homalg

An abstract MAPLE-package for homological algebra

Mohamed Barakat and Daniel Robertz

Lehrstuhl B für Mathematik
RWTH-Aachen University
Germany

September 21, 2006
Introduction

homalg

The Ring Packages

Todo
Q: what does “abstract” MAPLE-package mean?
A: homalg depends on a “ring package” that implements the ring-specific arithmetics
the package **homalg**

- Q: what does “abstract” MAPLE-package mean?
  - A: **homalg** depends on a “ring package” that implements the ring-specific arithmetics

- available ring packages: PIR, Involutive, Janet, JanetOre, OreModules...
Q: what does “abstract” MAPLE-package mean?  
A: homalg depends on a “ring package” that implements the ring-specific arithmetics

available ring packages: PIR, Involutive, Janet, JanetOre, OreModules...

homological algebra in MAPLE:  
the full subcategory of *finitely presented modules* of the abelian category of modules over a ring $R$, in which one can *algorithmically* solve the *ideal membership problem* (e.g. *reduce* via a *basis*) and compute *syzygies*
Q: what does “abstract” MAPLE-package mean?
A: homalg depends on a “ring package” that implements the ring-specific arithmetics

available ring packages: PIR, Involutive, Janet, JanetOre, OreModules...

homological algebra in MAPLE:
the full subcategory of finitely presented modules of the abelian category of modules over a ring \( R \), in which one can algorithmically solve the ideal membership problem (e.g. reduce via a basis) and compute syzygies

modules are given by a finite presentation:
\( \text{coker}(R^{l_1} \xrightarrow{A} R^{l_0}), \ A \in R^{l_1 \times l_0} \)
presentation of a module

\[
[[1, 0, 0] = \begin{bmatrix} 0 & y & 0 \\ 0 & -y & 0 \end{bmatrix}, [0, 1, 0] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, [0, 0, 1] = \begin{bmatrix} 0 & 0 & -y \\ 0 & 0 & x \end{bmatrix}],
\]

\[
[[x - y, 0, 0], [y, xy, 0], [0, 0, z^3]],
\]

“Presentation”,

3 + 8s + 14s^2 + s^3 \left( \frac{14}{1-s} + \frac{6}{(1-s)^2} \right),

\[
[[14, 6, 0]]
\]

HILBERT-series

CARTAN-characters
basic procedures in *homalg*

the ring package provides: *BasisOfModule* and *Reduce*

- RightDivide=PreImage, ReduceHomomorphism, Presentation
- Intersection, CompleteImSq
- ConnectingHom, Cokernel, TensorProduct
- OnMorphisms
basic procedures in homalg

the ring package provides: **BasisOfModule** and **Reduce**

- RightDivide=PreImage, ReduceHomomorphism, Presentation
- Intersection, CompleteImSq
- ConnectingHom, Cokernel, TensorProduct
- OnMorphisms

the ring package also provides: **SyzygiesGenerators**

- SubfactorModule, ResolutionOfModule, SyzygiesOp
- Kernel, DefectOfHoms, ParametrizeModule
- TorsionSubmodule, Pullback, Hom_R, Hom
- HomologyModules, CohomologyModules
- IsExactSeq, IsExactCoseq
- LongExactHomolSeq, LongExactCohomolSeq
advanced procedures in homalg

building upon the above

- ResolveShortExactSeq
- LeftDerivedFunctor, LeftDerivedRightExactFunctor, RightDerivedFunctor, RightDerivedLeftExactCofunctor
- Tor, Ext_R, Ext
- ...

