XModAlg

Crossed Modules and Cat1-Algebras

1.23

3 December 2022

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

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

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 [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 of 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 $\mathscr{K}(.) : Gr \to Alg$ is left adjoint to the unit group functor $\mathscr{U}(.) : Alg \to Gr$.

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.

2.1 Multipliers

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

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

The *regular multipliers* of A are the functions

$$\mu_a : A \to A \;:\; \mu_a b = ab \quad \forall\, b \in A.$$

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 $\mu_a : I \to I$ on an ideal I of A.

Example

```
gap> A5c6 := GroupRing( GF(5), Group( (1,2,3,4,5,6) ) );;
gap> vecA := BasisVectors( Basis( A5c6 ) );;
gap> v := vecA[1] + vecA[3] + vecA[5];
(Z(5)^0)*()+(Z(5)^0)*(1,3,5)(2,4,6)+(Z(5)^0)*(1,5,3)(2,6,4)
gap> I5c6 := Ideal( A5c6, [v] );;
gap> v2 := vecA[2];
(Z(5)^0)*(1,2,3,4,5,6)
gap> m2 := RegularAlgebraMultiplier( A5c6, I5c6, v2 );
[ (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(m)` (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( m2 );
true
gap> one := One( A5c6 );;
gap> L := List( vecA, v -> one );;
gap> m1 := LeftModuleHomomorphismByImages( A5c6, A5c6, vecA, L );
[ (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( m1 );
false
```

2.1.3 MultiplierAlgebraOfIdealBySubalgebra

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

The regular multipliers $\mu_b : I \to I$ 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> v3 := vecA[3];
(Z(5)^0)*(1,3,5)(2,4,6)
gap> B5c3 := Subalgebra( A5c6, [ v3 ] );;
gap> M := MultiplierAlgebraOfIdealBySubalgebra( A5c6, I5c6, B5c3 );
<algebra of dimension 1 over GF(5)>
gap> vecM := BasisVectors( Basis( M ) );;
gap> vecM[1];
<linear mapping by matrix,
<two-sided ideal in <algebra-with-one of dimension 6 over GF(5)>, (dimension 2
 )> -> <two-sided ideal in <algebra-with-one of dimension 6 over GF(5)>,
  (dimension 2)>>
```

2.1.4 MultiplierAlgebra

▷ `MultiplierAlgebra(A)` (attribute)

The regular multipliers $\mu_a : A \to A$ 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> MA5c6 := RegularMultiplierAlgebra( A5c6 );
<algebra of dimension 6 over GF(5)>
gap> vecM := BasisVectors( Basis( MA5c6 ) );;
gap> vecM[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 algebra A multiplying an ideal I then this operation returns the homomorphism from A to M mapping a to μ_a.

Example

```
gap> hom := MultiplierHomomorphism( MA5c6 );;
gap> ImageElm( hom, vecA[2] );
Basis( <two-sided ideal in <algebra-with-one of dimension 6 over GF(5)>,
  (dimension 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)
 ] ) ->
[ (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) ]
```

2.2 Commutative actions

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

$$R \times S \;\to\; S, \qquad (r,s) \;\mapsto\; r \cdot s$$

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 $0_R \cdot s = 0_S \; \forall \; s \in S$),
- $r \cdot (ss') \;=\; (r \cdot s)s' = s(r \cdot s')$,
- $(rr') \cdot s \;=\; r \cdot (r' \cdot s)$, (so $1_R \cdot s = s \; \forall \; s \in S$ when R has a one),

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

2.2.1 AlgebraActionByMultipliers

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

When I is an ideal in A we have seen that the multiplier homomorphism from A to `MultiplierAlgebraOf(Ideal(A,I)` 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 $r \cdot v = (1,2,3,4,5,6) + (1,4)(2,5)(3,6) + (1,6,5,4,3,2)$.

Example

