XModAlg

Crossed modules and cat1-algebras of commutative algebras

Version 1.12

14/11/2015

Zekeriya Arvasi
Alper Odabaş

Zekeriya Arvasi Email: zarvasi@ogu.edu.tr
Homepage: http://fef.ogu.edu.tr/zarvasi
Address: Department of Mathematics and Computer Science,
Osmangazi University, Eskişehir, Turkey

Alper Odabaş Email: aodabas@ogu.edu.tr
Homepage: http://fef.ogu.edu.tr/matbil/aodabas/
Address: Department of Mathematics and Computer Science,
Osmangazi University, Eskişehir, Turkey

Abstract

The XModAlg package provides functions for computation with crossed modules of commutative algebras and cat^1-algebras). The current version is 1.12, released 14th November 2015 for GAP 4.7.

Bug reports, suggestions and comments are, of course, welcome. Please contact the second author at aodabas@ogu.edu.tr.

Copyright

Acknowledgements

This documentation was prepared with the GAPDoc package of Frank Lübeck and Max Neunhöffer.

Both authors are very grateful to Chris Wensley 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, [AOUW15], [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 [AOUW15].

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].

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

Crossed modules

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

2.1 Definition and Examples

Let K be a fixed commutative ring with $1 \neq 0$. From now on, all K-algebras will be associative and commutative.

A *crossed module* is a K-algebra morphism $\mathscr{X} := (\partial : S \to R)$ with an 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 $\mathscr{X}$

In this definition we used the left action notation. In the category of commutative algebras the right and the left actions coincide.

We can produce crossed modules by using the following methods.

2.1.1 XModAlgebra

▷ `XModAlgebra(args)` (function)
▷ `XModAlgebraByBoundaryAndAction(bdy, act)` (operation)
▷ `XModAlgebraByIdeal(A, I)` (operation)
▷ `XModAlgebraByModule(M, R)` (operation)
▷ `XModAlgebraByCentralExtension(f)` (operation)
▷ `XModAlgebraByMultipleAlgebra(A)` (operation)

Here are the standard constructions which these operations implement:

- Let A be an algebra and I an ideal of A. Then $\mathscr{X} = (inc : I \to A)$ is a crossed module with the multiplication action 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.
- 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.
- Let $\partial : S \to R$ be a surjective algebra homomorphism. Define the action of R on S by $r \cdot s = \widetilde{r}s$ where $\widetilde{r} \in \partial^{-1}(r)$. Then $\mathscr{X} = (\partial : S \to R)$ is a crossed module with the defined action.

- Let S be a K-algebra such that $Ann(S) = 0$ or $S^2 = S$. Then $\partial : S \to M(S)$ is a crossed module, where $M(S)$ is the algebra of multipliers of S and ∂ is the canonical homomorphism, [AE03].

2.1.2 Source

▷ `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.

The following standard GAP operations have special XModAlg implementations:

- `Display(X)` is used to list the components of $\mathscr{X}$;
- `Size(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> A := GroupRing(GF(5),DihedralGroup(4));
<algebra-with-one over GF(5), with 2 generators>
gap> Size(A);
625
gap> SetName(A,"GF5[D4]");
gap> I := AugmentationIdeal(A);
<two-sided ideal in GF5[D4], (2 generators)>
gap> Size(I);
125
gap> SetName(I,"Aug");
gap> CM := XModAlgebraByIdeal(A,I);
[Aug->GF5[D4]]
gap> Display(CM);

