The **homalg** project and its related packages

The homalg project authors

2007-2009

The idea: A homological algebra meta-package

Homological algebra is linear algebra over rings.

The idea: A homological algebra meta-package

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity

The idea: A homological algebra meta-package

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$

The idea: A homological algebra meta-package

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact

The idea: A homological algebra meta-package

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact: $\mathrm{Hom}_R(-, N)$

The idea: A homological algebra meta-package

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact: $\mathrm{Hom}_R(-,N)$, $\mathrm{Hom}_R(M,-)$

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact: $\mathrm{Hom}_R(-,N)$, $\mathrm{Hom}_R(M,-)$, $- \otimes_R N$

The idea: A homological algebra meta-package

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact: $\mathrm{Hom}_R(-,N)$, $\mathrm{Hom}_R(M,-)$, $- \otimes_R N$, $M \otimes_R -$

The idea: A homological algebra meta-package

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact: $\mathrm{Hom}_R(-,N)$, $\mathrm{Hom}_R(M,-)$, $-\otimes_R N$, $M \otimes_R -$
- An R-module M is not necessarily free

The idea: A homological algebra meta-package

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact: $\mathrm{Hom}_R(-,N)$, $\mathrm{Hom}_R(M,-)$, $-\otimes_R N$, $M \otimes_R -$
- An R-module M is not necessarily free, i.e. $\nexists\, R^r \xrightarrow{\pi} M$ injective.

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact: $\mathrm{Hom}_R(-, N)$, $\mathrm{Hom}_R(M, -)$, $- \otimes_R N$, $M \otimes_R -$
- An R-module M is not necessarily free, i.e. $\nexists\, R^r \overset{\pi}{\twoheadrightarrow} M$ injective. We obtain resolutions:

$$M \overset{\pi}{\twoheadleftarrow} F_0$$

The idea: A homological algebra meta-package

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact: $\mathrm{Hom}_R(-,N)$, $\mathrm{Hom}_R(M,-)$, $- \otimes_R N$, $M \otimes_R -$
- An R-module M is not necessarily free, i.e. $\nexists\, R^r \xrightarrow{\pi} M$ injective. We obtain resolutions:

$$M \overset{\pi}{\twoheadleftarrow} F_0 \leftarrow F_1$$

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact: $\mathrm{Hom}_R(-,N)$, $\mathrm{Hom}_R(M,-)$, $- \otimes_R N$, $M \otimes_R -$
- An R-module M is not necessarily free, i.e. $\nexists\, R^r \overset{\pi}{\twoheadrightarrow} M$ injective. We obtain resolutions:

$$M \overset{\pi}{\twoheadleftarrow} F_0 \leftarrow F_1 \leftarrow F_2$$

The idea: A homological algebra meta-package

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact: $\mathrm{Hom}_R(-,N)$, $\mathrm{Hom}_R(M,-)$, $- \otimes_R N$, $M \otimes_R -$
- An R-module M is not necessarily free, i.e. $\nexists\, R^r \overset{\pi}{\twoheadrightarrow} M$ injective. We obtain resolutions:

$$M \overset{\pi}{\twoheadleftarrow} F_0 \leftarrow F_1 \leftarrow F_2 \leftarrow \cdots$$

The idea: A homological algebra meta-package

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact: $\mathrm{Hom}_R(-,N)$, $\mathrm{Hom}_R(M,-)$, $- \otimes_R N$, $M \otimes_R -$
- An R-module M is not necessarily free, i.e. $\nexists\, R^r \xrightarrow{\pi} M$ injective. We obtain resolutions:

$$M \overset{\pi}{\twoheadleftarrow} F_0 \leftarrow F_1 \leftarrow F_2 \leftarrow \cdots$$

- New vocabulary: homology

The idea: A homological algebra meta-package

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact: $\mathrm{Hom}_R(-,N)$, $\mathrm{Hom}_R(M,-)$, $-\otimes_R N$, $M \otimes_R -$
- An R-module M is not necessarily free, i.e. $\nexists\, R^r \xrightarrow{\pi} M$ injective. We obtain resolutions:

$$M \xleftarrow{\pi} F_0 \leftarrow F_1 \leftarrow F_2 \leftarrow \cdots$$

- New vocabulary: homology, cohomology

The idea: A homological algebra meta-package

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact: $\mathrm{Hom}_R(-, N)$, $\mathrm{Hom}_R(M, -)$, $- \otimes_R N$, $M \otimes_R -$
- An R-module M is not necessarily free, i.e. $\nexists\, R^r \overset{\pi}{\twoheadrightarrow} M$ injective. We obtain resolutions:

$$M \overset{\pi}{\twoheadleftarrow} F_0 \leftarrow F_1 \leftarrow F_2 \leftarrow \cdots$$

- New vocabulary: homology, cohomology, derived functors

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact: $\mathrm{Hom}_R(-,N)$, $\mathrm{Hom}_R(M,-)$, $- \otimes_R N$, $M \otimes_R -$
- An R-module M is not necessarily free, i.e. $\nexists\, R^r \xrightarrow{\pi} M$ injective. We obtain resolutions:

$$M \xleftarrow{\pi} F_0 \leftarrow F_1 \leftarrow F_2 \leftarrow \cdots$$

- New vocabulary: homology, cohomology, derived functors, Ext

The idea: A homological algebra meta-package

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact: $\mathrm{Hom}_R(-,N)$, $\mathrm{Hom}_R(M,-)$, $- \otimes_R N$, $M \otimes_R -$
- An R-module M is not necessarily free, i.e. $\nexists\, R^r \overset{\pi}{\twoheadrightarrow} M$ injective. We obtain resolutions:

$$M \overset{\pi}{\twoheadleftarrow} F_0 \leftarrow F_1 \leftarrow F_2 \leftarrow \cdots$$

- New vocabulary: homology, cohomology, derived functors, Ext, Tor

Homological algebra is linear algebra over rings.

