows( A, B ); end,

DecideZeroColumns_Effectively :=
  function( M, A, B ) return M = DecideZeroColumns( A, B ); end,

DecideZeroRowsEffectively :=
  function( M, A, T, B ) return M = A + T * B; end,

DecideZeroColumnsEffectively :=
  function( M, A, B, T ) return M = A + B * T; end,

DecideZeroRowsWRTNonBasis :=
  function( B )
    local R;
    R := HomalgRing( B );
    if not ( HasIsBasisOfRowsMatrix( B ) and
              IsBasisOfRowsMatrix( B ) ) and
        IsBound( R!.DecideZeroWRTNonBasis ) then
      if R!.DecideZeroWRTNonBasis = "warn" then
        Info( InfoWarning, 1,
              "about to reduce with respect to a matrix",
              "with IsBasisOfRowsMatrix not set to true" );
      elif R!.DecideZeroWRTNonBasis = "error" then
        Error( "about to reduce with respect to a matrix",
              "with IsBasisOfRowsMatrix not set to true\n" );
      fi;
    fi;
  end,

DecideZeroColumnsWRTNonBasis :=
  function( B )
    local R;
    R := HomalgRing( B );
    if not ( HasIsBasisOfColumnsMatrix( B ) and
              IsBasisOfColumnsMatrix( B ) ) and
        IsBound( R!.DecideZeroWRTNonBasis ) then
      if R!.DecideZeroWRTNonBasis = "warn" then
        Info( InfoWarning, 1,
              "about to reduce with respect to a matrix",
              "with IsBasisOfColumnsMatrix not set to true" );
      elif R!.DecideZeroWRTNonBasis = "error" then
        Error( "about to reduce with respect to a matrix",
              "with IsBasisOfColumnsMatrix not set to true\n" );
      fi;
    fi;
  end,

```

```

ReducedBasisOfRowModule :=
function( M, B )
  return GenerateSameRowModule( B, BasisOfRowModule( M ) );
end,

ReducedBasisOfColumnModule :=
function( M, B )
  return GenerateSameColumnModule( B, BasisOfColumnModule( M ) );
end,

ReducedSyzygiesGeneratorsOfRows :=
function( M, S )
  return GenerateSameRowModule( S, SyzygiesGeneratorsOfRows( M ) );
end,

ReducedSyzygiesGeneratorsOfColumns :=
function( M, S )
  return GenerateSameColumnModule( S, SyzygiesGeneratorsOfColumns( M ) );
end,

);

```

## E.2 Using homalgMode

### E.2.1 homalgMode

▷ `homalgMode(str[, str2])` (method)

This function sets different modes which influence how much of the basic matrix operations and the logical matrix methods become visible (→ Appendices A, C). Handling the string *str* is *not* case-sensitive. If a second string *str2* is given, then `homalgMode( str2 )` is invoked at the end. In case you let `homalg` delegate matrix operations to an external system the you might also want to check `homalgIOMode` in the `HomalgToCAS` package manual.

<i>str</i>	<i>str</i> (long form)	mode description
""	""	the default mode, i.e. the computation protocol won't be visible ( <code>homalgMode( )</code> is a short form for <code>homalgMode( "" )</code> )
"b"	"basic"	make the basic matrix operations visible + <code>homalgMode( "logic" )</code>
"d"	"debug"	same as "basic" but also makes <code>Row/ColumnReducedEchelonForm</code> visible
"l"	"logic"	make the logical methods in <code>LIMAT</code> and <code>COLEM</code> visible

All modes other than the "default"-mode only set their specific values and leave the other values untouched, which allows combining them to some extent. This also means that in order to get from



one mode to a new mode (without the aim to combine them) one needs to reset to the "default"-mode first. This can be done using `homalgMode( "", new_mode );`

```

Code
InstallGlobalFunction( homalgMode,
function( arg )
  local nargs, mode, s;

  nargs := Length( arg );

  if nargs = 0 or ( IsString( arg[1] ) and arg[1] = "" ) then
    mode := "default";
  elif IsString( arg[1] ) then      ## now we know, the string is not empty
    s := arg[1];
    if LowercaseString( s{[1]} ) = "b" then
      mode := "basic";
    elif LowercaseString( s{[1]} ) = "d" then
      mode := "debug";
    elif LowercaseString( s{[1]} ) = "l" then
      mode := "logic";
    else
      mode := "";
    fi;
  else
    Error( "the first argument must be a string\n" );
  fi;