Mohamed Barakat and Daniel Robertz

homalg: An abstract package for homological algebra
the **basic functors** in **homalg**

using **FunctorMap** we have the following **functors**:

- Cokernel, CokernelMap
- Kernel, KernelMap
- DefectOfHoms, DefectOfHomsMap
- DirectSum, DirectSumMap
- Pullback, PullbackMap
- TorsionSubmodule, TorsionSubmoduleMap
- TensorProduct, TensorProductMap, TensorProduct2Map
- Hom_R, HomMap_R
- Hom, HomMap, Hom2Map
the use of `FunctorMap` in `homalg`

`homalg/HomMap` := proc(M,A,N,L,var::list,RingPackage) ### Main: "Functors"
llocal RP, nar, optional, cofunctor;

RP := `homalg/tablename`(procname,args[-1],nar);

if type(RP,procedure) then RETURN(RP(args[1..-1-nar])) fi;

optional := args[6..-1-nar];

#=====# begin of the core procedure #=====#

cofunctor := proc(M) `homalg/Hom`(M,L,args[1+1..-1]) end;

`homalg/FunctorMap`(cofunctor,
  [M], A, [N]
  ,var,optional,RP)
end:
the use of **FunctorMap** in *homalg*

\[
{\text{'homalg}/Hom2Map'} := \text{proc}(L,M,A,N,\text{var}::\text{list},\text{RingPackage}) \quad \text{### Main: "FunctorMap"} \\
\text{local } \text{RP}, \text{nar}, \text{optional}, \text{functor}; \\
\]

\[
\text{RP} := {\text{'homalg}/tablename'}}(\text{procname},\text{args}[{-1}],\text{nar}); \\
\]

\[
\text{if type}(\text{RP},\text{procedure}) \text{ then RETURN}(\text{RP}(\text{args}[1..{-1}-\text{nar}])) \text{ fi}; \\
\]

\[
\text{optional} := \text{args}[6..{-1}-\text{nar}]; \\
\]

\[
\text{begin of the core procedure} \text{ begin of the core procedure} \\
\text{functor} := \text{proc}(M) {\text{'homalg}/Hom'}}(L,M,\text{args}[1+1..{-1}]) \text{ end;} \\
\]

\[
{\text{'homalg}/FunctorMap'}}(\text{functor}, \\
\quad \text{[M], A, [N] } \\
\quad ,\text{var,optional,RP}) \\
\text{end:}
\]
The bi-functor \texttt{Functor\_Hom} in \texttt{homalg}

\begin{verbatim}
`homalg/Functor_Hom` := proc(B) ### Main: "Functors"
   [ proc(M) `homalg/Hom`(M,B,args[1+1..-1]) end,
     proc(M,phi,N) `homalg/HomMap`(M,phi,N,B,args[3+1..-1]) end,
     ## take care of multi-functoriality:
     B,
     proc(P)
       proc(M,phi,N) `homalg/Hom2Map`(P,M,phi,N,args[3+1..-1]) end
     end ]
end:
\end{verbatim}
The bi-functor `Functor_Hom2` in homalg

```
'homalg/Functor_Hom2' := proc(A) ### Main: "Functors"

  [ proc(M) 'homalg/Hom'(A,M,args[1+1..-1]) end,
    proc(M,phi,N) 'homalg/Hom2Map'(A,M,phi,N,args[3+1..-1]) end,
    ## take care of multi-functoriality:
    A,
    proc(P)
      proc(M,phi,N) 'homalg/HomMap'(M,phi,N,P,args[3+1..-1]) end
    end
  ]
end:
```

Mohamed Barakat and Daniel Robertz

homalg: An abstract package for homological algebra
the use of \texttt{CofunctorOnSeqs} in \texttt{homalg}

\begin{verbatim}
`homalg/HomOnSeqs` := proc(L, alpha::{table, list}, _var, RingPackage) ###
    local RP, nar, cofunctor;

    RP := `homalg/tablename`(procname,args[-1],nar);

    if type(RP, procedure) then RETURN(RP(args[1..-1-nar])) fi;

    #=====# begin of the core procedure #=====#

    cofunctor := `homalg/Functor_Hom`(L);

