

XModAlg

Crossed Modules and Cat1-Algebras

1.32

11 April 2025

Zekeriya Arvasi

Alper Odabas

Zekeriya Arvasi

Email: zarvasi@ogu.edu.tr

Address: Prof. Dr. Z. Arvasi

Osmangazi University

Arts and Sciences Faculty

Department of Mathematics and Computer Science

Eskisehir

Turkey

Alper Odabas

Email: aodabas@ogu.edu.tr

Address: Dr. A. Odabas

Osmangazi University

Arts and Sciences Faculty

Department of Mathematics and Computer Science

Eskisehir

Turkey

Abstract

The XModAlg package provides functions for computation with crossed modules of commutative algebras and cat^1 -algebras.

Bug reports, suggestions and comments are, of course, welcome. Please submit an issue on GitHub at <https://github.com/gap-packages/xmodalg/issues/> or contact the second author at aodabas@ogu.edu.tr.

Copyright

© 2014–2025, Zekeriya Arvasi and Alper Odabas.

The XModAlg package is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

Acknowledgements

This documentation was prepared with the GAPDoc [LN17] and AutoDoc [GH17] packages.

The procedure used to produce new releases uses the package GitHubPagesForGAP [Hor17] and the package ReleaseTools.

Both authors are very grateful to Chris Wensley (<https://github.com/cdwensley>) for helpful suggestions.

This work was partially supported by TÜBİTAK (The Scientific and Technical Research Council of Turkey), project number 107T542.

Contents

1	Introduction	4
2	Algebras and their Actions	5
2.1	Multipliers	5
2.2	Commutative actions	7
2.3	Algebra modules	10
2.4	Actions on direct sums of algebras	12
2.5	Other operations on algebras	13
2.6	Lists of algebra homomorphisms	15
3	Cat1-algebras	17
3.1	Definitions and examples	17
3.2	Cat ¹ -algebra morphisms	22
4	Crossed modules	26
4.1	Definition and Examples	26
4.2	(Pre-)Crossed Module Morphisms	31
5	Conversion between cat1-algebras and crossed modules	35
5.1	Equivalent Categories	35
	References	39
	Index	40

Chapter 1

Introduction

In 1950 S. MacLane and J.H.C. Whitehead, [Whi49] suggested that crossed modules modeled homotopy 2-types. Later crossed modules have been considered as *2-dimensional groups*, [Bro82], [Bro87]. The commutative algebra version of this construction has been adapted by T. Porter, [AP96], [Por87]. This algebraic version is called *combinatorial algebra theory*, which contains potentially important new ideas (see [Sha92], [AP96], [AP98], [AE03]).

A share package XMod, [AOUW17], [AW00], was prepared by M. Alp and C.D. Wensley for the GAP computational group theory language, initially for GAP3 then revised for GAP4. The 2-dimensional part of this programme contains functions for computing crossed modules and cat^1 -groups and their morphisms [AOUW17].

This package includes functions for computing crossed modules of algebras, cat^1 -algebras and their morphisms by analogy with *computational group theory*. We will concentrate on group rings over abelian groups over finite fields because these algebras are conveniently implemented in GAP. The tools needed are the group algebras in which the group algebra functor $\mathcal{K}(\cdot) : Gr \rightarrow Alg$ is left adjoint to the unit group functor $\mathcal{U}(\cdot) : Alg \rightarrow Gr$.

The categories XModAlg (crossed modules of algebras) and Cat1Alg (cat^1 -algebras) are equivalent, and we include functions to convert objects and morphisms between them. The algorithms implemented in this package are analyzed in A. Odabas's Ph.D. thesis, [Oda09] and described in detail in the paper [AO16].

There are aspects of commutative algebras for which no GAP functions yet exist, for example semidirect products. We have included here functions for all homomorphisms of algebras.

Chapter 2

Algebras and their Actions

All the algebras considered in this package will be associative and commutative. Scalars belong to a commutative ring K with $1 \neq 0$.

(Why not a field? A group ring over the integers is not an algebra. [CDW])

2.1 Multipliers

A *multiplier* in a commutative algebra A is a function $\mu : A \rightarrow A$ such that

$$\mu(ab) = (\mu a)b = a(\mu b) \quad \forall a, b \in A.$$

The *regular multipliers* of A are the functions

$$\mu_a : A \rightarrow A : \mu_a b = ab \quad \forall b \in A.$$

When A has a one, it follows from the defining condition that $\mu(b1) = (\mu 1)b$ and so $\mu = \mu_a$ where $a = \mu 1$. Since an ideal I of A is closed under multiplication, a multiplier μ may be restricted to I .

QUESTION: Is there an example of an algebra A *without* a one which has multipliers *not* of the form μ_a ?

2.1.1 RegularAlgebraMultiplier

▷ RegularAlgebraMultiplier(A, I, a) (operation)

This operation defines the multiplier $\mu_a : I \rightarrow I$ on an ideal I of A .

Example

```
gap> A1 := GroupRing( GF(5), Group( (1,2,3,4,5,6) ) );
gap> SetName( A1, "A1" );
gap> BA1 := BasisVectors( Basis( A1 ) );
gap> v := BA1[1] + BA1[3] + BA1[5];
(Z(5)^0)*()+(Z(5)^0)*(1,3,5)(2,4,6)+(Z(5)^0)*(1,5,3)(2,6,4)
gap> I1 := Ideal( A1, [v] );
gap> SetName( I1, "I1" );
gap> v1 := BA1[2];
(Z(5)^0)*(1,2,3,4,5,6)
```

```
gap> m1 := RegularAlgebraMultiplier( A1, I1, v1 );
[ (Z(5)^0)*()+ (Z(5)^0)*(1,3,5)(2,4,6)+ (Z(5)^0)*(1,5,3)(2,6,4),
  (Z(5)^0)*(1,2,3,4,5,6)+ (Z(5)^0)*(1,4)(2,5)(3,6)+ (Z(5)^0)*(1,6,5,4,3,2) ] ->
[ (Z(5)^0)*(1,2,3,4,5,6)+ (Z(5)^0)*(1,4)(2,5)(3,6)+ (Z(5)^0)*(1,6,5,4,3,2),
  (Z(5)^0)*()+ (Z(5)^0)*(1,3,5)(2,4,6)+ (Z(5)^0)*(1,5,3)(2,6,4) ]
```

2.1.2 IsAlgebraMultiplier

▷ IsAlgebraMultiplier(*mu*)

(operation)

This function tests the condition $\mu(ab) = (\mu a)b = a(\mu b)$ for all a, b in the basis for A .

Example

```
gap> IsAlgebraMultiplier( m1 );
true
gap> id1 := One( A1 );;
gap> L1 := List( BA1, v -> id1 );;
gap> h1 := LeftModuleHomomorphismByImages( A1, A1, BA1, L1 );
[ (Z(5)^0)*(), (Z(5)^0)*(1,2,3,4,5,6), (Z(5)^0)*(1,3,5)(2,4,6),
  (Z(5)^0)*(1,4)(2,5)(3,6), (Z(5)^0)*(1,5,3)(2,6,4), (Z(5)^0)*(1,6,5,4,3,2)
] -> [ (Z(5)^0)*(), (Z(5)^0)*(), (Z(5)^0)*(), (Z(5)^0)*(), (Z(5)^0)*(),
  (Z(5)^0)*() ]
gap> IsAlgebraMultiplier( h1 );
false
```

2.1.3 MultiplierAlgebraOfIdealBySubalgebra

▷ MultiplierAlgebraOfIdealBySubalgebra (*A*, *I*, *B*)

(operation)

The regular multipliers $\mu_b : I \rightarrow I$ for all $b \in B$, where I is an ideal in A and B is a subalgebra of A , form an algebra with product $\mu_b \circ \mu_{b'} = \mu_{bb'}$.

Example

```
gap> u1 := BA1[3];
(Z(5)^0)*(1,3,5)(2,4,6)
gap> S1 := Subalgebra( A3, [ u1 ] );;
gap> SetName( S1, "S1" );
gap> MS1 := MultiplierAlgebraOfIdealBySubalgebra( A1, I1, S1 );
<algebra of dimension 1 over GF(5)>
gap> SetName( MS1, "MS1" );
gap> BMS1 := BasisVectors( Basis( MS1 ) );;
gap> BMS1[1];
<linear mapping by matrix, I1 -> I1>
```

2.1.4 MultiplierAlgebra

▷ `MultiplierAlgebra(A)` (attribute)

The regular multipliers $\mu_a : A \rightarrow A$ for all $a \in A$ form an algebra isomorphic to A by the map $a \mapsto \mu_a$. This operation returns `MultiplierAlgebraOfIdealBySubalgebra(A,A,A);`.

Example

```
gap> MA1 := MultiplierAlgebra( A1 );
<algebra of dimension 6 over GF(5)>
gap> BMA1 := BasisVectors( Basis( MA1 ) );
gap> BMA1[3];
<linear mapping by matrix, <algebra-with-one of dimension
6 over GF(5)> -> <algebra-with-one of dimension 6 over GF(5)>>
```

2.1.5 MultiplierHomomorphism

▷ `MultiplierHomomorphism(M)` (attribute)

If M is a multiplier algebra with elements of a subalgebra B of an algebra A multiplying an ideal I then this operation returns the homomorphism from B to M mapping b to μ_b .

Example

```
gap> hom1 := MultiplierHomomorphism( MA1 );
gap> ImageElm( hom1, BA1[2] );
Basis( A1, [ (Z(5)^0)*(), (Z(5)^0)*(1,2,3,4,5,6), (Z(5)^0)*(1,3,5)(2\
,4,6),
(Z(5)^0)*(1,4)(2,5)(3,6), (Z(5)^0)*(1,5,3)(2,6,4), (Z(5)^0)*(1,6,5,4,3,2)
] ) -> [ (Z(5)^0)*(1,2,3,4,5,6), (Z(5)^0)*(1,3,5)(2,4,6),
(Z(5)^0)*(1,4)(2,5)(3,6), (Z(5)^0)*(1,5,3)(2,6,4), (Z(5)^0)*(1,6,5,4,3,2),
(Z(5)^0)*() ]
```

2.2 Commutative actions

If S and R are commutative K -algebras, a map

$$R \times S \rightarrow S, \quad (r, s) \mapsto r \cdot s$$

is a commutative action if and only if the following five axioms hold:

- $k(r \cdot s) = (kr) \cdot s = r \cdot (ks)$,
- $r \cdot (s + s') = r \cdot s + r \cdot s'$, (so $r \cdot 0_S = 0_S \forall r \in R$),
- $(r + r') \cdot s = r \cdot s + r' \cdot s$, (so $0_R \cdot s = 0_S \forall s \in S$),
- $r \cdot (ss') = (r \cdot s)s' = s(r \cdot s')$,

$$\bullet (rr') \cdot s = r \cdot (r' \cdot s), \quad (\text{so } 1_R \cdot s = s \forall s \in S \text{ when } R \text{ has a one}),$$

for all $k \in K$, $r, r' \in R$, and $s, s' \in S$.

Notice in particular that, for fixed $r \in R$, the map $s \mapsto r \cdot s$ is a vector space homomorphism, but not in general an algebra homomorphism.

2.2.1 AlgebraAction

▷ `AlgebraAction(args)` (function)

This global function calls one of the following operations, depending on the arguments supplied.

2.2.2 AlgebraActionByMultipliers

▷ `AlgebraActionByMultipliers(A, I, B)` (operation)

When I is an ideal in A and B is a subalgebra of A , we have seen that the multiplier homomorphism from A to `MultiplierAlgebraOfIdealBySubalgebra(A, I, B)` is an action.

