RingsForHomalg

Dictionaries of external rings

2022.11-01

3 November 2022

Mohamed Barakat

Simon Görtzen

Markus Kirschmer

Markus Lange-Hegermann

Oleksandr Motsak

Daniel Robertz

Hans Schönemann

Andreas Steenpaß

Vinay Wagh

Mohamed Barakat

Email: mohamed.barakat@uni-siegen.de

Homepage: https://mohamed-barakat.github.io

Address: Walter-Flex-Str. 3
   57072 Siegen
   Germany

Simon Görtzen

Email: simon.goertzen@rwth-aachen.de

Homepage: https://www.linkedin.com/in/simongoertzen/

Address: Simon Görtzen
   Lehrstuhl B fuer Mathematik, RWTH Aachen
   Templergraben 64
   52062 Aachen
   Germany

Markus Kirschmer

Email: markus.kirschmer@math.rwth-aachen.de

Homepage: http://www.math.rwth-aachen.de/~Markus.Kirschmer/

Address: Markus Kirschmer
   Lehrstuhl D fuer Mathematik, RWTH Aachen
   Templergraben 64
   52062 Aachen
   Germany

Markus Lange-Hegermann

Email: markus.lange-hegermann@hs-owl.de

Homepage: https://www.th-owl.de/eecs/fachbereich/team/markus-lange-hegermann/

Address: Markus Lange-Hegermann
   Hochschule Ostwestfalen-Lippe
   Liebigstraße 87
   32657 Lemgo
   Germany

Oleksandr Motsak

Email: motsak@mathematik.uni-kl.de

Homepage: http://www.mathematik.uni-kl.de/~motsak/

Address: Department of Mathematics
   University of Kaiserslautern
   67653 Kaiserslautern
   Germany

Daniel Robertz

Email: daniel@momo.math.rwth-aachen.de

Homepage: http://wwwb.math.rwth-aachen.de/~daniel/

Address: Daniel Robertz
        Lehrstuhl B fuer Mathematik, RWTH Aachen
        Templergraben 64
        52062 Aachen
        Germany

Hans Schönemann

Email: hannes@mathematik.uni-kl.de

Homepage: http://www.mathematik.uni-kl.de/~hannes/

Address: Department of Mathematics
        University of Kaiserslautern
        67653 Kaiserslautern
        Germany

Andreas Steenpaß

Email: steenpass@mathematik.uni-kl.de

Address: Department of Mathematics
        University of Kaiserslautern
        67653 Kaiserslautern
        Germany

Vinay Wagh

Email: waghoba@gmail.com

Homepage: http://www.iitg.ernet.in/vinay.wagh/

Address: E-102, Department of Mathematics,
        Indian Institute of Technology Guwahati,
        Guwahati, Assam, India.
        PIN: 781 039.
        India

Copyright

Contents

Chapter 1

Introduction

This package is part of the homalg project [hom22]. The role of the package is described in the manual of the homalg package.

1.1 Ring Constructions for Supported External Computer Algebra Systems

Here are some of the supported ring constructions:

1.1.1 external **GAP**

```
——————————————— Example ———————————————
  gap> ZZ := HomalgRingOfIntegersInExternalGAP( );
  Z
  gap> Display( ZZ );
  <An external ring residing in the CAS GAP>
  gap> F2 := HomalgRingOfIntegersInExternalGAP( 2, ZZ );
  GF(2)
  gap> Display( F2 );
  <An external ring residing in the CAS GAP>
```

`F2 := HomalgRingOfIntegersInExternalGAP( 2 )` would launch another GAP.

```
——————————————— Example ———————————————
  gap> Z4 := HomalgRingOfIntegersInExternalGAP( 4, ZZ );
  Z/4Z
  gap> Display( Z4 );
  <An external ring residing in the CAS GAP>
  gap> Z_4 := HomalgRingOfIntegersInExternalGAP( ZZ ) / 4;
  Z/( 4 )
  gap> Display( Z_4 );
  <A residue class ring>
  gap> Q := HomalgFieldOfRationalsInExternalGAP( ZZ );
  Q
  gap> Display( Q );
  <An external ring residing in the CAS GAP>
```

