LinearAlgebraFor-CAP

Category of Matrices over a Field for CAP

2024.10-01

24 October 2024

Sebastian Gutsche

Sebastian Posur

Fabian Zickgraf

Sebastian Gutsche
Email: gutsche@mathematik.uni-siegen.de
Homepage: https://sebasguts.github.io/
Address: Department Mathematik
Universität Siegen
Walter-Flex-Straße 3
57068 Siegen
Germany

Sebastian Posur
Email: sebastian.posur@uni-siegen.de
Homepage: https://sebastianpos.github.io
Address: Department Mathematik
Universität Siegen
Walter-Flex-Straße 3
57068 Siegen
Germany

Fabian Zickgraf
Email: fabian.zickgraf@uni-siegen.de
Homepage: https://github.com/zickgraf/
Address: Walter-Flex-Str. 3
57068 Siegen
Germany

Contents

Chapter 1

Category of Matrices

1.1 Constructors

1.1.1 MatrixCategory (for IsFieldForHomalg)

▷ `MatrixCategory(F)` (operation)

Returns: a category

The argument is a homalg field F. The output is the matrix category over F. Objects in this category are non-negative integers. Morphisms from a non-negative integer m to a non-negative integer n are given by $m \times n$ matrices.

1.1.2 VectorSpaceMorphism (for IsVectorSpaceObject, IsHomalgMatrix, IsVectorSpaceObject)

▷ `VectorSpaceMorphism(S, M, R)` (operation)

Returns: a morphism in $\mathrm{Hom}(S,R)$

The arguments are an object S in the category of matrices over a homalg field F, a homalg matrix M over F, and another object R in the category of matrices over F. The output is the morphism $S \to R$ in the category of matrices over F whose underlying matrix is given by M.

1.1.3 VectorSpaceObject (for IsInt, IsFieldForHomalg)

▷ `VectorSpaceObject(d, F)` (operation)

Returns: an object

The arguments are a non-negative integer d and a homalg field F. The output is an object in the category of matrices over F of dimension d. This function delegates to `MatrixCategoryObject`.

1.1.4 MatrixCategoryObject (for IsMatrixCategory, IsInt)

▷ `MatrixCategoryObject(cat, d)` (operation)

Returns: an object

The arguments are a matrix category *cat* over a field and a non-negative integer d. The output is an object in *cat* of dimension d.

1.1.5 MatrixCategory_as_CategoryOfRows (for IsFieldForHomalg)

▷ `MatrixCategory_as_CategoryOfRows(F)` (operation)

Returns: a category

The argument is a homalg field F. The output is the matrix category over F, constructed internally as a wrapper category of the `CategoryOfRows` of F. Only available if the package `FreydCategoriesForCAP` is available.

1.2 Attributes

1.2.1 UnderlyingFieldForHomalg (for IsVectorSpaceMorphism)

▷ `UnderlyingFieldForHomalg(alpha)` (attribute)

Returns: a homalg field

The argument is a morphism α in the matrix category over a homalg field F. The output is the field F.

1.2.2 UnderlyingMatrix (for IsVectorSpaceMorphism)

▷ `UnderlyingMatrix(alpha)` (attribute)

Returns: a homalg matrix

The argument is a morphism α in a matrix category. The output is its underlying matrix M.

1.2.3 UnderlyingFieldForHomalg (for IsVectorSpaceObject)

▷ `UnderlyingFieldForHomalg(A)` (attribute)

Returns: a homalg field

The argument is an object A in the matrix category over a homalg field F. The output is the field F.

1.2.4 Dimension (for IsVectorSpaceObject)

▷ `Dimension(A)` (attribute)

Returns: a non-negative integer

The argument is an object A in a matrix category. The output is the dimension of A.

1.3 GAP Categories

1.3.1 IsVectorSpaceMorphism (for IsCapCategoryMorphism)

▷ `IsVectorSpaceMorphism(object)` (filter)

Returns: `true` or `false`

The GAP category of morphisms in the category of matrices of a field F.

1.3.2 IsVectorSpaceObject (for IsCapCategoryObject)

▷ `IsVectorSpaceObject(object)` (filter)

Returns: `true` or `false`

The GAP category of objects in the category of matrices of a field F.

Chapter 2

Examples and Tests

2.1 Basic Commands

Example
```
gap> LoadPackage( "LinearAlgebraForCAP", false );
true
gap> Q := HomalgFieldOfRationals();;
gap> vec := MatrixCategory( Q );;
gap> a := MatrixCategoryObject( vec, 3 );
<A vector space object over Q of dimension 3>
gap> IsProjective( a );
true
gap> ap := 3/vec;;
gap> IsEqualForObjects( a, ap );
true
gap> b := MatrixCategoryObject( vec, 4 );
<A vector space object over Q of dimension 4>
gap> homalg_matrix := HomalgMatrix( [ [ 1, 0, 0, 0 ],
>                                    [ 0, 1, 0, -1 ],
>                                    [ -1, 0, 2, 1 ] ], 3, 4, Q );;
gap> alpha := VectorSpaceMorphism( a, homalg_matrix, b );
<A morphism in Category of matrices over Q>
```

Example
```
gap> # @drop_example_in_Julia: view/print/display strings of matrices differ between GAP and Juli
> Display( alpha );
[ [   1,   0,   0,   0 ],
  [   0,   1,   0,  -1 ],
  [  -1,   0,   2,   1 ] ]

A morphism in Category of matrices over Q
```