  if mode = "default" then
    HOMALG_MATRICES.color_display := false;
    for s in HOMALG_MATRICES.matrix_logic_infolevels do
      SetInfoLevel( s, 1 );
    od;
    SetInfoLevel( InfoHomalgBasicOperations, 1 );
  elif mode = "basic" then
    SetInfoLevel( InfoHomalgBasicOperations, 3 );
    homalgMode( "logic" );
  elif mode = "debug" then
    SetInfoLevel( InfoHomalgBasicOperations, 4 );
    homalgMode( "logic" );
  elif mode = "logic" then
    HOMALG_MATRICES.color_display := true;
    for s in HOMALG_MATRICES.matrix_logic_infolevels do
      SetInfoLevel( s, 2 );
    od;
  fi;

  if nargs > 1 and IsString( arg[2] ) then
    homalgMode( arg[2] );
  fi;

end );

```

## Appendix F

# Overview of the `MatricesForHomalg` Package Source Code

### F.1 Rings, Ring Maps, Matrices, Ring Relations

Filename .gd/.gi	Content
<code>homalg</code>	definitions of the basic GAP4 categories and some tool functions (e.g. <code>homalgMode</code> )
<code>homalgTable</code>	dictionaries between <code>MatricesForHomalg</code> and the computing engines
<code>HomalgRing</code>	internal and external rings
<code>HomalgRingMap</code>	ring maps
<code>HomalgMatrix</code>	internal and external matrices
<code>HomalgRingRelations</code>	a set of ring relations

**Table:** *The `MatricesForHomalg` package files*

### F.2 The Low Level Algorithms

In the following CAS or CASystem mean computer algebra systems.

Filename .gd/.gi	Content
Tools	the elementary matrix operations that can be overwritten using the homalgTable (and hence delegable even to other CASystems)
Service	the three operations: basis, reduction, and syzygies; they can also be overwritten using the homalgTable (and hence delegable even to other CASystems)
Basic	higher level operations for matrices (cannot be overwritten using the homalgTable)

**Table:** *The MatricesForHomalg package files (continued)*

### F.3 Logical Implications for MatricesForHomalg Objects

Filename .gd/.gi	Content
LIRNG	logical implications for rings
LIMAP	logical implications for ring maps
LIMAT	logical implications for matrices
COLEM	clever operations for lazy evaluated matrices

**Table:** *The MatricesForHomalg package files (continued)*

### F.4 The subpackage ResidueClassRingForHomalg

Filename .gd/.gi	Content
ResidueClassRingForHomalg	some global variables
ResidueClassRing	residue class rings, their elements, and matrices, together with their constructors and operations
ResidueClassRingTools	the elementary matrix operations for matrices over residue class rings
ResidueClassRingBasic	the three operations: basis, reduction, and syzygies for matrices over residue class rings

**Table:** *The MatricesForHomalg package files (continued)*

## F.5 The homalgTable for GAP4 built-in rings

For the purposes of homalg, the ring of integers is, at least up till now, the only ring which is properly supported in GAP4. The GAP4 built-in capabilities for polynomial rings (also univariate) and group rings do not satisfy the minimum requirements of homalg. The GAP4 package Gauss enables GAP to fulfill the homalg requirements for prime fields, and  $\mathbb{Z}/p^n$ .

Filename .gi	Content
Integers	the homalgTable for the ring of integers

**Table:** *The MatricesForHomalg package files (continued)*

# References

- [BR08] Mohamed Barakat and Daniel Robertz. homalg – A Meta-Package for Homological Algebra. *J. Algebra Appl.*, 7(3):299–317, 2008. arXiv:math.AC/0701146. [49](#), [50](#)
- [MR01] J. C. McConnell and J. C. Robson. *Noncommutative Noetherian rings*, volume 30 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, revised edition, 2001. With the cooperation of L. W. Small. [22](#)