In the example the algebra is the group ring of the cyclic group C_6 over the field $GF(5)$. The ideal is generated by $v = () + (1, 3, 5)(2, 4, 6) + (1, 5, 3)(2, 6, 4)$. The generator $r = (1, 2, 3, 4, 5, 6)$ acts on v by multiplication to give the vector $r \cdot v = (1, 2, 3, 4, 5, 6) + (1, 4)(2, 5)(3, 6) + (1, 6, 5, 4, 3, 2)$, as shown in `AlgebraActionByHomomorphism` (2.2.4)

Example

```
gap> A1 := GroupRing( GF(5), Group( (1,2,3,4,5,6) ) );;
gap> BA1 := BasisVectors( Basis( A1 ) );;
gap> v := BA1[1] + BA1[3] + BA1[5];
(Z(5)^0)*()+(Z(5)^0)*(1,3,5)(2,4,6)+(Z(5)^0)*(1,5,3)(2,6,4)
gap> I1 := Ideal( A1, [v] );;
gap> act1 := AlgebraActionByMultipliers( A1, I1, A1 );;
gap> act12 := Image( act1, BA1[2] );;
gap> Image( act12, v );
(Z(5)^0)*(1,2,3,4,5,6)+(Z(5)^0)*(1,4)(2,5)(3,6)+(Z(5)^0)*(1,6,5,4,3,2)
```

2.2.3 AlgebraActionBySurjection

▷ `AlgebraActionBySurjection(hom)` (operation)

Let $\theta : B \rightarrow A$ be a surjective algebra homomorphism such that $\ker \theta$ is contained in the annihilator of B . Then A acts on B by $a \cdot b = pb$ where $p \in (\theta^{-1}a)$. Note that this action is well defined since $\theta^{-1}a = \{p + k \mid k \in \ker \theta\}$ and $(p + k)b = pb + kb = pb + 0$.

Continuing with the previous example, we construct the quotient algebra $Q1 = A1/I1$, and the natural homomorphism $\theta_1 : A1 \rightarrow Q1$. The kernel of θ is not contained in the annihilator of $A1$, so an attempt to form the action fails.

An alternative example involves a matrix algebra A_2 with generator m_2 , basis $\{m_2, m_2^2, m_2^3\}$, and where $m_2^4 = 0$. The ideal I_2 is generated by m_2^3 and the quotient Q_2 has basis $\{[m_2], [m_2^2]\}$.

Example

```

gap> theta1 := NaturalHomomorphismByIdeal( A1, I1 );
<linear mapping by matrix, <algebra-with-one of dimension
6 over GF(5)> -> <algebra of dimension 4 over GF(5)>>
gap> List( BA1, v -> ImageElm( theta1, v ) );
[ v.1, v.2, v.3, v.4, (Z(5)^2)*v.1+(Z(5)^2)*v.3, (Z(5)^2)*v.2+(Z(5)^2)*v.4 ]
gap> AlgebraActionBySurjection( theta1 );
kernel of hom is not in the annihilator of A
fail
gap> ## an example which does not fail:
gap> m2 := [ [0,1,2,3], [0,0,1,2], [0,0,0,1], [0,0,0,0] ];;
gap> m2^2;
[ [ 0, 0, 1, 4 ], [ 0, 0, 0, 1 ], [ 0, 0, 0, 0 ], [ 0, 0, 0, 0 ] ]
gap> m2^3;
[ [ 0, 0, 0, 1 ], [ 0, 0, 0, 0 ], [ 0, 0, 0, 0 ], [ 0, 0, 0, 0 ] ]
gap> A2 := Algebra( Rationals, [m2] );;
gap> SetName( A2, "A2" );
gap> S2 := Subalgebra( A2, [m2^3] );;
gap> SetName( S2, "S2" );
gap> nat2 := NaturalHomomorphismByIdeal( A2, S2 );
<linear mapping by matrix, A2 -> <algebra of dimension 2 over Ration\
als>>
gap> Q2 := Image( nat2 );;
gap> SetName( Q2, "Q2" );
gap> Display( nat2 );
LeftModuleHomomorphismByMatrix( Basis( A2,
[ [ [ 0, 0, 0, 1 ], [ 0, 0, 0, 0 ], [ 0, 0, 0, 0 ], [ 0, 0, 0, 0 ] ],
  [ [ 0, 1, 2, 3 ], [ 0, 0, 1, 2 ], [ 0, 0, 0, 1 ], [ 0, 0, 0, 0 ] ],
  [ [ 0, 0, 1, 4 ], [ 0, 0, 0, 1 ], [ 0, 0, 0, 0 ], [ 0, 0, 0, 0 ] ] ] ),
[ [ 0, 0 ], [ 1, 0 ], [ 0, 1 ] ], CanonicalBasis( Q2 ) )
gap> act2 := AlgebraActionBySurjection( nat2 );;
gap> I2 := Image( act2 );;
gap> BI2 := BasisVectors( Basis( I2 ) );;
gap> b1 := BI2[1];; b2 := BI2[2];;
gap> [ Image(b1,m2)=m2^2, Image(b1,m2^2)=m2^3, Image(b1,m2^3)=Zero(A2) ];
[ true, true, true ]
gap> [ Image(b2,m2)=m2^3, b2=b1^2 ];
[true, true ]

```

2.2.4 AlgebraActionByHomomorphism

▷ `AlgebraActionByHomomorphism(hom, alg)`

(operation)

If $\alpha : A \rightarrow C$ is an algebra homomorphism where C is an algebra of left module isomorphisms of an algebra B , then `AlgebraActionByHomomorphism(alpha, B)` attempts to return an action of A on B .

In the example the matrix algebra A_3 and the group algebra Rc_3 are isomorphic algebras, so the resulting action is equivalent to the multiplier action of Rc_3 on itself.

Example

```

gap> m3 := [ [0,1,0], [0,0,1], [1,0,0] ];;
gap> A3 := Algebra( Rationals, [m3] );;
gap> SetName( A3, "A3" );;
gap> c3 := Group( (1,2,3) );;
gap> Rc3 := GroupRing( Rationals, c3 );;
gap> SetName( Rc3, "GR(c3)" );;
gap> g3 := GeneratorsOfAlgebra( Rc3 )[2];;
gap> mg3 := RegularAlgebraMultiplier( Rc3, Rc3, g3 );;
gap> Amg3 := AlgebraByGenerators( Rationals, [ mg3 ] );;
gap> homg3 := AlgebraHomomorphismByImages( A3, Amg3, [ m3 ], [ mg3 ] );;
gap> actg3 := AlgebraActionByHomomorphism( homg3, Rc3 );
[ [ [ 0, 1, 0 ], [ 0, 0, 1 ], [ 1, 0, 0 ] ] ->
  [ [ (1)*(), (1)*(1,2,3), (1)*(1,3,2) ] -> [ (1)*(1,2,3), (1)*(1,3,2), (1)*()
    ] ]

```

2.3 Algebra modules

Recall that a module can be made into an algebra by defining every product to be zero. When we apply this construction to a (left) algebra module, we obtain an algebra action on an algebra.

Recall the construction of algebra modules from Chapter 62 of the GAP reference manual. In the example, the vector space $V3$ becomes an algebra module $M3$ with a left action by $A3$. Conversion between vectors in $V3$ and those in $M3$ is achieved using the operations `ObjByExtRep` and `ExtRepOfObj`. These vectors are indistinguishable when printed.

Example

```

gap> V3 := Rationals^3;;
gap> M3 := LeftAlgebraModule( A3, \*, V3 );;
gap> SetName( M3, "M3" );;
gap> famM3 := ElementsFamily( FamilyObj( M3 ) );;
gap> v3 := [3,4,5];;
gap> v3 := ObjByExtRep( famM3, v3 );
[ 3, 4, 5 ]
gap> m3*v3;
[ 4, 5, 3 ]
gap> genM3 := GeneratorsOfLeftModule( M3 );;
gap> u4 := 6*genM3[1] + 7*genM3[2] + 8*genM3[3];
[ 6, 7, 8 ]
gap> u4 := ExtRepOfObj( u4 );
[ 6, 7, 8 ]

```

2.3.1 ModuleAsAlgebra

▷ `ModuleAsAlgebra(leftmod)`

(attribute)

To form an algebra B from M with zero products we may construct an algebra with the correct dimension using an empty structure constants table, as shown below. In doing so, the remaining information about M is lost, so it is essential to form isomorphisms between the corresponding underlying vector spaces.

If the module M has been given a name, then the operation `ModuleAsAlgebra` assigns a name to the resulting algebra. The operation `AlgebraByStructureConstants` assigns names v_i to the basis vectors unless a list of names is provided. The operation `ModuleAsAlgebra` converts the basis elements of M into strings, with additional brackets added, and uses these as the names for the basis vectors. Note that these `[[i,j,k]]` are just strings, and not vectors.

Example

```
gap> D3 := LeftActingDomain( M3 );;
gap> T3 := EmptySCTable( Dimension(M3), Zero(D3), "symmetric" );;
gap> B3a := AlgebraByStructureConstants( D3, T3 );
<algebra of dimension 3 over Rationals>
gap> GeneratorsOfAlgebra( B3a );
[ v.1, v.2, v.3 ]
gap> B3 := ModuleAsAlgebra( M3 );
A(M3)
gap> GeneratorsOfAlgebra( B3 );
[ [[ 1, 0, 0 ]], [[ 0, 1, 0 ]], [[ 0, 0, 1 ]] ]
```

2.3.2 IsModuleAsAlgebra

▷ `IsModuleAsAlgebra(alg)`

(operation)

This is the property acquired when a module is converted into an algebra.

Example

```
gap> IsModuleAsAlgebra( B3 );
true
gap> IsModuleAsAlgebra( A3 );
false
```

2.3.3 ModuleToAlgebraIsomorphism

▷ `ModuleToAlgebraIsomorphism(alg)`

(operation)

▷ `AlgebraToModuleIsomorphism(alg)`

(operation)

These two algebra mappings are attributes of a module converted into an algebra. They are required for the process of converting the action of A on M into an action on B . Note that these left module homomorphisms have as source or range the underlying module V , not M .

Example

```
gap> KnownAttributesOfObject( B3 );
[ "Name", "ZeroImmutable", "LeftActingDomain", "Dimension",
  "GeneratorsOfLeftOperatorAdditiveGroup", "GeneratorsOfLeftOperatorRing",
```

```

"ModuleToAlgebraIsomorphism", "AlgebraToModuleIsomorphism" ]
gap> M2B3 := ModuleToAlgebraIsomorphism( B3 );
[ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ] -> [ [ 1, 0, 0 ], [ 0, 1, 0 ],
[ 0, 0, 1 ] ]
gap> Source( M2B3 ) = M3;
false
gap> Source( M2B3 ) = V3;
true
gap> B2M3 := AlgebraToModuleIsomorphism( B3 );
[ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ] ->
[ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ]
gap> Range( B2M3 ) = M3;
false
gap> Range( B2M3 ) = V3;
true

```

2.3.4 AlgebraActionByModule

▷ AlgebraActionByModule(*alg*, *leftmod*)

(operation)

This operation converts the action of A on M into an action of A on B .

Example

```

gap> act3 := AlgebraActionByModule( A3, M3 );
[ [ [ 0, 1, 0 ], [ 0, 0, 1 ], [ 1, 0, 0 ] ] ] ->
[ [ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ] ] ->
[ [ [ 0, 0, 1 ], [ 1, 0, 0 ], [ 0, 1, 0 ] ] ]
gap> a3 := 2*m3 + 3*m3^2;
[ [ 0, 2, 3 ], [ 3, 0, 2 ], [ 2, 3, 0 ] ]
gap> Image( act3, a3 );
Basis( A(M3), [ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ] ) ->
[ (3)*[ 0, 1, 0 ]+(2)*[ 0, 0, 1 ], (2)*[ 1, 0, 0 ]+(3)*[ 0, 0, 1 ],
(3)*[ 1, 0, 0 ]+(2)*[ 0, 1, 0 ] ]
gap> Image( act3 );
<algebra over Rationals, with 1 generator>

```

2.4 Actions on direct sums of algebras

2.4.1 DirectSumOfAlgebrasWithInfo

▷ DirectSumOfAlgebrasWithInfo(*A1*, *A2*)

(operation)

▷ DirectSumOfAlgebrasInfo(*A*)

(attribute)

This attribute for direct sums of algebras is missing from the main library, and is added here to be used in methods for Embedding and Projection. In order to construct a direct sum with this information attribute the operation DirectSumOfAlgebrasWithInfo may be used. This just calls DirectSumOfAlgebras and sets up the attribute.