1.1.2 Singular

```
──────────────────── Example ────────────────────
  #@if IsBound( TryLaunchCAS_IO_ForHomalg( HOMALG_IO_Singular ).stdout )
  gap> F2 := HomalgRingOfIntegersInSingular( 2 );
  GF(2)
  gap> Display( F2 );
  <An external ring residing in the CAS Singular>
  gap> F2s := HomalgRingOfIntegersInSingular( 2, "s" ,F2 );
  GF(2)(s)
  gap> Display( F2s );
  <An external ring residing in the CAS Singular>
  gap> ZZ := HomalgRingOfIntegersInSingular( F2 );
  Z
  gap> Display( ZZ );
  <An external ring residing in the CAS Singular>
  gap> Q := HomalgFieldOfRationalsInSingular( F2 );
  Q
  gap> Display( Q );
  <An external ring residing in the CAS Singular>
  gap> Qs := HomalgFieldOfRationalsInSingular( "s", F2 );
  Q(s)
  gap> Display( Qs );
  <An external ring residing in the CAS Singular>
  gap> Qi := HomalgFieldOfRationalsInSingular( "i", "i^2+1", Q );
  Q[i]/(i^2+1)
  gap> Display( Qi );
  <An external ring residing in the CAS Singular>
  #@fi
```

`Q := HomalgFieldOfRationalsInSingular( )` would launch another Singular.

```
──────────────────── Example ────────────────────
  #@if IsBound( TryLaunchCAS_IO_ForHomalg( HOMALG_IO_Singular ).stdout )
  gap> F2xyz := F2 * "x,y,z";
  GF(2)[x,y,z]
  gap> Display( F2xyz );
  <An external ring residing in the CAS Singular>
  gap> F2sxyz := F2s * "x,y,z";
  GF(2)(s)[x,y,z]
  gap> Display( F2sxyz );
  <An external ring residing in the CAS Singular>
  gap> F2xyzw := F2xyz * "w";
  GF(2)[x,y,z][w]
  gap> Display( F2xyzw );
  <An external ring residing in the CAS Singular>
  gap> F2sxyzw := F2sxyz * "w";
  GF(2)(s)[x,y,z][w]
  gap> Display( F2sxyzw );
  <An external ring residing in the CAS Singular>
  gap> ZZxyz := ZZ * "x,y,z";
  Z[x,y,z]
  gap> Display( ZZxyz );
  <An external ring residing in the CAS Singular>
```

```
gap> ZZxyzw := ZZxyz * "w";
Z[x,y,z][w]
gap> Display( ZZxyzw );
<An external ring residing in the CAS Singular>
gap> Qxyz := Q * "x,y,z";
Q[x,y,z]
gap> Display( Qxyz );
<An external ring residing in the CAS Singular>
gap> Qsxyz := Qs * "x,y,z";
Q(s)[x,y,z]
gap> Display( Qsxyz );
<An external ring residing in the CAS Singular>
gap> Qixyz := Qi * "x,y,z";
(Q[i]/(i^2+1))[x,y,z]
gap> Display( Qixyz );
<An external ring residing in the CAS Singular>
gap> Qxyzw := Qxyz * "w";
Q[x,y,z][w]
gap> Display( Qxyzw );
<An external ring residing in the CAS Singular>
gap> Qsxyzw := Qsxyz * "w";
Q(s)[x,y,z][w]
gap> Display( Qsxyzw );
<An external ring residing in the CAS Singular>
gap> Dxyz := RingOfDerivations( Qxyz, "Dx,Dy,Dz" );
Q[x,y,z]<Dx,Dy,Dz>
gap> Display( Dxyz );
<An external ring residing in the CAS Singular>
gap> Exyz := ExteriorRing( Qxyz, "e,f,g" );
Q{e,f,g}
gap> Display( Exyz );
<An external ring residing in the CAS Singular>
gap> Dsxyz := RingOfDerivations( Qsxyz, "Dx,Dy,Dz" );
Q(s)[x,y,z]<Dx,Dy,Dz>
gap> Display( Dsxyz );
<An external ring residing in the CAS Singular>
gap> Esxyz := ExteriorRing( Qsxyz, "e,f,g" );
Q(s){e,f,g}
gap> Display( Esxyz );
<An external ring residing in the CAS Singular>
gap> Dixyz := RingOfDerivations( Qixyz, "Dx,Dy,Dz" );
(Q[i]/(i^2+1))[x,y,z]<Dx,Dy,Dz>
gap> Display( Dixyz );
<An external ring residing in the CAS Singular>
gap> Eixyz := ExteriorRing( Qixyz, "e,f,g" );
(Q[i]/(i^2+1)){e,f,g}
gap> Display( Eixyz );
<An external ring residing in the CAS Singular>
gap> qring := HomalgQRingInSingular( Qxyz, "x*y" );
Q[x,y,z]/( x*y )
gap> Display( qring );
<An external ring residing in the CAS Singular>
```

```
gap> "z + x*y" / qring = "z" / qring;
true
#@fi
```