Crossed module [Aug->GF5[D4]] :-
: Source algebra Aug has generators:
  [ (Z(5)^2)*<identity> of ...+(Z(5)^0)*f1, (Z(5)^2)*<identity> of ...
  +(Z(5)^0)*f2 ]
: Range algebra GF5[D4] 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> Size(CM);
[ 125, 625 ]
gap> f := Boundary(CM);
MappingByFunction( Aug, GF5[D4], function( i ) ... end )
gap> Print( RepresentationsOfObject(CM), "\n" );
[ "IsComponentObjectRep", "IsAttributeStoringRep", "IsPreXModAlgebraObj" ]
gap> props := [ "CanEasilyCompareElements", "CanEasilySortElements",
>  "IsDuplicateFree", "IsLeftActedOnByDivisionRing", "IsAdditivelyCommutative",
>  "IsLDistributive", "IsRDistributive", "IsPreXModDomain", "Is2dAlgebraObject",
>  "IsPreXModAlgebra", "IsXModAlgebra" ];;
gap> known := KnownPropertiesOfObject(CM);;
gap> ForAll( props, p -> (p in known) );
true
gap> Print( KnownAttributesOfObject(CM), "\n" );
[ "Name", "Size", "Range", "Source", "Boundary", "XModAlgebraAction" ]
```

2.1.3 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(I)[4];
(Z(5)^0)*<identity> of ...+(Z(5)^0)*f1+(Z(5)^2)*f2+(Z(5)^2)*f1*f2
gap> J := Ideal( I, [e4] );
<two-sided ideal in Aug, (1 generators)>
gap> Size(J);
5
gap> SetName( J, "<e4>" );
gap> PM := XModAlgebraByIdeal( A, J );
[<e4>->GF5[D4]]
gap> Display( PM );

Crossed module [<e4>->GF5[D4]] :-
: 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[D(4)] 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( CM, PM );
true
```

2.1.4 PreXModAlgebraByBoundaryAndAction

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

An R-algebra homomorphism $\mathscr{X} := (\partial : S \to R)$ which satisfy 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] );
<algebra over GF(2^2), with 1 generators>
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 ]
```

2.2 (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:

$$\begin{array}{ccc} S & \xrightarrow{\theta} & S' \\ {\scriptstyle\partial}\downarrow & & \downarrow{\scriptstyle\partial'} \\ R & \xrightarrow[\varphi]{} & R' \end{array} \qquad \begin{array}{ccc} R \times S & \xrightarrow{\varphi\times\theta} & R' \times S' \\ \downarrow & & \downarrow \\ S & \xrightarrow[\theta]{} & S'. \end{array}$$

In GAP we define the morphisms between algebraic structures such as cat^1-algebras and crossed modules and they are investigated by the function `Make2AlgMorphism`.

2.2.1 XModAlgebraMorphism

▷ `XModAlgebraMorphism(arg)` (function)
▷ `IdentityMapping(X0)` (operation)
▷ `PreXModAlgebraMorphismByHoms(f, g)` (operation)
▷ `XModAlgebraMorphismByHoms(f, g)` (operation)
▷ `IsPreXModAlgebraMorphism(f)` (property)
▷ `IsXModAlgebraMorphism(f)` (property)
▷ `Source(m)` (attribute)
▷ `Range(m)` (attribute)
▷ `IsTotal(m)` (property)
▷ `IsSingleValued(m)` (property)
▷ `Name(m)` (attribute)

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

Example

```
gap> A:=GroupRing(GF(2),CyclicGroup(4));
<algebra-with-one over GF(2), with 2 generators>
gap> B:=AugmentationIdeal(A);
<two-sided ideal in <algebra-with-one over GF(2), with 2 generators>,
(dimension 3)>
gap> X1:=XModAlgebra(A,B);
[Algebra( GF(2), [ (Z(2)^0)*<identity> of ...+(Z(2)^0)*f2,
(Z(2)^0)*f1+(Z(2)^0)*f2, (Z(2)^0)*f2+(Z(2)^0)*f1*f2
 ] )->AlgebraWithOne( GF(2), [ (Z(2)^0)*f1, (Z(2)^0)*f2 ] )]
gap> C:=GroupRing(GF(2),SmallGroup(4,2));
<algebra-with-one over GF(2), with 2 generators>
gap> D:=AugmentationIdeal(C);
```

```
<two-sided ideal in <algebra-with-one over GF(2), with 2 generators>,
(dimension 3)>
gap> X2:=XModAlgebra(C,D);
[Algebra( GF(2), [ (Z(2)^0)*<identity> of ...+(Z(2)^0)*f2,
(Z(2)^0)*f1+(Z(2)^0)*f2, (Z(2)^0)*f2+(Z(2)^0)*f1*f2
 ] )->AlgebraWithOne( GF(2), [ (Z(2)^0)*f1, (Z(2)^0)*f2 ] )]
gap> B = D;
false
gap> all_f := AllHomsOfAlgebras(A,C);;
gap> all_g := AllHomsOfAlgebras(B,D);;
gap> mor := XModAlgebraMorphism(X1,X2,all_g[1],all_f[2]);
[[..] => [..]]
gap> Display(mor);

