HomalgToCAS

A window to the outer world

Version 2019.12.08

September 2015

Mohamed Barakat

Thomas Breuer

Simon Görtzen

Frank Lübeck

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/HomalgToCAS/chap0.html

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/homalgtocas

Mohamed Barakat
Email: mohamed.barakat@uni-siegen.de
Homepage: https://mohamed-barakat.github.io
Address: Department of Mathematics,
University of Siegen,
57072 Siegen,
Germany

Thomas Breuer
Email: sam@math.rwth-aachen.de
Homepage: http://www.math.rwth-aachen.de/~Thomas.Breuer/
Address: Lehrstuhl D für Mathematik, RWTH-Aachen,
Templergraben 64
52062 Aachen
Germany

Simon Görtzen
Email: simon.goertzen@rwth-aachen.de
Homepage: http://wwwb.math.rwth-aachen.de/goertzen/
Address: Lehrstuhl B für Mathematik, RWTH-Aachen,
Templergraben 64
52062 Aachen
Germany

Frank Lübeck
Email: frank.luebeck@math.rwth-aachen.de
Homepage: http://www.math.rwth-aachen.de/~Frank.Luebeck/
Address: Lehrstuhl D für Mathematik, RWTH-Aachen,
Templergraben 64
52062 Aachen
Germany

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

Copyright

Acknowledgements

We are very much indebted to Max Neunhöffer who provided the first piece of code around which the package IO_ForHomalg was built. The package HomalgToCAS provides a further abstraction layer preparing the communication.

Contents

Chapter 1

Introduction

HomalgToCAS is one of the core packages of the homalg project [hpa10]. But as one of the rather technical packages, this manual is probably not of interest for the average uers. The average user will usually not get in direct contact with the operations provided by this package.

Quoting from the Appendix (**homalg: The Core Packages and the Idea behind their Splitting**) of the homalg package manual ($\rightarrow$ (**homalg: HomalgToCAS**)):

"The package HomalgToCAS (which needs the homalg package) includes all what is needed to let the black boxes used by homalg reside in external computer algebra systems. So as mentioned above, HomalgToCAS is the right place to declare the three GAP representations external rings, external ring elements, and external matrices. Still, HomalgToCAS is independent from the external computer algebra system with which GAP will communicate *and* independent of how this communication physically looks like."

1.1 HomalgToCAS provides ...

- Declaration and construction of
  - external objects (which are pointers to data (rings,ring elements, matrices, ...) residing in external systems)
  - external rings (as a new representation for the GAP4-category of homalg rings)
  - external ring elements (as a new representation for the GAP4-category of homalg ring elements)
  - external matrices (as a new representation for the GAP4-category of homalg matrices)
- `LaunchCAS`: the standard interface used by homalg to launch external systems
- `TerminateCAS`: the standard interface used by homalg to terminate external systems
- `homalgSendBlocking`: the standard interface used by homalg to send commands to external systems
- External garbage collection: delete the data in the external systems that became obsolete for homalg
- `homalgIOMode`: decide how much of the communication you want to see

Chapter 2

Installation of the HomalgToCAS Package

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

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

```
LoadPackage( "HomalgToCAS" );
```

before its functions become available.

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

Mohamed Barakat, Thomas Breuer, Simon Görtzen, and Frank Lübeck

Chapter 3

Watch and Influence the Communication

3.1 Functions

3.1.1 homalgIOMode

▷ `homalgIOMode(`*`str[, str2[, str3]]`*`)` (function)

This function sets different modes which influence how much of the communication becomes visible. Handling the string *`str`* is *not* case-sensitive. `homalgIOMode` invokes the global function `homalgMode` defined in the homalg package with an "appropriate" argument (see code below). Alternatively, if a second or more strings are given, then `homalgMode` is invoked with the remaining strings *`str2`*, *`str3`*, ... at the end. In particular, you can use `homalgIOMode(` *`str`*`, "" )` to reset the effect of invoking `homalgMode`.

`str`	*`str`* (long form)	mode description
""	""	the default mode, i.e. the communication protocol won't be visible (`homalgIOMode( )` is a short form for `homalgIOMode( "" )`)
"a"	"all"	combine the modes "debug" and "file"
"b"	"basic"	the same as "picto" + `homalgMode( "basic" )`
"d"	"debug"	view the complete communication protocol
"f"	"file"	dump the communication protocol into a file with the name `Concatenation( "commands_file_of_", CAS, "_with_PID_", PID )`
"p"	"picto"	view the abbreviated communication protocol using the preassigned pictograms

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.