```
gap> A5c6 := GroupRing( GF(5), Group( (1,2,3,4,5,6) ) );;
gap> vecA := BasisVectors( Basis( A5c6 ) );;
gap> v := vecA[1] + vecA[3] + vecA[5];
(Z(5)^0)*()+(Z(5)^0)*(1,3,5)(2,4,6)+(Z(5)^0)*(1,5,3)(2,6,4)
gap> I5c6 := Ideal( A5c6, [v] );;
gap> actm := AlgebraActionByMultipliers( A5c6, I5c6 );;
gap> actm2 := Image( actm, vecA[2] );;
gap> Image( actm2, 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)
```

2.2.2 AlgebraActionBySurjection

▷ `AlgebraActionBySurjection(hom)` (operation)

Let $\theta : S \to R$ be a surjective algebra homomorphism such that $ks = 0_S \ \forall \ k \in K = \ker\theta$. Then R acts on S with $r \cdot s = (\theta^{-1}r)s$. Note that thus action is well defined since if $\theta p = r$ then $\theta^{-1}r = \{p+k \mid k \in \ker\theta\}$ and $(p+k)s = ps + ks = ps + 0$.

Continuing with the previous example, we construct the quotient algebra $Q5c6 = A5c6/I5c6$, and the natural homomorphism $\theta : A5c6 \to Q5c6$. The kernel of θ is not contained in the annihilator of $A5c6$, so an attempt to form the action fails.

An alternative example involves a single-generator matrix algebra.

Example

```
gap> theta := NaturalHomomorphismByIdeal( A5c6, I5c6 );
<linear mapping by matrix, <algebra-with-one of dimension
6 over GF(5)> -> <algebra of dimension 4 over GF(5)>>
gap> List( vecA, v -> ImageElm( theta, 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> actp := AlgebraActionBySurjection( theta );
kernel of hom is not in the annihilator of A
fail
gap> ## an example which does not fail:
gap> m := [ [0,1,2,3], [0,0,1,2], [0,0,0,1], [0,0,0,0] ];;
gap> m^2;
[ [ 0, 0, 1, 4 ], [ 0, 0, 0, 1 ], [ 0, 0, 0, 0 ], [ 0, 0, 0, 0 ] ]
gap> m^3;
[ [ 0, 0, 0, 1 ], [ 0, 0, 0, 0 ], [ 0, 0, 0, 0 ], [ 0, 0, 0, 0 ] ]
gap> A1 := Algebra( Rationals, [m] );;
```

```
gap> A3 := Subalgebra( A1, [m^3] );;
gap> nat3 := NaturalHomomorphismByIdeal( A1, A3 );
<linear mapping by matrix, <algebra of dimension
3 over Rationals> -> <algebra of dimension 2 over Rationals>>
gap> act3 := AlgebraActionBySurjection( nat3 );;
gap> a3 := Image( act3, BasisVectors( Basis( Image( nat3 ) ) )[1] );;
gap> [ Image( a3, m ) = m^2, Image( a3, m^2 ) = m^3 ];
[ true, true ]
```

2.2.3 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:

$$(r_1,s_1)(r_2,s_2) \;=\; (r_1r_2,\; r_1\cdot s_2 + r_2\cdot s_1 + s_1s_2).$$

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

$$(r_1r_2r_3,\;(r_1r_2)\cdot s3 + (r_1r_3)\cdot s_2 + (r_2r_3)\cdot s_1 + r_1\cdot(s_2s_3) + r_2\cdot(s_1s_3) + r_3\cdot(s_1s_2) + s_1s_2s_3)\,.$$

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

$$(r_1,0_S)(r_2,0_S) = (r_1r_2,0_S), \qquad (r,0_S)(0_R,s) = (0_R,r\cdot s), \qquad (0_R,s_1)(0_R,s_2) = (0_R,s_1s_2).$$

Continuing the example above,

Example

```
gap> P := SemidirectProductOfAlgebras( A5c6, actm, I5c6 );
gap> Embedding( P, 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( P, 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( P, 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 ... ]
```

2.2.4 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`.

2.3 Lists of algebra homomorphisms

2.3.1 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 );
14
```

Chapter 3

Cat1-algebras

3.1 Definitions and examples

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

- cat^1-algebras, (Ellis, [Ell88]);
- 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 cat^1-groups was defined as an algebraic model of 2-types by Loday in [Lod82]. Then Ellis defined the cat^1-algebras in [Ell88].

