Computing 2-Dimensional Algebras: Crossed Modules and Cat\(^1\)-Algebras

Z. Arvasi and A. Odabaş

Abstract

In this paper, we described the GAP implementation of crossed modules of commutative algebras and cat\(^1\)-algebras and their equivalence.

Keywords: GAP, group algebra, crossed module, cat\(^1\)-algebra.

1 Introduction

In 1950 S. MacLane and J. H. C. Whitehead, [15], suggested that crossed modules modeled homotopy 2-types. Later crossed modules had been considered as “2-dimensional groups”, [7, 8]. The commutative algebra version of this construction has been adapted by T. Porter, [4, 14]. This algebraic version is called “combinatorial algebra theory” which contains potentially important new ideas (see [4, 5, 6].)

A share package XMod, [1, 3], for the GAP3 computational group theory language was described by M. Alp and C. Wensley. The 2-dimensional part of this programme contains functions for computing crossed modules and cat\(^1\)-groups and their morphisms. This package was rewritten for GAP4, [10], containing functions for crossed square with utility functions, [2].

In this paper we describe for GAP with XModAlg including functions for computing crossed module of algebras, cat\(^1\)-algebras and their morphisms by analogy with “computational group theory”. The tools needed are the group algebras in which the group algebra functor \(K(.) : Gr \rightarrow Alg\) is left adjoint to the unit group functor \(U(.) : Alg \rightarrow Gr\).

One of the main result is the GAP implementation of the equivalent categories XModAlg (crossed modules algebras) and Cat\(^1\)-Alg (cat\(^1\)-algebras) has been shown in this work. Algorithms of the GAP implementations in this paper are deeply analyzed in A. Odabas’s Ph.D. thesis, [13].
Unfortunately the GAP is not quiet suitable for commutative algebras since no standard GAP function yet exist for computing semidirect products, endomorphisms and multiplier algebras. But we only manage the multiplier algebras.

2 Crossed Modules

In this section we will present the notion of crossed modules on commutative algebra and adapted in to computer environment.

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 \( \mathcal{X} := (\partial : S \to R) \) with an action of \( R \) on \( S \) satisfying

\[
\text{XModAlg1 : } \partial(r \cdot s) = r\partial(s), \quad \text{XModAlg2 : } \partial(s) \cdot s' = ss',
\]

for all \( s, s' \in S \), \( r \in R \), \( \partial \) is called the boundary map of \( \mathcal{X} \).

We can produce crossed modules by using many methods as follows:

| Method                                           | Function/Operation |
|--------------------------------------------------|--------------------|
| XModAlg(arg)                                    | (function)         |
| XModAlgByBoundaryAndAction(bdy,act)              | (operation)        |
| XModAlgByIdeal(A,I)                             | (operation)        |
| XModAlgByModule(M,R)                            | (operation)        |
| XModAlgByCentralExtension(f)                    | (operation)        |
| XModAlgByMultipleAlgebra(A)                     | (operation)        |

Examples of crossed modules:

i. Let \( A \) be an algebra and \( I \) is an ideal of \( A \). Then \( \mathcal{X} = (\text{inc} : I \to A) \) is a crossed module with the multiplication action of \( A \) on \( I \). Conversely, we induce an ideal from a given crossed module. Indeed, for a given crossed module \( \mathcal{X} = (\partial : S \to R) \), \( \partial(S) \) is an ideal of \( R \).

ii. Let \( M \) be a \( R \)-module then \( \mathcal{X} = (0 : M \to R) \) is a crossed module. Conversely given \( \mathcal{X} = (\partial : M \to R) \) a crossed module one can get that \( \text{Ker}\partial \) is a \( R/\partial M \)-module.

iii. Let \( \partial : S \to R \) be a surjective algebra homomorphism. Define the action of \( R \) on \( S \) by \( r \cdot s = \tilde{r}s \) where \( \tilde{r} \in \partial^{-1}(r) \). Then \( \mathcal{X} = (\partial : S \to R) \) is a crossed module with the defined action.

iv. Let \( S \) be a \( k \)-algebra such that \( \text{Ann}(S) = 0 \) 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 \( \partial \) is the canonical homomorphism, [6].
The implementations,

\begin{verbatim}
Source(X) (attribute)
Range(X) (attribute)
Boundary(X) (attribute)
XModAlgAction(X) (attribute)
\end{verbatim}

are the attributes which used for 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** \((X)\), boundary map of the crossed module \(\mathcal{X}\),

- **XModAlgAction** \((X)\) is the action used in the crossed module.