1.1.3 MAGMA

```
 ─────────────────────── Example ───────────────────────
#@if IsBound( TryLaunchCAS_IO_ForHomalg( HOMALG_IO_MAGMA ).stdout )
gap> ZZ := HomalgRingOfIntegersInMAGMA( );
Z
gap> Display( ZZ );
<An external ring residing in the CAS MAGMA>
gap> F2 := HomalgRingOfIntegersInMAGMA( 2, ZZ );
GF(2)
gap> Display( F2 );
<An external ring residing in the CAS MAGMA>
#@fi
```

`F2 := HomalgRingOfIntegersInMAGMA( 2 )` would launch another MAGMA.

```
 ─────────────────────── Example ───────────────────────
#@if IsBound( TryLaunchCAS_IO_ForHomalg( HOMALG_IO_MAGMA ).stdout )
gap> Z_4 := HomalgRingOfIntegersInMAGMA( ZZ ) / 4;
Z/( 4 )
gap> Display( Z_4 );
<A residue class ring>
gap> Q := HomalgFieldOfRationalsInMAGMA( ZZ );
Q
gap> Display( Q );
<An external ring residing in the CAS MAGMA>
gap> F2xyz := F2 * "x,y,z";
GF(2)[x,y,z]
gap> Display( F2xyz );
<An external ring residing in the CAS MAGMA>
gap> Qxyz := Q * "x,y,z";
Q[x,y,z]
gap> Display( Qxyz );
<An external ring residing in the CAS MAGMA>
gap> Exyz := ExteriorRing( Qxyz, "e,f,g" );
Q{e,f,g}
gap> Display( Exyz );
<An external ring residing in the CAS MAGMA>
#@fi
```

1.1.4 Macaulay2

```
 ─────────────────────── Example ───────────────────────
#@if IsBound( TryLaunchCAS_IO_ForHomalg( HOMALG_IO_Macaulay2 ).stdout )
gap> ZZ := HomalgRingOfIntegersInMacaulay2( );
Z
gap> Display( ZZ );
<An external ring residing in the CAS Macaulay2>
gap> F2 := HomalgRingOfIntegersInMacaulay2( 2, ZZ );
GF(2)
```

```
gap> Display( F2 );
<An external ring residing in the CAS Macaulay2>
#@fi
```

F2 := HomalgRingOfIntegersInMacaulay2(2) would launch another Macaulay2.

```
————————————— Example —————————————
#@if IsBound( TryLaunchCAS_IO_ForHomalg( HOMALG_IO_Macaulay2 ).stdout )
gap> Z_4 := HomalgRingOfIntegersInMacaulay2( ZZ ) / 4;
Z/( 4 )
gap> Display( Z_4 );
<A residue class ring>
gap> Q := HomalgFieldOfRationalsInMacaulay2( ZZ );
Q
gap> Display( Q );
<An external ring residing in the CAS Macaulay2>
gap> F2xyz := F2 * "x,y,z";
GF(2)[x,y,z]
gap> Display( F2xyz );
<An external ring residing in the CAS Macaulay2>
gap> Qxyz := Q * "x,y,z";
Q[x,y,z]
gap> Display( Qxyz );
<An external ring residing in the CAS Macaulay2>
gap> Dxyz := RingOfDerivations( Qxyz, "Dx,Dy,Dz" );
Q[x,y,z]<Dx,Dy,Dz>
gap> Display( Dxyz );
<An external ring residing in the CAS Macaulay2>
gap> Exyz := ExteriorRing( Qxyz, "e,f,g" );
Q{e,f,g}
gap> Display( Exyz );
<An external ring residing in the CAS Macaulay2>
#@fi
```