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

$$\begin{tikzcd} A \arrow[d, shift right, "t"'] \arrow[d, shift left, "h"] \\ R \arrow[u, bend left=60, "e"] \end{tikzcd}$$

If the conditions,

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

are satisfied, then the algebraic system $\mathscr{C} := (e;t,h : A \to R)$ is called a *cat^1-algebra*. A system which satisfies the condition **Cat1Alg1** is called a *precat^1-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*`)` (operation)

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

The operations listed above are used for construction of precat^1- and cat^1-algebra structures. The function `Cat1Algebra` selects the operation from the above implementations up to user's input. The operations `PreCat1AlgebraByEndomorphisms` and `PreCat1AlgebraByTailHeadEmbedding` are used with particular choices of algebra homomorphisms.

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)
▷ Boundary(C) (attribute)
▷ Size2d(C) (attribute)

These are the eight main attributes of a pre-cat^1-algebra.

In the example we use homomorphisms between `A2c6` and `I2c6` constructed in section 2.3.

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> 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(n, gpsize, gpnum, num)` (operation)

The `Cat1Algebra` (3.1.1) function may also be used to select a cat^1-algebra from a data file. All cat^1-structures on commutative algebras are stored in a list in file `cat1algdata.g`. The 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 sizes of groups for group algebras with Galois field $GF(n)$.
- `Cat1AlgebraSelect( n, m )` returns the list of allowable group numbers with given Galois field $GF(n)$ and groups of size m.
- `Cat1AlgebraSelect( n, m, k )` returns the list of possible cat^1-algebra structures with given Galois field $GF(n)$ and group number k of size m.
- `Cat1AlgebraSelect( n, m, k, j )` (or simply `Cat1Algebra( n, m, k, j )`) returns the j-th cat^1-algebra structure with these other parameters.

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

Example

```
gap> C := Cat1AlgebraSelect( 11 );
|--------------------------------------------------------|
| 11 is invalid number for Galois Field (GFnum)           |
| Possible numbers 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 number for size of group (gpsize)        |
| Possible numbers for the gpsize for GF(4) in the Data: |
|--------------------------------------------------------|
 [ 1, 2, 3, 4, 5, 6, 7, 8, 9 ]
Usage: Cat1Algebra( GFnum, gpsize, gpnum, num );
fail
gap> C := Cat1AlgebraSelect( 2, 6, 3 );
|--------------------------------------------------------|
| 3 is invalid number for group of order 6               |
```

```
| Possible numbers for the gpnum in the Data :             |
|----------------------------------------------------------|
 [ 1, 2 ]
Usage: Cat1Algebra( GFnum, gpsize, gpnum, num );
fail
gap> C := Cat1AlgebraSelect( 2, 6, 2 );
There are 4 cat1-structures for the 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
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> 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(arg)` (operation)
▷ `SubPreCat1Algebra(arg)` (operation)
▷ `IsSubCat1Algebra(arg)` (property)
▷ `IsSubPreCat1Algebra(arg)` (property)

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

$$t' = t|_{A'} : A' \to R', \qquad h' = h|_{A'} : A' \to R', \qquad e' = e|_{R'} : R' \to A'$$

satisfy the **Cat1Alg1** and **Cat1Alg2** conditions, then the system $\mathscr{C}' = (e';t',h' : A' \to R')$ is called a *subcat^1-algebra* of $\mathscr{C} = (e;t,h : A \to R)$.

If the morphisms satisfy only the **Cat1Alg1** condition then $\mathscr{C}'$ is called a *sub-precat^1-algebra* of $\mathscr{C}$.

The operations in this subsection are used for constructing subcat^1-algebras of a given cat^1-algebra.

Example