Morphism of crossed modules :-
: Source = [Algebra( GF(2), [ (Z(2)^0)*<identity> of ...+(Z(2)^0)*f2,
  (Z(2)^0)*f1+(Z(2)^0)*f2, (Z(2)^0)*f2+(Z(2)^0)*f1*f2 ] )->AlgebraWithOne( GF(2),
  [ (Z(2)^0)*f1, (Z(2)^0)*f2 ] )] with generating sets:
  [ (Z(2)^0)*<identity> of ...+(Z(2)^0)*f2, (Z(2)^0)*f1+(Z(2)^0)*f2,
  (Z(2)^0)*f2+(Z(2)^0)*f1*f2 ]
  [ (Z(2)^0)*<identity> of ..., (Z(2)^0)*f1, (Z(2)^0)*f2 ]
:  Range = [Algebra( GF(2), [ (Z(2)^0)*<identity> of ...+(Z(2)^0)*f2,
  (Z(2)^0)*f1+(Z(2)^0)*f2, (Z(2)^0)*f2+(Z(2)^0)*f1*f2 ] )->AlgebraWithOne( GF(2),
  [ (Z(2)^0)*f1, (Z(2)^0)*f2 ] )] with generating sets:
  [ (Z(2)^0)*<identity> of ...+(Z(2)^0)*f2, (Z(2)^0)*f1+(Z(2)^0)*f2,
  (Z(2)^0)*f2+(Z(2)^0)*f1*f2 ]
  [ (Z(2)^0)*<identity> of ..., (Z(2)^0)*f1, (Z(2)^0)*f2 ]
: Source Homomorphism maps source generators to:
  [ <zero> of ..., <zero> of ..., <zero> of ... ]
: Range Homomorphism maps range generators to:
  [ (Z(2)^0)*<identity> of ..., (Z(2)^0)*<identity> of ...,
  (Z(2)^0)*<identity> of ... ]

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

2.2.2 Kernel

▷ `Kernel(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

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

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

Example

```
gap> X3 := Kernel(mor);
```

```
[Algebra( GF(2), [ (Z(2)^0)*<identity> of ...+(Z(2)^0)*f2,
(Z(2)^0)*f1+(Z(2)^0)*f2, (Z(2)^0)*f2+(Z(2)^0)*f1*f2
 ] )->Algebra( GF(2), [ (Z(2)^0)*f1+(Z(2)^0)*f2,
 (Z(2)^0)*f1+(Z(2)^0)*f1*f2, (Z(2)^0)*<identity> of ...+(Z(2)^0)*f1
 ] )]
gap> IsXModAlgebra(X3);
true
gap> Size(X3);
[ 8, 8 ]
gap> IsSubXModAlgebra(X1,X3);
true
```

2.2.3 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

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

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

In our package, the image of a crossed module homomorphism can be obtained by the command `Image2dAlgMapping`. The operation `Sub2dAlgObject` is effectively used for finding the kernel and image crossed modules induced from a given crossed module homomorphism.

2.2.4 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> theta := SourceHom(mor);
[ (Z(2)^0)*<identity> of ...+(Z(2)^0)*f2, (Z(2)^0)*f1+(Z(2)^0)*f2,
(Z(2)^0)*f2+(Z(2)^0)*f1*f2 ] ->
[ <zero> of ..., <zero> of ..., <zero> of ... ]
gap> phi := RangeHom(mor);
[ (Z(2)^0)*f1 ] -> [ (Z(2)^0)*<identity> of ... ]
gap> IsInjective(mor);
false
gap> IsSurjective(mor);
false
```

Chapter 3

Cat1-algebras

3.1 Definitions and examples

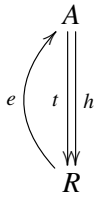
Algebraic structures which are equivalent to crossed modules of algebras include :

- 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.



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. The system which satisfy the condition **Cat1Alg1** is called a *precat^1-algebra*. The homomorphisms t,h and e are called the *tail*, *head* and *embedding* homomorphisms, respectively.

3.1.1 Cat1

▷ `Cat1(args)` (function)
▷ `PreCat1ByTailHeadEmbedding(t, h, e)` (operation)
▷ `PreCat1ByEndomorphisms(t, h)` (operation)
▷ `PreCat1AlgebraObj(C)` (operation)
▷ `PreCat1Algebra(C)` (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 operation PreCat1AlgebraObj is used for preserving the implementations,

3.1.2 Source

▷ Source(*C*) (attribute)
▷ Range(*C*) (attribute)
▷ Tail(*C*) (attribute)
▷ Head(*C*) (attribute)
▷ Embedding(*C*) (attribute)
▷ Kernel(*C*) (attribute)
▷ Boundary(*C*) (attribute)

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

Example