    `homalg/CofunctorOnSeqs`(cofunctor, args[1+1..-1])
end:
\end{verbatim}
the use of \texttt{FunctorOnSeqs} in \texttt{homalg}

\begin{verbatim}
'homalg/Hom2OnSeqs' := proc(L, alpha::{table,list}, _var, RingPackage) 
    local RP, nar, functor;
    RP := 'homalg/tablename'(procname,args[-1],nar);
    if type(RP,procedure) then RETURN(RP(args[1..-1-nar])) fi;
    functor := 'homalg/Functor_Hom2'(L);
    'homalg/FunctorOnSeqs'(functor, args[1+1..-1])
end:
\end{verbatim}
composed functors in \texttt{homalg}

using \texttt{ComposeFunctors} we have the following (multi)-functors:

- $\text{HomHom}_R$, $\text{HomHomMap}_R$
- $\text{HomHom}$, $\text{HomHomMap}$, $\text{HomHom2Map}$, $\text{HomHom3Map}$
the use of **ComposeFunctors** in **homalg**

```plaintext
`homalg/HomHom` := proc(M,A,B,var::list,RingPackage) ### Main: "Functors"
    local RP, nar, optional, cofunctor1, cofunctor2;

    RP := `homalg/tablename`(procname,args[-1],nar);
    if type(RP,procedure) then RETURN(RP(args[1..-1-nar])) fi;

    optional := args[5..-1-nar];

    #=====# begin of the core procedure #=====#
    cofunctor1 := `homalg/Functor_Hom`(B);
    cofunctor2 := `homalg/Functor_Hom`(A);
    eval(
        `homalg/ComposeFunctors`(cofunctor1,cofunctor2,var,optional,RP)(M,var,RP)
    )
end:
```

Mohamed Barakat and Daniel Robertz  **homalg**: An abstract package for homological algebra
'homalg/HomHomMap' := proc(M,phi,N,A,B,var::list,RingPackage) ### Main
local RP, nar, optional, functor;

RP := 'homalg/tablename'(procname,args[-1],nar);
if type(RP,procedure) then RETURN(RP(args[1..-1-nar])) fi;
optional := args[7..-1-nar];

#=====# begin of the core procedure #=====#
functor := proc(M) 'homalg/HomHom'(M,A,B,args[1+1..-1]) end;

'homalg/FunctorMap'(functor,
  [M], phi, [N]
  ,var,optional,RP)
end:
the tri-functor \texttt{Functor\_HomHom} in \texttt{homalg}

```plaintext
# The functor Hom(Hom(-,A),B)
`homalg/Functor_HomHom` := proc(A,B) ### Main: "Functors"
   proc(M) `homalg/HomHom`(M,A,B,args[1+1..-1]) end,
   proc(M,phi,N) `homalg/HomHomMap`(M,phi,N,A,B,args[3+1..-1]) end,
   ## take care of multi-functoriality:
   A,
   proc(P)
      proc(M,phi,N) `homalg/HomHom2Map`(P,M,phi,N,B,args[3+1..-1]) end
   end,
   B,
   proc(P)
      proc(M,phi,N) `homalg/HomHom3Map`(P,A,M,phi,N,args[3+1..-1]) end
   end
end:
```

Mohamed Barakat and Daniel Robertz

\texttt{homalg}: An abstract package for homological algebra
right derived cofunctors in homalg

using `RightDerivedLeftExactCofunctor` we have the following *multi-functor*:

- `Ext`, `ExtMap`, `Ext2Map`
right derived cofunctors in \texttt{homalg}

Using \texttt{RightDerivedLeftExactCofunctor} we have the following \textit{multi-functor}:

- $\text{Ext, ExtMap, Ext2Map}$

Using \texttt{RightDerivedCofunctor} we have the following \textit{multi-functors}:

- $\text{Ext}_R, \text{ExtMap}_R$
- $\text{Extq, ExtqMap, Extq2Map}$
- $\text{RHomHom, RHomHomMap, RHomHom2Map, RHomHom3Map}$
definition of the $\text{Ext}^i(-, D)$-modules

take a free resolution of $M$:

$$
\ldots \xrightarrow{d_3} D^r \xrightarrow{d_2} D^q \xrightarrow{d_1} D^p \twoheadrightarrow M \twoheadrightarrow 0
$$
definition of the $\text{Ext}^i(\mathord{-}, D)$-modules

take a free resolution of $M$:

$$\ldots \xrightarrow{d_3} D^r \xrightarrow{d_2} D^q \xrightarrow{d_1} D^p \rightarrow M \rightarrow 0$$

apply the functor $\text{Hom}_D(\mathord{-}, D)$:

$$\ldots \xleftarrow{d_3^*} \text{Hom}_D(D^r, D) \xleftarrow{d_2^*} \text{Hom}_D(D^q, D) \xleftarrow{d_1^*} \text{Hom}_D(D^p, D) \xleftarrow{0}$$
definition of the \( \text{Ext}^i(-, D) \)-modules

take a free resolution of \( M \):
\[
\ldots \xrightarrow{d_3} D^r \xrightarrow{d_2} D^q \xrightarrow{d_1} D^p \longrightarrow M \longrightarrow 0
\]
apply the functor \( \text{Hom}_D(-, D) \):
\[
\ldots \xleftarrow{d_3^*} \text{Hom}_D(D^r, D) \xleftarrow{d_2^*} \text{Hom}_D(D^q, D) \xleftarrow{d_1^*} \text{Hom}_D(D^p, D) \xleftarrow{0}
\]
define
\[
\text{Ext}^0_D(M, D) := \ker(d_1^*) = \text{Hom}_D(M, D),
\]
\[
\text{Ext}^i_D(M, D) := \ker(d_{i+1}^*) / \text{im}(d_i^*), \quad i \geq 1.
\]
definition of the $\text{Ext}^i(−, D)$-modules

take a free resolution of $M$:

$$\cdots \xrightarrow{d_3} D^r \xrightarrow{d_2} D^q \xrightarrow{d_1} D^p \longrightarrow M \longrightarrow 0$$

apply the functor $\text{Hom}_D(−, D)$:

$$\cdots \leftarrow ^{d_3^*} \text{Hom}_D(D^r, D) \leftarrow ^{d_2^*} \text{Hom}_D(D^q, D) \leftarrow ^{d_1^*} \text{Hom}_D(D^p, D) \leftarrow 0$$

define

$$\text{Ext}_D^0(M, D) := \ker(d_1^*) = \text{Hom}_D(M, D),$$

$$\text{Ext}_D^i(M, D) := \ker(d_{i+1}^*) / \text{im}(d_i^*), \quad i \geq 1.$$  

**Theorem**

$\text{Ext}_D^i(M, D)$ is independent of the choice of the projective resolution of $M$. 

Mohamed Barakat and Daniel Robertz

homalg: An abstract package for homological algebra
`homalg/Ext` := proc(q::nonnegint,A,B,var::list,RingPackage) ### Main:
local RP, nar, optional, cofunctor;

RP := `homalg/tablename`(procname,args[-1],nar);

if type(RP,procedure) then RETURN(RP(args[1..-1-nar])) fi;

optional := args[5..-1-nar];

### begin of the core procedure ###

cofunctor := `homalg/Functor_Hom`(B);

`homalg/RightDerivedLeftExactCofunctor`(
    q,cofunctor,A
    ,var,optional,RP)
end:
the use of RightDerivedCofunctor in homalg

`homalg/Extq` := proc(q::nonnegint,A,B,var::list,RingPackage) ### Main:
    local RP, nar, optional, cofunctor;

    RP := `homalg/tablename`('homalg/procname',args[-1],nar);

    if type(RP,procedure) then RETURN(RP(args[1..-1-nar])) fi;

    optional := args[5..-1-nar];

    #=====# begin of the core procedure #=====#

    cofunctor := `homalg/Functor_Hom`(B);