Caution:

- In case you choose one of the modes "file" or "all" you might want to set the global variable `HOMALG_IO.DoNotDeleteTemporaryFiles := true`; this is only important if during the computations some matrices get converted via files (using `ConvertHomalgMatrixViaFile`), as reading these files will be part of the protocol!
- It makes sense for the dumped communication protocol to be (re)executed with the respective external system, only in case the latter is deterministic (i.e. same-input-same-output).

Code

```
InstallGlobalFunction( homalgIOMode,
  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]} ) = "a" then
            mode := "all";
        elif LowercaseString( s{[1]} ) = "b" then
            mode := "basic";
        elif LowercaseString( s{[1]} ) = "d" then
            mode := "debug";
        elif LowercaseString( s{[1]} ) = "f" then
            mode := "file";
        elif LowercaseString( s{[1]} ) = "p" then
            mode := "picto";
        else
            mode := "";
        fi;
    else
        Error( "the first argument must be a string\n" );
    fi;

    if mode = "default" then
        ## reset to the default values
        HOMALG_IO.color_display := false;
        HOMALG_IO.show_banners := true;
        HOMALG_IO.save_CAS_commands_to_file := false;
        HOMALG_IO.DoNotDeleteTemporaryFiles := false;
        HOMALG_IO.SaveHomalgMaximumBackStream := false;
        HOMALG_IO.InformAboutCASystemsWithoutActiveRings := true;
        SetInfoLevel( InfoHomalgToCAS, 1 );
        homalgMode( );
    elif mode = "all" then
        homalgIOMode( "debug" );
        homalgIOMode( "file" );
    elif mode = "basic" then
        HOMALG_IO.color_display := true;
```

```
        HOMALG_IO.show_banners := true;
        SetInfoLevel( InfoHomalgToCAS, 4 );
        homalgMode( "basic" );        ## use homalgIOMode( "basic", "" ) to reset
    elif mode = "debug" then
        HOMALG_IO.color_display := true;
        HOMALG_IO.show_banners := true;
        SetInfoLevel( InfoHomalgToCAS, 8 );
        homalgMode( "debug" );        ## use homalgIOMode( "debug", "" ) to reset
    elif mode = "file" then
        HOMALG_IO.save_CAS_commands_to_file := true;
    elif mode = "picto" then
        HOMALG_IO.color_display := true;
        HOMALG_IO.show_banners := true;
        SetInfoLevel( InfoHomalgToCAS, 4 );
        homalgMode( "logic" );        ## use homalgIOMode( "picto", "" ) to reset
    fi;

    if nargs > 1 and IsString( arg[2] ) then
        CallFuncList( homalgMode, arg{[ 2 .. nargs ]} );
    fi;

end );
```

This is the part of the global function homalgSendBlocking that controls the visibility of the communication.

Code

```
io_info_level := InfoLevel( InfoHomalgToCAS );

if not IsBound( pictogram ) then
    pictogram := HOMALG_IO.Pictograms.unknown;
    picto := pictogram;
elif io_info_level >= 3 then
    picto := pictogram;
    ## add colors to the pictograms
    if pictogram = HOMALG_IO.Pictograms.ReducedEchelonForm and
       IsBound( HOMALG_MATRICES.color_BOE ) then
        pictogram := Concatenation( HOMALG_MATRICES.color_BOE, pictogram, "\033[0m" );
    elif pictogram = HOMALG_IO.Pictograms.BasisOfModule and
      IsBound( HOMALG_MATRICES.color_BOB ) then
        pictogram := Concatenation( HOMALG_MATRICES.color_BOB, pictogram, "\033[0m" );
    elif pictogram = HOMALG_IO.Pictograms.DecideZero and
      IsBound( HOMALG_MATRICES.color_BOD ) then
        pictogram := Concatenation( HOMALG_MATRICES.color_BOD, pictogram, "\033[0m" );
    elif pictogram = HOMALG_IO.Pictograms.SyzygiesGenerators and
      IsBound( HOMALG_MATRICES.color_BOH ) then
        pictogram := Concatenation( HOMALG_MATRICES.color_BOH, pictogram, "\033[0m" );
    elif pictogram = HOMALG_IO.Pictograms.BasisCoeff and
      IsBound( HOMALG_MATRICES.color_BOC ) then
        pictogram := Concatenation( HOMALG_MATRICES.color_BOC, pictogram, "\033[0m" );
    elif pictogram = HOMALG_IO.Pictograms.DecideZeroEffectively and
      IsBound( HOMALG_MATRICES.color_BOP ) then
        pictogram := Concatenation( HOMALG_MATRICES.color_BOP, pictogram, "\033[0m" );
    elif need_output or need_display then
```

```
        pictogram := Concatenation( HOMALG_IO.Pictograms.color_need_output,
                              pictogram, "\033[0m" );
    else
        pictogram := Concatenation( HOMALG_IO.Pictograms.color_need_command,
                              pictogram, "\033[0m" );
    fi;
else
    picto := pictogram;
fi;

if io_info_level >= 3 then
    if ( io_info_level >= 7 and not need_display ) or io_info_level >= 8 then
        ## print the pictogram, the prompt of the external system,
        ## and the sent command
        Info( InfoHomalgToCAS, 7, pictogram, " ", stream.prompt,
              L{[ 1 .. Length( L ) - 1 ]} );
    elif io_info_level >= 4 then
        ## print the pictogram and the prompt of the external system
        Info( InfoHomalgToCAS, 4, pictogram, " ", stream.prompt, "..." );
    else
        ## print the pictogram only
        Info( InfoHomalgToCAS, 3, pictogram );
    fi;

fi;
```