```
gap> A := GroupRing(GF(2),Group((1,2,3)(4,5)));
<algebra-with-one over GF(2), with 1 generators>
gap> R := GroupRing(GF(2),Group((1,2,3)));
<algebra-with-one over GF(2), with 1 generators>
gap> f := AllHomsOfAlgebras(A,R);
[ [ (Z(2)^0)*(1,3,2)(4,5) ] -> [ <zero> of ... ],
  [ (Z(2)^0)*(1,3,2)(4,5) ] -> [ (Z(2)^0)*() ],
  [ (Z(2)^0)*(1,3,2)(4,5) ] -> [ (Z(2)^0)*()+(Z(2)^0)*(1,2,3) ],
  [ (Z(2)^0)*(1,3,2)(4,5) ] -> [ (Z(2)^0)*()+(Z(2)^0)*(1,2,3)+(Z(2)^0)*(1,3,2) ],
  [ (Z(2)^0)*(1,3,2)(4,5) ] -> [ (Z(2)^0)*()+(Z(2)^0)*(1,3,2) ],
  [ (Z(2)^0)*(1,3,2)(4,5) ] -> [ (Z(2)^0)*(1,2,3) ],
  [ (Z(2)^0)*(1,3,2)(4,5) ] -> [ (Z(2)^0)*(1,2,3)+(Z(2)^0)*(1,3,2) ],
  [ (Z(2)^0)*(1,3,2)(4,5) ] -> [ (Z(2)^0)*(1,3,2)
    ] ]
gap> g := AllHomsOfAlgebras(R,A);
[ [ (Z(2)^0)*(1,2,3) ] -> [ <zero> of ... ],
  [ (Z(2)^0)*(1,2,3) ] -> [ (Z(2)^0)*() ],
  [ (Z(2)^0)*(1,2,3) ] -> [ (Z(2)^0)*()+(Z(2)^0)*(1,2,3) ],
  [ (Z(2)^0)*(1,2,3) ] -> [ (Z(2)^0)*()+(Z(2)^0)*(1,2,3)+(Z(2)^0)*(1,3,2) ],
  [ (Z(2)^0)*(1,2,3) ] -> [ (Z(2)^0)*()+(Z(2)^0)*(1,3,2) ],
  [ (Z(2)^0)*(1,2,3) ] -> [ (Z(2)^0)*(1,2,3) ],
  [ (Z(2)^0)*(1,2,3) ] -> [ (Z(2)^0)*(1,2,3)+(Z(2)^0)*(1,3,2) ],
  [ (Z(2)^0)*(1,2,3) ] -> [ (Z(2)^0)*(1,3,2) ] ]
gap> C4 := PreCat1ByTailHeadEmbedding(f[6],f[6],g[8]);
[AlgebraWithOne( GF(2), [ (Z(2)^0)*(1,2,3)(4,5) ] ) -> AlgebraWithOne( GF(2),
  [ (Z(2)^0)*(1,2,3) ] )]
gap> IsCat1Algebra(C4);
true
gap> Size(C4);
[ 64, 8 ]
```

```
gap> Display(C4);

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

3.1.3 Cat1AlgebraSelect

▷ `Cat1AlgebraSelect(`*gf, gpsize, gpnum, num*`)` (operation)

The Cat1Algebra 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(gf) returns the list of possible size of Galois field or the list of possible size of groups with given Galois field.
- Cat1AlgebraSelect(gf, gpsize) returns the list of possible size of group with given Galois fields or the list of possible number of groups with given Galois field and size of group.
- Cat1AlgebraSelect(gf, gpsize, gpnum) returns the list of possible number of group with given Galois field and size of group or the list of possible cat^1-structures with given Galois field and group.
- Cat1AlgebraSelect(gf, gpsize, gpnum, num) (or just Cat1Algebra(gf, gpsize, gpnum, num)) returns the chosen cat^1-algebra.

