

Linear Algebra For- CAP

**Category of Matrices over a Field for
CAP**

2017.07.25

25 July 2017

Sebastian Gutsche

Sebastian Posur

Sebastian Gutsche

Email: gutsche@mathematik.uni-siegen.de

Homepage: <http://www.uni-siegen.de/fb6/rmi/>

Address: Department Mathematik
Universität Siegen
Walter-Flex-Straße 3
57068 Siegen
Germany

Sebastian Posur

Email: sebastian.posur@uni-siegen.de

Homepage: <http://www.uni-siegen.de/fb6/rmi/>

Address: Department Mathematik
Universität Siegen
Walter-Flex-Straße 3
57068 Siegen
Germany

Contents

1	Category of Matrices	3
1.1	Constructors	3
1.2	GAP Categories	3
1.3	Attributes	4
2	Examples and Tests	5
2.1	Basic Commands	5
	Index	7

Chapter 1

Category of Matrices

1.1 Constructors

1.1.1 MatrixCategory (for IsFieldForHomalg)

▷ MatrixCategory(F) (attribute)

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 $\text{Hom}(S, R)$

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 $S \rightarrow R$ 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 .

1.2 GAP Categories

1.2.1 IsVectorSpaceMorphism (for IsCapCategoryMorphism and IsCellOfSkeletalCategory)

▷ IsVectorSpaceMorphism($object$) (filter)

Returns: true or false

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

1.2.2 IsVectorSpaceObject (for IsCapCategoryObject and IsCellOfSkeletalCategory)

▷ IsVectorSpaceObject(*object*) (filter)

Returns: true or false

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

1.3 Attributes

1.3.1 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 .

1.3.2 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 .

1.3.3 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 .

1.3.4 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 .

Chapter 2

Examples and Tests

2.1 Basic Commands

Example

```
gap> Q := HomalgFieldOfRationals();;
gap> a := VectorSpaceObject( 3, Q );
<A vector space object over Q of dimension 3>
gap> b := VectorSpaceObject( 4, Q );
<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 );
<A 3 x 4 matrix over an internal ring>
gap> alpha := VectorSpaceMorphism( a, homalg_matrix, b );
<A morphism in Category of matrices over Q>
gap> Display( alpha );
[ [ 1, 0, 0, 0 ],
  [ 0, 1, 0, -1 ],
  [ -1, 0, 2, 1 ] ]

A morphism in Category of matrices over Q
gap> homalg_matrix := HomalgMatrix( [ [ 1, 1, 0, 0 ],
>                                     [ 0, 1, 0, -1 ],
>                                     [ -1, 0, 2, 1 ] ], 3, 4, Q );
<A 3 x 4 matrix over an internal ring>
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( c );
[ [ 0 ],
  [ 1 ],
  [ -1/2 ],
  [ 1 ] ]

A split epimorphism in Category of matrices over Q
gap> gamma := UniversalMorphismIntoDirectSum( [ c, c ] );;
gap> Display( gamma );
```

```
[ [ 0, 0 ],
  [ 1, 1 ],
  [ -1/2, -1/2 ],
  [ 1, 1 ] ]
```

A morphism in Category of matrices over Q

```
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( PreCompose( p1, alpha ) );
[ [ 0, 1, 0, -1 ],
  [ -1, 0, 2, 1 ] ]
```

A morphism in Category of matrices over Q

```
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>
gap> Display( 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 ] ]
```

A morphism in Category of matrices over Q

```
gap> KernelObjectFunctorial( u, IdentityMorphism( Source( u ) ), u ) = IdentityMorphism( VectorSpace( u ) );
true
gap> IsZero( CokernelObjectFunctorial( u, IdentityMorphism( Range( u ) ), u ) );
true
gap> DirectProductFunctorial( [ u, u ] ) = DirectSumFunctorial( [ u, u ] );
true
gap> CoproductFunctorial( [ u, u ] ) = DirectSumFunctorial( [ u, u ] );
true
gap> IsOne( FiberProductFunctorial( [ [ u, IdentityMorphism( Source( u ) ) ], u ], [ u, IdentityMorphism( Source( u ) ) ] ) );
true
gap> IsOne( PushoutFunctorial( [ [ u, IdentityMorphism( Range( u ) ) ], u ], [ u, IdentityMorphism( Range( u ) ) ] ) );
true
```

Index

Dimension

for `IsVectorSpaceObject`, 4

`IsVectorSpaceMorphism`

for `IsCapCategoryMorphism` and `IsCellOfSkeletalCategory`, 3

`IsVectorSpaceObject`

for `IsCapCategoryObject` and `IsCellOfSkeletalCategory`, 4

`MatrixCategory`

for `IsFieldForHomalg`, 3

`UnderlyingFieldForHomalg`

for `IsVectorSpaceMorphism`, 4

for `IsVectorSpaceObject`, 4

`UnderlyingMatrix`

for `IsVectorSpaceMorphism`, 4

`VectorSpaceMorphism`

for `IsVectorSpaceObject`, `IsHomalgMatrix`,
`IsVectorSpaceObject`, 3

`VectorSpaceObject`

for `IsInt`, `IsFieldForHomalg`, 3