Example
```
gap> alphap := homalg_matrix/vec;;
gap> IsCongruentForMorphisms( alpha, alphap );
true
gap> homalg_matrix := HomalgMatrix( [ [ 1, 1, 0, 0 ],
>                                    [ 0, 1, 0, -1 ],
>                                    [ -1, 0, 2, 1 ] ], 3, 4, Q );;
gap> beta := VectorSpaceMorphism( a, homalg_matrix, b );
<A morphism in Category of matrices over Q>
```

```
gap> CokernelObject( alpha );
<A vector space object over Q of dimension 1>
gap> c := CokernelProjection( alpha );;
gap> Display( EntriesOfHomalgMatrixAsListList( UnderlyingMatrix( c ) ) );
[ [ 0 ], [ 1 ], [ -1/2 ], [ 1 ] ]
gap> gamma := UniversalMorphismIntoDirectSum( [ c, c ] );;
gap> Display( EntriesOfHomalgMatrixAsListList( UnderlyingMatrix( gamma ) ) );
[ [ 0, 0 ], [ 1, 1 ], [ -1/2, -1/2 ], [ 1, 1 ] ]
gap> colift := CokernelColift( alpha, gamma );;
gap> IsEqualForMorphisms( PreCompose( c, colift ), gamma );
true
gap> FiberProduct( alpha, beta );
<A vector space object over Q of dimension 2>
gap> F := FiberProduct( alpha, beta );
<A vector space object over Q of dimension 2>
gap> p1 := ProjectionInFactorOfFiberProduct( [ alpha, beta ], 1 );
<A morphism in Category of matrices over Q>
gap> Display( EntriesOfHomalgMatrixAsListList( UnderlyingMatrix( PreCompose( p1, alpha ) ) ) );
[ [ 0, 1, 0, -1 ], [ -1, 0, 2, 1 ] ]
gap> Pushout( alpha, beta );
<A vector space object over Q of dimension 5>
gap> i1 := InjectionOfCofactorOfPushout( [ alpha, beta ], 1 );
<A morphism in Category of matrices over Q>
gap> i2 := InjectionOfCofactorOfPushout( [ alpha, beta ], 2 );
<A morphism in Category of matrices over Q>
gap> u := UniversalMorphismFromDirectSum( [ b, b ], [ i1, i2 ] );
<A morphism in Category of matrices over Q>
```

Example

```
gap> # @drop_example_in_Julia: differences in the output of SyzygiesOfRows, see https://github.co
> Display( EntriesOfHomalgMatrixAsListList( UnderlyingMatrix( u ) ) );
[ [ 0, 1, 1, 0, 0 ],\
 [ 1, 0, 1, 0, -1 ],\
 [ -1/2, 0, 1/2, 1, 1/2 ],\
 [ 1, 0, 0, 0, 0 ],\
 [ 0, 1, 0, 0, 0 ],\
 [ 0, 0, 1, 0, 0 ],\
 [ 0, 0, 0, 1, 0 ],\
 [ 0, 0, 0, 0, 1 ] ]
```