# Index

- $\backslash*$ 
  - copy a matrix over a different ring, 30
  - copy a matrix over a different ring (right), 30
  - for composable matrices, 42
  - for ring elements and matrices, 42
- $\backslash+$ 
  - for matrices, 42
- $\backslash-$ 
  - for matrices, 42
- $\backslash/$ 
  - constructor for residue class rings, 10
- $\backslash=$ 
  - for matrices, 42
- MatricesForHomalg, 5
- AddMat
  - homalgTable entry, 58
  - ResidueClassRing, 97
- AffineDegree, 37
- AffineDimension, 37
- AreEqualMatrices
  - homalgTable entry, 62
  - ResidueClassRing, 99
- AreUnitsCentral, 18
- AssociatedGradedRing, 22
- BasisAlgorithmRespectsPrincipalIdeals, 18
- BasisOfColumnModule
  - for matrices, 44
  - ResidueClassRing, 87
- BasisOfColumns
  - for matrices, 47
  - for pairs of matrices, 47
- BasisOfColumnsCoeff
  - for matrices, 46
  - ResidueClassRing, 90
- BasisOfRowModule
  - for matrices, 44
  - ResidueClassRing, 87
- BasisOfRows
  - for matrices, 47
  - for pairs of matrices, 47
- BasisOfRowsCoeff
  - for matrices, 46
  - ResidueClassRing, 90
- CanBeUsedToDecideZero, 53
- CertainColumns
  - for matrices, 41
  - homalgTable entry, 57
  - ResidueClassRing, 95
- CertainRows
  - for matrices, 41
  - homalgTable entry, 57
  - ResidueClassRing, 94
- CoefficientsOfNumeratorOfHilbertPoincareSeries, 37
- CoefficientsOfUnreducedNumeratorOfHilbertPoincareSeries, 37
- CoefficientsRing, 21
- ColumnRankOfMatrix, 36
- Compose
  - homalgTable entry, 58
  - ResidueClassRing, 98
- ConstantTermOfHilbertPolynomialn, 38
- ConstructorForHomalgMatrices, 19
- ContainsAField, 11
- CoordinateRingOfGraph
  - for ring maps, 25
- DecideZero
  - for matrices and relations, 47
- DecideZeroColumns
  - for pairs of matrices, 44
  - ResidueClassRing, 88
- DecideZeroColumnsEffectively
  - for pairs of matrices, 47

- ResidueClassRing, 91
- DecideZeroRows
  - for pairs of matrices, 44
  - ResidueClassRing, 88
- DecideZeroRowsEffectively
  - for pairs of matrices, 46
  - ResidueClassRing, 91
- DegreeOfMorphism
  - for ring maps, 25
- Determinant
  - homalgTable entry, 61
  - ResidueClassRing, 99
- DeterminantMat, 35
- DiagMat
  - for matrices, 41
  - homalgTable entry, 58
  - ResidueClassRing, 96
- ElementaryRank, 22
- Eliminate, 43
- Eval
  - for matrices created with AddMat, 84
  - for matrices created with CertainColumns, 79
  - for matrices created with CertainRows, 78
  - for matrices created with Compose, 86
  - for matrices created with DiagMat, 81
  - for matrices created with HomalgIdentityMatrix, 75
  - for matrices created with HomalgInitialIdentityMatrix, 73
  - for matrices created with HomalgInitialMatrix, 72
  - for matrices created with HomalgZeroMatrix, 74
  - for matrices created with Involution, 77
  - for matrices created with KroneckerMat, 82
  - for matrices created with LeftInverseLazy, 76
  - for matrices created with MulMat, 83
  - for matrices created with RightInverseLazy, 77
  - for matrices created with SubMat, 85
  - for matrices created with UnionOfColumns, 80
  - for matrices created with UnionOfRows, 79
- GeneralLinearRank, 22
- GenerateSameColumnModule
  - for pairs of matrices, 51
- GenerateSameRowModule
  - for pairs of matrices, 51
- GetColumnIndependentUnitPositions
  - for matrices, 43
  - homalgTable entry, 66
- GetRowIndependentUnitPositions
  - for matrices, 43
  - homalgTable entry, 67
- GetUnitPosition
  - for matrices, 43
  - homalgTable entry, 68
- GlobalDimension, 22
- HasInvariantBasisProperty, 13
- HasLeftInvariantBasisProperty, 13
- HasRightInvariantBasisProperty, 14
- HilbertPoincareSeries, 37
- HilbertPolynomial, 37
- HomalgDiagonalMatrix
  - constructor for diagonal matrices, 30