3.2 The Pictograms

3.2.1 HOMALG_IO.Pictograms

▷ `HOMALG_IO.Pictograms` (global variable)

The record of pictograms is a component of the record `HOMALG_IO`.

Code

```
Pictograms := rec(

  ##
  ## colors:
  ##

  ## pictogram color of a "need_command" or assignment operation:
  color_need_command                    := "\033[1;33;44m",

  ## pictogram color of a "need_output" or "need_display" operation:
  color_need_output                     := "\033[1;34;43m",

  ##
  ## good morning computer algebra system:
  ##

  ## initialize:
```

```
initialize                              := "ini",

## define macros:
define                                  := "def",

## get time:
time                                    := ":ms",

## memory usage:
memory                                  := "mem",

## unknown:
unknown                                 := "???",

##
## external garbage collection:
##

## delete a variable:
delete                                  := "xxx",

## delete serveral variables:
multiple_delete                         := "XXX",

## trigger the garbage collector:
garbage_collector                       := "grb",

##
## create lists:
##

## define a list:
CreateList                              := "lst",

##
## create rings:
##

## define a ring:
CreateHomalgRing                        := "R:=",

## get the names of the "variables" defining the ring:
variables                               := "var",

## define zero:
Zero                                    := "0:=",

## define one:
One                                     := "1:=",

## define minus one:
MinusOne                                := "-:=",
```

```
##
## mandatory ring operations:
##

## get the name of an element:
## (important if the CAS pretty-prints ring elements,
##  we need names that can be used as input!)
## (install a method instead of a homalgTable entry)
homalgSetName                           := "\"a\"",

## a = 0 ?
IsZero                                  := "a=0",

## a = 1 ?
IsOne                                   := "a=1",

## substract two ring elements
## (needed by SimplerEquivalentMatrix in case
##  CopyRow/ColumnToIdentityMatrix are not defined):
Minus                                   := "a-b",

## divide the element a by the unit u
## (needed by SimplerEquivalentMatrix in case
##  DivideEntryByUnit is not defined):
DivideByUnit                            := "a/u",

## important ring operations:
## (important for performance since existing
##  fallback methods cause a lot of traffic):

## is u a unit?
## (mainly needed by the fallback methods for matrices, see below):
IsUnit                                  := "?/u",

##
## optional ring operations:
##

## copy an element:
CopyElement                             := "a>a",

## add two ring elements:
Sum                                     := "a+b",

## multiply two ring elements:
Product                                 := "a*b",

## the (greatest) common divisor:
Gcd                                     := "gcd",

## cancel the (greatest) common divisor:
CancelGcd                               := "ccd",
```

```
## random polynomial:
RandomPol                               := "rpl",

## numerator:
Numerator                               := "num",

## denominator:
Denominator                             := "den",

## evaluate polynomial:
Evaluate                                := "evl",

## degree of a multivariate polynomial
DegreeOfRingElement                     := "deg",

## maximal degree part of a polynomial
MaximalDegreePart                       := "mdp",

## is irreducible:
IsIrreducible                           := "irr",

##
## create matrices:
##

## define a matrix:
HomalgMatrix                            := "A:=",

## copy a matrix:
CopyMatrix                              := "A>A",

## load a matrix from file:
LoadHomalgMatrixFromFile                := "A<<",

## save a matrix to file:
SaveHomalgMatrixToFile                  := "A>>",

## get a matrix entry as a string:
MatElm                  := "<ij",

## set a matrix entry from a string:
SetMatElm                   := ">ij",

## add to a matrix entry from a string:
AddToMatElm                 := "+ij",

## get a list of the matrix entries as a string:
GetListOfHomalgMatrixAsString           := "\"A\"",

## get a listlist of the matrix entries as a string:
GetListListOfHomalgMatrixAsString       := "\"A\"",

## get a "sparse" list of the matrix entries as a string:
```

```
GetSparseListOfHomalgMatrixAsString     := ".A.",

## assign a "sparse" list of matrix entries to a variable:
sparse                                  := "spr",

## list of assumed inequalities:
Inequalities                            := "<>0",