Example

```
gap> KernelObjectFunctorial( u, IdentityMorphism( Source( u ) ), u ) = IdentityMorphism( MatrixCa
true
gap> IsZeroForMorphisms( CokernelObjectFunctorial( u, IdentityMorphism( Range( u ) ), u ) );
true
gap> DirectProductFunctorial( [ u, u ] ) = DirectSumFunctorial( [ u, u ] );
true
gap> CoproductFunctorial( [ u, u ] ) = DirectSumFunctorial( [ u, u ] );
true
gap> IsCongruentForMorphisms(
>     FiberProductFunctorial( [ u, u ], [ IdentityMorphism( Source( u ) ), IdentityMorphism( Sour
>     IdentityMorphism( FiberProduct( [ u, u ] ) )
> );
true
```

```
gap> IsCongruentForMorphisms(
>     PushoutFunctorial( [ u, u ], [ IdentityMorphism( Range( u ) ), IdentityMorphism( Range( u )
>     IdentityMorphism( Pushout( [ u, u ] ) )
> );
true
gap> IsCongruentForMorphisms( ((1/2) / Q) * alpha, alpha * ((1/2) / Q) );
true
gap> Dimension( HomomorphismStructureOnObjects( a, b ) ) = Dimension( a ) * Dimension( b );
true
gap> IsCongruentForMorphisms(
>     PreCompose( [ u, DualOnMorphisms( i1 ), DualOnMorphisms( alpha ) ] ),
>     InterpretMorphismFromDistinguishedObjectToHomomorphismStructureAsMorphism( Source( u ), Sou
>         PreCompose(
>             InterpretMorphismAsMorphismFromDistinguishedObjectToHomomorphismStructure( DualOnM
>             HomomorphismStructureOnMorphisms( u, DualOnMorphisms( alpha ) )
>         )
>     )
> );
true
gap> op := Opposite( vec );;
gap> alpha_op := Opposite( op, alpha );
<A morphism in Opposite( Category of matrices over Q )>
gap> basis := BasisOfExternalHom( Source( alpha_op ), Range( alpha_op ) );;
gap> coeffs := CoefficientsOfMorphism( alpha_op );;
gap> Display( coeffs );
[ 1, 0, 0, 0, 0, 1, 0, -1, -1, 0, 2, 1 ]
gap> IsEqualForMorphisms( alpha_op, LinearCombinationOfMorphisms( Source( alpha_op ), coeffs, bas
true
gap> vec := CapCategory( alpha );;
gap> t := TensorUnit( vec );;
gap> z := ZeroObject( vec );;
gap> IsCongruentForMorphisms(
>     ZeroObjectFunctorial( vec ),
>     InterpretMorphismFromDistinguishedObjectToHomomorphismStructureAsMorphism( z, z, ZeroMorphi
> );
true
gap> IsCongruentForMorphisms(
>     ZeroObjectFunctorial( vec ),
>     InterpretMorphismFromDistinguishedObjectToHomomorphismStructureAsMorphism(
>         z, z,
>         InterpretMorphismAsMorphismFromDistinguishedObjectToHomomorphismStructure( ZeroObjectFu
>     )
> );
true
gap> right_side := PreCompose( [ i1, DualOnMorphisms( u ), u ] );;
gap> x := SolveLinearSystemInAbCategory( [ [ i1 ] ], [ [ u ] ], [ right_side ] )[1];;
gap> IsCongruentForMorphisms( PreCompose( [ i1, x, u ] ), right_side );
true
gap> a_otimes_b := TensorProductOnObjects( a, b );
<A vector space object over Q of dimension 12>
gap> hom_ab := InternalHomOnObjects( a, b );
<A vector space object over Q of dimension 12>
```

```
gap> cohom_ab := InternalCoHomOnObjects( a, b );
<A vector space object over Q of dimension 12>
gap> hom_ab = cohom_ab;
true
gap> unit_ab := VectorSpaceMorphism(
>           a_otimes_b,
>           HomalgIdentityMatrix( Dimension( a_otimes_b ), Q ),
>           a_otimes_b
>           );
<A morphism in Category of matrices over Q>
gap> unit_hom_ab := VectorSpaceMorphism(
>               hom_ab,
>               HomalgIdentityMatrix( Dimension( hom_ab ), Q ),
>               hom_ab
>             );
<A morphism in Category of matrices over Q>
gap> unit_cohom_ab := VectorSpaceMorphism(
>                 cohom_ab,
>                 HomalgIdentityMatrix( Dimension( cohom_ab ), Q ),
>                 cohom_ab
>               );
<A morphism in Category of matrices over Q>
gap> ev_ab := ClosedMonoidalLeftEvaluationMorphism( a, b );
<A morphism in Category of matrices over Q>
gap> coev_ab := ClosedMonoidalLeftCoevaluationMorphism( a, b );
<A morphism in Category of matrices over Q>
gap> coev_ba := ClosedMonoidalLeftCoevaluationMorphism( b, a );
<A morphism in Category of matrices over Q>
gap> cocl_ev_ab := CoclosedMonoidalLeftEvaluationMorphism( a, b );
<A morphism in Category of matrices over Q>
gap> cocl_ev_ba := CoclosedMonoidalLeftEvaluationMorphism( b, a );
<A morphism in Category of matrices over Q>
gap> cocl_coev_ab := CoclosedMonoidalLeftCoevaluationMorphism( a, b );
<A morphism in Category of matrices over Q>
gap> cocl_coev_ba := CoclosedMonoidalLeftCoevaluationMorphism( b, a );
<A morphism in Category of matrices over Q>
gap> UnderlyingMatrix( ev_ab ) = TransposedMatrix( UnderlyingMatrix( cocl_ev_ab ) );
true
gap> UnderlyingMatrix( coev_ab ) = TransposedMatrix( UnderlyingMatrix( cocl_coev_ab ) );
true
gap> UnderlyingMatrix( coev_ba ) = TransposedMatrix( UnderlyingMatrix( cocl_coev_ba ) );
true
gap> tensor_hom_adj_1_hom_ab := InternalHomToTensorProductLeftAdjunctMorphism( a, b, unit_hom_ab
<A morphism in Category of matrices over Q>
gap> cohom_tensor_adj_1_cohom_ab := InternalCoHomToTensorProductLeftAdjunctMorphism( a, b, unit_c
<A morphism in Category of matrices over Q>
gap> tensor_hom_adj_1_ab := TensorProductToInternalHomLeftAdjunctMorphism( a, b, unit_ab );
<A morphism in Category of matrices over Q>
gap> cohom_tensor_adj_1_ab := TensorProductToInternalCoHomLeftAdjunctMorphism( a, b, unit_ab );
<A morphism in Category of matrices over Q>
gap> ev_ab = tensor_hom_adj_1_hom_ab;
true
```

```
gap> cocl_ev_ba = cohom_tensor_adj_1_cohom_ab;
true
gap> coev_ba = tensor_hom_adj_1_ab;
true
gap> cocl_coev_ba = cohom_tensor_adj_1_ab;
true
gap> c := MatrixCategoryObject( vec, 2 );
<A vector space object over Q of dimension 2>
gap> d := MatrixCategoryObject( vec, 1 );
<A vector space object over Q of dimension 1>
```