The effect of working over rings

- Natural maps, which are isomorphisms over fields, often lose injectivity and surjectivity: $M \xrightarrow{\varepsilon} M^{**}$
- Functors which are exact over fields, often become half exact: $\mathrm{Hom}_R(-,N)$, $\mathrm{Hom}_R(M,-)$, $- \otimes_R N$, $M \otimes_R -$
- An R-module M is not necessarily free, i.e. $\nexists\, R^r \overset{\pi}{\twoheadrightarrow} M$ injective. We obtain resolutions:

$$M \overset{\pi}{\twoheadleftarrow} F_0 \leftarrow F_1 \leftarrow F_2 \leftarrow \cdots$$

- New vocabulary: homology, cohomology, derived functors, Ext, Tor, ...

The idea: A homological algebra meta-package

Linear algebra over computable rings

To solve **linear inhomogeneous systems** $XA = B$ (resp. $AX = B$) with coefficients in R we need to

The idea: A homological algebra meta-package

Linear algebra over computable rings

To solve **linear inhomogeneous systems** $XA = B$ (resp. $AX = B$) with coefficients in R we need to

- Decide if the system is solvable

The idea: A homological algebra meta-package

Linear algebra over computable rings

To solve **linear inhomogeneous systems** $XA = B$ (resp. $AX = B$) with coefficients in R we need to

- Decide if the system is solvable, i.e. decide if B is zero modulo A.

DecideZero	dc0

Linear algebra over computable rings

To solve **linear inhomogeneous systems** $XA = B$ (resp. $AX = B$) with coefficients in R we need to

- Decide if the system is solvable, i.e. decide if B is zero modulo A.
- Compute a particular solution X

DecideZero dc0

Linear algebra over computable rings

To solve **linear inhomogeneous systems** $XA = B$ (resp. $AX = B$) with coefficients in R we need to

- Decide if the system is solvable, i.e. decide if B is zero modulo A.
- Compute a particular solution X, i.e. **effectively** decide if B is zero module A.

DecideZero	`dc0`
DecideZero**Effectively**	`DC0`

The idea: A homological algebra meta-package

Linear algebra over computable rings

To solve **linear inhomogeneous systems** $XA = B$ (resp. $AX = B$) with coefficients in R we need to

- Decide if the system is solvable, i.e. decide if B is zero modulo A.
- Compute a generating set of homogeneous solutions
- Compute a particular solution X, i.e. **effectively** decide if B is zero module A.

DecideZero	dc0
DecideZero**Effectively**	DC0

The idea: A homological algebra meta-package

Linear algebra over computable rings

To solve **linear inhomogeneous systems** $XA = B$ (resp. $AX = B$) with coefficients in R we need to

- Decide if the system is solvable, i.e. decide if B is zero modulo A.
- Compute a generating set of homogeneous solutions, i.e. compute a generating set of syzygies.
- Compute a particular solution X, i.e. **effectively** decide if B is zero module A.

DecideZero	`dc0`
SyzygiesGenerators	`syz`
DecideZero**Effectively**	`DC0`

The idea: A homological algebra meta-package

Structures `homalg` provides

The idea: A homological algebra meta-package

Structures homalg provides

Matrices with lazy evaluated operations

The idea: A homological algebra meta-package

Structures homalg provides

Matrices with lazy evaluated operations, modules (intrinsic)

The idea: A homological algebra meta-package

Structures `homalg` provides

Matrices with lazy evaluated operations, modules (intrinsic), maps

The idea: A homological algebra meta-package

Structures homalg provides

Matrices with lazy evaluated operations, modules (intrinsic), maps, filtrations

The idea: A homological algebra meta-package

Structures `homalg` provides

Matrices with lazy evaluated operations, modules (intrinsic), maps, filtrations, complexes (of modules and of complexes)

The idea: A homological algebra meta-package

Structures `homalg` provides

Matrices with lazy evaluated operations, modules (intrinsic), maps, filtrations, complexes (of modules and of complexes), chain maps

The idea: A homological algebra meta-package

Structures homalg provides

Matrices with lazy evaluated operations, modules (intrinsic), maps, filtrations, complexes (of modules and of complexes), chain maps, bicomplexes

The idea: A homological algebra meta-package

Structures `homalg` provides

Matrices with lazy evaluated operations, modules (intrinsic), maps, filtrations, complexes (of modules and of complexes), chain maps, bicomplexes, bigraded (differential) objects

The idea: A homological algebra meta-package

Structures homalg provides

Matrices with lazy evaluated operations, modules (intrinsic), maps, filtrations, complexes (of modules and of complexes), chain maps, bicomplexes, bigraded (differential) objects, spectral sequences

The idea: A homological algebra meta-package

Structures homalg provides

Matrices with lazy evaluated operations, modules (intrinsic), maps, filtrations, complexes (of modules and of complexes), chain maps, bicomplexes, bigraded (differential) objects, spectral sequences, functors.

The idea: A homological algebra meta-package

Based on `dc0`, `syz` and `DC0` `homalg` provides

The idea: A homological algebra meta-package

Based on `dc0`, `syz` and `DC0` homalg provides

- **applying** functors (like Ext, Tor, $\ldots$) to modules, maps, complexes and chain maps

The idea: A homological algebra meta-package

Based on `dc0`, `syz` and `DC0` `homalg` provides

- **applying** functors (like Ext, Tor, . . .) to modules, maps, complexes and chain maps
- **derivation** and **composition** of functors

The idea: A homological algebra meta-package

Based on `dc0`, `syz` and `DC0` `homalg` provides

- **applying** functors (like Ext, Tor, $\ldots$) to modules, maps, complexes and chain maps
- **derivation** and **composition** of functors
- connecting homomorphisms and **long exact sequences**

The idea: A homological algebra meta-package

Based on `dc0`, `syz` and `DC0` `homalg` provides

- **applying** functors (like Ext, Tor, $\ldots$) to modules, maps, complexes and chain maps
- **derivation** and **composition** of functors
- connecting homomorphisms and **long exact sequences**
- Cartan-Eilenberg resolution of complexes and **hyper (co)homology**

The idea: A homological algebra meta-package

Based on `dc0`, `syz` and `DC0` homalg provides

- **applying** functors (like Ext, Tor, $\ldots$) to modules, maps, complexes and chain maps
- **derivation** and **composition** of functors
- connecting homomorphisms and **long exact sequences**
- Cartan-Eilenberg resolution of complexes and **hyper (co)homology**
- spectral sequences of bicomplexes and the **Grothendieck spectral sequences** associated to two composable functors