The implementations

- **Display** \((X)\), is used for the details of \(\mathcal{X}\),

- **Size** \((X)\), is used for calculating the order of the crossed module \(\mathcal{X}\),

- **Name** \((X)\), is used for giving a name for the crossed module \(\mathcal{X}\) by associating the names of source and range algebras.

are additional attributes of a crossed module \(\mathcal{X}\).

In the following example, we construct a crossed module by using the algebra \(GF_5\text{[D}(4)\text{]}\) and its augmentation ideal. Also we show usage of the previous attributes.

\begin{verbatim}
gap> A:=GroupRing(GF(5),DihedralGroup(4));
<algebra-with-one over GF(5), with 2 generators>
gap> Size(A);
625
gap> eA:=Elements(A);;
gap> SetName(A,"GF5[D(4)]");
gap> I:=AugmentationIdeal(A);
<two-sided ideal in GF5[D(4)], (2 generators)>
gap> Size(I);
125
gap> SetName(I,"Aug");
gap> CM:=XModAlgByIdeal(A,I);
[Aug->GF5[D(4)]]
gap> Display(CM);
\end{verbatim}
Crossed module \([\text{Aug}\rightarrow\text{GF5}[D(4)]]\) :-

: Source group \(\text{Aug}\) has generators:
\[
\text{[ (Z(5)^2)*<identity> of ...+(Z(5)^0)*f1, (Z(5)^2)*<identity> of ...+(Z(5)^0)*f2 ]}
\]

: Range group \(\text{GF5}[D(4)]\) has generators:
\[
\text{[ (Z(5)^0)*<identity> of ..., (Z(5)^0)*f1, (Z(5)^0)*f2 ]}
\]

: Boundary homomorphism maps source generators to:
\[
\text{[ (Z(5)^2)*<identity> of ...+(Z(5)^0)*f1, (Z(5)^2)*<identity> of ...+(Z(5)^0)*f2 ]}
\]

gap> \text{Size(CM)};
\[ 125, 625 \]
gap> \text{Print(RepresentationsOfObject(CM),"n");}
\[
\text{[ "IsComponentObjectRep", "IsAttributeStoringRep", "IsPreXModAlgObj" ]}
\]
gap> \text{Print(KnownPropertiesOfObject(CM),"n");}
\[
\text{[ "Is2dAlgObject", "IsPreXModAlg", "IsXModAlg" ]}
\]
gap> \text{Print(KnownAttributesOfObject(K),"n");}
\[
\text{[ "Range", "Source", "Boundary", , "XModAlgAction", "Cat1AlgOfXModAlg" ]}
\]

**Definition 1** A crossed module \(\mathcal{X}' = (\partial' : S' \rightarrow R')\) is a crossed submodule 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 implementations

| SubXModAlg(X) | (operation) |
| IsSubXModAlg(X) | (attribute) |

are for constructing the crossed submodules of a given crossed module \(\mathcal{X}\) and for additional attributes. \text{Display}(X)\ can be used for the details of \(\mathcal{X}'\).

\begin{verbatim}
gap> eI:=\text{Elements}(I);
gap> J:=\text{Ideal}(I,[eI[4]]);
<two-sided ideal in Aug, (1 generators)>
gap> J=I;
false
gap> \text{Size}(J);
5
gap> \text{IsIdeal}(I,J);
true
gap> \text{IsIdeal}(A,J);
\end{verbatim}
true

\texttt{gap> PM:=XModAlg(A,J);}
\[
\text{[Algebra( GF(5),}
\begin{align*}
&\text{[ (Z(5)^0)*<identity> of ...+(Z(5)^0)*f1+(Z(5)^2)*f2+(Z(5)^2)*f1*f2} \\
&\quad \text{ ] )->GF5[D(4)]]}
\end{align*}
\]
\texttt{gap> Display(PM);}

Crossed module \(\[\rightarrow GF5[D(4)]\] :-

: Source group has generators:
\[
\text{[ (Z(5)^0)*<identity> of ...+(Z(5)^0)*f1+(Z(5)^2)*f2+}
\begin{align*}
&\text{ (Z(5)^2)*f1*f2 ]}
\end{align*}
\]

: Range group \(GF5[D(4)]\) has generators:
\[
\text{[ (Z(5)^0)*<identity> of ..., (Z(5)^0)*f1,}
\begin{align*}
&\text{ (Z(5)^0)*f2 ]}
\end{align*}
\]