- HomalgFieldOfRationals
  - constructor for the field of rationals, 10
- HomalgIdentityMatrix
  - constructor for identity matrices, 28
- HomalgInitialIdentityMatrix
  - constructor for initial quadratic matrices with ones on the diagonal, 27
- HomalgInitialMatrix
  - constructor for initial matrices filled with zeros, 26
- HomalgMatrix
  - constructor for matrices using a list, 29
  - constructor for matrices using a listlist, 29
  - constructor for matrices using a string of a list, 29
  - constructor for matrices using a string of a listlist, 29
- homalgMode, 103
- HomalgRing
  - for matrices, 38
- HomalgRingOfIntegers
  - constructor for the integers, 9
  - constructor for the residue class rings of the integers, 9
- homalgTable, 19

- HomalgVoidMatrix
  - constructor for void matrices, 28
- HomalgZeroMatrix
  - constructor for zero matrices, 28
- IdentityMatrix
  - homalgTable entry, 57
  - ResidueClassRing, 94
- IndeterminateAntiCommutingVariablesOf-
  - ExteriorRing, 21
- IndeterminateCoordinatesOfRingOf-
  - Derivations, 20
- IndeterminateDerivationsOfRingOf-
  - Derivations, 21
- IndeterminatesOfExteriorRing, 21
- IndeterminatesOfPolynomialRing, 20
- InitialIdentityMatrix
  - homalgTable entry for initial identity matrices, 56
  - ResidueClassRing, 93
- InitialMatrix
  - homalgTable entry for initial matrices, 56
  - ResidueClassRing, 93
- Inverse
  - for homalg ring elements, 19
- Involution
  - for matrices, 40
  - homalgTable entry, 57
  - ResidueClassRing, 94
- IsArtinian
  - for rings, 16
- IsAutomorphism
  - for ring maps, 24
- IsBasisOfColumnsMatrix, 34
- IsBasisOfRowsMatrix, 34
- IsBezoutRing, 12
- IsCohenMacaulay, 15
- IsDedekindDomain, 12
- IsDiagonalMatrix, 33
  - homalgTable entry, 64
  - ResidueClassRing, 99
- IsDiscreteValuationRing, 12
- IsDivisionRingForHomalg, 11
- IsEmptyMatrix, 33
- IsEpimorphism
  - for ring maps, 24
- IsFieldForHomalg, 11
- IsFiniteFreePresentationRing, 17
- IsFreePolynomialRing, 13
- IsGlobalDimensionFinite, 13
- IsGorenstein, 15
- IsHereditary, 14
- IsHermite, 14
- IsHomalgInternalMatrixRep, 26
- IsHomalgInternalRingRep, 9
- IsHomalgMatrix, 26
- IsHomalgRing, 8
- IsHomalgRingElement, 8
- IsHomalgRingMap, 23
- IsHomalgRingMapRep, 23
- IsHomalgRingRelations, 52
- IsHomalgRingRelationsAsGeneratorsOf-
  - LeftIdeal, 52
- IsHomalgRingRelationsAsGeneratorsOf-
  - RightIdeal, 52
- IsHomalgRingSelfMap, 23
- IsIdentityMatrix
  - homalgTable entry, 63
- IsIdentityMorphism
  - for ring maps, 24
- IsInitialIdentityMatrix, 34
- IsInitialMatrix, 34
- IsInjectivePresentation, 53
- IsIntegersForHomalg, 12
- IsIntegralDomain, 14
- IsIntegrallyClosedDomain, 12
- IsInvertibleMatrix, 32
- IsIsomorphism
  - for ring maps, 24
- IsKaplanskyHermite, 12
- IsKoszul, 15
- IsLeftArtinian, 16
- IsLeftFiniteFreePresentationRing, 17
- IsLeftGlobalDimensionFinite, 13
- IsLeftHereditary, 14
- IsLeftHermite, 15
- IsLeftInvertibleMatrix, 32
- IsLeftNoetherian, 15
- IsLeftOreDomain, 16
- IsLeftPrincipalIdealRing, 16
- IsLeftRegular, 32
  - for homalg ring elements, 18
- IsLocal, 14
- IsLocalizedWeylRing, 13



- IsLowerStairCaseMatrix, 34
- IsLowerTriangularMatrix, 33
- IsMinusOne, 18
- IsMonic
  - for homalg ring elements, 18
- IsMonicUptoUnit
  - for homalg ring elements, 18
- IsMonomorphism
  - for ring maps, 24
- IsMorphism
  - for ring maps, 24
- IsNoetherian, 15
- IsOne, 31
  - ResidueClassRing, 99
- IsOreDomain, 16
- IsPermutationMatrix, 32
- IsPreHomalgRing, 8
- IsPrincipalIdealRing, 16
- IsRationalsForHomalg, 11
- IsReducedBasisOfColumnsMatrix, 34
- IsReducedBasisOfRowsMatrix, 34
- IsRegular, 17
  - for homalg ring elements, 18
- IsResidueClassRingOfTheIntegers, 12