    `homalg/RightDerivedCofunctor`('homalg/Extq'(
        q,cofunctor,A
        ,var,optional,RP)
    end:
linear control theory

characterizing system/module properties

<table>
<thead>
<tr>
<th>system</th>
<th>module</th>
<th>homological algebra</th>
</tr>
</thead>
<tbody>
<tr>
<td>autonomous elements</td>
<td>$t(M) \neq 0$</td>
<td>$\text{Ext}^1_D(M^\top , D) \neq 0$</td>
</tr>
<tr>
<td>controllability, parametrizability</td>
<td>$t(M) = 0$</td>
<td>$\text{Ext}^1_D(M^\top , D) = 0$</td>
</tr>
<tr>
<td>parametrizability of the parametrization</td>
<td>reflexive</td>
<td>$\text{Ext}^i_D(M^\top , D) = 0$, $i = 1, 2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>int. stabilizability, BÉZOUT-identity, chain of $n$ parametrizations</td>
<td>projective</td>
<td>$\text{Ext}^i_D(M^\top , D) = 0$, $1 \leq i \leq n$</td>
</tr>
<tr>
<td>flatness</td>
<td>free</td>
<td>in general no criteria but for PIR: torsion free = free</td>
</tr>
</tbody>
</table>

Mohamed Barakat and Daniel Robertz; homalg: An abstract package for homological algebra
left derived functors in homalg

using `LeftDerivedRightExactFunctor` we have the following multi-functor:

- Tor, TorMap, Tor2Map
left derived functors in \texttt{homalg}

using \texttt{LeftDerivedRightExactFunctor} we have the following \textit{multi-functor}:

- Tor, TorMap, Tor2Map

using \texttt{LeftDerivedFunctor} we have the following \textit{multi-functors}:

- Torq, TorqMap, Torq2Map
- LHom, LHomMap, LHom2Map
- LHomHom\_R, LHomHomMap\_R
- LHomHom, LHomHomMap, LHomHom2Map, LHomHom3Map
the use of `LeftDerivedFunctor` in *homalg*

```plaintext
'homalg/LHomHom' := proc(q::nonnegint,M,A,B,var::list,RingPackage) ###
    local RP, nar, optional, functor;

    RP := 'homalg/tablename'(procname, args[1..-1-nar]);

    if type(RP, procedure) then RETURN(RP(args[1..-1-nar])) fi;

    optional := args[6..-1-nar];

    #=====# begin of the core procedure #=====#

    functor := 'homalg/Functor_HomHom'(A, B);

    'homalg/LeftDerivedFunctor'(q, functor, M, var, optional, RP)
end:
```

*Mohamed Barakat and Daniel Robertz*  
*homalg: An abstract package for homological algebra*
LHomHomMap using FunctorMap

`
`homalg/LHomHomMap` := proc(q,M,phi,N,A,B,\text{var::list},RingPackage) ### Main:
local RP, nar, optional, functor;

RP := `homalg/tablename`(procname,args[\text{\text{-1}}],nar);

if type(RP,procedure) then RETURN(RP(args[1..\text{\text{-1}}-nar])) fi;

optional := args[8..\text{\text{-1}}-nar];

#=====# begin of the core procedure #=====#

functor := proc(M) `homalg/LHomHom`(q,M,A,B,args[1+1..\text{\text{-1}}]) end;

`homalg/FunctorMap`(functor,
 [M], phi, [N],
,\text{var},optional,RP)
end:

Mohamed Barakat and Daniel Robertz homalg: An abstract package for homological algebra
the tri-functor $L\text{Hom}\text{Hom}$ in homalg

```plaintext
# The functor $L\text{Hom}Hom_q(-,A,B)$

'homalg/Functor_LHomHom' := proc(q,A,B)
### Main: "Functors"

[ proc(M) 'homalg/LHomHom'(q,M,A,B,args[1+1..-1]) end,
  proc(M,phi,N) 'homalg/LHomHomMap'(q,M,phi,N,A,B,args[3+1..-1]) end,
  ## take care of multi-functoriality:
  A,
  proc(P)
    proc(M,phi,N) 'homalg/LHomHom2Map'(q,P,M,phi,N,B,args[3+1..-1]) end
  end,
  B,
  proc(P)
    proc(M,phi,N) 'homalg/LHomHom3Map'(q,P,A,M,phi,N,args[3+1..-1]) end
  end ]

end:
```

Mohamed Barakat and Daniel Robertz
homalg: An abstract package for homological algebra
natural transformations in *homalg*

we also have the following *natural transformations*:

- \( \text{NatTrTargetToCokernel} = \text{CokernelEpi} \)
- \( \text{NatTrKernelToSource} = \text{KernelEmb} \)
- \( \text{NatTrTorsionSubmoduleToId} = \text{TorsionSubmoduleEmb} \)
- \( \text{NatTrPullbackToTwoSources} = \text{PullbackPairOfMaps} \)
- \( \text{NatTrIdToHomHom}_R \)
- \( \text{NatIsoKerOfSqToImOfSq} = \text{Lambek} \)
Outline

1. Introduction
2. homalg
3. The Ring Packages
4. Todo

Mohamed Barakat and Daniel Robertz

homalg: An abstract package for homological algebra
features of the ring package PIR

- $R$ is one of the fields $\{\mathbb{F}_p, \mathbb{Q}\}$, or one of the Euclidean domains (and hence principal ideal domains) $\{\mathbb{Z}, \mathbb{Z}[I], \mathbb{Q}[x], \mathbb{F}_p[x]\}$ or one of its quotients, which are in general only principle ideal rings
features of the ring package \texttt{PIR}

- $R$ is one of the fields $\{\mathbb{F}_p, \mathbb{Q}\}$, or one of the Euclidean domains (and hence principal ideal domains) $\{\mathbb{Z}, \mathbb{Z}[I], \mathbb{Q}[x], \mathbb{F}_p[x]\}$ or one of its quotients, which are in general only principle ideal rings.
- the global dimension of the non-trivial principal ideal domains is 1, that of their semi-simple quotients is 0 and that of their non-semi-simple quotients is $\infty$
features of the ring package \texttt{PIR}

- \( R \) is one of the fields \( \{ \mathbb{F}_p, \mathbb{Q} \} \), or one of the Euclidean domains (and hence principal ideal domains) \( \{ \mathbb{Z}, \mathbb{Z}[I], \mathbb{Q}[x], \mathbb{F}_p[x] \} \) or one of its quotients, which are in general only principle ideal rings.

- the global dimension of the non-trivial principal ideal domains is 1, that of their semi-simple quotients is 0 and that of their non-semi-simple quotients is \( \infty \).

- essentially a sample ring package: \texttt{homalg} provides extras to easily use packages for principal ideal rings.
features of the ring package \textbf{PIR}

- the triangular basis is given by the \textsc{Hermite} normal form
The Ring Package PIR

- The triangular basis is given by the HERMITE normal form
- the SMITH normal form (elementary divisors) provides a normal form for a presentation of a module
features of the ring package PIR

- the triangular basis is given by the HERMITE normal form
- the SMITH normal form (elementary divisors) provides a normal form for a presentation of a module
- applications to algebraic topology and CONLEY-index-theory
'PIR/homalg':=table(
  [  ## Must only then be provided by the RingPackage in case the default
    ## "service" function does not match the Ring
    GlobalDim='PIR/PGlobalDim',
    PresentationInfo=
      proc(M,var) 'homalg/DiagonalElementsAndRank'(M,var,'PIR') end,

    ## Can optionally be provided by the RingPackage
    ## (homalg functions check if these functions are defined or not)
    ## (‘homalg/tablename’ gives no default value)
    BestBasis='PIR/PBestBasis',
    RingElementNormalForm='PIR/PRingElementNormalForm',

    ## Must be defined if other functions are not defined
    TriangularBasis='PIR/PTriangularBasis', ## needed by ‘homalg/BasisOfModule’
    QuotientWithRemainder='PIR/PQuo' ## needed by ‘homalg/Reduce’

    ## Must only then be provided by the RingPackage in case the default
    ## value provided by ‘homalg/tablename’ does not match the Ring
  ]):

Mohamed Barakat and Daniel Robertz  homalg: An abstract package for homological algebra
features of the ring package \texttt{Involutive}

\[ R = K[x_1, \ldots, x_n], \text{ where } K \geq \mathbb{Q} \text{ or } K = \mathbb{Z}/p\mathbb{Z} \]
features of the ring package **Involutive**