: Boundary homomorphism maps source generators to:
\[
\text{[ (Z(5)^0)*<identity> of ...+(Z(5)^0)*f1+(Z(5)^2)*f2+}
\begin{align*}
&\text{ (Z(5)^2)*f1*f2 ]}
\end{align*}
\]

\texttt{gap> IsSubXModAlg(CM,PM);}
true

\textbf{Definition 2} An \(R\)-algebra homomorphism \(X := (S \xrightarrow{\partial} R)\) which satisfy the condition \(XModAlg1\) is called a precrossed module.

\[\text{PreXModAlgByBoundaryAndAction(bdy,act)}\quad \text{(operation)}
\]
\[\text{IsPreXModAlg(X)}\quad \text{(operation)}
\]

\texttt{gap> G:=SmallGroup(4,2);}
\texttt{<pc group of size 4 with 2 generators>}
\texttt{gap> F:=GaloisField(4);
GF(2^2)}
\texttt{gap> A:=GroupRing(F,G);
<algebra-with-one over GF(2^2), with 2 generators>}
\texttt{gap> Size(A);}
256
\texttt{gap> eA:=Elements(A);;}
\texttt{gap> B:=Subalgebra(A,[eA[5]]);}
\texttt{<algebra over GF(2^2), with 1 generators>}
\texttt{gap> AB:=Cartesian(A,B);;}
\texttt{gap> act:=AlgebraAction(A,AB,B);;}
\texttt{gap> bdy:=AlgebraHomomorphismByFunction(B,A,i->i);}
\texttt{MappingByFunction( <algebra of dimension 1 over GF(2^2)>, <algebra-with-one of dimension 4 over GF(2^2)>,
function( i ) ... end )}
gap> IsAlgebraAction(act); true
gap> IsAlgebraHomomorphism(bdy); true
gap> PM:=PreXModAlgByBoundaryAndAction(bdy,act); <enumerator>
gap> IsPreXModAlg(PM); true

The details of these implementations can be found in [13].

2.1 (Pre) Crossed Module Morphism

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

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

for all $r \in R$, $s \in S$, then the pair $(\theta, \varphi)$ is called a morphism between $\mathcal{X} = (\partial : S \rightarrow R)$ and $\mathcal{X}' = (\partial' : S' \rightarrow R')$.

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

\[
\begin{array}{ccc}
S & \xrightarrow{\theta} & S' \\
\downarrow{\partial} & & \downarrow{\partial'} \\
R & \xrightarrow{\varphi} & R'
\end{array}
\hspace{1cm}
\begin{array}{ccc}
R \times S & \xrightarrow{\varphi \times \theta} & R' \times S' \\
\downarrow{\partial \times \theta} & & \downarrow{\partial' \times \theta} \\
S & \xrightarrow{\theta} & S'
\end{array}
\]

In GAP, we define the morphisms between algebraic structures such as $\text{cat}^1$-algebras and crossed modules and they are investigated by the implementation $\text{Make2AlgMorphism}$. The main part of the implementation $\text{Make2AlgMorphism}$ given below.

```gap
InstallMethod( Make2dAlgMapping, 
    "for 2d-object, 2d-object, homomorphism, homomorphism", true, 
    [ Is2dAlgObject, Is2dAlgObject, 
      IsAlgebraHomomorphism, IsAlgebraHomomorphism ], 0,
    function( src, rng, shom, rhom )
      local filter, fam, mor, ok;
      fam := FamilyObj( [ src, rng, shom, rhom ] );
      filter := Is2dAlgMappingRep;
      ok := ( ( Source( src ) = Source( shom ) ) and
```
( \text{Range}( \text{src} ) = \text{Source}( \text{rhom} ) ) \text{ and } \\
( \text{Source}( \text{rng} ) = \text{Range}( \text{shom} ) ) \text{ and } \\
( \text{Range}( \text{rng} ) = \text{Range}( \text{rhom} ) ) ; \\
\text{if not ok then} \\
\text{Info}( \text{InfoXModAlg}, 2, "sources and ranges do not match" ); \\
\text{return fail;} \\
\text{fi;} \\
\text{mor := rec();} \\
\text{ObjectifyWithAttributes}( \text{mor}, \\
\text{NewType}( \text{fam}, \text{filter} ), \\
\text{Source}, \text{src}, \\
\text{Range}, \text{rng}, \\
\text{SourceHom}, \text{shom}, \\
\text{RangeHom}, \text{rhom} ); \\
\text{return mor;} \\
\end

We have the following implementations for constructing crossed module homomorphisms.