Example

```

gap> A3Rc3 := DirectSumOfAlgebrasWithInfo( A3, Rc3 );;
gap> SetName( A3Rc3, Concatenation( Name(A3), "(+)", Name(Rc3) ) );
gap> DirectSumOfAlgebrasInfo( A3Rc3 );
rec( algebras := [ A3, GR(c3) ],
      embeddings :=
      [
        Basis( A3, [ [ [ 0, 1, 0 ], [ 0, 0, 1 ], [ 1, 0, 0 ] ],
                  [ [ 0, 0, 1 ], [ 1, 0, 0 ], [ 0, 1, 0 ] ],
                  [ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ] ] ) -> [ v.1, v.2, v.3 ],
        CanonicalBasis( GR(c3) ) -> [ v.4, v.5, v.6 ] ], first := [ 1, 4 ],
      projections :=
      [ CanonicalBasis( A3(+)GR(c3) ) ->
        [ [ [ 0, 1, 0 ], [ 0, 0, 1 ], [ 1, 0, 0 ] ],
          [ [ 0, 0, 1 ], [ 1, 0, 0 ], [ 0, 1, 0 ] ],
          [ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ],
          [ [ 0, 0, 0 ], [ 0, 0, 0 ], [ 0, 0, 0 ] ],
          [ [ 0, 0, 0 ], [ 0, 0, 0 ], [ 0, 0, 0 ] ],
          [ [ 0, 0, 0 ], [ 0, 0, 0 ], [ 0, 0, 0 ] ] ],
        CanonicalBasis( A3(+)GR(c3) ) -> [ <zero> of ..., <zero> of ...,
          <zero> of ..., (1)*(), (1)*(1,2,3), (1)*(1,3,2) ] ] )

```

2.4.2 Embedding (for direct sum of algebras)

- ▷ Embedding(*A*, *nr*) (operation)
- ▷ Projection(*A*, *nr*) (operation)

Methods for Embedding and Projection for direct sums of algebras are missing from the main library, and so are included here.

Example

```

gap> Embedding( A3Rc3, 1 );
Basis( A3, [ [ [ 0, 1, 0 ], [ 0, 0, 1 ], [ 1, 0, 0 ] ],
            [ [ 0, 0, 1 ], [ 1, 0, 0 ], [ 0, 1, 0 ] ],
            [ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ] ] ) -> [ v.1, v.2, v.3 ]
gap> Projection( A3Rc3, 2 );
CanonicalBasis( A3(+)GR(c3) ) -> [ <zero> of ..., <zero> of ...,
  <zero> of ..., (1)*(), (1)*(1,2,3), (1)*(1,3,2) ]

```

2.5 Other operations on algebras

2.5.1 SemidirectProductOfAlgebras

- ▷ SemidirectProductOfAlgebras(*R*, *act*, *S*) (operation)

When R, S are commutative algebras and R acts on S then we can form the semidirect product $R \ltimes S$, where the product is given by:

$$(r_1, s_1)(r_2, s_2) = (r_1 r_2, r_1 \cdot s_2 + r_2 \cdot s_1 + s_1 s_2).$$

This product, as well as being commutative, is associative: $(r_1, s_1)(r_2, s_2)(r_3, s_3)$ expands as:

$$(r_1 r_2 r_3, (r_1 r_2) \cdot s_3 + (r_1 r_3) \cdot s_2 + (r_2 r_3) \cdot s_1 + r_1 \cdot (s_2 s_3) + r_2 \cdot (s_1 s_3) + r_3 \cdot (s_1 s_2) + s_1 s_2 s_3).$$

If B_R, B_S are the sets of basis vectors for R and S then $R \ltimes S$ has basis

$$\{(r, 0_S) \mid r \in B_R\} \cup \{(0_R, s) \mid s \in B_S\}$$

with defining products

$$(r_1, 0_S)(r_2, 0_S) = (r_1 r_2, 0_S), \quad (r, 0_S)(0_R, s) = (0_R, r \cdot s), \quad (0_R, s_1)(0_R, s_2) = (0_R, s_1 s_2).$$

Continuing the example above,

Example

```
gap> P1 := SemidirectProductOfAlgebras( A1, act1, I1 );
<algebra of dimension 8 over GF(5)>
gap> Embedding( P1, 1 );
[ (Z(5)^0)*(), (Z(5)^0)*(1,2,3,4,5,6), (Z(5)^0)*(1,3,5)(2,4,6),
  (Z(5)^0)*(1,4)(2,5)(3,6), (Z(5)^0)*(1,5,3)(2,6,4), (Z(5)^0)*(1,6,5,4,3,2)
] -> [ v.1, v.2, v.3, v.4, v.5, v.6 ]
gap> Embedding( P1, 2 );
[ (Z(5)^0)*()+(Z(5)^0)*(1,3,5)(2,4,6)+(Z(5)^0)*(1,5,3)(2,6,4),
  (Z(5)^0)*(1,2,3,4,5,6)+(Z(5)^0)*(1,4)(2,5)(3,6)+(Z(5)^0)*(1,6,5,4,3,2) ] ->
[ v.7, v.8 ]
gap> Projection( P1, 1 );
[ v.1, v.2, v.3, v.4, v.5, v.6, v.7, v.8 ] ->
[ (Z(5)^0)*(), (Z(5)^0)*(1,2,3,4,5,6), (Z(5)^0)*(1,3,5)(2,4,6),
  (Z(5)^0)*(1,4)(2,5)(3,6), (Z(5)^0)*(1,5,3)(2,6,4), (Z(5)^0)*(1,6,5,4,3,2),
  <zero> of ..., <zero> of ... ]
gap> P2 := SemidirectProductOfAlgebras( Q2, act2, A2 );
Q2 |X A2
gap> Embedding( P2, 1 );
[ v.1, v.2 ] -> [ v.1, v.2 ]
gap> Embedding( P2, 2 );
[ [ [ 0, 1, 2, 3 ], [ 0, 0, 1, 2 ], [ 0, 0, 0, 1 ], [ 0, 0, 0, 0 ] ],
  [ [ 0, 0, 1, 4 ], [ 0, 0, 0, 1 ], [ 0, 0, 0, 0 ], [ 0, 0, 0, 0 ] ],
  [ [ 0, 0, 0, 1 ], [ 0, 0, 0, 0 ], [ 0, 0, 0, 0 ], [ 0, 0, 0, 0 ] ] ] ->
[ v.3, v.4, v.5 ]
```

2.5.2 SemidirectProductOfAlgebrasInfo

▷ SemidirectProductOfAlgebrasInfo(P)

(attribute)

The SemidirectProductOfAlgebrasInfo(P) for $P = R \ltimes S$ is a record with fields $P.action$; $P.algebras$; $P.embeddings$; and $P.projections$.

2.6 Lists of algebra homomorphisms

2.6.1 AllAlgebraHomomorphisms

- ▷ AllAlgebraHomomorphisms(A , B) (operation)
- ▷ AllBijectiveAlgebraHomomorphisms(A , B) (operation)
- ▷ AllIdempotentAlgebraHomomorphisms(A , B) (operation)

These three operations list all the homomorphisms from A to B of the specified type. These lists can get very long, so the operations should only be used with small algebras.

Example

```
gap> A2c6 := GroupRing( GF(2), Group( (1,2,3,4,5,6) ) );;
gap> R2c3 := GroupRing( GF(2), Group( (7,8,9) ) );;
gap> homAR := AllAlgebraHomomorphisms( A2c6, R2c3 );;
gap> List( homAR, h -> MappingGeneratorsImages(h) );
[ [ [ (Z(2)^0)*(1,6,5,4,3,2) ], [ <zero> of ... ] ],
  [ [ (Z(2)^0)*(1,6,5,4,3,2) ], [ (Z(2)^0)*() ] ],
  [ [ (Z(2)^0)*(1,6,5,4,3,2) ], [ (Z(2)^0)*()+(Z(2)^0)*(7,8,9) ] ],
  [ [ (Z(2)^0)*(1,6,5,4,3,2) ],
    [ (Z(2)^0)*()+(Z(2)^0)*(7,8,9)+(Z(2)^0)*(7,9,8) ] ],
  [ [ (Z(2)^0)*(1,6,5,4,3,2) ], [ (Z(2)^0)*()+(Z(2)^0)*(7,9,8) ] ],
  [ [ (Z(2)^0)*(1,6,5,4,3,2) ], [ (Z(2)^0)*(7,8,9) ] ],
  [ [ (Z(2)^0)*(1,6,5,4,3,2) ], [ (Z(2)^0)*(7,8,9)+(Z(2)^0)*(7,9,8) ] ],
  [ [ (Z(2)^0)*(1,6,5,4,3,2) ], [ (Z(2)^0)*(7,9,8) ] ] ]
gap> homRA := AllAlgebraHomomorphisms( R2c3, A2c6 );;
gap> List( homRA, h -> MappingGeneratorsImages(h) );
[ [ [ (Z(2)^0)*(7,8,9) ], [ <zero> of ... ] ],
  [ [ (Z(2)^0)*(7,8,9) ], [ (Z(2)^0)*() ] ],
  [ [ (Z(2)^0)*(7,8,9) ], [ (Z(2)^0)*()+(Z(2)^0)*(1,3,5)(2,4,6) ] ],
  [ [ (Z(2)^0)*(7,8,9) ],
    [ (Z(2)^0)*()+(Z(2)^0)*(1,3,5)(2,4,6)+(Z(2)^0)*(1,5,3)(2,6,4) ] ],
  [ [ (Z(2)^0)*(7,8,9) ], [ (Z(2)^0)*()+(Z(2)^0)*(1,5,3)(2,6,4) ] ],
  [ [ (Z(2)^0)*(7,8,9) ], [ (Z(2)^0)*(1,3,5)(2,4,6) ] ],
  [ [ (Z(2)^0)*(7,8,9) ], [ (Z(2)^0)*(1,3,5)(2,4,6)+(Z(2)^0)*(1,5,3)(2,6,4) ] ],
  [ [ (Z(2)^0)*(7,8,9) ], [ (Z(2)^0)*(1,5,3)(2,6,4) ] ] ]
gap> bijAA := AllBijectiveAlgebraHomomorphisms( A2c6, A2c6 );;
gap> List( bijAA, h -> MappingGeneratorsImages(h) );
[ [ [ (Z(2)^0)*(1,6,5,4,3,2) ],
    [ (Z(2)^0)*()+(Z(2)^0)*(1,3,5)(2,4,6)+(Z(2)^0)*(1,4)(2,5)(3,6) ] ],
  [ [ (Z(2)^0)*(1,6,5,4,3,2) ],
    [ (Z(2)^0)*()+(Z(2)^0)*(1,4)(2,5)(3,6)+(Z(2)^0)*(1,5,3)(2,6,4) ] ],
  [ [ (Z(2)^0)*(1,6,5,4,3,2) ], [ (Z(2)^0)*(1,2,3,4,5,6) ] ],
  [ [ (Z(2)^0)*(1,6,5,4,3,2) ],
    [ (Z(2)^0)*(1,2,3,4,5,6)+(Z(2)^0)*(1,3,5)(2,4,6)+(Z(2)^0)*(1,5,3)(2,6,4) ] ],
  [ [ (Z(2)^0)*(1,6,5,4,3,2) ],
    [ (Z(2)^0)*(1,3,5)(2,4,6)+(Z(2)^0)*(1,5,3)(2,6,4)+(Z(2)^0)*
      (1,6,5,4,3,2) ] ],
  [ [ (Z(2)^0)*(1,6,5,4,3,2) ], [ (Z(2)^0)*(1,6,5,4,3,2) ] ] ]
gap> ideAA := AllIdempotentAlgebraHomomorphisms( A2c6, A2c6 );;
gap> Length( ideAA );
```


Chapter 3

Cat1-algebras

3.1 Definitions and examples

Algebraic structures which are equivalent to crossed modules of algebras include :

- cat^1 -algebras, (Ellis, [Eli88]);
- simplicial algebras with Moore complex of length 1, (Z. Arvasi and T.Porter, [AP96]);
- algebra-algebroids, (Gaffar Musa's Ph.D. thesis, [Mos86]).