- \( R = K[x_1, \ldots, x_n] \), where \( K \geq \mathbb{Q} \) or \( K = \mathbb{Z}/p\mathbb{Z} \)
- the basis is given by the involutive basis
features of the ring package \textbf{Involution} 

- $R = K[x_1, \ldots, x_n]$, where $K \geq \mathbb{Q}$ or $K = \mathbb{Z}/p\mathbb{Z}$
- the basis is given by the involutive basis
- application to commutative algebra and algebraic geometry
the **homalg** conversion table of **Involutive**

```plaintext
`Involutive/homalg`:=table(
    [
        ## Must only then be provided by the RingPackage in case the default
        ## "service" function does not match the Ring
        BasisOfModule=`homalg/Involutive/IBasis`,
        PresentationInfo=`homalg/Involutive/PolHilbertCartan`,
        Reduce=`homalg/Involutive/PolInvoReduce`,
        SimplifyBasis=`Involutive/jetsdepcheck`,
        SyzygiesGenerators=`homalg/Involutive/PolSyzygies`
    ]):
```
features of the ring package \texttt{Janet}

- $R = K[\partial_1, \ldots, \partial_n]$, where $K$ is a differential field
- the basis is given by the JANET basis
features of the ring package Janet

- $R = K[\partial_1, \ldots, \partial_n]$, where $K$ is a differential field
- the basis is given by the JANET basis
- in the univariate case (ODEs) the JACOBSON normal form (with just one elementary divisor!) provides a normal form for a presentation of a module: $R$ is then a left principal ideal ring
features of the ring package \textit{Janet}

- \( R = K[\partial_1, \ldots, \partial_n] \), where \( K \) is a differential field
- the basis is given by the JANET basis
- in the univariate case (ODEs) the JACOBSON normal form (with just one elementary divisor!) provides a normal form for a presentation of a module: \( R \) is then a left principal ideal ring
- application to linear control theory
the homalg conversion table of Janet

`Janet/homalg`:=table(
    [
        ## Must only then be provided by the RingPackage in case the default
        ## "service" function does not match the Ring
        AddMat=`Janet/jaddmat`,
        BasisOfModule=`homalg/Janet/JBasis`,
        Compose=`Janet/CmpOp`,
        Involution=`homalg/Janet/Involution`,
        MulMat=`Janet/jmulmat`,
        PresentationInfo=`homalg/Janet/JanHilbertCartan`,
        Reduce=`homalg/Janet/InvoReduce`,
        SimplifyBasis=`homalg/Janet/Jandepcheck`,
        SyzygiesGenerators=`homalg/Janet/Syzygies`,
        SubMat=`Janet/jsubmat`,

        ## Can optionally be provided by the RingPackage
        ## (homalg functions check if these functions are defined or not)
        ## (`homalg/tablename` gives no default value)
        IsRingElement=proc(a) `Janet/jchkdop`(a,"") end,
    ]
)

Mohamed Barakat and Daniel Robertz homalg: An abstract package for homological algebra
## Must only then be provided by the RingPackage in case the default
## value provided by `homalg/tablename` does not match the Ring
DivideByUnit=`Janet/jdividebyunit`,
matrix=`Janet/jmkmat`,
Minus=`Janet/jsubcon`,
One=[[1, []]]
}):
the (pseudo) package Janet1

`Janet1/homalg`:=table(
    [
        ## Must only then be provided by the RingPackage in case the default
        ## "service" function does not match the Ring
        AddMat=`Janet/jaddmat`,
        BasisOfModule=`homalg/Janet/JBasis`,
        Compose=`Janet/CmpOp`,
        Involution=`homalg/Janet/Involution`,
        MulMat=`Janet/jmulmat`,
        PresentationInfo=`homalg/Janet/JanHilbertCartan`,
        Reduce=`homalg/Janet/InvoReduce`,
        SimplifyBasis=`homalg/Janet/Jandepcheck`,
        SyzygiesGenerators=`homalg/Janet/Syzygies`,
        SubMat=`Janet/jsubmat`,