## list of assumed inequalities:
MaximalIndependentSet                   := "idp",

##
## mandatory matrix operations:
##

## test if a matrix is the zero matrix:
## CAUTION: the external system must be able to check
##          if the matrix is zero modulo possible ring relations
##          only known to the external system!
IsZeroMatrix                            := "A=0",

## number of rows:
NrRows                                  := "#==",

## number of columns:
NrColumns                               := "#||",

## determinant of a matrix over a (commutative) ring:
Determinant                             := "det",

## create a zero matrix:
ZeroMatrix                              := "(0)",

## create a initial zero matrix:
InitialMatrix                           := "[0]",

## create an identity matrix:
IdentityMatrix                          := "(1)",

## create an initial identity matrix:
InitialIdentityMatrix                   := "[1]",

## "transpose" a matrix (with "the" involution of the ring):
Involution                              := "A^*",

## transpose a matrix
TransposedMatrix                        := "A^t",

## get certain rows of a matrix:
CertainRows                             := "===",

## get certain columns of a matrix:
CertainColumns                          := "|||",
```

```
## stack to matrices vertically:
UnionOfRows                             := "A_B",

## glue to matrices horizontally:
UnionOfColumns                          := "A|B",

## create a block diagonal matrix:
DiagMat                                 := "A\\B",

## the Kronecker (tensor) product of two matrices:
KroneckerMat                            := "AoB",

## multiply a ring element with a matrix:
MulMat                                  := "a*A",

## multiply a matrix with a ring element:
MulMatRight                             := "A*a",

## add two matrices:
AddMat                                  := "A+B",

## substract two matrices:
SubMat                                  := "A-B",

## multiply two matrices:
Compose                                 := "A*B",

## pullback a matrix by a ring map:
Pullback                                := "pbk",

##
## important matrix operations:
## (important for performance since existing
##  fallback methods cause a lot of traffic):
##

## test if two matrices are equal:
## CAUTION: the external system must be able to check
##          equality of the two matrices modulo possible ring relations
##          only known to the external system!
AreEqualMatrices                        := "A=B",

## test if a matrix is the identity matrix:
IsIdentityMatrix                        := "A=1",

## test if a matrix is diagonal (needed by the display method):
IsDiagonalMatrix                        := "A=\\",

## get the positions of the zero rows:
ZeroRows                                := "0==",

## get the positions of the zero columns:
```

```
ZeroColumns                             := "0||",

## get "column-independent" unit positions
## (needed by ReducedBasisOfModule):
GetColumnIndependentUnitPositions       := "ciu",

## get "row-independent" unit positions
## (needed by ReducedBasisOfModule):
GetRowIndependentUnitPositions          := "riu",

## get the position of the "first" unit in the matrix
## (needed by SimplerEquivalentMatrix):
GetUnitPosition                         := "gup",

## position of the first non-zero entry per row
PositionOfFirstNonZeroEntryPerRow       := "fnr",

## position of the first non-zero entry per column
PositionOfFirstNonZeroEntryPerColumn    := "fnc",

## indicator matrix of non-zero entries
IndicatorMatrixOfNonZeroEntries         := "<>0",

## transposed matrix:
TransposedMatrix                        := "^tr",

## divide an entry of a matrix by a unit
## (needed by SimplerEquivalentMatrix in case
##  DivideRow/ColumnByUnit are not defined):
DivideEntryByUnit                       := "ij/",

## divide a row by a unit
## (needed by SimplerEquivalentMatrix):
DivideRowByUnit                         := "-/u",

## divide a column by a unit
## (needed by SimplerEquivalentMatrix):
DivideColumnByUnit                      := "|/u",

## divide a row by a unit
## (needed by SimplerEquivalentMatrix):
CopyRowToIdentityMatrix                 := "->-",

## divide a column by a unit
## (needed by SimplerEquivalentMatrix):
CopyColumnToIdentityMatrix              := "|>|",

## set a column (except a certain row) to zero
## (needed by SimplerEquivalentMatrix):
SetColumnToZero                         := "|=0",

## get the positions of the rows with a single one
## (needed by SimplerEquivalentMatrix):
```

```
GetCleanRowsPositions                   := "crp",