The idea: A homological algebra meta-package

Based on `dc0`, `syz` and `DC0` `homalg` provides

- **applying** functors (like Ext, Tor, $\ldots$) to modules, maps, complexes and chain maps
- **derivation** and **composition** of functors
- connecting homomorphisms and **long exact sequences**
- Cartan-Eilenberg resolution of complexes and **hyper (co)homology**
- spectral sequences of bicomplexes and the **Grothendieck spectral sequences** associated to two composable functors
- test if a module is torsion-free, reflexive, projective, stably free, free, pure

The idea: A homological algebra meta-package

Based on dc0, syz and DC0 homalg provides

- **applying** functors (like Ext, Tor, ...) to modules, maps, complexes and chain maps
- **derivation** and **composition** of functors
- connecting homomorphisms and **long exact sequences**
- Cartan-Eilenberg resolution of complexes and **hyper (co)homology**
- spectral sequences of bicomplexes and the **Grothendieck spectral sequences** associated to two composable functors
- test if a module is torsion-free, reflexive, projective, stably free, free, pure and determine the rank, codimension, projective dimension, degree of torsion-freeness, and codegree of purity of a module

The idea: A homological algebra meta-package

`homalg`

Candidates: There are several systems that could host `homalg`

`homalg`

Maple MAGMA Macaulay2 GAP Sage Singular

Candidates: There are several systems that could host `homalg`, each supporting certain kinds of rings

`homalg`

Maple	MAGMA	Macaulay2	GAP	Sage	Singular
$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$
$\mathbb{Z}[x,\partial]$, $\ldots$	$\mathbb{Z}[x]$, $\mathbb{F}[x]$	$\mathbb{F}[x]$, $\mathbb{F}[x,\partial]$, $\ldots$	$\mathbb{Z}$	$\mathbb{Z}$, $\ldots$	$\mathbb{F}[x]$, $\mathbb{F}[x,\partial]$, $\ldots$

(GAP4)

homalg

Maple	MAGMA	Macaulay2	GAP	Sage	Singular
$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$
$\mathbb{Z}[x, \partial]$, ...	$\mathbb{Z}[x]$, $\mathbb{F}[x]$	$\mathbb{F}[x]$, $\mathbb{F}[x, \partial]$, ...	$\mathbb{Z}$	$\mathbb{Z}$, ...	$\mathbb{F}[x]$, $\mathbb{F}[x, \partial]$, ...

homalg: As a GAP4 package and *independent* of any ring arithmetic

$$\text{(GAP4)} \downarrow \textbf{homalg}$$

Maple	MAGMA	Macaulay2	GAP	Sage	Singular
$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$
$\mathbb{Z}[x, \partial]$, ...	$\mathbb{Z}[x]$, $\mathbb{F}[x]$	$\mathbb{F}[x]$, $\mathbb{F}[x, \partial]$, ...	$\mathbb{Z}$	$\mathbb{Z}$, ...	$\mathbb{F}[x]$, $\mathbb{F}[x, \partial]$, ...

homalg: GAP “sufficiently supports” the ring of integers

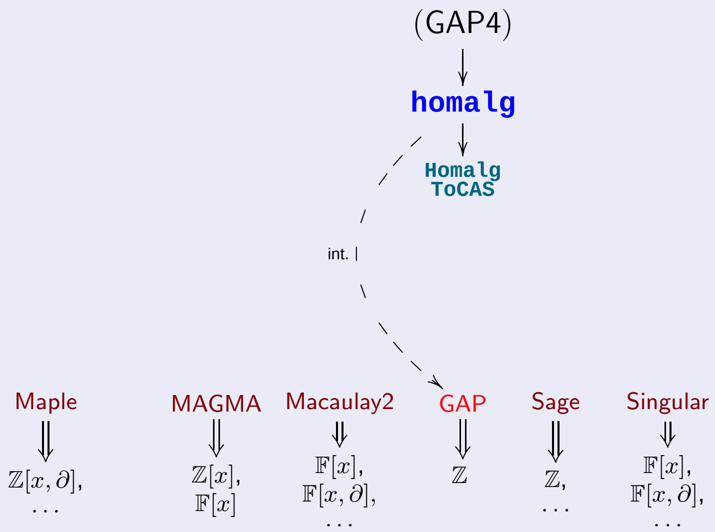
(GAP4)

$\downarrow$

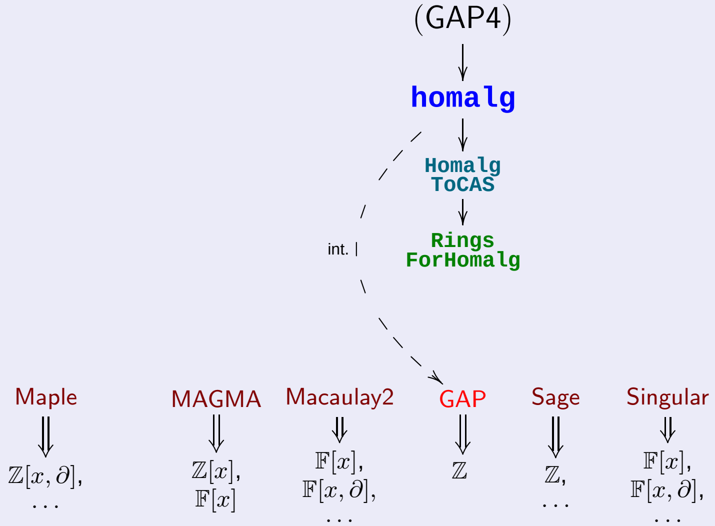
homalg

int.