In this section we describe an implementation of cat^1 -algebras and their morphisms.

The notion of a cat^1 -group was defined as an algebraic model of 2-types by Loday in [Lod82]. Then Ellis defined cat^1 -algebras in [Eli88].

Let A and R be k -algebras, let $t, h : A \rightarrow R$ be surjections, and let $e : R \rightarrow A$ be an inclusion.

$$\begin{array}{c} A \\ \begin{array}{c} \uparrow e \\ \parallel t \\ \downarrow h \\ R \end{array} \end{array}$$

If the conditions,

$$\mathbf{Cat1Alg1} : te = id_R = he, \quad \mathbf{Cat1Alg2} : (\ker t)(\ker h) = \{0_A\}$$

are satisfied, then the algebraic system $\mathcal{C} := (e; t, h : A \rightarrow R)$ is called a *cat¹-algebra*. A system which satisfies the condition **Cat1Alg1** is called a *precat¹-algebra*. The homomorphisms t, h and e are called the *tail map*, *head map* and *range embedding* homomorphisms, respectively.

3.1.1 Cat1Algebra

- | | |
|--|-------------|
| ▷ Cat1Algebra(args) | (function) |
| ▷ PreCat1AlgebraByEndomorphisms(t, h) | (operation) |
| ▷ PreCat1AlgebraByTailHeadEmbedding(t, h, e) | (operation) |
| ▷ PreCat1Algebra(args) | (function) |

- ▷ `IsIdentityCat1Algebra(C)` (property)
- ▷ `IsCat1Algebra(C)` (property)
- ▷ `IsPreCat1Algebra(C)` (property)

The operations `PreCat1AlgebraByEndomorphisms` and `PreCat1AlgebraByTailHeadEmbedding` are used with particular choices of algebra homomorphisms. The operations listed above are used for the construction of `precat1`- and `cat1`-algebras. The functions `PreCat1Algebra` and `Cat1Algebra` select the appropriate operation to suit the user's input.

3.1.2 Source (for cat1-algebras)

- ▷ `Source(C)` (attribute)
- ▷ `Range(C)` (attribute)
- ▷ `TailMap(C)` (attribute)
- ▷ `HeadMap(C)` (attribute)
- ▷ `RangeEmbedding(C)` (attribute)
- ▷ `Kernel(C)` (method)
- ▷ `KernelEmbedding(C)` (attribute)
- ▷ `Boundary(C)` (attribute)
- ▷ `Size2d(C)` (attribute)
- ▷ `Dimension(C)` (attribute)

These are the nine main attributes of a pre-cat¹-algebra.

In the example we use homomorphisms between `A2c6` and `I2c6` constructed in section 2.6.

Example

```
gap> t4 := homAR[8];
[ (Z(2)^0)*(1,6,5,4,3,2) ] -> [ (Z(2)^0)*(7,9,8) ]
gap> e4 := homRA[8];
[ (Z(2)^0)*(7,8,9) ] -> [ (Z(2)^0)*(1,5,3)(2,6,4) ]
gap> C4 := PreCat1AlgebraByTailHeadEmbedding( t4, t4, e4 );
[AlgebraWithOne( GF(2), [ (Z(2)^0)*(1,2,3,4,5,6)
  ] ) -> AlgebraWithOne( GF(2), [ (Z(2)^0)*(7,8,9) ] )]
gap> IsCat1Algebra( C4 );
true
gap> Size2d( C4 );
[ 64, 8 ]
gap> Dimension( C4 );
[ 6, 3 ]
gap> Display( C4 );

Cat1-algebra [..=>..] :-
: source algebra has generators:
  [ (Z(2)^0)*(), (Z(2)^0)*(1,2,3,4,5,6) ]
: range algebra has generators:
  [ (Z(2)^0)*(), (Z(2)^0)*(7,8,9) ]
: tail homomorphism maps source generators to:
  [ (Z(2)^0)*(), (Z(2)^0)*(7,8,9) ]
: head homomorphism maps source generators to:
  [ (Z(2)^0)*(), (Z(2)^0)*(7,8,9) ]
```

```

: range embedding maps range generators to:
[ (Z(2)^0)*(), (Z(2)^0)*(1,5,3)(2,6,4) ]
: kernel has generators:
[ (Z(2)^0)*()+(Z(2)^0)*(1,4)(2,5)(3,6), (Z(2)^0)*(1,2,3,4,5,6)+(Z(2)^0)*
  (1,5,3)(2,6,4), (Z(2)^0)*(1,3,5)(2,4,6)+(Z(2)^0)*(1,6,5,4,3,2) ]
: boundary homomorphism maps generators of kernel to:
[ <zero> of ..., <zero> of ..., <zero> of ... ]
: kernel embedding maps generators of kernel to:
[ (Z(2)^0)*()+(Z(2)^0)*(1,4)(2,5)(3,6), (Z(2)^0)*(1,2,3,4,5,6)+(Z(2)^0)*
  (1,5,3)(2,6,4), (Z(2)^0)*(1,3,5)(2,4,6)+(Z(2)^0)*(1,6,5,4,3,2) ]

```

3.1.3 Cat1AlgebraSelect

▷ Cat1AlgebraSelect(GFnum, gpsize, gpnum, num) (operation)

The Cat1Algebra (3.1.1) function may also be used to select certain cat^1 -group-algebras from the data file included with this package. Data for these cat^1 -structures on commutative group algebras is stored in a list in file cat1alldata.g. This data is read into the list CAT1ALG_LIST only when this function is called.

The function Cat1AlgebraSelect may be used in four ways:

- Cat1AlgebraSelect(n) returns the list of possible group orders when Galois field $GF(n)$, with $n \in [2, 3, 4, 5, 7]$, is used to form cat^1 -group-algebra structures.
- Cat1AlgebraSelect(n, m) returns the list of available group numbers of size m with which to form cat^1 -group-algebra structures with given Galois field $GF(n)$.
- Cat1AlgebraSelect(n, m, k) prints the list of possible cat^1 -group-algebra structures with given Galois field $GF(n)$ and group number k of size m . The number of these structures is returned.
- Cat1AlgebraSelect(n, m, k, j) (or simply Cat1Algebra(n, m, k, j)) returns the j -th cat^1 -group-algebra structure with these other parameters.

Now, we give examples of the use of this function.

Example

```

gap> C := Cat1AlgebraSelect( 11 );
|-----|
| 11 is invalid value for the Galois Field (GFnum) |
| Available values for GFnum in the data :         |
|-----|
[ 2, 3, 4, 5, 7 ]
Usage: Cat1Algebra( GFnum, gpsize, gpnum, num );
fail
gap> C := Cat1AlgebraSelect( 4, 12 );
|-----|
| 12 is invalid value for size of group (gpsize)   |
| Available values for gpsize with GF(4) in the data: |
|-----|

```

```

Usage: Cat1Algebra( GFnum, gpsize, gpnum, num );
[ 1, 2, 3, 4, 5, 6, 7, 8, 9 ]
gap> C := Cat1AlgebraSelect( 2, 6, 3 );
|-----|
| 3 is invalid value for groups of order 6 |
| Available values for gpnum for groups of size 6 : |
|-----|
Usage: Cat1Algebra( GFnum, gpsize, gpnum, num );
[ 1, 2 ]
gap> C := Cat1AlgebraSelect( 2, 6, 2 );
There are 4 cat1-structures for the group algebra GF(2)_c6.
  Range Alg      Tail      Head
|-----|
| GF(2)_c6      identity map      identity map |
| -----      [ 2, 10 ]          [ 2, 10 ] |
| -----      [ 2, 14 ]          [ 2, 14 ] |
| -----      [ 2, 50 ]          [ 2, 50 ] |
|-----|
Usage: Cat1Algebra( GFnum, gpsize, gpnum, num );
Algebra has generators [ (Z(2)^0)*(), (Z(2)^0)*(1,2,3)(4,5) ]
4

```

The algebra $\text{GF}(n)_{\text{gp}}$ has a list of $n^{|gp|}$ elements. The $[2, 10]$ in the second structure above indicates that the tail map, and also the head map, of the cat^1 -algebra maps the two generators of c_6 to the second and tenth elements of this algebra respectively.

Example

```

gap> C0 := Cat1AlgebraSelect( 4, 6, 2, 2 );
[GF(2^2)_c6 -> Algebra( GF(2^2),
[ (Z(2)^0)*(), (Z(2)^0)*()+ (Z(2)^0)*(1,3,5)(2,4,6)+ (Z(2)^0)*(1,4)(2,5)(3,6)+
  Z(2)^0*(1,5,3)(2,6,4)+ (Z(2)^0)*(1,6,5,4,3,2) ] )]
gap> Size2d( C0 );
[ 4096, 1024 ]
gap> Dimension( C0 );
[ 6, 5 ]
gap> Display( C0 );
Cat1-algebra [GF(2^2)_c6=>..] :-
: source algebra has generators:
[ (Z(2)^0)*(), (Z(2)^0)*(1,2,3,4,5,6) ]
: range algebra has generators:
[ (Z(2)^0)*(), (Z(2)^0)*()+ (Z(2)^0)*(1,3,5)(2,4,6)+ (Z(2)^0)*(1,4)(2,5)
  (3,6)+ (Z(2)^0)*(1,5,3)(2,6,4)+ (Z(2)^0)*(1,6,5,4,3,2) ]
: tail homomorphism maps source generators to:
[ (Z(2)^0)*(), (Z(2)^0)*()+ (Z(2)^0)*(1,3,5)(2,4,6)+ (Z(2)^0)*(1,4)(2,5)
  (3,6)+ (Z(2)^0)*(1,5,3)(2,6,4)+ (Z(2)^0)*(1,6,5,4,3,2) ]
: head homomorphism maps source generators to:
[ (Z(2)^0)*(), (Z(2)^0)*()+ (Z(2)^0)*(1,3,5)(2,4,6)+ (Z(2)^0)*(1,4)(2,5)
  (3,6)+ (Z(2)^0)*(1,5,3)(2,6,4)+ (Z(2)^0)*(1,6,5,4,3,2) ]
: range embedding maps range generators to:
[ (Z(2)^0)*(), (Z(2)^0)*()+ (Z(2)^0)*(1,3,5)(2,4,6)+ (Z(2)^0)*(1,4)(2,5)
  (3,6)+ (Z(2)^0)*(1,5,3)(2,6,4)+ (Z(2)^0)*(1,6,5,4,3,2) ]

```

```

: kernel has generators:
[ (Z(2)^0)*()+ (Z(2)^0)*(1,2,3,4,5,6)+ (Z(2)^0)*(1,3,5)(2,4,6)+ (Z(2)^0)*(1,4)
  (2,5)(3,6)+ (Z(2)^0)*(1,5,3)(2,6,4)+ (Z(2)^0)*(1,6,5,4,3,2) ]
: boundary homomorphism maps generators of kernel to:
[ <zero> of ... ]
: kernel embedding maps generators of kernel to:
[ (Z(2)^0)*()+ (Z(2)^0)*(1,2,3,4,5,6)+ (Z(2)^0)*(1,3,5)(2,4,6)+ (Z(2)^0)*(1,4)
  (2,5)(3,6)+ (Z(2)^0)*(1,5,3)(2,6,4)+ (Z(2)^0)*(1,6,5,4,3,2) ]

```

3.1.4 SubCat1Algebra

- ▷ SubCat1Algebra(*alg*, *src*, *rng*) (operation)
- ▷ SubPreCat1Algebra(*alg*, *src*, *rng*) (operation)
- ▷ IsSubCat1Algebra(*alg*, *sub*) (operation)
- ▷ IsSubPreCat1Algebra(*alg*, *sub*) (operation)

Let $\mathcal{C} = (e; t, h : A \rightarrow R)$ be a cat^1 -algebra, and let A', R' be subalgebras of A and R respectively. If the restriction morphisms

$$t' = t|_{A'} : A' \rightarrow R', \quad h' = h|_{A'} : A' \rightarrow R', \quad e' = e|_{R'} : R' \rightarrow A'$$

satisfy the **Cat1Alg1** and **Cat1Alg2** conditions, then the system $\mathcal{C}' = (e'; t', h' : A' \rightarrow R')$ is called a *subcat¹-algebra* of $\mathcal{C} = (e; t, h : A \rightarrow R)$.