Now, we will give an example for the usage of this function.

Example

```
gap> C := Cat1AlgebraSelect(11);
|-------------------------------------------------------|
| 11 is invalid number for Galois Field (gf)            |
```

```
| Possible numbers for the gf in the Data :               |
|---------------------------------------------------------|
 [ 2, 3, 4, 5, 7 ]
Usage: Cat1Algebra( gf, 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( gf, 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( gf, 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( gf, gpsize, gpnum, num );
Algebra has generators [ (Z(2)^0)*(), (Z(2)^0)*(1,2,3)(4,5) ]
4
gap> C2 := 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> Size( C2 );
[ 4096, 1024 ]
gap> Display( C2 );

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

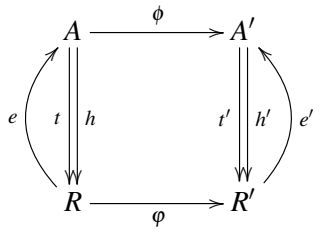
```
gap> C3 := Cat1AlgebraSelect( 2, 6, 2, 4 );;
gap> A3 := Source( C3 );
GF(2)_c6
gap> B3 := Range( C3 );
GF(2)_c3
gap> eA3 := Elements( A3 );;
gap> eB3 := Elements( B3 );;
gap> AA3 := Subalgebra( A3, [ eA3[1], eA3[2], eA3[3] ] );
<algebra over GF(2), with 3 generators>
gap> [ Size(A3), Size(AA3) ];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



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 $(\phi, \varphi)$is called a cat^1-algebra morphism.

3.2.1 Cat1AlgebraMorphism

▷ `Cat1AlgebraMorphism(arg)` (operation)
▷ `IdentityMapping(C)` (operation)
▷ `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

▷ Source(*m*) (attribute)
▷ Range(*m*) (attribute)
▷ IsTotal(*m*) (attribute)
▷ IsSingleValued(*m*) (property)
▷ Name(*m*) (attribute)
▷ 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> Image2dAlgMapping(m);
[GF(3)_c2^3=>GF(3)_c2^3]
gap> IsSurjective(m);
false
gap> IsInjective(m);
true
gap> IsBijective(m);
false
```

3.3 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 $(\partial : A \to R)$ we can construct the semidirect product $R \ltimes A$ thanks to the action of R on A. If we define $t,h : R \ltimes A \to R$

and $e : R \to R \ltimes A$ by

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

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

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 and ∂ is the restriction of h to $\ker t$.

3.3.1 PreCat1ByPreXMod

▷ `PreCat1ByPreXMod(`*X0*`)` (operation)
▷ `PreXModAlgebraByPreCat1Algebra(`*C*`)` (operation)
▷ `Cat1AlgebraByXModAlgebra(`*X0*`)` (operation)
▷ `XModAlgebraByCat1Algebra(`*C*`)` (operation)

These operations are used for constructing a cat^1-algebra from a given crossed module, and conversely.

Example

```
gap> CXM := Cat1AlgebraByXModAlgebra( XM );
[GF(2^2)[k4] IX <e5> -> GF(2^2)[k4]]
gap> X3 := XModAlgebraByCat1Algebra( C3 );
[Algebra( GF(2), [ <zero> of ..., <zero> of ..., <zero> of ...
 ] )->Algebra( GF(2),
[ (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) ] )]
gap> Display( X3 );

Crossed module [..->..] :-
: Source algebra has generators:
  [ <zero> of ..., <zero> of ..., <zero> of ... ]
: Range 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) ]
: Boundary homomorphism maps source generators to:
  [ <zero> of ..., <zero> of ..., <zero> of ... ]
```

Since all these operations are linked to the functions `Cat1Algebra` and `XModAlgebra`, all of them can be done by using these two functions. We may also use the function `Cat1Algebra` instead of the operation `Cat1AlgebraSelect`.

References

[AE03] Z. Arvasi and U. Ege. Annihilators, multipliers and crossed modules. *Applied Categorical Structures*, 11:487–506, 2003. 4, 6

[AOUW15] M. Alp, A. Odabas, E. O. Uslu, and C. D. Wensley. *XMod: Crossed Modules and Cat1-groups in GAP*, 2015. 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, 19

[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, 8, 17

[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