Example

```
gap> # @drop_example_in_Julia: MonoidalPreComposeMorphism is very slow because multiplication of
> pre_compose := MonoidalPreComposeMorphism( a, b, c );
<A morphism in Category of matrices over Q>
gap> post_compose := MonoidalPostComposeMorphism( a, b, c );
<A morphism in Category of matrices over Q>
gap> pre_cocompose := MonoidalPreCoComposeMorphism( c, b, a );
<A morphism in Category of matrices over Q>
gap> post_cocompose := MonoidalPostCoComposeMorphism( c, b, a );
<A morphism in Category of matrices over Q>
gap> UnderlyingMatrix( pre_compose ) = TransposedMatrix( UnderlyingMatrix( pre_cocompose ) );
true
gap> UnderlyingMatrix( post_compose ) = TransposedMatrix( UnderlyingMatrix( post_cocompose ) );
true
gap> tp_hom_comp := TensorProductInternalHomCompatibilityMorphism( [ a, b, c, d ] );
<A morphism in Category of matrices over Q>
gap> cohom_tp_comp := InternalCoHomTensorProductCompatibilityMorphism( [ b, d, a, c ] );
<A morphism in Category of matrices over Q>
gap> UnderlyingMatrix( tp_hom_comp ) = TransposedMatrix( UnderlyingMatrix( cohom_tp_comp ) );
true
gap> lambda := LambdaIntroduction( alpha );
<A morphism in Category of matrices over Q>
gap> lambda_elim := LambdaElimination( a, b, lambda );
<A morphism in Category of matrices over Q>
gap> alpha = lambda_elim;
true
gap> alpha_op := VectorSpaceMorphism( b, TransposedMatrix( UnderlyingMatrix( alpha ) ), a );
<A morphism in Category of matrices over Q>
gap> colambda := CoLambdaIntroduction( alpha_op );
<A morphism in Category of matrices over Q>
gap> colambda_elim := CoLambdaElimination( b, a, colambda );
<A morphism in Category of matrices over Q>
gap> alpha_op = colambda_elim;
true
gap> UnderlyingMatrix( lambda ) = TransposedMatrix( UnderlyingMatrix( colambda ) );
true
gap> delta := PreCompose( colambda, lambda);
<A morphism in Category of matrices over Q>
gap> Display( EntriesOfHomalgMatrixAsListList( UnderlyingMatrix( TraceMap( delta ) ) ) );
[ [ 9 ] ]
gap> Display( EntriesOfHomalgMatrixAsListList( UnderlyingMatrix( CoTraceMap( delta ) ) ) );
[ [ 9 ] ]
```

```
gap> TraceMap( delta ) = CoTraceMap( delta );
true
gap> RankMorphism( a ) = CoRankMorphism( a );
true
```

2.2 Functors

Example

```
gap> LoadPackage( "LinearAlgebraForCAP", false );
true
gap> ring := HomalgFieldOfRationals( );;
gap> vec := MatrixCategory( ring );;
gap> F := CapFunctor( "CohomForVec", [ vec, [ vec, true ] ], vec );;
gap> obj_func := function( A, B ) return TensorProductOnObjects( A, DualOnObjects( B ) ); end;;
gap> mor_func := function( source, alpha, beta, range ) return TensorProductOnMorphismsWithGivenT
gap> AddObjectFunction( F, obj_func );;
gap> AddMorphismFunction( F, mor_func );;
gap> Display( InputSignature( F ) );
[ [ Category of matrices over Q, false ], [ Category of matrices over Q, true ] ]
gap> V1 := TensorUnit( vec );;
gap> V3 := DirectSum( V1, V1, V1 );;
gap> pi1 := ProjectionInFactorOfDirectSum( [ V1, V1 ], 1 );;
gap> pi2 := ProjectionInFactorOfDirectSum( [ V3, V1 ], 1 );;
gap> value1 := ApplyFunctor( F, pi1, pi2 );;
gap> input := ProductCategoryMorphism( AsCapCategory( Source( F ) ), [ pi1, Opposite( pi2 ) ] );;
gap> value2 := ApplyFunctor( F, input );;
gap> IsCongruentForMorphisms( value1, value2 );
true
gap> InstallFunctor( F, "F_installation" );;
gap> F_installation( pi1, pi2 );;
gap> F_installation( input );;
gap> F_installationOnObjects( V1, V1 );;
gap> F_installationOnObjects( ProductCategoryObject( AsCapCategory( Source( F ) ), [ V1, Opposite
gap> F_installationOnMorphisms( pi1, pi2 );;
gap> F_installationOnMorphisms( input );;
gap> F2 := CapFunctor( "CohomForVec2", ProductCategory( [ vec, Opposite( vec ) ] ), vec );;
gap> AddObjectFunction( F2, a -> obj_func( a[1], Opposite( a[2] ) ) );;
gap> AddMorphismFunction( F2, function( source, datum, range ) return mor_func( source, datum[1],
gap> input := ProductCategoryMorphism( AsCapCategory( Source( F2 ) ), [ pi1, Opposite( pi2 ) ] );
gap> value3 := ApplyFunctor( F2, input );;
gap> IsCongruentForMorphisms( value1, value3 );
true
gap> Display( InputSignature( F2 ) );
[ [ Product of: Category of matrices over Q, Opposite( Category of matrices over Q ), false ] ]
gap> InstallFunctor( F2, "F_installation2" );;
gap> F_installation2( input );;
gap> F_installation2OnObjects( ProductCategoryObject( AsCapCategory( Source( F2 ) ), [ V1, Opposi
gap> F_installation2OnMorphisms( input );;
```