1.1.5 Sage

```
————————————— Example —————————————
#@if IsBound( TryLaunchCAS_IO_ForHomalg( HOMALG_IO_Sage ).stdout )
gap> ZZ := HomalgRingOfIntegersInSage( );
Z
gap> Display( ZZ );
<An external ring residing in the CAS Sage>
gap> F2 := HomalgRingOfIntegersInSage( 2, ZZ );
GF(2)
gap> Display( F2 );
<An external ring residing in the CAS Sage>
#@fi
```

F2 := HomalgRingOfIntegersInSage(2) would launch another Sage.

```
————————————— Example —————————————
#@if IsBound( TryLaunchCAS_IO_ForHomalg( HOMALG_IO_Sage ).stdout )
gap> Z_4 := HomalgRingOfIntegersInSage( ZZ ) / 4;
Z/( 4 )
```

```
gap> Display( Z_4 );
<A residue class ring>
gap> Q := HomalgFieldOfRationalsInSage( ZZ );
Q
gap> Display( Q );
<An external ring residing in the CAS Sage>
gap> F2x := F2 * "x";
GF(2)[x]
gap> Display( F2x );
<An external ring residing in the CAS Sage>
gap> Qx := Q * "x";
Q[x]
gap> Display( Qx );
<An external ring residing in the CAS Sage>
#@fi
```

1.1.6 Maple

─────────── Example ───────────
```
#@if IsBound( TryLaunchCAS_IO_ForHomalg( HOMALG_IO_Maple ).stdout )
gap> ZZ := HomalgRingOfIntegersInMaple( );
Z
gap> Display( ZZ );
<An external ring residing in the CAS Maple>
gap> F2 := HomalgRingOfIntegersInMaple( 2, ZZ );
GF(2)
gap> Display( F2 );
<An external ring residing in the CAS Maple>
#@fi
```

`F2 := HomalgRingOfIntegersInMaple( 2 )` would launch another Maple.

─────────── Example ───────────
```
#@if IsBound( TryLaunchCAS_IO_ForHomalg( HOMALG_IO_Maple ).stdout )
gap> Z4 := HomalgRingOfIntegersInMaple( 4, ZZ );
Z/4Z
gap> Display( Z4 );
<An external ring residing in the CAS Maple>
gap> Z_4 := HomalgRingOfIntegersInMaple( ZZ ) / 4;
Z/( 4 )
gap> Display( Z_4 );
<A residue class ring>
gap> Q := HomalgFieldOfRationalsInMaple( ZZ );
Q
gap> Display( Q );
<An external ring residing in the CAS Maple>
gap> F2xyz := F2 * "x,y,z";
GF(2)[x,y,z]
gap> Display( F2xyz );
<An external ring residing in the CAS Maple>
gap> Qxyz := Q * "x,y,z";
Q[x,y,z]
gap> Display( Qxyz );
```

```
<An external ring residing in the CAS Maple>
gap> Dxyz := RingOfDerivations( Qxyz, "Dx,Dy,Dz" );
Q[x,y,z]<Dx,Dy,Dz>
gap> Display( Dxyz );
<An external ring residing in the CAS Maple>
gap> Exyz := ExteriorRing( Qxyz, "e,f,g" );
Q{e,f,g}
gap> Display( Exyz );
<An external ring residing in the CAS Maple>
#@fi
```

Chapter 2

Installation of the **RingsForHomalg** Package

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

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

```
LoadPackage( "RingsForHomalg" );
```

before its functions become available.

Please, send us 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.

The authors.

Chapter 3

The Ring Table

3.1 An Example for a Ring Table - Singular

todo: introductory text, mention: transposed matrices, the macros, refer to the philosophy

3.1.1 BasisOfRowModule (in the homalg table for Singular)

▷ BasisOfRowModule(*M*)   (function)

Returns:

This is the entry of the homalg table, which calls the corresponding macro BasisOfRowModule (3.1.2) inside the computer algebra system.

```
 ──────────────── Code ────────────────
  BasisOfRowModule :=
    function( M )
      local N;

      N := HomalgVoidMatrix(
        "unknown_number_of_rows",
        NrColumns( M ),
        HomalgRing( M )
      );

      homalgSendBlocking(
        [ "matrix ", N, " = BasisOfRowModule(", M, ")" ],
        "need_command",
        "BasisOfModule"
      );

      return N;

    end,
```