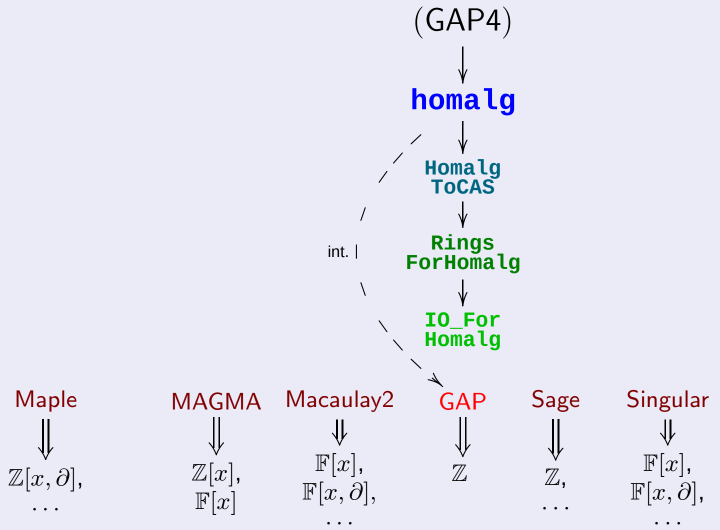
Maple	MAGMA	Macaulay2	GAP	Sage	Singular
$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$
$\mathbb{Z}[x,\partial]$, ...	$\mathbb{Z}[x]$, $\mathbb{F}[x]$	$\mathbb{F}[x]$, $\mathbb{F}[x,\partial]$, ...	$\mathbb{Z}$	$\mathbb{Z}$, ...	$\mathbb{F}[x]$, $\mathbb{F}[x,\partial]$, ...


(GAP4)
homalg
Homalg ToCAS
int.
Maple
$\mathbb{Z}[x, \partial]$, ...
MAGMA
$\mathbb{Z}[x]$, $\mathbb{F}[x]$
Macaulay2
$\mathbb{F}[x]$, $\mathbb{F}[x, \partial]$, ...
GAP
$\mathbb{Z}$
Sage
$\mathbb{Z}$, ...
Singular
$\mathbb{F}[x]$, $\mathbb{F}[x, \partial]$, ...

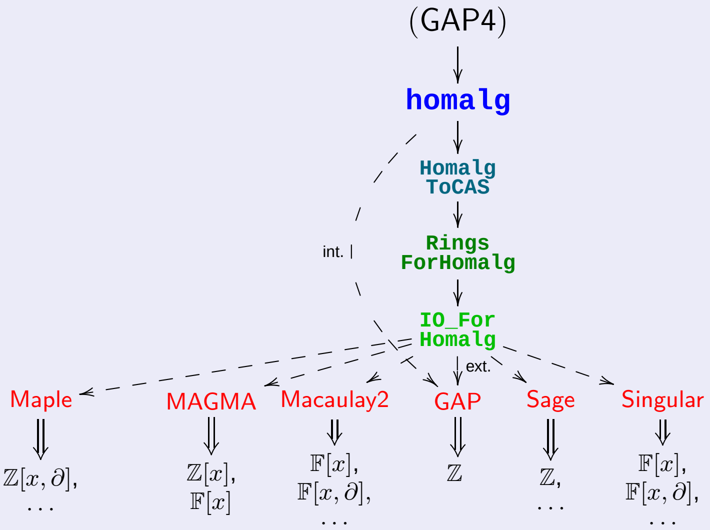
RingsForHomalg: The dictionaries homalg uses.



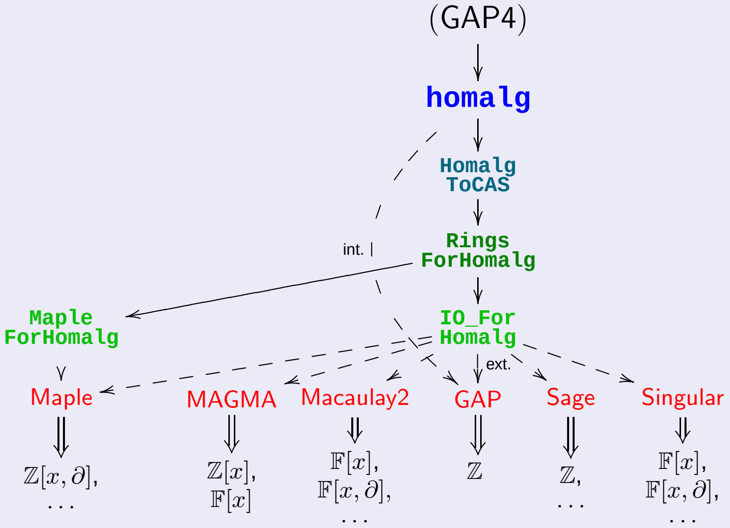
IO_ForHomalg: Communicate via streams



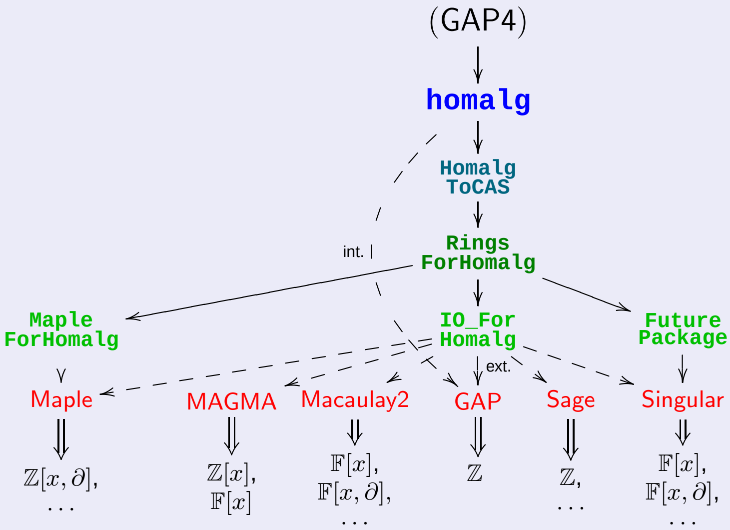
IO_ForHomalg: Communicate via streams with various CA systems through their command line interface



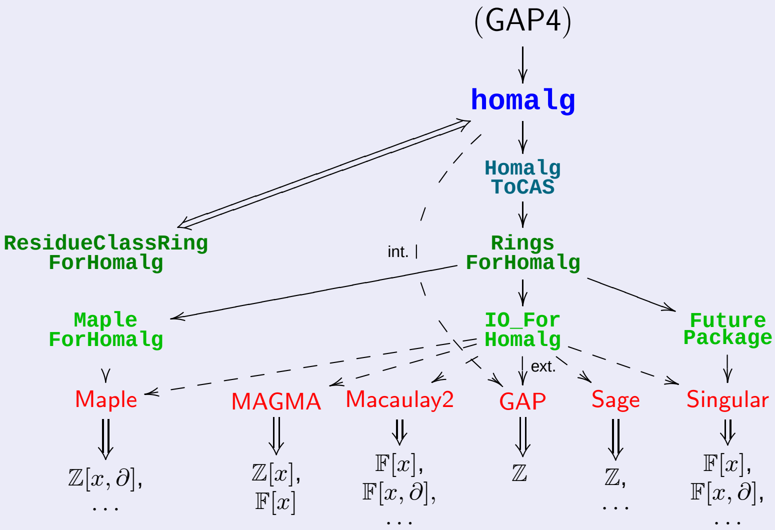
MapleForHomalg: Communicate with Maple's interpreter, shortcutting its command line interface.