```
gap> C3 := Cat1AlgebraSelect( 2, 6, 2, 4 );;
gap> A3 := Source( C3 );
GF(2)_c6
gap> B3 := Range( C3 );
GF(2)_c3
gap> eA3 := Elements( A3 );;
gap> eB3 := Elements( B3 );;
gap> AA3 := Subalgebra( A3, [ eA3[1], eA3[2], eA3[3] ] );
<algebra over GF(2), with 3 generators>
gap> [ Size(A3), Size(AA3) ];
[ 64, 4 ]
gap> BB3 := Subalgebra( B3, [ eB3[1], eB3[2] ] );
<algebra over GF(2), with 2 generators>
gap> [ Size(B3), Size(BB3) ];
[ 8, 2 ]
gap> CC3 := SubCat1Algebra( C3, AA3, BB3 );
[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( CC3 );

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) ]
```

3.2 Cat$^1-$algebra morphisms

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

$$\begin{tikzcd}
A \arrow[r, "\phi"] \arrow[d, shift right, "t"'] \arrow[d, shift left, "h"] & A' \arrow[d, shift right, "t'"'] \arrow[d, shift left, "h'"] \\
R \arrow[r, "\varphi"'] \arrow[u, bend left=60, "e"] & R' \arrow[u, bend right=60, "e'"']
\end{tikzcd}$$

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)` (operation)
▷ `IdentityMapping(C)` (method)
▷ `PreCat1AlgebraMorphismByHoms(f, g)` (operation)
▷ `Cat1AlgebraMorphismByHoms(f, g)` (operation)
▷ `IsPreCat1AlgebraMorphism(C)` (property)
▷ `IsCat1AlgebraMorphism(arg)` (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 := Cat1Algebra( 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 := Cat1Algebra( 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> R1 := Source( C1 );;
gap> R2 := Source( C2 );;
gap> S1 := Range( C1 );;
gap> S2 := Range( C2 );;
gap> gR1 := GeneratorsOfAlgebra( R1 );
[ (Z(2)^0)*(), (Z(2)^0)*() ]
gap> gR2 := GeneratorsOfAlgebra( R2 );
[ (Z(2)^0)*(), (Z(2)^0)*(1,2) ]
gap> gS1 := GeneratorsOfAlgebra( S1 );
[ (Z(2)^0)*(), (Z(2)^0)*() ]
```

```
gap> gS2 := GeneratorsOfAlgebra( S2 );
[ (Z(2)^0)*(), (Z(2)^0)*() ]
gap> im1 := [ gR2[1], gR2[1] ];
[ (Z(2)^0)*(), (Z(2)^0)*() ]
gap> f1 := AlgebraHomomorphismByImages( R1, R2, gR1, im1 );
[ (Z(2)^0)*(), (Z(2)^0)*() ] -> [ (Z(2)^0)*(), (Z(2)^0)*() ]
gap> im2 := [ gS2[1], gS2[1] ];
[ (Z(2)^0)*(), (Z(2)^0)*() ]
gap> f2 := AlgebraHomomorphismByImages( S1, S2, gS1, im2 );
[ (Z(2)^0)*(), (Z(2)^0)*() ] -> [ (Z(2)^0)*(), (Z(2)^0)*() ]
gap> m := Cat1AlgebraMorphism( C1, C2, f1, f2 );
[[GF(2)_triv=>GF(2)_triv] => [GF(2)_c2=>GF(2)_triv]]
gap> Display( m );
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> IsSurjective( m );
false
gap> IsInjective( m );
true
gap> IsBijective( m );
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> imm := ImagesSource2DimensionalMapping( m );;
gap> Display( imm );

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 K-algebra morphism $\mathscr{X} := (\partial : S \to R)$ with a left action of R on S satisfying

$$\textbf{XModAlg 1} \;:\; \partial(r \cdot s) = r(\partial s), \qquad \textbf{XModAlg 2} \;:\; (\partial s) \cdot s' = ss',$$

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

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

4.1.1 XModAlgebra

▷ `XModAlgebra(args)` (function)

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

4.1.2 XModAlgebraByIdeal

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

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

Example