2.3 Homology object

Example

```
gap> field := HomalgFieldOfRationals( );;
gap> vec := MatrixCategory( field );;
gap> A := MatrixCategoryObject( vec, 1 );;
gap> B := MatrixCategoryObject( vec, 2 );;
gap> C := MatrixCategoryObject( vec, 3 );;
gap> alpha := VectorSpaceMorphism( A, HomalgMatrix( [ [ 1, 0, 0 ] ], 1, 3, field ), C );;
gap> beta := VectorSpaceMorphism( C, HomalgMatrix( [ [ 1, 0 ], [ 1, 1 ], [ 1, 2 ] ], 3, 2, field
gap> IsZeroForMorphisms( PreCompose( alpha, beta ) );
false
gap> IsCongruentForMorphisms(
>     IdentityMorphism( HomologyObject( alpha, beta ) ),
>     HomologyObjectFunctorial( alpha, beta, IdentityMorphism( C ), alpha, beta )
> );
true
gap> kernel_beta := KernelEmbedding( beta );;
gap> K := Source( kernel_beta );;
gap> IsIsomorphism(
>     HomologyObjectFunctorial(
>         MorphismFromZeroObject( K ),
>         MorphismIntoZeroObject( K ),
>         kernel_beta,
>         MorphismFromZeroObject( Source( beta ) ),
>         beta
>     )
> );
true
gap> cokernel_alpha := CokernelProjection( alpha );;
gap> Co := Range( cokernel_alpha );;
gap> IsIsomorphism(
>     HomologyObjectFunctorial(
>         alpha,
>         MorphismIntoZeroObject( Range( alpha ) ),
>         cokernel_alpha,
>         MorphismFromZeroObject( Co ),
>         MorphismIntoZeroObject( Co )
>     )
> );
true
gap> op := Opposite( vec );;
gap> alpha_op := Opposite( op, alpha );;
gap> beta_op := Opposite( op, beta );;
gap> IsCongruentForMorphisms(
>     IdentityMorphism( HomologyObject( beta_op, alpha_op ) ),
>     HomologyObjectFunctorial( beta_op, alpha_op, IdentityMorphism( Opposite( C ) ), beta_op, al
> );
true
gap> kernel_beta := KernelEmbedding( beta_op );;
gap> K := Source( kernel_beta );;
gap> IsIsomorphism(
>     HomologyObjectFunctorial(
```

```
>         MorphismFromZeroObject( K ),
>         MorphismIntoZeroObject( K ),
>         kernel_beta,
>         MorphismFromZeroObject( Source( beta_op ) ),
>         beta_op
>     )
> );
true
gap> cokernel_alpha := CokernelProjection( alpha_op );;
gap> Co := Range( cokernel_alpha );;
gap> IsIsomorphism(
>     HomologyObjectFunctorial(
>         alpha_op,
>         MorphismIntoZeroObject( Range( alpha_op ) ),
>         cokernel_alpha,
>         MorphismFromZeroObject( Co ),
>         MorphismIntoZeroObject( Co )
>     )
> );
true
```

2.4 Liftable

Example

```
gap> field := HomalgFieldOfRationals( );;
gap> vec := MatrixCategory( field );;
gap> V := MatrixCategoryObject( vec, 1 );;
gap> W := MatrixCategoryObject( vec, 2 );;
gap> alpha := VectorSpaceMorphism( V, HomalgMatrix( [ [ 1, -1 ] ], 1, 2, field ), W );;
gap> beta := VectorSpaceMorphism( W, HomalgMatrix( [ [ 1, 2 ], [ 3, 4 ] ], 2, 2, field ), W );;
gap> IsLiftable( alpha, beta );
true
gap> IsLiftable( beta, alpha );
false
gap> IsLiftableAlongMonomorphism( beta, alpha );
true
gap> gamma := VectorSpaceMorphism( W, HomalgMatrix( [ [ 1 ], [ 1 ] ], 2, 1, field ), V );;
gap> IsColiftable( beta, gamma );
true
gap> IsColiftable( gamma, beta );
false
gap> IsColiftableAlongEpimorphism( beta, gamma );
true
gap> PreCompose( PreInverseForMorphisms( gamma ), gamma ) = IdentityMorphism( V );
true
gap> PreCompose( alpha, PostInverseForMorphisms( alpha ) ) = IdentityMorphism( V );
true
```