3.1.2 BasisOfRowModule (Singular macro)

▷ BasisOfRowModule(*M*)   (function)

Returns:

```
─── Code ───
     BasisOfRowModule := "\n\
 proc BasisOfRowModule (matrix M)\n\
 {\n\
   return(std(M));\n\
 }\n\n",
```

3.1.3 BasisOfColumnModule (in the homalg table for Singular)

▷ BasisOfColumnModule(*M*)   (function)

    Returns:

This is the entry of the homalg table, which calls the corresponding macro
BasisOfColumnModule (3.1.4) inside the computer algebra system.

```
─── Code ───
  BasisOfColumnModule :=
    function( M )
      local N;

      N := HomalgVoidMatrix(
        NrRows( M ),
        "unknown_number_of_columns",
        HomalgRing( M )
      );

      homalgSendBlocking(
        [ "matrix ", N, " = BasisOfColumnModule(", M, ")" ],
        "need_command",
        "BasisOfModule"
      );

      return N;

    end,
```

3.1.4 BasisOfColumnModule (Singular macro)

▷ BasisOfColumnModule(*M*)   (function)

    Returns:

```
─── Code ───
     BasisOfColumnModule := "\n\
 proc BasisOfColumnModule (matrix M)\n\
 {\n\
   return(Involution(BasisOfRowModule(Involution(M))));\n\
 }\n\n",
```

3.1.5 DecideZeroRows (in the homalg table for Singular)

▷ DecideZeroRows(*A, B*)   (function)

    Returns:

This is the entry of the homalg table, which calls the corresponding macro `DecideZeroRows` (3.1.6) inside the computer algebra system. The rows of B must form a basis (see `BasisOfRowModule` (3.1.1)).

```
DecideZeroRows :=
  function( A, B )
    local N;

    N := HomalgVoidMatrix(
      NrRows( A ),
      NrColumns( A ),
      HomalgRing( A )
    );

    homalgSendBlocking(
      [ "matrix ", N, " = DecideZeroRows(", A, B, ")" ],
      "need_command",
      "DecideZero"
    );

    return N;

  end,
```

3.1.6 DecideZeroRows (Singular macro)

▷ `DecideZeroRows(`*A, B*`)`   (function)

Returns:

```
    DecideZeroRows := "\n\
proc DecideZeroRows (matrix A, module B)\n\
{\n\
  attrib(B,\"isSB\",1);\n\
  return(reduce(A,B));\n\
}\n\n",
```

3.1.7 DecideZeroColumns (in the homalg table for Singular)

▷ `DecideZeroColumns(`*A, B*`)`   (function)
   Returns:

This is the entry of the homalg table, which calls the corresponding macro `DecideZeroColumns` (3.1.8) inside the computer algebra system. The columns of B must form a basis (see `BasisOfColumnModule` (3.1.3)).

```
DecideZeroColumns :=
  function( A, B )
    local N;

    N := HomalgVoidMatrix(
      NrRows( A ),
```

```
      NrColumns( A ),
      HomalgRing( A )
    );

    homalgSendBlocking(
      [ "matrix ", N, " = DecideZeroColumns(", A, B, ")" ],
      "need_command",
      "DecideZero"
    );

    return N;

  end,
```

3.1.8 DecideZeroColumns (Singular macro)

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

    Returns:

——————————— Code ———————————
```
    DecideZeroColumns := "\n\
proc DecideZeroColumns (matrix A, matrix B)\n\
{\n\
  return(Involution(DecideZeroRows(Involution(A),Involution(B))));\n\
}\n\n",
```

3.1.9 SyzygiesGeneratorsOfRows (in the homalg table for Singular)

▷ SyzygiesGeneratorsOfRows(*M*)   (function)

    Returns:

    This is the entry of the homalg table, which calls the corresponding macro SyzygiesGeneratorsOfRows (3.1.10) inside the computer algebra system.