```
gap> F := GF(5);;
gap> one := One(F);;
gap> two := Z(5);;
gap> z := Zero( F );;
gap> l := [ [one,z,z], [z,one,z], [z,z,one] ];;
gap> m := [ [z,one,two^3], [z,z,one], [z,z,z] ];;
```

```
gap> n := [ [z,z,one], [z,z,z], [z,z,z] ];;
gap> A := Algebra( F, [l,m] );;
gap> SetName( A, "A(l,m)" );
gap> B := Subalgebra( A, [m] );;
gap> SetName( B, "A(m)" );
gap> IsIdeal( A, B );
true
gap> act := AlgebraActionByMultipliers( A, B );;
gap> XAB := XModAlgebraByIdeal( A, B );
[ A(m) -> A(l,m) ]
gap> SetName( XAB, "XAB" );
```

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 XModAlgebraByMultiplierAlgebra

▷ `XModAlgebraByMultiplierAlgebra(A)` (operation)

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

Example

```
gap> XA := XModAlgebraByMultiplierAlgebra( A );
[ A(l,m) -> <algebra of dimension 3 over GF(5)> ]
gap> XModAlgebraAction( XA );
IdentityMapping( <algebra of dimension 3 over GF(5)> )
```

4.1.5 XModAlgebraBySurjection

▷ `XModAlgebraBySurjection(f)` (operation)

Let $\partial : S \to R$ be a surjective algebra homomorphism whose kernel lise 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` (2.2.2) Then $\mathscr{X} = (\partial : S \to R)$ is a crossed module with the defined action.

Continuing with the example in that section,

Example

```
gap> X3 := XModAlgebraBySurjection( nat3 );;
gap> Display( X3 );

Crossed module [..->..] :-
: Source algebra has generators:
  [ [ [ 0, 1, 2, 3 ], [ 0, 0, 1, 2 ], [ 0, 0, 0, 1 ], [ 0, 0, 0, 0 ] ] ]
: Range algebra has generators:
  [ v.1, v.2 ]
: Boundary homomorphism maps source generators to:
  [ v.1 ]
```

4.1.6 XModAlgebraByBoundaryAndAction

▷ `XModAlgebraByBoundaryAndAction(bdy, act)` (operation)
▷ `PreXModAlgebraByBoundaryAndAction(bdy, act)` (operation)
▷ `IsPreXModAlgebra(X0)` (property)

An R-algebra homomorphism $\mathscr{X} := (\partial : S \to R)$ which satisfies the condition **XModAlg 1** is called a *precrossed module*. The details of these implementations can be found in [Oda09].

Example

```
gap> G := SmallGroup( 4, 2 );
```

```
<pc group of size 4 with 2 generators>
gap> F := GaloisField( 4 );
GF(2^2)
gap> R := GroupRing( F, G );
<algebra-with-one over GF(2^2), with 2 generators>
gap> Size( R );
256
gap> SetName( R, "GF(2^2)[k4]" );
gap> e5 := Elements( R )[5];
(Z(2)^0)*<identity> of ...+(Z(2)^0)*f1+(Z(2)^0)*f2+(Z(2)^0)*f1*f2
gap> S := Subalgebra( R, [e5] );;
gap> SetName( S, "<e5>" );
gap> RS := Cartesian( R, S );;
gap> SetName( RS, "GF(2^2)[k4] x <e5>" );
gap> act := AlgebraAction( R, RS, S );;
gap> bdy := AlgebraHomomorphismByFunction( S, R, r->r );
MappingByFunction( <e5>, GF(2^2)[k4], function( r ) ... end )
gap> IsAlgebraAction( act );
true
gap> IsAlgebraHomomorphism( bdy );
true
gap> XM := PreXModAlgebraByBoundaryAndAction( bdy, act );
[<e5>->GF(2^2)[k4]]
gap> IsXModAlgebra( XM );
true
gap> Display( XM );