- IsRightArtinian, 16
- IsRightFiniteFreePresentationRing, 17
- IsRightGlobalDimensionFinite, 13
- IsRightHereditary, 14
- IsRightHermite, 15
- IsRightInvertibleMatrix, 32
- IsRightNoetherian, 15
- IsRightOreDomain, 16
- IsRightPrincipalIdealRing, 17
- IsRightRegular, 32
  - for homalg ring elements, 18
- IsRingRelationsRep, 52
- IsScalarlMatrix, 33
- IsSemiLocalRing, 14
- IsSemiSimpleRing, 17
- IsSimpleRing, 17
- IsSpecialSubidentityMatrix, 32
- IsStrictLowerTriangularMatrix, 33
- IsStrictUpperTriangularMatrix, 33
- IsSubidentityMatrix, 32
- IsSuperCommutative, 17
- IsTriangularMatrix, 34
- IsUniqueFactorizationDomain, 12
- IsUnitFree, 31
- IsUpperStairCaseMatrix, 33
- IsUpperTriangularMatrix, 33
- IsVoidMatrix, 35
- IsWeylRing, 13
- IsZero
  - for matrices, 31
  - for rings, 11
- IsZeroMatrix
  - homalgTable entry, 59
  - ResidueClassRing, 98
- KroneckerMat
  - for matrices, 41
  - homalgTable entry, 58
  - ResidueClassRing, 97
- KrullDimension, 21
- LeftDivide
  - for pairs of matrices, 48
  - for triples of matrices, 50
- LeftGlobalDimension, 22
- LeftInverse, 36
  - for matrices, 38
- LeftInverseLazy
  - for matrices, 40
- MatrixOfSymbols, 38
- MinusOne, 20
- MulMat
  - homalgTable entry, 58
  - ResidueClassRing, 97
- NonZeroColumns, 36
- NonZeroRows, 35
- NrColumns, 35
  - homalgTable entry, 60
  - ResidueClassRing, 99
- NrRows, 35
  - homalgTable entry, 59
  - ResidueClassRing, 98
- NumeratorOfHilbertPoincareSeries, 37
- One
  - for homalg rings, 20
- PositionOfFirstNonZeroEntryPerColumn, 36

- homalgTable entry, 70
- PositionOfFirstNonZeroEntryPerRow, 36
  - homalgTable entry, 69
- ProductOfIndeterminates, 20
- ProjectiveDegree, 38
- Range
  - for ring maps, 25
- RationalParameters, 20
- ReducedBasisOfColumnModule
  - for matrices, 45
- ReducedBasisOfRowModule
  - for matrices, 45
- ReducedSyzygiesGeneratorsOfColumns
  - for matrices, 46
- ReducedSyzygiesGeneratorsOfRows
  - for matrices, 46
- ReducedSyzygiesOfColumns
  - for matrices, 48
  - for pairs of matrices, 48
- ReducedSyzygiesOfRows
  - for matrices, 48
  - for pairs of matrices, 48
- RelativeIndeterminateAntiCommuting-VariablesOfExteriorRing, 21
- RelativeIndeterminateCoordinatesOfRingOfDerivations, 21
- RelativeIndeterminateDerivationsOfRingOfDerivations, 21
- RelativeIndeterminatesOfPolynomialRing, 20
- RelativeSyzygiesGeneratorsOfColumnsResidueClassRing, 92
- RelativeSyzygiesGeneratorsOfRowsResidueClassRing, 92
- RightDivide
  - for pairs of matrices, 48
  - for triples of matrices, 49
- RightGlobalDimension, 22
- RightInverse, 36
  - for matrices, 39
- RightInverseLazy
  - for matrices, 40
- RingElementConstructor, 19
- RingMap
  - constructor for ring maps, 23
- RowRankOfMatrix, 36
- Source
  - for ring maps, 25
- StableRank, 22
- SubMat
  - homalgTable entry, 58
  - ResidueClassRing, 98
- SyzygiesGeneratorsOfColumns
  - for matrices, 45
  - for pairs of matrices, 45
  - ResidueClassRing, 89
- SyzygiesGeneratorsOfRows
  - for matrices, 45
  - for pairs of matrices, 45
  - ResidueClassRing, 88
- SyzygiesOfColumns
  - for matrices, 48
  - for pairs of matrices, 48
- SyzygiesOfRows
  - for matrices, 47
  - for pairs of matrices, 47
- TypeOfHomalgMatrix, 19
- UnionOfColumns
  - for matrices, 41
  - homalgTable entry, 57
  - ResidueClassRing, 96
- UnionOfRows
  - for matrices, 41
  - homalgTable entry, 57
  - ResidueClassRing, 95
- UnreducedNumeratorOfHilbertPoincareSeries, 37
- Zero
  - for homalg rings, 20
- ZeroColumns, 35
  - homalgTable entry, 65
  - ResidueClassRing, 100
- ZeroMatrix
  - homalgTable entry, 56
  - ResidueClassRing, 93
- ZeroRows, 35
  - homalgTable entry, 65
  - ResidueClassRing, 100