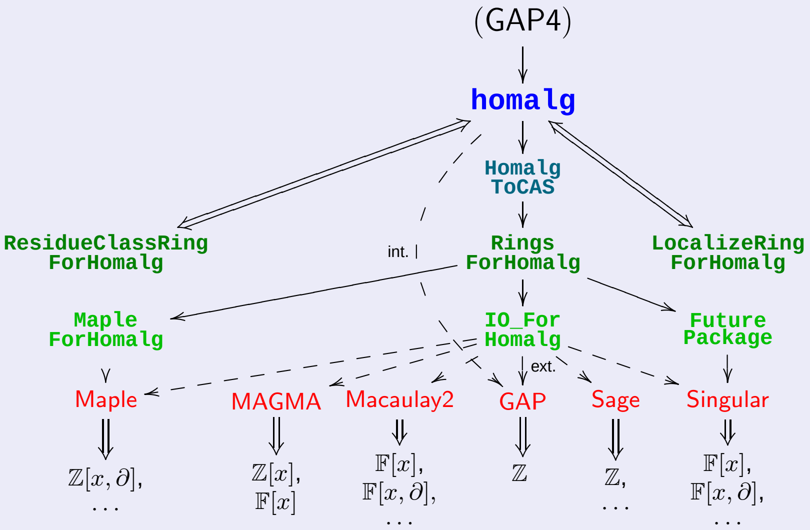
Future: Communicate with interpreters of various CASs shortcutting their command line interface.



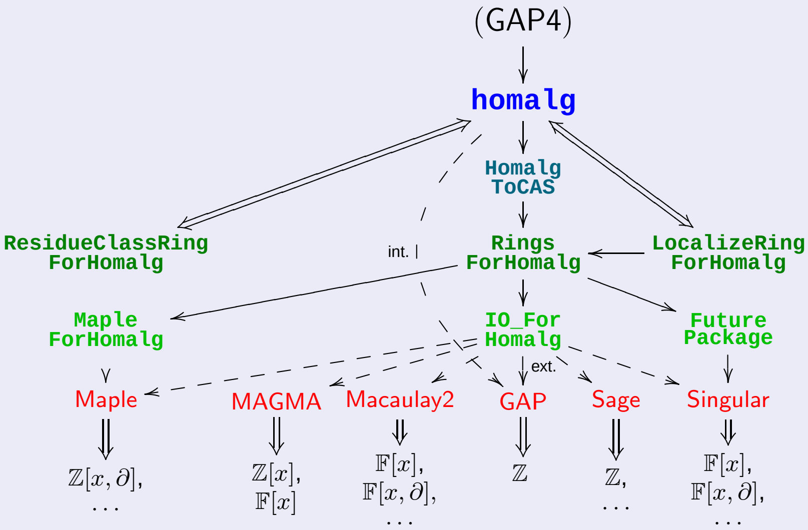
ResidueClassRingForHomalg: Support for residue class rings



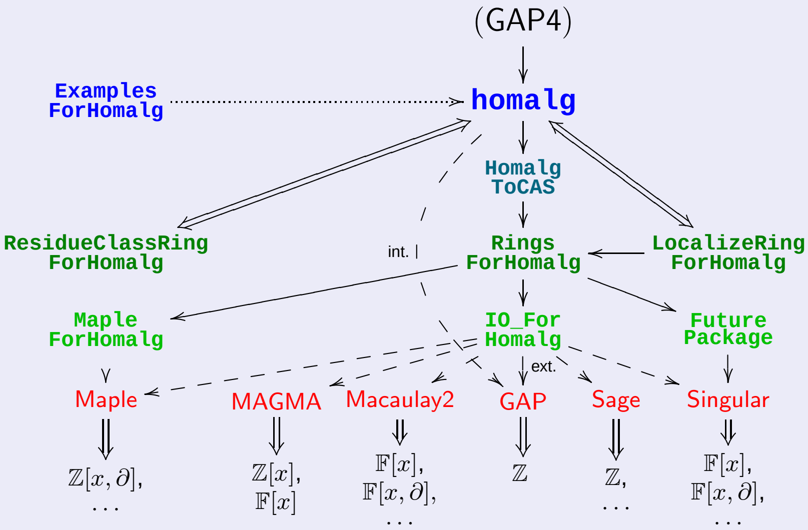
LocalizeRingForHomalg: Localizations of commutative rings in homalg at maximal ideals.

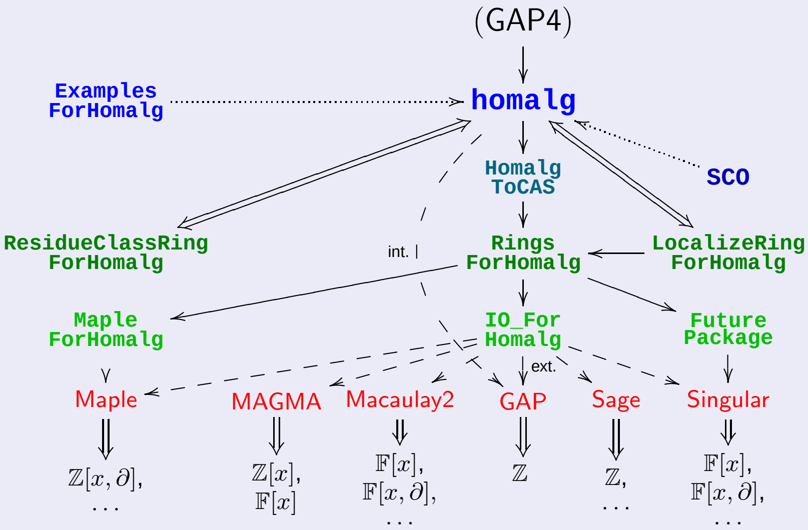


LocalizeRingForHomalg: Use MORA's algorithm in Singular to localize polynomial rings at max. ideals.