If the morphisms satisfy only the **Cat1Alg1** condition then \mathcal{C}' is called a *sub-precat¹-algebra* of \mathcal{C} .

The operations in this subsection are used for constructing subcat¹-algebras of a given cat¹-algebra.

Example

```

gap> C6 := Cat1AlgebraSelect( 2, 6, 2, 4 );;
gap> A6 := Source( C6 );
GF(2)_c6
gap> B6 := Range( C6 );
<algebra of dimension 3 over GF(2)>
gap> eA6 := Elements( A6 );;
gap> eB6 := Elements( B6 );;
gap> SA6 := Subalgebra( A6, [ eA6[1], eA6[2], eA6[3] ] );
<algebra over GF(2), with 3 generators>
gap> [ Size(A6), Size(SA6) ];
[ 64, 4 ]
gap> SB6 := Subalgebra( B6, [ eB6[1], eB6[2] ] );
<algebra over GF(2), with 2 generators>
gap> [ Size(B6), Size(SB6) ];
[ 8, 2 ]
gap> SC6 := SubCat1Algebra( C6, SA6, SB6 );
[Algebra( GF(2), [ <zero> of ..., (Z(2)^0)*(), (Z(2)^0)*()+ (Z(2)^0)*(4,5)
  ] ) -> Algebra( GF(2), [ <zero> of ..., (Z(2)^0)*() ] )]
gap> Display( SC6 );
Cat1-algebra [..=>..] :-

```

```

: source algebra has generators:
[ <zero> of ..., (Z(2)^0)*(), (Z(2)^0)*()+(Z(2)^0)*(4,5) ]
: range algebra has generators:
[ <zero> of ..., (Z(2)^0)*() ]
: tail homomorphism maps source generators to:
[ <zero> of ..., (Z(2)^0)*(), <zero> of ... ]
: head homomorphism maps source generators to:
[ <zero> of ..., (Z(2)^0)*(), <zero> of ... ]
: range embedding maps range generators to:
[ <zero> of ..., (Z(2)^0)*() ]
: kernel has generators:
[ <zero> of ..., (Z(2)^0)*()+(Z(2)^0)*(4,5) ]
: boundary homomorphism maps generators of kernel to:
[ <zero> of ..., <zero> of ... ]
: kernel embedding maps generators of kernel to:
[ <zero> of ..., (Z(2)^0)*()+(Z(2)^0)*(4,5) ]
gap> IsSubCat1Algebra( C6, SC6 );
true

```

3.2 Cat^1 –algebra morphisms

Let $\mathcal{C} = (e; t, h : A \rightarrow R)$, $\mathcal{C}' = (e'; t', h' : A' \rightarrow R')$ be cat^1 -algebras, and let $\phi : A \rightarrow A'$ and $\varphi : R \rightarrow R'$ be algebra homomorphisms. If the diagram

$$\begin{array}{ccc}
 A & \xrightarrow{\phi} & A' \\
 \begin{array}{c} \curvearrowright e \\ \parallel t \\ \parallel h \\ \curvearrowright \end{array} & & \begin{array}{c} \parallel t' \\ \parallel h' \\ \curvearrowright e' \end{array} \\
 R & \xrightarrow{\varphi} & R'
 \end{array}$$

commutes, (i.e. $t' \circ \phi = \varphi \circ t$, $h' \circ \phi = \varphi \circ h$ and $e' \circ \varphi = \phi \circ e$), then the pair (ϕ, φ) is called a cat^1 -algebra morphism.

3.2.1 Cat1AlgebraMorphism

- ▷ Cat1AlgebraMorphism(*arg*) (function)
- ▷ PreCat1AlgebraMorphism(*arg*) (function)
- ▷ IdentityMapping(*C*) (method)
- ▷ PreCat1AlgebraMorphismByHoms(*src*, *rng*, *srchom*, *rnghom*) (operation)
- ▷ Cat1AlgebraMorphismByHoms(*src*, *rng*, *srchom*, *rnghom*) (operation)
- ▷ IsPreCat1AlgebraMorphism(*mor*) (property)
- ▷ IsCat1AlgebraMorphism(*mor*) (property)

These operations are used for constructing cat^1 -algebra morphisms. Details of the implementations can be found in [Oda09].

3.2.2 Source (for morphisms of cat1-algebras)

▷ Source(m)	(attribute)
▷ Range(m)	(attribute)
▷ IsTotal(m)	(method)
▷ IsSingleValued(m)	(method)
▷ Name(m)	(method)
▷ Boundary(m)	(attribute)

These are the six main attributes of a cat^1 -algebra morphism.

Example

```
gap> C1 := Cat1AlgebraSelect( 2, 1, 1, 1 );
[GF(2)_triv -> GF(2)_triv]
gap> Display( C1 );
Cat1-algebra [GF(2)_triv=>GF(2)_triv] :-
: source algebra has generators:
  [ (Z(2)^0)*(), (Z(2)^0)*() ]
: range algebra has generators:
  [ (Z(2)^0)*(), (Z(2)^0)*() ]
: tail homomorphism maps source generators to:
  [ (Z(2)^0)*(), (Z(2)^0)*() ]
: head homomorphism maps source generators to:
  [ (Z(2)^0)*(), (Z(2)^0)*() ]
: range embedding maps range generators to:
  [ (Z(2)^0)*(), (Z(2)^0)*() ]
: the kernel is trivial.
gap> C2 := Cat1AlgebraSelect( 2, 2, 1, 2 );
[GF(2)_c2 -> GF(2)_triv]
gap> Display( C2 );
Cat1-algebra [GF(2)_c2=>GF(2)_triv] :-
: source algebra has generators:
  [ (Z(2)^0)*(), (Z(2)^0)*(1,2) ]
: range algebra has generators:
  [ (Z(2)^0)*(), (Z(2)^0)*() ]
: tail homomorphism maps source generators to:
  [ (Z(2)^0)*(), (Z(2)^0)*() ]
: head homomorphism maps source generators to:
  [ (Z(2)^0)*(), (Z(2)^0)*() ]
: range embedding maps range generators to:
  [ (Z(2)^0)*(), (Z(2)^0)*() ]
: kernel has generators:
  [ (Z(2)^0)*()+(Z(2)^0)*(1,2) ]
: boundary homomorphism maps generators of kernel to:
  [ <zero> of ... ]
: kernel embedding maps generators of kernel to:
  [ (Z(2)^0)*()+(Z(2)^0)*(1,2) ]
gap> C1 = C2;
false
gap> SC1 := Source( C1 );
gap> SC2 := Source( C2 );
GF(2)_c2
```

```

gap> RC1 := Range( C1 );;
gap> RC2 := Range( C2 );;
gap> gSC1 := GeneratorsOfAlgebra( SC1 );
[ (Z(2)^0)*(), (Z(2)^0)*() ]
gap> gSC2 := GeneratorsOfAlgebra( SC2 );
[ (Z(2)^0)*(), (Z(2)^0)*(1,2) ]
gap> gRC1 := GeneratorsOfAlgebra( RC1 );
[ (Z(2)^0)*(), (Z(2)^0)*() ]
gap> gRC2 := GeneratorsOfAlgebra( RC2 );
[ (Z(2)^0)*(), (Z(2)^0)*() ]
gap> imS := [ gSC2[1], gSC2[1] ];
[ (Z(2)^0)*(), (Z(2)^0)*() ]
gap> homS := AlgebraHomomorphismByImages( SC1, SC2, gSC1, imS );
[ (Z(2)^0)*(), (Z(2)^0)*() ] -> [ (Z(2)^0)*(), (Z(2)^0)*() ]
gap> imR := [ gRC2[1], gRC2[1] ];
[ (Z(2)^0)*(), (Z(2)^0)*() ]
gap> homR := AlgebraHomomorphismByImages( RC1, RC2, gRC1, imR );
[ (Z(2)^0)*(), (Z(2)^0)*() ] -> [ (Z(2)^0)*(), (Z(2)^0)*() ]
gap> m12 := Cat1AlgebraMorphism( C1, C2, homS, homR );
[[GF(2)_triv=>GF(2)_triv] => [GF(2)_c2=>GF(2)_triv]]
gap> Display( m12 );

```

```

Morphism of cat1-algebras :-
: Source = [GF(2)_triv=>GF(2)_triv] with generating sets:
  [ (Z(2)^0)*(), (Z(2)^0)*() ]
  [ (Z(2)^0)*(), (Z(2)^0)*() ]
: Range = [GF(2)_c2=>GF(2)_triv] with generating sets:
  [ (Z(2)^0)*(), (Z(2)^0)*(1,2) ]
  [ (Z(2)^0)*(), (Z(2)^0)*() ]
: Source Homomorphism maps source generators to:
  [ (Z(2)^0)*(), (Z(2)^0)*() ]
: Range Homomorphism maps range generators to:
  [ (Z(2)^0)*(), (Z(2)^0)*() ]

```

```

## gap> Image2dAlgMapping( m12 );
## [GF(3)_c2^3=>GF(3)_c2^3]
gap> IsSurjective( m12 );
false
gap> IsInjective( m12 );
true
gap> IsBijective( m12 );
false

```

3.2.3 ImagesSource2DimensionalMapping

▷ ImagesSource2DimensionalMapping(m) (operation)

When (θ, φ) is a homomorphism of cat1-algebras (or crossed modules) this operation returns the image crossed module.

Example


```
gap> im12 := ImagesSource2DimensionalMapping( m12 );;  
gap> Display( im12 );  
Cat1-algebra [..=>..] :-  
: source algebra has generators:  
  [ (Z(2)^0)*(), (Z(2)^0)*() ]  
: range algebra has generators:  
  [ (Z(2)^0)*() ]  
: tail homomorphism maps source generators to:  
  [ (Z(2)^0)*(), (Z(2)^0)*() ]  
: head homomorphism maps source generators to:  
  [ (Z(2)^0)*(), (Z(2)^0)*() ]  
: range embedding maps range generators to:  
  [ (Z(2)^0)*() ]  
: the kernel is trivial.
```

Chapter 4

Crossed modules

In this chapter we will present the notion of crossed modules of commutative algebras and their implementation in this package.

4.1 Definition and Examples

A *crossed module* is a \mathbb{K} -algebra morphism $\mathcal{X} := (\partial : S \rightarrow R)$ with a left action of R on S satisfying

$$\mathbf{XModAlg\ 1} : \partial(r \cdot s) = r(\partial s), \quad \mathbf{XModAlg\ 2} : (\partial s) \cdot s' = ss',$$

for all $s, s' \in S$, $r \in R$. The morphism ∂ is called the *boundary map* of \mathcal{X}

Note that, although in this definition we have used a left action, in the category of commutative algebras left and right actions coincide.

When only the first axiom is satisfied, it is a *precrossed module* which is constructed.

The details of these implementations can be found in [Oda09].

4.1.1 XModAlgebra

- ▷ `XModAlgebra(args)` (function)
- ▷ `PreXModAlgebra(args)` (function)
- ▷ `IsXModAlgebra(X0)` (property)
- ▷ `IsPreXModAlgebra(X0)` (property)

These two global function call one of the following six operations, depending on the arguments supplied. The two properties listed are assigned as appropriate to the resulting structures.