——————————— Code ———————————
```
  SyzygiesGeneratorsOfRows :=
    function( M )
      local N;

      N := HomalgVoidMatrix(
        "unknown_number_of_rows",
        NrRows( M ),
        HomalgRing( M )
      );

      homalgSendBlocking(
        [ "matrix ", N, " = SyzygiesGeneratorsOfRows(", M, ")" ],
        "need_command",
        "SyzygiesGenerators"
      );

      return N;

    end,
```

3.1.10 SyzygiesGeneratorsOfRows (Singular macro)

▷ SyzygiesGeneratorsOfRows(*M*) (function)
    Returns:

```
———————————————————— Code ————————————————————
    SyzygiesGeneratorsOfRows := "\n\
  proc SyzygiesGeneratorsOfRows (matrix M)\n\
  {\n\
    return(SyzForHomalg(M));\n\
  }\n\n",
```

3.1.11 SyzygiesGeneratorsOfColumns (in the homalg table for Singular)

▷ SyzygiesGeneratorsOfColumns(*M*) (function)
    Returns:

    This is the entry of the homalg table, which calls the corresponding macro
SyzygiesGeneratorsOfColumns (3.1.12) inside the computer algebra system.

```
———————————————————— Code ————————————————————
  SyzygiesGeneratorsOfColumns :=
    function( M )
      local N;

      N := HomalgVoidMatrix(
        NrColumns( M ),
        "unknown_number_of_columns",
        HomalgRing( M )
      );

      homalgSendBlocking(
        [ "matrix ", N, " = SyzygiesGeneratorsOfColumns(", M, ")" ],
        "need_command",
        "SyzygiesGenerators"
      );

      return N;

    end,
```

3.1.12 SyzygiesGeneratorsOfColumns (Singular macro)

▷ SyzygiesGeneratorsOfColumns(*M*) (function)
    Returns:

```
———————————————————— Code ————————————————————
    SyzygiesGeneratorsOfColumns := "\n\
  proc SyzygiesGeneratorsOfColumns (matrix M)\n\
  {\n\
    return(Involution(SyzForHomalg(Involution(M))));\n\
  }\n\n",
```

3.1.13 BasisOfRowsCoeff (in the homalg table for Singular)

▷ BasisOfRowsCoeff(*M, T*) (function)
  Returns:
   This is the entry of the homalg table, which calls the corresponding macro BasisOfRowsCoeff (3.1.14) inside the computer algebra system.

```
——————————————————————— Code ———————————————————————
  BasisOfRowsCoeff :=
    function( M, T )
      local v, N;

      v := homalgStream( HomalgRing( M ) )!.variable_name;

      N := HomalgVoidMatrix(
        "unknown_number_of_rows",
        NrColumns( M ),
        HomalgRing( M )
      );

      homalgSendBlocking(
        [ "matrix ", N, T, " = BasisOfRowsCoeff(", M, ")" ],
        "need_command",
        "BasisCoeff"
      );

      return N;

    end,
```

3.1.14 BasisOfRowsCoeff (Singular macro)

▷ BasisOfRowsCoeff(*M, T*) (function)
  Returns:

```
——————————————————————— Code ———————————————————————
    BasisOfRowsCoeff := """
  proc BasisOfRowsCoeff (matrix M)
  {
    matrix B = BasisOfRowModule(M);
    option(noredSB);
    matrix T = lift(M,B);
    option(redSB);
    return(B,T);
  }

   """,
```

3.1.15 BasisOfColumnsCoeff (in the homalg table for Singular)

▷ BasisOfColumnsCoeff(*M, T*) (function)
  Returns:

This is the entry of the homalg table, which calls the corresponding macro BasisOfColumnsCoeff (3.1.16) inside the computer algebra system.

```
—————————————————— Code ——————————————————
  BasisOfColumnsCoeff :=
    function( M, T )
      local v, N;

      v := homalgStream( HomalgRing( M ) )!.variable_name;

      N := HomalgVoidMatrix(
        NrRows( M ),
        "unknown_number_of_columns",
        HomalgRing( M )
      );

      homalgSendBlocking(
        [ "matrix ", N, T, " = BasisOfColumnsCoeff(", M, ")" ],
        "need_command",
        "BasisCoeff"
      );

      return N;

    end,
```

3.1.16 BasisOfColumnsCoeff (Singular macro)

▷ BasisOfColumnsCoeff(*M, T*) (function)