Crossed module [<e5>->GF(2^2)[k4]] :-
: Source algebra has generators:
  [ (Z(2)^0)*<identity> of ...+(Z(2)^0)*f1+(Z(2)^0)*f2+(Z(2)^0)*f1*f2 ]
: Range algebra GF(2^2)[k4] has generators:
  [ (Z(2)^0)*<identity> of ..., (Z(2)^0)*f1, (Z(2)^0)*f2 ]
: Boundary homomorphism maps source generators to:
  [ (Z(2)^0)*<identity> of ...+(Z(2)^0)*f1+(Z(2)^0)*f2+(Z(2)^0)*f1*f2 ]
```

4.1.7 XModAlgebraByModule

▷ XModAlgebraByModule(M, R) (operation)

Let M be a R-module. Then $\mathscr{X} = (0 : M \to R)$ is a crossed module. Conversely, given a crossed module $\mathscr{X} = (\partial : M \to R)$, one can get that $\ker\partial$ is a $(R/\partial M)$-module.

Example

```
gap> ## example needed
```

4.1.8 Source (for crossed modules of commutative algebras)

▷ `Source(X0)` (attribute)
▷ `Range(X0)` (attribute)
▷ `Boundary(X0)` (attribute)
▷ `XModAlgebraAction(X0)` (attribute)

These four attributes are used in the construction of a crossed module $\mathscr{X}$ 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 $\mathscr{X}$;
- `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 $\mathscr{X}$;
- `Size2d(X)` is used for calculating the order of the crossed module $\mathscr{X}$;
- `Name(X)` is used for giving a name to the crossed module $\mathscr{X}$ 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> f := Boundary( XIAk4 );
MappingByFunction( I(GF5[k4]), GF5[k4], function( i ) ... end )
gap> Print( RepresentationsOfObject(XIAk4), "\n" );
[ "IsComponentObjectRep", "IsAttributeStoringRep", "IsPreXModAlgebraObj" ]
gap> props := [ "CanEasilyCompareElements", "CanEasilySortElements",
>  "IsDuplicateFree", "IsLeftActedOnByDivisionRing", "IsAdditivelyCommutative",
>  "IsLDistributive", "IsRDistributive", "IsPreXModDomain", "Is2dAlgebraObject",
>  "IsPreXModAlgebra", "IsXModAlgebra" ];;
gap> known := KnownPropertiesOfObject( XIAk4 );;
gap> ForAll( props, p -> (p in known) );
true
gap> Print( KnownAttributesOfObject(XIAk4), "\n" );
[ "Name", "Range", "Source", "Boundary", "Size2d", "XModAlgebraAction" ]
```

4.1.9 SubXModAlgebra

▷ `SubXModAlgebra(X0)` (operation)
▷ `IsSubXModAlgebra(X0)` (operation)

A crossed module $\mathscr{X}' = (\partial' : S' \to R')$ is a subcrossed module of the crossed module $\mathscr{X} = (\partial : S \to R)$ 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> Size( Je4 );
5
gap> SetName( Je4, "<e4>" );
gap> XJe4 := XModAlgebraByIdeal( Ak4, Je4 );
[ <e4> -> GF5[k4] ]
gap> Display( XJe4 );

Crossed module [<e4>->GF5[k4]] :-
: 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 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)^0)*<identity> of ...+(Z(5)^0)*f1+(Z(5)^2)*f2+(Z(5)^2)*f1*f2 ]