        ## Can optionally be provided by the RingPackage
        ## (homalg functions check if these functions are defined or not)
        ## (`homalg/tablename` gives no default value)
        BestBasis=`homalg/Janet/Jacobson`,
        IsRingElement=proc(a) `Janet/jchkdop`(a,"") end,
    ]
)

Mohamed Barakat and Daniel Robertz  homalg: An abstract package for homological algebra
the (pseudo) ring package Janet1

## Must only then be provided by the RingPackage in case the default
## value provided by 'homalg/tablename' does not match the Ring
DivideByUnit='Janet/jdividebyunit',
matrix='Janet/jmkmat',
Minus='Janet/jsubcon',
One=[[1, []]]
features of the ring package JanetOre

- $R$ is an Ore-domain
features of the ring package *JanetOre*

- $R$ is an ORE-domain
- the basis is given by the involutive basis
features of the ring package \texttt{JanetOre}

- $R$ is an \texttt{ORE}-domain
- the basis is given by the involutive basis
- application to linear control theory
features of the ring package \texttt{JanetOre}

- $R$ is an O\textsc{re}-domain
- the basis is given by the involutive basis
- application to linear control theory
- using the BGG correspondence to compute the cohomology of coherent sheaves over projective spaces
Mohamed Barakat and Daniel Robertz  

`JanetOre/homalg`:table(  
[  
  ## Must only then be provided by the RingPackage in case the default  
  ## "service" function does not match the Ring  
  BasisOfModule=`homalg/JanetOre/Basis`,  
  Compose=`homalg/JanetOre/Mult`,  
  Involution=`homalg/JanetOre/Involution`,  
  PresentationInfo=`homalg/JanetOre/HilbertCartan`,  
  Reduce=`homalg/JanetOre/Reduce`,  
  SimplifyBasis=`JanetOre/jets_depcheck`,  
  SyzygiesGenerators=`homalg/JanetOre/Syzygies`,  
  IsUnit=`homalg/JanetOre/IsUnit`  
]):
features of the ring package **OreModules**

- $R$ is an ORE-domain
features of the ring package OreModules

- $R$ is an ORE-domain
- the basis is given by the GRÖBNER basis
features of the ring package \texttt{OreModules}

- $R$ is an ORE-domain
- the basis is given by the GRÖBNER basis
- application to linear control theory
features of the ring package **OreModules**

- $R$ is an ORE-domain
- the basis is given by the GRÖBNER basis
- application to linear control theory
the homalg conversion table of OreModules

`OreModules/homalg`:=table(
    [   
        ## Must only then be provided by the RingPackage in case the default
        ## "service" function does not match the Ring
        BasisOfModule=`homalg/OreModules/Basis`,
        Compose=`OreModules/Mult`,
        Involution=`homalg/OreModules/Involution`,
        PresentationInfo=`homalg/OreModules/HilbertSeries`,
        Reduce=`homalg/OreModules/Reduce`,
        SimplifyBasis=`homalg/OreModules/jetsdepcheck`,
        SyzygiesGenerators=`homalg/OreModules/SyzygiesGen`
    ]):

Mohamed Barakat and Daniel Robertz  homalg: An abstract package for homological algebra
Mohamed Barakat and Daniel Robertz  homalg: An abstract package for homological algebra
applications: topology, cohomology of groups, number theory, control theory ...
applications: topology, cohomology of groups, number theory, control theory ...

spectral sequences (simply more applications)
applications: topology, cohomology of groups, number theory, control theory ...

spectral sequences (simply more applications)

create and link more ring packages to homalg
applications: topology, cohomology of groups, number theory, control theory ... 
spectral sequences (simply more applications) 
create and link more ring packages to homalg 
handle all functors on an equal footing, regardless of the (source and) target abelian category
applications: topology, cohomology of groups, number theory, control theory ...

spectral sequences (simply more applications)

create and link more ring packages to homalg

handle all functors on an equal footing, regardless of the (source and) target abelian category

use homalg with other CASs: Singular, Kash, GAP, ...