2.5 Monoidal structure

Example

```
gap> LoadPackage( "LinearAlgebraForCAP", false );
true
gap> Q := HomalgFieldOfRationals();;
gap> vec := MatrixCategory( Q );;
gap> a := MatrixCategoryObject( vec, 1 );
<A vector space object over Q of dimension 1>
gap> b := MatrixCategoryObject( vec, 2 );
<A vector space object over Q of dimension 2>
gap> c := MatrixCategoryObject( vec, 3 );
<A vector space object over Q of dimension 3>
gap> z := ZeroObject( vec );
<A vector space object over Q of dimension 0>
gap> alpha := VectorSpaceMorphism( a, [ [ 1, 0 ] ], b );
<A morphism in Category of matrices over Q>
gap> beta := VectorSpaceMorphism( b,
>                   [ [ 1, 0, 0 ], [ 0, 1, 0 ] ], c );
<A morphism in Category of matrices over Q>
gap> gamma := VectorSpaceMorphism( c,
>                    [ [ 0, 1, 1 ], [ 1, 0, 1 ], [ 1, 1, 0 ] ], c );
<A morphism in Category of matrices over Q>
gap> IsCongruentForMorphisms(
>      TensorProductOnMorphisms( alpha, beta ),
>      TensorProductOnMorphisms( beta, alpha )
> );
false
gap> IsCongruentForMorphisms(
>      AssociatorRightToLeft( a, b, c ),
>      IdentityMorphism( TensorProductOnObjects( a, TensorProductOnObjects( b, c ) ) )
> );
true
gap> IsCongruentForMorphisms(
>      gamma,
>      LambdaElimination( c, c, LambdaIntroduction( gamma ) )
> );
true
gap> IsZeroForMorphisms( TraceMap( gamma ) );
true
gap> IsCongruentForMorphisms(
>      RankMorphism( DirectSum( a, b ) ),
>      RankMorphism( c )
> );
true
gap> IsCongruentForMorphisms(
>      Braiding( b, c ),
>      IdentityMorphism( TensorProductOnObjects( b, c ) )
> );
false
gap> IsCongruentForMorphisms(
>      PreCompose( Braiding( b, c ), Braiding( c, b ) ),
>      IdentityMorphism( TensorProductOnObjects( b, c ) )
> );
```

```
true
```

2.6 MorphismFromSourceToPushout and MorphismFromFiberProductToSink

Example

```
gap> field := HomalgFieldOfRationals( );;
gap> vec := MatrixCategory( field );;
gap> A := MatrixCategoryObject( vec, 3 );;
gap> B := MatrixCategoryObject( vec, 2 );;
gap> alpha := VectorSpaceMorphism( B, HomalgMatrix( [ [ 1, -1, 1 ], [ 1, 1, 1 ] ], 2, 3, field ),
gap> beta := VectorSpaceMorphism( B, HomalgMatrix( [ [ 1, 2, 1 ], [ 2, 1, 1 ] ], 2, 3, field ), A
gap> m := MorphismFromFiberProductToSink( [ alpha, beta ] );;
gap> IsCongruentForMorphisms(
>     m,
>     PreCompose( ProjectionInFactorOfFiberProduct( [ alpha, beta ], 1 ), alpha )
> );
true
gap> IsCongruentForMorphisms(
>     m,
>     PreCompose( ProjectionInFactorOfFiberProduct( [ alpha, beta ], 2 ), beta )
> );
true
gap> IsCongruentForMorphisms(
> MorphismFromKernelObjectToSink( alpha ),
>     PreCompose( KernelEmbedding( alpha ), alpha )
> );
true
gap> alpha_p := DualOnMorphisms( alpha );;
gap> beta_p := DualOnMorphisms( beta );;
gap> m_p := MorphismFromSourceToPushout( [ alpha_p, beta_p ] );;
gap> IsCongruentForMorphisms(
>     m_p,
>     PreCompose( alpha_p, InjectionOfCofactorOfPushout( [ alpha_p, beta_p ], 1 ) )
> );
true
gap> IsCongruentForMorphisms(
>     m_p,
>     PreCompose( beta_p, InjectionOfCofactorOfPushout( [ alpha_p, beta_p ], 2 ) )
> );
true
gap> IsCongruentForMorphisms(
>     MorphismFromSourceToCokernelObject( alpha_p ),
>     PreCompose( alpha_p, CokernelProjection( alpha_p ) )
> );
true
```

2.7 Opposite category

Example

```
gap> LoadPackage( "LinearAlgebraForCAP", ">= 2024.01-04", false );
true
gap> QQ := HomalgFieldOfRationals();;
gap> vec := MatrixCategory( QQ );;
gap> op := Opposite( vec );;
gap> Display( ListKnownCategoricalProperties( op ) );
[ "IsAbCategory", "IsAbelianCategory", "IsAbelianCategoryWithEnoughInjectives",\
 "IsAbelianCategoryWithEnoughProjectives", "IsAdditiveCategory",\
 "IsBraidedMonoidalCategory", "IsCategoryWithInitialObject",\
 "IsCategoryWithTerminalObject", "IsCategoryWithZeroObject",\
 "IsClosedMonoidalCategory", "IsCoclosedMonoidalCategory",\
 "IsEnrichedOverCommutativeRegularSemigroup",\
 "IsEquippedWithHomomorphismStructure", "IsLinearCategoryOverCommutativeRing",\
 "IsLinearCategoryOverCommutativeRingWithFinitelyGeneratedFreeExternalHoms",\
 "IsMonoidalCategory", "IsPreAbelianCategory",\
 "IsRigidSymmetricClosedMonoidalCategory",\
 "IsRigidSymmetricCoclosedMonoidalCategory", "IsSkeletalCategory",\
 "IsStrictMonoidalCategory", "IsSymmetricClosedMonoidalCategory",\
 "IsSymmetricCoclosedMonoidalCategory", "IsSymmetricMonoidalCategory" ]
gap> V1 := Opposite( TensorUnit( vec ) );;
gap> V2 := DirectSum( V1, V1 );;
gap> V3 := DirectSum( V1, V2 );;
gap> V4 := DirectSum( V1, V3 );;
gap> V5 := DirectSum( V1, V4 );;
gap> IsWellDefined( MorphismBetweenDirectSums( op, [ ], [ ], [ V1 ] ) );
true
gap> IsWellDefined( MorphismBetweenDirectSums( op, [ V1 ], [ [ ] ], [ ] ) );
true
gap> alpha13 := InjectionOfCofactorOfDirectSum( [ V1, V2 ], 1 );;
gap> alpha14 := InjectionOfCofactorOfDirectSum( [ V1, V2, V1 ], 3 );;
gap> alpha15 := InjectionOfCofactorOfDirectSum( [ V2, V1, V2 ], 2 );;
gap> alpha23 := InjectionOfCofactorOfDirectSum( [ V2, V1 ], 1 );;
gap> alpha24 := InjectionOfCofactorOfDirectSum( [ V1, V2, V1 ], 2 );;
gap> alpha25 := InjectionOfCofactorOfDirectSum( [ V2, V2, V1 ], 1 );;
gap> mat := [
>     [ alpha13, alpha14, alpha15 ],
>     [ alpha23, alpha24, alpha25 ]
> ];;
gap> mor := MorphismBetweenDirectSums( mat );;
gap> IsWellDefined( mor );
true
gap> IsWellDefined( Opposite( mor ) );
true
gap> IsCongruentForMorphisms(
>     UniversalMorphismFromImage( mor, [ CoastrictionToImage( mor ), ImageEmbedding( mor ) ] ),
>     IdentityMorphism( ImageObject( mor ) )
> );
true
```