4.1.2 XModAlgebraByIdeal

- ▷ `XModAlgebraByIdeal(A, I)` (operation)

Let A be an algebra and I an ideal of A . Then $\mathcal{X} = (inc : I \rightarrow A)$ is a crossed module whose action is left multiplication of A on I . Conversely, given a crossed module $\mathcal{X} = (\partial : S \rightarrow R)$, it is the case that $\partial(S)$ is an ideal of R .

Example

```

gap> F5 := GF(5);;
gap> id5 := One( F5 );;
gap> two := Z(5);;
gap> z5 := Zero( F5 );;
gap> n0 := [ [id5,z5,z5], [z5,id5,z5], [z5,z5,id5] ];;
gap> n1 := [ [z5,id5,two^3], [z5,z5,id5], [z5,z5,z5] ];;
gap> n2 := [ [z5,z5,id5], [z5,z5,z5], [z5,z5,z5] ];;
gap> An := Algebra( F5, [ n0, n1 ] );;
gap> SetName( An, "An" );
gap> Bn := Subalgebra( An, [ n1 ] );;
gap> SetName( Bn, "Bn" );
gap> IsIdeal( An, Bn );
true
gap> actn := AlgebraActionByMultipliers( An, Bn, An );;
gap> Xn := XModAlgebraByIdeal( An, Bn );
[ Bn -> An ]
gap> SetName( Xn, "Xn" );

```

4.1.3 AugmentationXMod

▷ AugmentationXMod(A)

(attribute)

As a special case of the previous operation, the attribute AugmentationXMod(A) of a group algebra A is the XModAlgebraByIdeal formed using the AugmentationIdeal of the group algebra.

Example

```

gap> Ak4 := GroupRing( GF(5), DihedralGroup(4) );
<algebra-with-one over GF(5), with 2 generators>
gap> Size( Ak4 );
625
gap> SetName( Ak4, "GF5[k4]" );
gap> IAK4 := AugmentationIdeal( Ak4 );
<two-sided ideal in GF5[k4], (2 generators)>
gap> Size( IAK4 );
125
gap> SetName( IAK4, "I(GF5[k4])" );
gap> XIAK4 := XModAlgebraByIdeal( Ak4, IAK4 );
[ I(GF5[k4]) -> GF5[k4] ]
gap> Display( XIAK4 );
Crossed module [I(GF5[k4])->GF5[k4]] :-
: Source algebra I(GF5[k4]) has generators:
  [ (Z(5)^2)*<identity> of ...+(Z(5)^0)*f1, (Z(5)^2)*<identity> of ...+(Z(5)^0)*f2 ]
: Range algebra GF5[k4] has generators:
  [ (Z(5)^0)*<identity> of ..., (Z(5)^0)*f1, (Z(5)^0)*f2 ]
: Boundary homomorphism maps source generators to:
  [ (Z(5)^2)*<identity> of ...+(Z(5)^0)*f1, (Z(5)^2)*<identity> of ...+(Z(5)^0)*f2 ]

```

```
gap> Size2d( XIAk4 );
[ 125, 625 ]
```

4.1.4 Source (for crossed modules of commutative algebras)

- ▷ Source(X) (attribute)
- ▷ Range(X) (attribute)
- ▷ Boundary(X) (attribute)
- ▷ XModAlgebraAction(X) (attribute)

These four attributes are used in the construction of a crossed module \mathcal{X} where:

- Source(X) and Range(X) are the *source* and the *range* of the boundary map respectively;
- Boundary(X) is the boundary map of the crossed module \mathcal{X} ;
- XModAlgebraAction(X) is the action used in the crossed module. This is an algebra homomorphism from Range(X) to an algebra of endomorphisms of Source(X).

The following standard GAP operations have special XModAlg implementations:

- Display(X) is used to list the components of \mathcal{X} ;
- Size2d(X) for a crossed module \mathcal{X} returns a 2-element list, the sizes of the source and range,
- Dimension(X) for a crossed module \mathcal{X} returns a 2-element list, the dimensions of the source and range,
- Name(X) is used for giving a name to the crossed module \mathcal{X} by associating the names of source and range algebras.

In the following example, we construct a crossed module by using the algebra GF_5D_4 and its augmentation ideal. We also show usage of the attributes listed above.

Example

```
gap> RepresentationsOfObject( XIAk4 );
[ "IsComponentObjectRep", "IsAttributeStoringRep", "IsPreXModAlgebraObj" ]
gap> KnownPropertiesOfObject( XIAk4 );
[ "CanEasilyCompareElements", "CanEasilySortElements", "IsDuplicateFree",
  "IsLeftActedOnByDivisionRing", "IsAdditivelyCommutative", "IsLDistributive",
  "IsRDistributive", "IsPreXModDomain", "Is2dAlgebraObject",
  "IsPreXModAlgebra", "IsXModAlgebra" ]
gap> KnownAttributesOfObject( XIAk4 );
[ "Name", "LeftActingDomain", "Range", "Source", "Boundary", "Size2d",
  "XModAlgebraAction" ]
```

4.1.5 XModAlgebraByMultiplierAlgebra

▷ `XModAlgebraByMultiplierAlgebra(A)` (operation)

When A is an algebra with multiplier algebra M , then the map $A \rightarrow M$, $a \mapsto \mu_a$ is the boundary of a crossed module in which the action is the identity map on M .

Example

```
gap> XAn := XModAlgebraByMultiplierAlgebra( An );
[ An -> <algebra of dimension 3 over GF(5)> ]
gap> XModAlgebraAction( XAn );
IdentityMapping( <algebra of dimension 3 over GF(5)> )
```

4.1.6 XModAlgebraBySurjection

▷ `XModAlgebraBySurjection(f)` (operation)

Let $\partial : S \rightarrow R$ be a surjective algebra homomorphism whose kernel lies in the annihilator of S . Define the action of R on S by $r \cdot s = \tilde{r}s$ where $\tilde{r} \in \partial^{-1}(r)$, as described in section [AlgebraActionBySurjection \(2.2.3\)](#). Then $\mathcal{X} = (\partial : S \rightarrow R)$ is a crossed module with the defined action.

Continuing with the example in that section,

Example

```
gap> X2 := XModAlgebraBySurjection( nat2 );;
gap> Display( X2 );
Crossed module [A2->Q2] :-
: Source algebra A2 has generators:
  [ [ 0, 1, 2, 3 ], [ 0, 0, 1, 2 ], [ 0, 0, 0, 1 ], [ 0, 0, 0, 0 ] ]
: Range algebra Q2 has generators:
  [ v.1, v.2 ]
: Boundary homomorphism maps source generators to:
  [ v.1 ]
```

4.1.7 XModAlgebraByBoundaryAndAction

▷ `XModAlgebraByBoundaryAndAction(bdy, act)` (operation)

▷ `PreXModAlgebraByBoundaryAndAction(bdy, act)` (operation)

When a suitable pair of algebra homomorphisms are available, these operations may be used. The example uses the algebra action created in section [2.2.4](#).

Example

```
gap> bdy3 := AlgebraHomomorphismByImages( Rc3, A3, [ g3 ], [ m3 ] );
[ (1)*(1,2,3) ] -> [ [ 0, 1, 0 ], [ 0, 0, 1 ], [ 1, 0, 0 ] ]
gap> X3 := XModAlgebraByBoundaryAndAction( bdy3, actg3 );
[ GR(c3) -> A3 ]
```

```

gap> Display( X3 );
Crossed module [GR(c3) -> A3] :-
: Source algebra GR(c3) has generators:
  [ (1)*(), (1)*(1,2,3) ]
: Range algebra A3 has generators:
  [ [ 0, 1, 0 ], [ 0, 0, 1 ], [ 1, 0, 0 ] ]
: Boundary homomorphism maps source generators to:
  [ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ],
  [ [ 0, 1, 0 ], [ 0, 0, 1 ], [ 1, 0, 0 ] ]

```

4.1.8 XModAlgebraByModule

▷ `XModAlgebraByModule(alg, leftmod)` (operation)

Let M be an A -module. Then $\mathcal{X} = (0 : A(M) \rightarrow A)$ is a crossed module, where $A(M)$ is M considered as an algebra with zero products (see section 2.3.1). The example uses the action `act3` constructed in section 2.3.4.

Conversely, given a crossed module $\mathcal{X} = (\partial : M \rightarrow R)$, one can get that $\ker \partial$ is a $(R/\partial M)$ -module.

Example

```

gap> Y3 := XModAlgebraByModule( A3, M3 );
[ A(M3) -> A3 ]
gap> Image( XModAlgebraAction( Y3 ), m3 ) = Image( act3, m3 );
true
gap> Display( Y3 );
Crossed module [A(M3) -> A3] :-
: Source algebra A(M3) has generators: [ [ 1, 0, 0 ], [ 0, 1, 0 ],
  [ 0, 0, 1 ] ]
: Range algebra A3 has generators:
  [ [ 0, 1, 0 ], [ 0, 0, 1 ], [ 1, 0, 0 ] ]
: Boundary homomorphism maps source generators to:
  [ [ 0, 0, 0 ], [ 0, 0, 0 ], [ 0, 0, 0 ] ],
  [ [ 0, 0, 0 ], [ 0, 0, 0 ], [ 0, 0, 0 ] ],
  [ [ 0, 0, 0 ], [ 0, 0, 0 ], [ 0, 0, 0 ] ]

```

4.1.9 SubXModAlgebra

▷ `SubXModAlgebra(alg, src, rng)` (operation)

▷ `IsSubXModAlgebra(alg, sub)` (operation)

A crossed module $\mathcal{X}' = (\partial' : S' \rightarrow R')$ is a subcrossed module of the crossed module $\mathcal{X} = (\partial : S \rightarrow R)$ if $S' \leq S$, $R' \leq R$, $\partial' = \partial|_{S'}$, and the action of S' on R' is induced by the action of R on S . The operation `SubXModAlgebra` is used to construct a subcrossed module of a given crossed module.

Example

```

gap> e4 := Elements( IAK4 )[4];

```

```

(Z(5)^0)*<identity> of ...+(Z(5)^0)*f1+(Z(5)^2)*f2+(Z(5)^2)*f1*f2
gap> Je4 := Ideal( IAK4, [e4] );
gap> SetName( Je4, "<e4>" );
gap> Ke4 := Subalgebra( Ak4, [e4] );
gap> [ Size( Je4 ), Size( Ke4 ) ];
[ 5, 5 ]
gap> Se4 := SubXModAlgebra( XIAk4, Je4, Ke4 );
[ <e4> -> <algebra of dimension 1 over GF(5)> ]
gap> IsSubXModAlgebra( XIAk4, Se4 );
true
gap> Display( Se4 );
Crossed module [<e4> -> ..] :-
: Source algebra <e4> has generators:
[ (Z(5)^0)*<identity> of ...+(Z(5)^0)*f1+(Z(5)^2)*f2+(Z(5)^2)*f1*f2 ]
: Range algebra has generators:
[ (Z(5)^0)*<identity> of ...+(Z(5)^0)*f1+(Z(5)^2)*f2+(Z(5)^2)*f1*f2 ]
: Boundary homomorphism maps source generators to:
[ (Z(5)^0)*<identity> of ...+(Z(5)^0)*f1+(Z(5)^2)*f2+(Z(5)^2)*f1*f2 ]

```

4.2 (Pre-)Crossed Module Morphisms

Let $\mathcal{X} = (\partial : S \rightarrow R)$ and $\mathcal{X}' = (\partial' : S' \rightarrow R')$ be (pre)crossed modules and let $\theta : S \rightarrow S'$, $\varphi : R \rightarrow R'$ be algebra homomorphisms. Then if

$$\varphi \circ \partial = \partial' \circ \theta, \quad \theta(r \cdot s) = \varphi(r) \cdot \theta(s),$$

for all $r \in R, s \in S$, the pair (θ, φ) is called a morphism between \mathcal{X} and \mathcal{X}'

The conditions can be thought as the commutativity of the following diagrams:

$$\begin{array}{ccc}
 S & \xrightarrow{\theta} & S' \\
 \partial \downarrow & & \downarrow \partial' \\
 R & \xrightarrow{\varphi} & R'
 \end{array}
 \quad
 \begin{array}{ccc}
 R \times S & \xrightarrow{\varphi \times \theta} & R' \times S' \\
 \downarrow & & \downarrow \\
 S & \xrightarrow{\theta} & S'
 \end{array}$$

In GAP we define the morphisms between algebraic structures such as cat^1 -algebras and crossed modules and they are investigated by the function `Make2dAlgebraMorphism`.