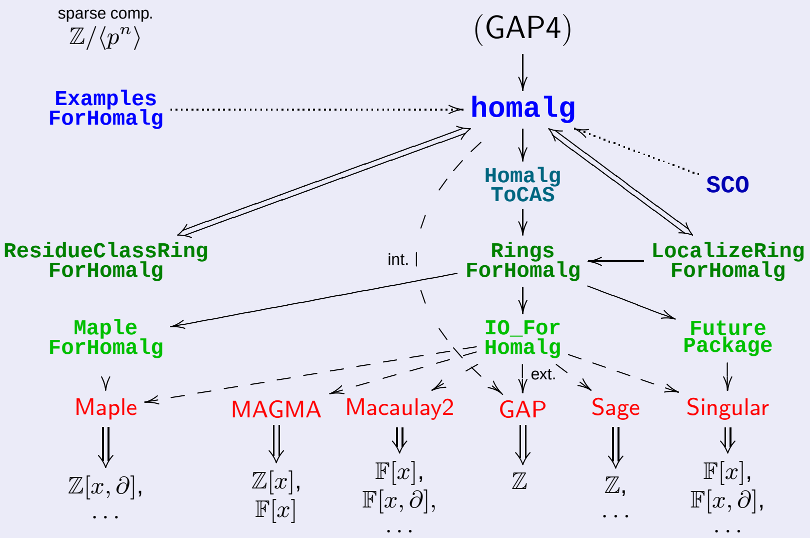


ExamplesForHomalg: Computing-engine-independent example using homalg.

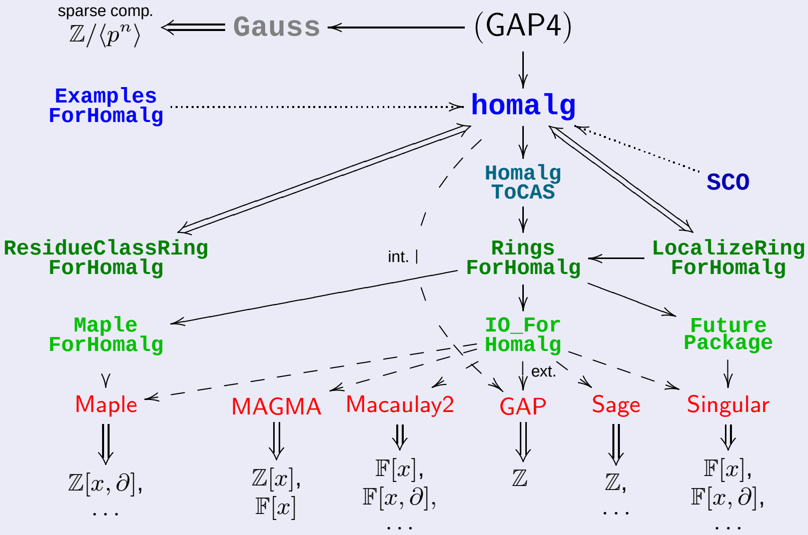



(GAP4)
homalg
ExamplesForHomalg
HomalgToCAS
SCO
ResidueClassRingForHomalg
int.
RingsForHomalg
LocalizeRingForHomalg
MapleForHomalg
IO_ForHomalg
FuturePackage
ext.
Maple
MAGMA
Macaulay2
GAP
Sage
Singular
$\mathbb{Z}[x, \partial]$, ...
$\mathbb{Z}[x]$, $\mathbb{F}[x]$
$\mathbb{F}[x]$, $\mathbb{F}[x, \partial]$, ...
$\mathbb{Z}$
$\mathbb{Z}$, ...
$\mathbb{F}[x]$, $\mathbb{F}[x, \partial]$, ...

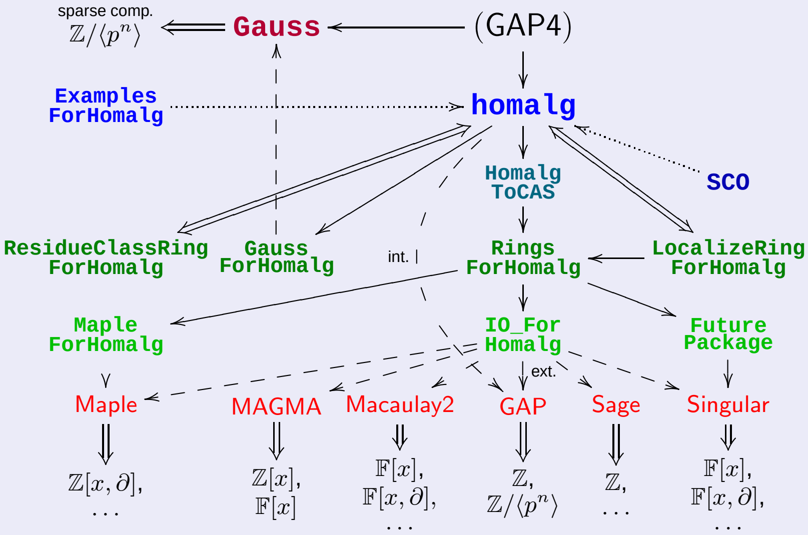
A neccessity not only for SCO: Sparse computations over p-adic numbers