2.8 PreComposeList and PostComposeList

Example

```
gap> field := HomalgFieldOfRationals( );;
gap> vec := MatrixCategory( field );;
gap> A := MatrixCategoryObject( vec, 1 );;
gap> B := MatrixCategoryObject( vec, 2 );;
gap> C := MatrixCategoryObject( vec, 3 );;
gap> alpha := VectorSpaceMorphism( A, HomalgMatrix( [ [ 1, 0, 0 ] ], 1, 3, field ), C );;
gap> beta := VectorSpaceMorphism( C, HomalgMatrix( [ [ 1, 0 ], [ 1, 1 ], [ 1, 2 ] ], 3, 2, field
gap> IsCongruentForMorphisms( PreCompose( alpha, beta ), PostCompose( beta, alpha ) );
true
gap> IsCongruentForMorphisms( PreComposeList( A, [ ], A ), IdentityMorphism( A ) );
true
gap> IsCongruentForMorphisms( PreComposeList( A, [ alpha ], C ), alpha );
true
gap> IsCongruentForMorphisms( PreComposeList( A, [ alpha, beta ], B ), PreCompose( alpha, beta )
true
gap> IsCongruentForMorphisms( PostComposeList( A, [ ], A ), IdentityMorphism( A ) );
true
gap> IsCongruentForMorphisms( PostComposeList( A, [ alpha ], C ), alpha );
true
gap> IsCongruentForMorphisms( PostComposeList( A, [ beta, alpha ], B ), PostCompose( beta, alpha
true
```

2.9 Split epi summand

Example

```
gap> LoadPackage( "LinearAlgebraForCAP", false );
true
gap> Q := HomalgFieldOfRationals();;
gap> Qmat := MatrixCategory( Q );;
gap> a := MatrixCategoryObject( Qmat, 3 );;
gap> b := MatrixCategoryObject( Qmat, 4 );;
gap> homalg_matrix := HomalgMatrix( [ [ 1, 0, 0, 0 ],
>                                     [ 0, 1, 0, -1 ],
>                                     [ -1, 0, 2, 1 ] ], 3, 4, Q );;
gap> alpha := VectorSpaceMorphism( a, homalg_matrix, b );;
gap> beta := SomeReductionBySplitEpiSummand( alpha );;
gap> IsWellDefinedForMorphisms( beta );
true
gap> Dimension( Source( beta ) );
0
gap> Dimension( Range( beta ) );
1
gap> gamma := SomeReductionBySplitEpiSummand_MorphismFromInputRange( alpha );;
gap> Display( EntriesOfHomalgMatrixAsListList( UnderlyingMatrix( gamma ) ) );
[ [ 0 ], [ 1 ], [ -1/2 ], [ 1 ] ]
```

Example

```
gap> # @drop_example_in_Julia: differences in the output of (Safe)RightDivide, see https://githu
> delta := SomeReductionBySplitEpiSummand_MorphismToInputRange( alpha );;
gap> Display( EntriesOfHomalgMatrixAsListList( UnderlyingMatrix( delta ) ) );
```

```
[ [ 0, 1, 0, 0 ] ]
```

2.10 Kernel

Example

```
gap> Q := HomalgFieldOfRationals();;
gap> vec := MatrixCategory( Q );;
gap> V := MatrixCategoryObject( vec, 2 );
<A vector space object over Q of dimension 2>
gap> W := MatrixCategoryObject( vec, 3 );
<A vector space object over Q of dimension 3>
gap> alpha := VectorSpaceMorphism( V, [ [ 1, 1, 1 ], [ -1, -1, -1 ] ], W );
<A morphism in Category of matrices over Q>
gap> k := KernelObject( alpha );
<A vector space object over Q of dimension 1>
gap> T := MatrixCategoryObject( vec, 2 );
<A vector space object over Q of dimension 2>
gap> tau := VectorSpaceMorphism( T, [ [ 2, 2 ], [ 2, 2 ] ], V );
<A morphism in Category of matrices over Q>
gap> k_lift := KernelLift( alpha, tau );
<A morphism in Category of matrices over Q>
gap> HasKernelEmbedding( alpha );
false
gap> KernelEmbedding( alpha );
<A split monomorphism in Category of matrices over Q>
```