4.2.1 XModAlgebraMorphism

- ▷ `XModAlgebraMorphism(arg)` (function)
- ▷ `IdentityMapping(Xalg)` (method)
- ▷ `PreXModAlgebraMorphismByHoms(src, rng, srchom, rnghom)` (operation)
- ▷ `XModAlgebraMorphismByHoms(src, rng, srchom, rnghom)` (operation)
- ▷ `IsPreXModAlgebraMorphism(mor)` (property)
- ▷ `IsXModAlgebraMorphism(mor)` (property)
- ▷ `Source(mor)` (attribute)

▷ Range(<i>mor</i>)	(attribute)
▷ IsTotal(<i>mor</i>)	(method)
▷ IsSingleValued(<i>mor</i>)	(method)
▷ Name(<i>mor</i>)	(method)

These operations construct crossed module homomorphisms, which may have the attributes listed.

Example

```
gap> c4 := CyclicGroup( 4 );;
gap> Ac4 := GroupRing( GF(2), c4 );
<algebra-with-one over GF(2), with 2 generators>
gap> SetName( Ac4, "GF2[c4]" );
gap> IAc4 := AugmentationIdeal( Ac4 );
<two-sided ideal in GF2[c4], (dimension 3)>
gap> SetName( IAc4, "I(GF2[c4])" );
gap> XIAc4 := XModAlgebra( Ac4, IAc4 );
[ I(GF2[c4]) -> GF2[c4] ]
gap> Bk4 := GroupRing( GF(2), SmallGroup( 4, 2 ) );
<algebra-with-one over GF(2), with 2 generators>
gap> SetName( Bk4, "GF2[k4]" );
gap> IBk4 := AugmentationIdeal( Bk4 );
<two-sided ideal in GF2[k4], (dimension 3)>
gap> SetName( IBk4, "I(GF2[k4])" );
gap> XIBk4 := XModAlgebra( Bk4, IBk4 );
[ I(GF2[k4]) -> GF2[k4] ]
gap> IAc4 = IBk4;
false
gap> homIAIB := AllAlgebraHomomorphisms( IAc4, IBk4 );;
gap> theta := homIAIB[3];;
gap> homAB := AllAlgebraHomomorphisms( Ac4, Bk4 );;
gap> phi := homAB[7];;
gap> mor := XModAlgebraMorphism( XIAc4, XIBk4, theta, phi );
[[I(GF2[c4])->GF2[c4]] => [I(GF2[k4])->GF2[k4]]]
gap> Display( mor );

Morphism of crossed modules :-
: Source = [I(GF2[c4])->GF2[c4]]
: Range = [I(GF2[k4])->GF2[k4]]
: Source Homomorphism maps source generators to:
  [ <zero> of ..., (Z(2)^0)*<identity> of ...+(Z(2)^0)*f1+(Z(2)^0)*f2+(Z(2)^0)*f1*f2,
    (Z(2)^0)*<identity> of ...+(Z(2)^0)*f1+(Z(2)^0)*f2+(Z(2)^0)*f1*f2 ]
: Range Homomorphism maps range generators to:
  [ (Z(2)^0)*<identity> of ..., (Z(2)^0)*f1+(Z(2)^0)*f2+(Z(2)^0)*f1*f2,
    (Z(2)^0)*<identity> of ... ]

gap> IsTotal( mor );
true
gap> IsSingleValued( mor );
true
```


4.2.2 Kernel (for morphisms of crossed modules of algebras)

▷ `Kernel(mor)`

(method)

Let $(\theta, \varphi) : \mathcal{X} = (\partial : S \rightarrow R) \rightarrow \mathcal{X}' = (\partial' : S' \rightarrow R')$ be a crossed module homomorphism. Then the crossed module

$$\ker(\theta, \varphi) = (\partial| : \ker \theta \rightarrow \ker \varphi)$$

is called the *kernel* of (θ, φ) . Also, $\ker(\theta, \varphi)$ is an ideal of \mathcal{X} . An example is given below.

Example

```
gap> Xmor := Kernel( mor );
[ <algebra of dimension 2 over GF(2)> -> <algebra of dimension 2 over GF(2)> ]
gap> IsXModAlgebra( Xmor );
true
gap> Size2d( Xmor );
[ 4, 4 ]
gap> IsSubXModAlgebra( XIAC4, Xmor );
true
```

4.2.3 Image

▷ `Image(mor)`

(operation)

Let $(\theta, \varphi) : \mathcal{X} = (\partial : S \rightarrow R) \rightarrow \mathcal{X}' = (\partial' : S' \rightarrow R')$ be a crossed module homomorphism. Then the crossed module

$$\mathfrak{I}(\theta, \varphi) = (\partial'| : \mathfrak{I}\theta \rightarrow \mathfrak{I}\varphi)$$

is called the *image* of (θ, φ) . Further, $\mathfrak{I}(\theta, \varphi)$ is a subcrossed module of (S', R', ∂') .

In this package, the image of a crossed module homomorphism can be obtained by the command `ImagesSource`. The operation `Sub2dAlgObject` is effectively used for finding the kernel and image crossed modules induced from a given crossed module homomorphism.

4.2.4 SourceHom

▷ `SourceHom(mor)`

(attribute)

▷ `RangeHom(mor)`

(attribute)

▷ `IsInjective(mor)`

(property)

▷ `IsSurjective(mor)`

(property)

▷ `IsBijective(mor)`

(property)

Let (θ, φ) be a homomorphism of crossed modules. If the homomorphisms θ and φ are injective (surjective) then (θ, φ) is injective (surjective).

The attributes `SourceHom` and `RangeHom` store the two algebra homomorphisms θ and φ .

Example

```
gap> ic4 := One( Ac4 );;
gap> e1 := ic4*c4.1 + ic4*c4.2;
(Z(2)^0)*f1+(Z(2)^0)*f2
```

```

gap> ImageElm( theta, e1 );
(Z(2)^0)*<identity> of ...+(Z(2)^0)*f1+(Z(2)^0)*f2+(Z(2)^0)*f1*f2
gap> e2 := ic4*c4.1;
(Z(2)^0)*f1
gap> ImageElm( phi, e2 );
(Z(2)^0)*f1+(Z(2)^0)*f2+(Z(2)^0)*f1*f2
gap> IsInjective( mor );
false
gap> IsSurjective( mor );
false
gap> immor := ImagesSource2DimensionalMapping( mor );
gap> Display( immor );

Crossed module [...->...] :-
: Source algebra has generators:
  [ (Z(2)^0)*<identity> of ...+(Z(2)^0)*f1+(Z(2)^0)*f2+(Z(2)^0)*f1*f2 ]
: Range algebra has generators:
  [ (Z(2)^0)*f1+(Z(2)^0)*f2+(Z(2)^0)*f1*f2, (Z(2)^0)*<identity> of ... ]
: Boundary homomorphism maps source generators to:
  [ (Z(2)^0)*<identity> of ...+(Z(2)^0)*f1+(Z(2)^0)*f2+(Z(2)^0)*f1*f2 ]

```

Chapter 5

Conversion between cat1-algebras and crossed modules

5.1 Equivalent Categories

The categories **Cat1Alg** (cat¹-algebras) and **XModAlg** (crossed modules) are naturally equivalent [EI88]. This equivalence is outlined in what follows. For a given crossed module $(\partial : S \rightarrow R)$ we can construct the semidirect product $R \ltimes S$ thanks to the action of R on S . If we define $t, h : R \ltimes S \rightarrow R$ and $e : R \rightarrow R \ltimes S$ by

$$t(r, s) = r, \quad h(r, s) = r + \partial(s), \quad e(r) = (r, 0),$$

respectively, then $\mathcal{C} = (e; t, h : R \ltimes S \rightarrow R)$ is a cat¹-algebra.

Notice that h is an algebra homomorphism, since:

$$h(r_1 r_2, r_1 \cdot s_2 + r_2 \cdot s_1 + s_1 s_2) = r_1 r_2 + r_1(\partial s_2) + r_2(\partial s_1) + (\partial s_1)(\partial s_2) = (r_1 + \partial s_1)(r_2 + \partial s_2).$$

Conversely, for a given cat¹-algebra $\mathcal{C} = (e; t, h : A \rightarrow R)$, the map $\partial : \ker t \rightarrow R$ is a crossed module, where the action is multiplication action by eR , and ∂ is the restriction of h to $\ker t$.

Since all of these operations are linked to the functions `Cat1Algebra` (3.1.1) and `XModAlgebra` (4.1.1), they can be performed by calling these two functions. We may also use the function `Cat1Algebra` (3.1.1) instead of the operation `Cat1AlgebraSelect` (3.1.3).

5.1.1 Cat1AlgebraOfXModAlgebra

- ▷ `Cat1AlgebraOfXModAlgebra(X0)` (operation)
- ▷ `PreCat1AlgebraOfPreXModAlgebra(X0)` (operation)

These operations are used for constructing a cat¹-algebra from a given crossed module of algebras. As an example we use the crossed module `XAB` constructed in section 4.1.2.

Example

```
gap> Cn := Cat1AlgebraOfXModAlgebra( Xn );
[An |X Bn -> An]
gap> Display( Cn );
Cat1-algebra [An |X Bn => An] :-
: range algebra has generators:
```

```

[
  [ [ Z(5)^0, 0*Z(5), 0*Z(5) ], [ 0*Z(5), Z(5)^0, 0*Z(5) ],
    [ 0*Z(5), 0*Z(5), Z(5)^0 ] ],
  [ [ 0*Z(5), Z(5)^0, Z(5)^3 ], [ 0*Z(5), 0*Z(5), Z(5)^0 ],
    [ 0*Z(5), 0*Z(5), 0*Z(5) ] ] ]
: tail homomorphism maps source generators to:
[
  [ [ Z(5)^0, 0*Z(5), 0*Z(5) ], [ 0*Z(5), Z(5)^0, 0*Z(5) ],
    [ 0*Z(5), 0*Z(5), Z(5)^0 ] ],
  [ [ 0*Z(5), Z(5)^0, Z(5)^3 ], [ 0*Z(5), 0*Z(5), Z(5)^0 ],
    [ 0*Z(5), 0*Z(5), 0*Z(5) ] ],
  [ [ 0*Z(5), 0*Z(5), Z(5)^0 ], [ 0*Z(5), 0*Z(5), 0*Z(5) ],
    [ 0*Z(5), 0*Z(5), 0*Z(5) ] ],
  [ [ 0*Z(5), 0*Z(5), 0*Z(5) ], [ 0*Z(5), 0*Z(5), 0*Z(5) ],
    [ 0*Z(5), 0*Z(5), 0*Z(5) ] ],
  [ [ 0*Z(5), 0*Z(5), 0*Z(5) ], [ 0*Z(5), 0*Z(5), 0*Z(5) ],
    [ 0*Z(5), 0*Z(5), 0*Z(5) ] ] ]
: head homomorphism maps source generators to:
[
  [ [ Z(5)^0, 0*Z(5), 0*Z(5) ], [ 0*Z(5), Z(5)^0, 0*Z(5) ],
    [ 0*Z(5), 0*Z(5), Z(5)^0 ] ],
  [ [ 0*Z(5), Z(5)^0, Z(5)^3 ], [ 0*Z(5), 0*Z(5), Z(5)^0 ],
    [ 0*Z(5), 0*Z(5), 0*Z(5) ] ],
  [ [ 0*Z(5), 0*Z(5), Z(5)^0 ], [ 0*Z(5), 0*Z(5), 0*Z(5) ],
    [ 0*Z(5), 0*Z(5), 0*Z(5) ] ],
  [ [ 0*Z(5), Z(5)^0, Z(5)^3 ], [ 0*Z(5), 0*Z(5), Z(5)^0 ],
    [ 0*Z(5), 0*Z(5), 0*Z(5) ] ],
  [ [ 0*Z(5), 0*Z(5), Z(5)^0 ], [ 0*Z(5), 0*Z(5), 0*Z(5) ],
    [ 0*Z(5), 0*Z(5), 0*Z(5) ] ] ]
: range embedding maps range generators to:      [ v.1, v.2 ]
: kernel has generators: [ v.4, v.5 ]