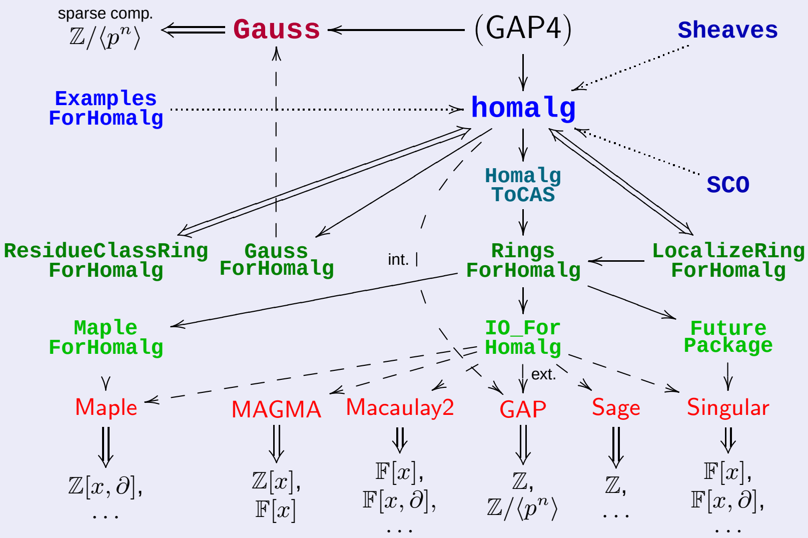


Gauss: Extended GAUSS Functionality for GAP



GaussForHomalg: Linking Gauss and homalg




sparse comp.
$\mathbb{Z}/\langle p^n \rangle$
Gauss
(GAP4)
Sheaves
Examples
ForHomalg
homalg
Homalg
ToCAS
SCO
ResidueClassRing
ForHomalg
Gauss
ForHomalg
int.
Rings
ForHomalg
LocalizeRing
ForHomalg
Maple
ForHomalg
IO_For
Homalg
Future
Package
ext.
Maple
MAGMA
Macaulay2
GAP
Sage
Singular
$\mathbb{Z}[x, \partial]$, ...
$\mathbb{Z}[x]$, $\mathbb{F}[x]$
$\mathbb{F}[x]$, $\mathbb{F}[x, \partial]$, ...
$\mathbb{Z}$, $\mathbb{Z}/\langle p^n \rangle$
$\mathbb{Z}$, ...
$\mathbb{F}[x]$, $\mathbb{F}[x, \partial]$, ...

A simple example

A simple example over $\mathbb{Z}$

```
> LoadPackage( "homalg" );
```

A simple example

A simple example over $\mathbb{Z}$

```
> LoadPackage( "homalg" );
> ZZ := HomalgRingOfIntegers( );;
```

A simple example

A simple example over $\mathbb{Z}$

```
> LoadPackage( "homalg" );
> ZZ := HomalgRingOfIntegers( );;
> mat := HomalgMatrix( "[ 4, 6, 10 ]", 1, 3, ZZ );;
```

A simple example

A simple example over $\mathbb{Z}$

```
> LoadPackage( "homalg" );
> ZZ := HomalgRingOfIntegers( );;
> mat := HomalgMatrix( "[ 4, 6, 10 ]", 1, 3, ZZ );;
> M := LeftPresentation( mat );
<A non-torsion left module presented by 1 relation for 3 generators>
```

A simple example

A simple example over $\mathbb{Z}$

```
> LoadPackage( "homalg" );
> ZZ := HomalgRingOfIntegers( );;
> mat := HomalgMatrix( "[ 4, 6, 10 ]", 1, 3, ZZ );;
> M := LeftPresentation( mat );
<A non-torsion left module presented by 1 relation for 3 generators>
> Display( M );
[ [ 4, 6, 10 ] ]
Cokernel of the map
Z^(1x1) --> Z^(1x3),
currently represented by the above matrix
```

A simple example

A simple example over $\mathbb{Z}$ (continued)

```
> IsTorsionFree( M );
  false
```

A simple example

A simple example over $\mathbb{Z}$ (continued)

```
> IsTorsionFree( M );
false
> M;
<A non-pure left module presented by 1 relation for 3 generators>
```

A simple example

A simple example over $\mathbb{Z}$ (continued)

```
> IsTorsionFree( M );
false
> M;
<A non-pure left module presented by 1 relation for 3 generators>
> ByASmallerPresentation( M );
<A non-pure left module presented by 1 relation for 3 generators>
```

A simple example

A simple example over $\mathbb{Z}$ (continued)

```
> IsTorsionFree( M );
  false
> M;
  <A non-pure left module presented by 1 relation for 3 generators>
> ByASmallerPresentation( M );
  <A non-pure left module presented by 1 relation for 3 generators>
> Display( M );
  Z/< 2 > + Z^(1 x 2)
```

Mathematical object-oriented programming

Mathematical way of thinking

1. TrueMethods

Mathematical way of thinking

1. TrueMethods
2. ImmediateMethods

Mathematical object-oriented programming

Mathematical way of thinking

1. TrueMethods
2. ImmediateMethods
3. Methods for operations

TrueMethods for properties

```
> InstallTrueMethod( IsTorsionFree, IsModule and IsReflexive );
```

Mathematical object-oriented programming

TrueMethods for properties

```
> InstallTrueMethod( IsTorsionFree, IsModule and IsReflexive );
> InstallTrueMethod( IsReflexive, IsModule and IsProjective );
```

TrueMethods for properties

```
> InstallTrueMethod( IsTorsionFree, IsModule and IsReflexive );
> InstallTrueMethod( IsReflexive, IsModule and IsProjective );
> InstallTrueMethod( IsProjective, IsModule and IsFree );
```

Mathematical object-oriented programming

TrueMethods for properties

```
> InstallTrueMethod( IsTorsionFree, IsModule and IsReflexive );
> InstallTrueMethod( IsReflexive, IsModule and IsProjective );
> InstallTrueMethod( IsProjective, IsModule and IsFree );
> ...
```

ImmediateMethods for properties and attributes

```
> InstallImmediateMethod( IsFree,
  IsModule and HasIsProjective, 0,
  function( M ) if not IsProjective( M ) then return false; fi;
  TryNextMethod( ); end );
```

ImmediateMethods for properties and attributes

- InstallImmediateMethod(IsFree, IsModule and HasIsProjective, 0, function(M) if not IsProjective(M) then return false; fi; TryNextMethod( ); end);
- InstallImmediateMethod(IsProjective, IsModule and HasProjectiveDimension, 0, M -> 0 = ProjectiveDimension(M));

ImmediateMethods for properties and attributes

```
> InstallImmediateMethod( IsFree,
  IsModule and HasIsProjective, 0,
  function( M ) if not IsProjective( M ) then return false; fi;
  TryNextMethod( ); end );
> InstallImmediateMethod( IsProjective,
  IsModule and HasProjectiveDimension, 0,
  M -> 0 = ProjectiveDimension( M ) );
> ...
```

The TrueMethods and ImmediateMethods enable GAP to draw *arbitrary long* chains of conclusions.