## convert a single row matrix into a matrix
## with specified number of rows/columns
## (needed by the display methods for homomorphisms):
ConvertRowToMatrix                      := "-%A",

## convert a single column matrix into a matrix
## with specified number of rows/columns
## (needed by the display methods for homomorphisms):
ConvertColumnToMatrix                   := "|%A",

## convert a matrix into a single row matrix:
ConvertMatrixToRow                      := "A%-",

## convert a matrix into a single column matrix:
ConvertMatrixToColumn                   := "A%|",

##
## basic matrix operations:
##

## compute a (r)educed (e)chelon (f)orm:
ReducedEchelonForm                      := "ref",

## compute a "(bas)is" of a given set of module elements:
BasisOfModule                           := "bas",

## compute a reduced "(Bas)is" of a given set of module elements:
ReducedBasisOfModule                    := "Bas",

## (d)e(c)ide the ideal/submodule membership problem,
## i.e. if an element is (0) modulo the ideal/submodule:
DecideZero                              := "dc0",

## compute a generating set of (syz)ygies:
SyzygiesGenerators                      := "syz",

## compute a generating set of reduced (Syz)ygies:
ReducedSyzygiesGenerators               := "Syz",

## compute a (R)educed (E)chelon (F)orm
## together with the matrix of coefficients:
ReducedEchelonFormC                     := "REF",

## compute a "(BAS)is" of a given set of module elements
## together with the matrix of coefficients:
BasisCoeff                              := "BAS",

## (D)e(C)ide the ideal/submodule membership problem,
## i.e. write an element effectively as (0) modulo the ideal/submodule:
DecideZeroEffectively                   := "DC0",
```

```
##
## optional matrix operations:
##

## Hilbert-Poincare series of a module:
HilbertPoincareSeries                   := "HPs",

## Hilbert polynomial of a module:
HilbertPolynomial                       := "Hil",

## affine dimension of a module:
AffineDimension                         := "dim",

## affine degree of a module:
AffineDegree                            := "adg",

## the constant term of the hilbert polynomial:
ConstantTermOfHilbertPolynomial         := "P_0",

## differentiate a matrix M w.r.t. a matrix D
Diff                                    := "dif",

## maximal dimensional radical subobject:
MaxDimensionalRadicalSubobject          := "V_d",

## radical subobject:
RadicalSubobject                        := "rad",

## radical decomposition:
RadicalDecomposition                    := "VxU",

## maximal dimensional subobject:
MaxDimensionalSubobject                 := "X_d",

## equi-dimensional decomposition:
EquiDimenionalDecomposition             := "XxY",

## primary decomposition:
PrimaryDecomposition                    := "YxZ",

## eliminate variables:
Eliminate                               := "eli",

## leading module:
LeadingModule                           := "led",

## the i-th monomial matrix
MonomialMatrix                          := "mon",

## matrix of symbols:
MatrixOfSymbols                         := "smb",

## leading module:
```

```
  ## coefficients:
  Coefficients                          := "cfs",

  ##
  ## optional module operations:
  ##

  ## compute a better equivalent matrix
  ## (field -> row+col Gauss, PIR -> Smith, Dedekind domain -> Krull, etc ...):
  BestBasis                             := "(\\)",

  ## compute elementary divisors:
  ElementaryDivisors                    := "div",

  ##
  ## for the eye:
  ##

  ## display objects:
  Display                               := "dsp",

  ## the LaTeX code of the mathematical entity:
  homalgLaTeX                           := "TeX",

)
```

Chapter 4

External Rings

4.1 External Rings: Representation

4.1.1 IsHomalgExternalRingRep

▷ `IsHomalgExternalRingRep(`*R*`)` (Representation)

Returns: `true` or `false`

The internal representation of homalg rings.

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

4.2 Rings: Constructors

4.3 External Rings: Operations and Functions

Appendix A

Overview of the homalg Package Source Code

The package HomalgToCAS is split in several files.

Filename .gd/.gi	Content
HomalgToCAS	the global variable HOMALG_IO and the global function homalgIOMode
homalgExternalObject	homalg external objects, homalgPointer, homalgExternalCASystem, homalgStream, ...
HomalgExternalRing	CreateHomalgExternalRing, HomalgExternalRingElement
HomalgExternalMatrix	ConvertHomalgMatrix, ConvertHomalgMatrixViaFile
homalgSendBlocking	homalgFlush, homalgSendBlocking
IO	LaunchCAS, TerminateCAS

Table: *The HomalgToCAS package files*

References

[hpa10] The homalg project authors. *The homalg project*, 2003-2010. http://homalg.math.rwth-aachen.de/. 4

Index