Example

```
gap> Q := HomalgFieldOfRationals();;
gap> vec := MatrixCategory( Q );;
gap> V := MatrixCategoryObject( vec, 2 );
<A vector space object over Q of dimension 2>
gap> W := MatrixCategoryObject( vec, 3 );
<A vector space object over Q of dimension 3>
gap> alpha := VectorSpaceMorphism( V, [ [ 1, 1, 1 ], [ -1, -1, -1 ] ], W );
<A morphism in Category of matrices over Q>
gap> k := KernelObject( alpha );
<A vector space object over Q of dimension 1>
gap> T := MatrixCategoryObject( vec, 2 );
<A vector space object over Q of dimension 2>
gap> tau := VectorSpaceMorphism( T, [ [ 2, 2 ], [ 2, 2 ] ], V );
<A morphism in Category of matrices over Q>
gap> k_lift := KernelLift( alpha, tau );
<A morphism in Category of matrices over Q>
gap> HasKernelEmbedding( alpha );
false
```

Example

```
gap> Q := HomalgFieldOfRationals();;
gap> vec := MatrixCategory( Q );;
gap> V := MatrixCategoryObject( vec, 2 );
<A vector space object over Q of dimension 2>
gap> W := MatrixCategoryObject( vec, 3 );
<A vector space object over Q of dimension 3>
```

```
gap> alpha := VectorSpaceMorphism( V, [ [ 1, 1, 1 ], [ -1, -1, -1 ] ], W );
<A morphism in Category of matrices over Q>
gap> k := KernelObject( alpha );
<A vector space object over Q of dimension 1>
gap> k_emb := KernelEmbedding( alpha );
<A split monomorphism in Category of matrices over Q>
gap> IsEqualForObjects( Source( k_emb ), k );
true
gap> V := MatrixCategoryObject( vec, 2 );
<A vector space object over Q of dimension 2>
gap> W := MatrixCategoryObject( vec, 3 );
<A vector space object over Q of dimension 3>
gap> beta := VectorSpaceMorphism( V, [ [ 1, 1, 1 ], [ -1, -1, -1 ] ], W );
<A morphism in Category of matrices over Q>
gap> k_emb := KernelEmbedding( beta );
<A split monomorphism in Category of matrices over Q>
gap> IsIdenticalObj( Source( k_emb ), KernelObject( beta ) );
true
```

2.11 FiberProduct

Example

```
gap> Q := HomalgFieldOfRationals();;
gap> vec := MatrixCategory( Q );;
gap> A := MatrixCategoryObject( vec, 1 );
<A vector space object over Q of dimension 1>
gap> B := MatrixCategoryObject( vec, 2 );
<A vector space object over Q of dimension 2>
gap> C := MatrixCategoryObject( vec, 3 );
<A vector space object over Q of dimension 3>
gap> AtoC := VectorSpaceMorphism( A, [ [ 1, 2, 0 ] ], C );
<A morphism in Category of matrices over Q>
gap> BtoC := VectorSpaceMorphism( B, [ [ 1, 0, 0 ], [ 0, 1, 0 ] ], C );
<A morphism in Category of matrices over Q>
gap> P := FiberProduct( AtoC, BtoC );
<A vector space object over Q of dimension 1>
gap> p1 := ProjectionInFactorOfFiberProduct( [ AtoC, BtoC ], 1 );
<A morphism in Category of matrices over Q>
gap> p2 := ProjectionInFactorOfFiberProduct( [ AtoC, BtoC ], 2 );
<A morphism in Category of matrices over Q>
```

2.12 WrapperCategory

Example

```
gap> LoadPackage( "LinearAlgebraForCAP", false );
true
gap> Q := HomalgFieldOfRationals( );;
gap> Qmat := MatrixCategory( Q );
Category of matrices over Q
gap> Wrapper := WrapperCategory( Qmat, rec( ) );
WrapperCategory( Category of matrices over Q )
```

```
gap> mor := ZeroMorphism( ZeroObject( Wrapper ), ZeroObject( Wrapper ) );;
gap> (2 / Q) * mor;;
gap> BasisOfExternalHom( Source( mor ), Range( mor ) );;
gap> CoefficientsOfMorphism( mor );;
gap> distinguished_object := DistinguishedObjectOfHomomorphismStructure( Wrapper );;
gap> object := HomomorphismStructureOnObjects( Source( mor ), Source( mor ) );;
gap> HomomorphismStructureOnMorphisms( mor, mor );;
gap> HomomorphismStructureOnMorphismsWithGivenObjects( object, mor, mor, object );;
gap> iota := InterpretMorphismAsMorphismFromDistinguishedObjectToHomomorphismStructure( mor );;
gap> InterpretMorphismAsMorphismFromDistinguishedObjectToHomomorphismStructureWithGivenObjects( d
gap> beta := InterpretMorphismFromDistinguishedObjectToHomomorphismStructureAsMorphism( Source( m
gap> IsCongruentForMorphisms( mor, beta );
true
```

Index