Mathematical object-oriented programming

The TrueMethods and ImmediateMethods enable GAP to draw *arbitrary long* chains of conclusions.

`homalg` rediscovered a proof of Serre

Mathematical object-oriented programming

The TrueMethods and ImmediateMethods enable GAP to draw *arbitrary long* chains of conclusions.

homalg rediscovered a proof of Serre

Let k be a field and M a module over $R := k[x, y]$. Then $\mathrm{Hom}(M, R)$ is free.

The TrueMethods and ImmediateMethods enable GAP to draw *arbitrary long* chains of conclusions.

homalg rediscovered a proof of SERRE

Let k be a field and M a module over $R := k[x,y]$. Then $\mathrm{Hom}(M,R)$ is free.

> LoadPackage("RingsForHomalg");

Mathematical object-oriented programming

The TrueMethods and ImmediateMethods enable GAP to draw *arbitrary long* chains of conclusions.

homalg rediscovered a proof of SERRE

Let k be a field and M a module over $R := k[x, y]$. Then $\mathrm{Hom}(M, R)$ is free.

```
> LoadPackage( "RingsForHomalg" );
> R := HomalgFieldOfRationalsInDefaultCAS( ) * "x,y";;
```

Mathematical object-oriented programming

The TrueMethods and ImmediateMethods enable GAP to draw *arbitrary long* chains of conclusions.

homalg rediscovered a proof of SERRE

Let k be a field and M a module over $R := k[x,y]$. Then $\mathrm{Hom}(M,R)$ is free.

```
> LoadPackage( "RingsForHomalg" );
> R := HomalgFieldOfRationalsInDefaultCAS( ) * "x,y";;
> M := HomalgMatrix( "[x^3,y^3,x+y,x-y, y,x,x,y]", 2, 4, R );;
```

Mathematical object-oriented programming

The TrueMethods and ImmediateMethods enable GAP to draw *arbitrary long* chains of conclusions.

homalg rediscovered a proof of SERRE

Let k be a field and M a module over $R := k[x, y]$. Then $\mathrm{Hom}(M, R)$ is free.

```
> LoadPackage( "RingsForHomalg" );
> R := HomalgFieldOfRationalsInDefaultCAS( ) * "x,y";;
> M := HomalgMatrix( "[x^3,y^3,x+y,x-y, y,x,x,y]", 2, 4, R );;
> M := LeftPresentation( M );
<A non-torsion left module presented by 2 relations for 4 generators>
```

Mathematical object-oriented programming

The TrueMethods and ImmediateMethods enable GAP to draw *arbitrary long* chains of conclusions.

homalg rediscovered a proof of SERRE

Let k be a field and M a module over $R := k[x, y]$. Then $\mathrm{Hom}(M, R)$ is free.

```
> LoadPackage( "RingsForHomalg" );
> R := HomalgFieldOfRationalsInDefaultCAS( ) * "x,y";;
> M := HomalgMatrix( "[x^3,y^3,x+y,x-y, y,x,x,y]", 2, 4, R );;
> M := LeftPresentation( M );
<A non-torsion left module presented by 2 relations for 4 generators>
> Hom( M, R );
<A free right module of rank 2 on 3 non-free generators satisfying a single relation>
```

RHom in `homalg`

```
> RightDerivedCofunctor( Functor_Hom );
```

RHom in homalg

> RightDerivedCofunctor(Functor_Hom);

This *single* call

1. creates an object named `Functor_RHom` for the bivariate functor RHom.

RHom in homalg

> RightDerivedCofunctor(Functor_Hom);

This *single* call

1. creates an object named `Functor_RHom` for the bivariate functor RHom.
2. installs methods for

RHom in homalg

> RightDerivedCofunctor(Functor_Hom);

This *single* call

1. creates an object named `Functor_RHom` for the bivariate functor RHom.
2. installs methods for
   - modules

RHom in homalg

> > RightDerivedCofunctor(Functor_Hom);

This *single* call

1. creates an object named `Functor_RHom` for the bivariate functor RHom.
2. installs methods for
   - modules
   - maps

RHom in `homalg`

> RightDerivedCofunctor(Functor_Hom);

This *single* call

1. creates an object named `Functor_RHom` for the bivariate functor RHom.
2. installs methods for
   - modules
   - maps
   - complexes of modules

Mathematical object-oriented programming

RHom in homalg

> RightDerivedCofunctor(Functor_Hom);

This *single* call

1. creates an object named `Functor_RHom` for the bivariate functor RHom.
2. installs methods for
   - modules
   - maps
   - complexes of modules
   - chain maps

Mathematical object-oriented programming

RHom in `homalg`

> RightDerivedCofunctor(Functor_Hom);

This *single* call

1. creates an object named `Functor_RHom` for the bivariate functor RHom.
2. installs methods for
   - modules
   - maps
   - complexes of modules and of complexes
   - chain maps

RHom in homalg

> \> RightDerivedCofunctor(Functor_Hom);

This *single* call

1. creates an object named `Functor_RHom` for the bivariate functor RHom.
2. installs methods for
   - modules
   - maps
   - complexes of modules and of complexes
   - chain maps
   - bicomplexes

Mathematical object-oriented programming

RHom in `homalg`

> RightDerivedCofunctor(Functor_Hom);

This *single* call

1. creates an object named `Functor_RHom` for the bivariate functor RHom.
2. installs methods for
   - modules
   - maps
   - complexes of modules and of complexes
   - chain maps
   - bicomplexes
   - short exact sequence of modules

Mathematical object-oriented programming

In the `homalg` project we use the power of the GAP language to program in a *conceptual* way.

In the `homalg` project we use the power of the GAP language to program in a *conceptual* way.

Purity filtration in `homalg`

```
> InstallMethod( PurityFiltration,
  [ IsFinitelyPresentedModuleRep ],
  function( M )
  local R, F, G, II_E;
  R := HomalgRing( M );
  F := DualizingFunctor( R ); G := DualizingFunctor( R );
  II_E := GrothendieckSpectralSequence( F, G, M );
  return FiltrationBySpectralSequence( II_E );
  end );
```