gap> IsSubXModAlgebra( XIAk4, XJe4 );
true
```

4.2 (Pre-)Crossed Module Morphisms

Let $\mathscr{X} = (\partial : S \to R)$, $\mathscr{X}' = (\partial' : S' \to R')$ be (pre)crossed modules and $\theta : S \to S'$, $\varphi : R \to R'$ be algebra homomorphisms. If

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

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

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

$$\begin{tikzcd} S \arrow[r, "\theta"] \arrow[d, "\partial"'] & S' \arrow[d, "\partial'"] \\ R \arrow[r, "\varphi"'] & R' \end{tikzcd} \qquad \begin{tikzcd} R \times S \arrow[r, "\varphi \times \theta"] \arrow[d] & R' \times S' \arrow[d] \\ S \arrow[r, "\theta"'] & S'. \end{tikzcd}$$

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(X0)` (method)
▷ `PreXModAlgebraMorphismByHoms(f, g)` (operation)
▷ `XModAlgebraMorphismByHoms(f, g)` (operation)

▷ IsPreXModAlgebraMorphism(*f*) (property)
▷ IsXModAlgebraMorphism(*f*) (property)
▷ Source(*m*) (attribute)
▷ Range(*m*) (attribute)
▷ IsTotal(*m*) (method)
▷ IsSingleValued(*m*) (method)
▷ Name(*m*) (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 := AllHomsOfAlgebras( IAc4, IBk4 );;
gap> theta := homIAIB[3];;
gap> homAB := AllHomsOfAlgebras( 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(X0)` (method)

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

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

is called the *kernel* of (θ,φ). Also, $\ker(\theta,\varphi)$ is an ideal of $\mathscr{X}$. 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(X0)` (operation)

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

$$\Im(\theta,\varphi) = (\partial'| : \Im\theta \to \Im\varphi)$$

is called the image of (θ,φ). Further, $\Im(\theta,\varphi)$ 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(m)` (attribute)
▷ `RangeHom(m)` (attribute)
▷ `IsInjective(m)` (property)
▷ `IsSurjective(m)` (property)
▷ `IsBijjective(m)` (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** (cat^1-algebras) and **XModAlg** (crossed modules) are naturally equivalent [Ell88]. This equivalence is outlined in what follows. For a given crossed module $(\partial : S \to R)$ we can construct the semidirect product $R \ltimes S$ thanks to the action of R on S. If we define $t,h : R \ltimes S \to R$ and $e : R \to R \ltimes S$ by

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

respectively, then $\mathscr{C} = (e;t,h : R \ltimes S \to R)$ is a cat$^1-$algebra.

Notice that h *is* an algebra homomorphism, since:

$$h(r_1r_2,\ r_1 \cdot s_2 + r_2 \cdot s_1 + s_1s_2) \ = \ r_1r_2 + r_1(\partial s_2) + r_2(\partial s_1) + (\partial s_1)(\partial s_2) \ = \ (r_1 + \partial s_1)(r_2 + \partial s_2).$$

Conversely, for a given cat^1-algebra $\mathscr{C} = (e;t,h : A \to R)$, the map $\partial : \ker t \to R$ 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 cat^1-algebra from a given crossed module of algebras. As an example we use the crossed module `XAB` constructed in `XModAlgebraByIdeal` (4.1.2) (The output from `Display` needs to be improved.)

Example

```
gap> CAB := Cat1AlgebraOfXModAlgebra( XAB );
[Algebra( GF(5), [ v.1, v.2, v.3, v.4, v.5 ] ) -> A(l,m)]
gap> Display( CAB );
```

```
Cat1-algebra [..=>A(l,m)] :-
:  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:
: range embedding maps range generators to:
  [ v.1, v.2 ]
: kernel has generators:
  Algebra( GF(5), [ v.4, v.5 ] )
```

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.

Example

```
gap> X3 := XModAlgebraOfCat1Algebra( C3 );
[ <algebra of dimension 3 over GF(2)> -> <algebra of dimension 3 over GF(2)> ]
gap> Display( X3 );

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, 12

[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. 12, 30

[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. 12

[Mos86] G. H. Mosa. *Higher dimensional algebroids and crossed complexes*. Ph.d. thesis, University of Wales, Bangor (U.K.), 1986. 12

[Oda09] A. Odabas. *Crossed Modules of Algebras with GAP*. Ph.d. thesis, Osmangazi University, Eskisehir, 2009. 4, 17, 23

[Por87] T. Porter. Some categorical results in the theory of crossed modules in commutative algebras. *J. Algebra*, 109:415–429, 1987. 4

[Whi49] J. H. C. Whitehead. Combinatorial homotopy II. *Bull. Amer. Math. Soc.*, 55:453–496, 1949. 4

Index