   Returns:

```
—————————————————— Code ——————————————————
    BasisOfColumnsCoeff := """
  proc BasisOfColumnsCoeff (matrix M)
  {
    matrix B,T = BasisOfRowsCoeff(Involution(M));
    return(Involution(B),Involution(T));
  }

   """,
```

3.1.17 DecideZeroRowsEffectively (in the homalg table for Singular)

▷ DecideZeroRowsEffectively(*A, B, T*) (function)

   Returns:

This is the entry of the homalg table, which calls the corresponding macro DecideZeroRowsEffectively (3.1.18) inside the computer algebra system. The rows of *B* must form a basis (see BasisOfRowModule (3.1.1)).

```
—————————————————— Code ——————————————————
  DecideZeroRowsEffectively :=
    function( A, B, T )
      local v, N;
```

```
        v := homalgStream( HomalgRing( A ) )!.variable_name;

        N := HomalgVoidMatrix(
          NrRows( A ),
          NrColumns( A ),
          HomalgRing( A )
        );

        homalgSendBlocking(
          [ "matrix ", N, T, " = DecideZeroRowsEffectively(", A, B, ")" ],
          "need_command",
          "DecideZeroEffectively"
        );

        return N;

      end,
```

3.1.18   DecideZeroRowsEffectively (Singular macro)

▷ DecideZeroRowsEffectively(*A, B, T*)   (function)

Returns:

———————————————————— Code ————————————————————

```
    DecideZeroRowsEffectively := """
proc DecideZeroRowsEffectively (matrix A, module B)
{
  attrib(B,"isSB",1);
  matrix M = reduce(A,B);
  matrix T = lift(B,M-A);
  return(M,T);
}

  """,
```

3.1.19   DecideZeroColumnsEffectively (in the homalg table for Singular)

▷ DecideZeroColumnsEffectively(*A, B, T*)   (function)

Returns:

This is the entry of the homalg table, which calls the corresponding macro
DecideZeroColumnsEffectively (3.1.20) inside the computer algebra system. The columns
of *B* must form a basis (see BasisOfColumnModule (3.1.3)).

———————————————————— Code ————————————————————

```
  DecideZeroColumnsEffectively :=
    function( A, B, T )
      local v, N;

      v := homalgStream( HomalgRing( A ) )!.variable_name;

      N := HomalgVoidMatrix(
        NrRows( A ),
        NrColumns( A ),
```

```
      HomalgRing( A )
    );

    homalgSendBlocking(
      [ "matrix ", N, T, " = DecideZeroColumnsEffectively(", A, B, ")" ],
      "need_command",
      "DecideZeroEffectively"
    );

    return N;

  end,
```

3.1.20   DecideZeroColumnsEffectively (Singular macro)

▷ DecideZeroColumnsEffectively(*A, B, T*)   (function)
   Returns:

————————————————— Code ——————————————————
```
    DecideZeroColumnsEffectively := """
proc DecideZeroColumnsEffectively (matrix A, matrix B)
{
  matrix M,T = DecideZeroRowsEffectively(Involution(A),Involution(B));
  return(Involution(M),Involution(T));
}

  """,
```

3.1.21   RelativeSyzygiesGeneratorsOfRows (in the homalg table for Singular)

▷ RelativeSyzygiesGeneratorsOfRows(*M, M2*)   (function)
   Returns:
   This is the entry of the homalg table, which calls the corresponding macro
RelativeSyzygiesGeneratorsOfRows (3.1.22) inside the computer algebra system.

————————————————— Code ——————————————————
```
  RelativeSyzygiesGeneratorsOfRows :=
    function( M, M2 )
      local N;

      N := HomalgVoidMatrix(
        "unknown_number_of_rows",
        NrRows( M ),
        HomalgRing( M )
      );

      homalgSendBlocking(
        [ "matrix ", N, " = RelativeSyzygiesGeneratorsOfRows(", M, M2, ")" ],
        "need_command",
        "SyzygiesGenerators"
      );

      return N;
```

```
    end,
```

3.1.22   RelativeSyzygiesGeneratorsOfRows (Singular macro)

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

   Returns:

```
─────────────────────── Code ───────────────────────
    RelativeSyzygiesGeneratorsOfRows := "\n\
 proc RelativeSyzygiesGeneratorsOfRows (matrix M1, matrix M2)\n\
 {\n\
   return(BasisOfRowModule(modulo(M1, M2)));\n\
 }\n\n",
```