```

As a second example, we convert the crossed module X_4 constructed in section 4.1.8

Example

```

gap> C3 := Cat1AlgebraOfXModAlgebra( X3 );
[A3 |X GR(c3) -> A3]
gap> Display( C3 );
Cat1-algebra [A3 |X GR(c3) => A3] :-
: range algebra has generators:[ [ [ 0, 1, 0 ], [ 0, 0, 1 ], [ 1, 0, 0 ] ] ]
: tail homomorphism maps source generators to:
[ [ [ 0, 1, 0 ], [ 0, 0, 1 ], [ 1, 0, 0 ] ],
  [ [ 0, 0, 1 ], [ 1, 0, 0 ], [ 0, 1, 0 ] ],
  [ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ],
  [ [ 0, 0, 0 ], [ 0, 0, 0 ], [ 0, 0, 0 ] ],
  [ [ 0, 0, 0 ], [ 0, 0, 0 ], [ 0, 0, 0 ] ],
  [ [ 0, 0, 0 ], [ 0, 0, 0 ], [ 0, 0, 0 ] ] ]
: head homomorphism maps source generators to:
[ [ [ 0, 1, 0 ], [ 0, 0, 1 ], [ 1, 0, 0 ] ],
  [ [ 0, 0, 1 ], [ 1, 0, 0 ], [ 0, 1, 0 ] ],
  [ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ],

```

```

[ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ],
[ [ 0, 1, 0 ], [ 0, 0, 1 ], [ 1, 0, 0 ] ],
[ [ 0, 0, 1 ], [ 1, 0, 0 ], [ 0, 1, 0 ] ] ]
: range embedding maps range generators to:    [ v.1 ]
: kernel has generators:  [ v.4, v.5, v.6 ]

```

5.1.2 XModAlgebraOfCat1Algebra

- ▷ XModAlgebraOfCat1Algebra(*C*) (operation)
- ▷ PreXModAlgebraOfPreCat1Algebra(*C*) (operation)

These operations are used for constructing a crossed module of algebras from a given cat^1 -algebra. The example uses the cat^1 -algebra C3 constructed in section 3.1.4.

Example

```

gap> X6 := XModAlgebraOfCat1Algebra( C6 );
[ <algebra of dimension 3 over GF(2)> -> <algebra of dimension 3 over GF(2)> ]
gap> Display( X6 );
Crossed module [..->..] :-
: Source algebra has generators:
  [ (Z(2)^0)*(), (Z(2)^0)*(4,5), (Z(2)^0)*(1,2,3)+(Z(2)^0)*(1,2,3)(4,5),
    (Z(2)^0)*(1,3,2)+(Z(2)^0)*(1,3,2)(4,5) ]
: Range algebra has generators:
  [ (Z(2)^0)*(), (Z(2)^0)*(1,2,3) ]
: Boundary homomorphism maps source generators to:
  [ <zero> of ..., <zero> of ..., <zero> of ... ]

```

References

- [AE03] Z. Arvasi and U. Ege. Annihilators, multipliers and crossed modules. *Applied Categorical Structures*, 11:487--506, 2003. 4
- [AO16] Z. Arvasi and A. Odabas. Computing 2-dimensional algebras: Crossed modules and cat1 -algebras. *Journal of Algebra and Its Applications*, 15:1650185 (12 pages), 2016. 4
- [AOUW17] M. Alp, A. Odabas, E. O. Uslu, and C. D. Wensley. *XMod: Crossed Modules and Cat1-groups in GAP*, 2017. GAP package, <https://gap-packages.github.io/xmod/>. 4
- [AP96] Z. Arvasi and T. Porter. Simplicial and crossed resolutions of commutative algebras. *J. Algebra*, 181:426--448, 1996. 4, 17
- [AP98] Z. Arvasi and T. Porter. Freeness conditions for 2-crossed modules of commutative algebras. *Applied Categorical Structures*, 6:455--471, 1998. 4
- [AW00] M. Alp and C. D. Wensley. Enumeration of cat1 -groups of low order. *Int. J. Algebra and Computation*, 10:407--424, 2000. 4
- [Bro82] R. Brown. Higher dimensional group theory. In *Low Dimensional Topology*, volume 48 of *London Math. Soc. Lecture Note Series*. London Mathematical Society, 1982. 4
- [Bro87] R. Brown. From groups to groupoids: a brief survey. *Bull. London Math. Soc.*, 19:113--134, 1987. 4
- [Ell88] G. J. Ellis. Higher dimensional crossed modules of algebras. *J. Pure and Appl. Algebra*, 52:277--282, 1988. 17, 35
- [GH17] S. Gutsche and M. Horn. *AutoDoc - Generate documentation from GAP source code (Version 2017.09.15)*, 2017. GAP package, <https://github.com/gap-packages/AutoDoc>. 2
- [Hor17] M. Horn. *GitHubPagesForGAP - a GitHub Pages generator for GAP packages*, 2017. GAP package, <https://gap-system.github.io/GitHubPagesForGAP/>. 2
- [LN17] F. Lübeck and M. Neunhöffer. *GAPDoc (version 1.6)*. RWTH Aachen, 2017. GAP package, <https://www.math.rwth-aachen.de/~Frank.Luebeck/GAPDoc/index.html>. 2
- [Lod82] J.-L. Loday. Spaces with finitely many non-trivial homotopy groups. *J. App. Algebra*, 24:179--202, 1982. 17

- [Mos86] G. H. Mosa. *Higher dimensional algebroids and crossed complexes*. PhD thesis, University of Wales, Bangor (U.K.), 1986. [17](#)
- [Oda09] A. Odabas. *Crossed Modules of Algebras with GAP*. PhD thesis, Osmangazi University, Eskisehir, 2009. [4](#), [22](#), [26](#)
- [Por87] T. Porter. Some categorical results in the theory of crossed modules in commutative algebras. *J. Algebra*, 109:415--429, 1987. [4](#)
- [Sha92] N. M. Shammu. *Algebraic and categorical structure of categories of crossed modules of algebras*. PhD thesis, University of Wales, Bangor (U.K.), 1992. [4](#)
- [Whi49] J. H. C. Whitehead. Combinatorial homotopy II. *Bull. Amer. Math. Soc.*, 55:453--496, 1949. [4](#)

Index

2d-algebra, [26](#)

AlgebraAction, [8](#)

AlgebraActionByHomomorphism, [9](#)

AlgebraActionByModule, [12](#)

AlgebraActionByMultipliers, [8](#)

AlgebraActionBySurjection, [8](#)

AlgebraToModuleIsomorphism, [11](#)

AllAlgebraHomomorphisms, [15](#)

AllBijectiveAlgebraHomomorphisms, [15](#)

AllIdempotentAlgebraHomomorphisms, [15](#)

AugmentationXMod, [27](#)

Boundary

for cat1-algebras, [18](#)

for crossed modules of commutative algebras, [28](#)

for morphisms of cat1-algebras, [23](#)

cat1-group, [17](#)

Cat1Algebra, [17](#)

Cat1AlgebraMorphism, [22](#)

Cat1AlgebraMorphismByHoms, [22](#)

Cat1AlgebraOfXModAlgebra, [35](#)

Cat1AlgebraSelect, [19](#)

crossed module, [26](#)

Dimension

for 2d-algebras, [18](#)

DirectSumOfAlgebrasInfo, [12](#)

DirectSumOfAlgebrasWithInfo, [12](#)

Embedding

for direct sum of algebras, [13](#)

HeadMap

for cat1-algebras, [18](#)

IdentityMapping

for cat1-algebras, [22](#)

for crossed modules of algebras, [31](#)

Image, [33](#)

ImagesSource2DimensionalMapping, [24](#)

IsAlgebraMultiplier, [6](#)

IsBijective, [33](#)

IsCat1Algebra, [18](#)

IsCat1AlgebraMorphism, [22](#)

IsIdentityCat1Algebra, [18](#)

IsInjective, [33](#)

IsModuleAsAlgebra, [11](#)

IsPreCat1Algebra, [18](#)

IsPreCat1AlgebraMorphism, [22](#)

IsPreXModAlgebra, [26](#)

IsPreXModAlgebraMorphism, [31](#)

IsSingleValued

for morphisms of cat1-algebras, [23](#)

for morphisms of crossed modules of algebras, [32](#)

IsSubCat1Algebra, [21](#)

IsSubPreCat1Algebra, [21](#)

IsSubXModAlgebra, [30](#)

IsSurjective, [33](#)

IsTotal

for morphisms of cat1-algebras, [23](#)

for morphisms of crossed modules of algebras, [32](#)

IsXModAlgebra, [26](#)

IsXModAlgebraMorphism, [31](#)

Kernel

for cat1-algebras, [18](#)

for morphisms of crossed modules of algebras, [33](#)

KernelEmbedding

for cat1-algebras, [18](#)

ModuleAsAlgebra, [10](#)

ModuleToAlgebraIsomorphism, [11](#)

MultiplierAlgebra, [7](#)

MultiplierAlgebraOfIdealBySubalgebra ,
 6
 MultiplierHomomorphism, 7
 Name
 for morphisms of cat1-algebras, 23
 for morphisms of crossed modules of algebras, 32
 PreCat1Algebra, 17
 PreCat1AlgebraByEndomorphisms, 17
 PreCat1AlgebraByTailHeadEmbedding, 17
 PreCat1AlgebraMorphism, 22
 PreCat1AlgebraMorphismByHoms, 22
 PreCat1AlgebraOfPreXModAlgebra, 35
 precrossed module, 26
 PreXModAlgebra, 26
 PreXModAlgebraByBoundaryAndAction, 29
 PreXModAlgebraMorphismByHoms, 31
 PreXModAlgebraOfPreCat1Algebra, 37
 Projection
 for direct sum of algebras, 13
 Range
 for cat1-algebras, 18
 for crossed modules of commutative algebras, 28
 for morphisms of cat1-algebras, 23
 for morphisms of crossed modules of algebras, 32
 RangeEmbedding
 for cat1-algebras, 18
 RangeHom, 33
 RegularAlgebraMultiplier, 5
 SemidirectProductOfAlgebras, 13
 SemidirectProductOfAlgebrasInfo, 14
 Size2d
 for 2d-algebras, 18
 Source
 for cat1-algebras, 18
 for crossed modules of commutative algebras, 28
 for morphisms of cat1-algebras, 23
 for morphisms of crossed modules of algebras, 31
 SourceHom, 33
 SubCat1Algebra, 21
 SubPreCat1Algebra, 21
 SubXModAlgebra, 30
 TailMap
 for cat1-algebras, 18
 XModAlgebra, 26
 XModAlgebraAction, 28
 XModAlgebraByBoundaryAndAction, 29
 XModAlgebraByIdeal, 26
 XModAlgebraByModule, 30
 XModAlgebraByMultiplierAlgebra, 29
 XModAlgebraBySurjection, 29
 XModAlgebraMorphism, 31
 XModAlgebraMorphismByHoms, 31
 XModAlgebraOfCat1Algebra, 37