3.1.23   RelativeSyzygiesGeneratorsOfColumns (in the homalg table for Singular)

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

   Returns:

   This is the entry of the homalg table, which calls the corresponding macro RelativeSyzygiesGeneratorsOfColumns (3.1.24) inside the computer algebra system.

```
─────────────────────── Code ───────────────────────
  RelativeSyzygiesGeneratorsOfColumns :=
    function( M, M2 )
      local N;

      N := HomalgVoidMatrix(
        NrColumns( M ),
        "unknown_number_of_columns",
        HomalgRing( M )
      );

      homalgSendBlocking(
        [ "matrix ", N, " = RelativeSyzygiesGeneratorsOfColumns(", M, M2, ")" ],
        "need_command",
        "SyzygiesGenerators"
      );

      return N;

    end,
```

3.1.24   RelativeSyzygiesGeneratorsOfColumns (Singular macro)

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

   Returns:

```
─────────────────────── Code ───────────────────────
    RelativeSyzygiesGeneratorsOfColumns := "\n\
 proc RelativeSyzygiesGeneratorsOfColumns (matrix M1, matrix M2)\n\
 {\n\
```

```
      return(Involution(RelativeSyzygiesGeneratorsOfRows(Involution(M1),Involution(M2))));\n\
  }\n\n",
```

3.1.25 ReducedSyzygiesGeneratorsOfRows (in the homalg table for Singular)

▷ ReducedSyzygiesGeneratorsOfRows(*M*)   (function)

Returns:

 This is the entry of the homalg table, which calls the corresponding macro
ReducedSyzygiesGeneratorsOfRows (3.1.26) inside the computer algebra system.

```
——————————————————————— Code ———————————————————————
  ReducedSyzygiesGeneratorsOfRows :=
    function( M )
      local N;

      N := HomalgVoidMatrix(
        "unknown_number_of_rows",
        NrRows( M ),
        HomalgRing( M )
      );

      homalgSendBlocking(
        [ "matrix ", N, " = ReducedSyzygiesGeneratorsOfRows(", M, ")" ],
        "need_command",
        "SyzygiesGenerators"
      );

      return N;

    end,
```

3.1.26 ReducedSyzygiesGeneratorsOfRows (Singular macro)

▷ ReducedSyzygiesGeneratorsOfRows(*M*)   (function)

Returns:

```
——————————————————————— Code ———————————————————————
      ReducedSyzForHomalg := "\n\
  proc ReducedSyzForHomalg (matrix M)\n\
  {\n\
    return(matrix(nres(M,2)[2]));\n\
  }\n\n",
      ReducedSyzygiesGeneratorsOfRows := "\n\
  proc ReducedSyzygiesGeneratorsOfRows (matrix M)\n\
  {\n\
    return(ReducedSyzForHomalg(M));\n\
  }\n\n",
```

3.1.27 ReducedSyzygiesGeneratorsOfColumns (in the homalg table for Singular)

▷ ReducedSyzygiesGeneratorsOfColumns(*M*)   (function)

Returns:

This is the entry of the homalg table, which calls the corresponding macro `ReducedSyzygiesGeneratorsOfColumns` (3.1.28) inside the computer algebra system.

```
———————————————————————————————— Code ————————————————————————————————
  ReducedSyzygiesGeneratorsOfColumns :=
    function( M )
      local N;

      N := HomalgVoidMatrix(
        NrColumns( M ),
        "unknown_number_of_columns",
        HomalgRing( M )
      );

      homalgSendBlocking(
        [ "matrix ", N, " = ReducedSyzygiesGeneratorsOfColumns(", M, ")" ],
        "need_command",
        "SyzygiesGenerators"
      );

      return N;

    end,
```

3.1.28   ReducedSyzygiesGeneratorsOfColumns (Singular macro)

▷ `ReducedSyzygiesGeneratorsOfColumns(`*M*`)`   (function)

Returns:

```
———————————————————————————————— Code ————————————————————————————————
      ReducedSyzygiesGeneratorsOfColumns := "\n\
  proc ReducedSyzygiesGeneratorsOfColumns (matrix M)\n\
  {\n\
    return(Involution(ReducedSyzForHomalg(Involution(M))));\n\
  }\n\n",
```

References

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

Index