\begin{verbatim}
XModAlgMorphism(arg)
IdentityMapping(CM)
PreXModAlgMorphismByHoms(f,g)
XModAlgMorphismByHoms(f,g)
IsPreXModAlgMorphism(t)
IsXModAlgMorphism(h)
\end{verbatim}

On the other hand, the following implementations has the attributes for constructing homomorphism \( m \) of the crossed module.

\begin{verbatim}
Source(m)
IsTotal(m)
IsSingleValued(m)
Name(m)
Boundary(m)
\end{verbatim}

In the following example, we construct a crossed module morphism

\begin{verbatim}
gap> A:=GroupRing(GF(2),CyclicGroup(6));
<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 5)>
\end{verbatim}
\texttt{gap> CM:=XModAlg(A,B);}
\texttt{[Algebra( GF(2), [ (Z(2)^0)*\text{id} of \ldots+(Z(2)^0)*f1,}
\texttt{ (Z(2)^0)*f1+(Z(2)^0)*f2, (Z(2)^0)*f2+(Z(2)^0)*f1*f2,}
\texttt{ (Z(2)^0)*f1*f2+(Z(2)^0)*f2^2, (Z(2)^0)*f2^2+(Z(2)^0)*f1*f2^2}
\texttt{ ] )->AlgebraWithOne( GF(2), [ (Z(2)^0)*f1, (Z(2)^0)*f2 ] )]}
\texttt{gap> SetName(CM,"GF_2C_6");}
\texttt{gap> Display(CM);}
\texttt{Crossed module GF_2C_6 :-}
\texttt{ : Source group has generators:}
\texttt{ [ (Z(2)^0)*\text{id} of \ldots+(Z(2)^0)*f1, (Z(2)^0)*f1+(Z(2)^0)*f2,}
\texttt{ (Z(2)^0)*f2+(Z(2)^0)*f1*f2, (Z(2)^0)*f1*f2+(Z(2)^0)*f2^2,}
\texttt{ (Z(2)^0)*f2^2+(Z(2)^0)*f1*f2^2 ]}
\texttt{ : Range group has generators:}
\texttt{ [ (Z(2)^0)*\text{id} of \ldots, (Z(2)^0)*f1, (Z(2)^0)*f2 ]}
\texttt{ : Boundary homomorphism maps source generators to:}
\texttt{ [ (Z(2)^0)*\text{id} of \ldots+(Z(2)^0)*f1, (Z(2)^0)*f1+(Z(2)^0)*f2,}
\texttt{ (Z(2)^0)*f2+(Z(2)^0)*f1*f2, (Z(2)^0)*f1*f2+(Z(2)^0)*f2^2,}
\texttt{ (Z(2)^0)*f2^2+(Z(2)^0)*f1*f2^2 ]}
\texttt{gap> f:=IdentityMapping(CM);}
\texttt{[[..] => [..]]}
\texttt{gap> IsPreXModAlgMorphism(f);}
\texttt{true}
\texttt{gap> IsXModAlgMorphism(f);}
\texttt{true}
\texttt{gap> Display(f);}
\texttt{Morphism of crossed modules :-}
\texttt{ : Source = [Algebra( GF(2), [ (Z(2)^0)*\text{id} of \ldots+(Z(2)^0)*f1,}
\texttt{ (Z(2)^0)*f1+(Z(2)^0)*f2, (Z(2)^0)*f2+(Z(2)^0)*f1*f2,}
\texttt{ (Z(2)^0)*f1*f2+(Z(2)^0)*f2^2, (Z(2)^0)*f2^2+(Z(2)^0)*f1*f2^2 ]
\texttt{ ] )]->AlgebraWithOne( GF(2), [ (Z(2)^0)*f1, (Z(2)^0)*f2 ] )]}
\texttt{ : Range = Source}
\texttt{ : Source Homomorphism maps source generators to:}
\texttt{ [ (Z(2)^0)*\text{id} of \ldots+(Z(2)^0)*f1, (Z(2)^0)*f1+(Z(2)^0)*f2,}
\texttt{ (Z(2)^0)*f2+(Z(2)^0)*f1*f2, (Z(2)^0)*f1*f2+(Z(2)^0)*f2^2,}
\texttt{ (Z(2)^0)*f2^2+(Z(2)^0)*f1*f2^2 ]}
\texttt{ : Range Homomorphism maps range generators to:}
\texttt{ [ (Z(2)^0)*\text{id} of \ldots, (Z(2)^0)*f1, (Z(2)^0)*f2 ]}
\texttt{gap> IsTotal(f);}
\texttt{true}
\texttt{gap> IsSingleValued(f);}
**Definition 3** Let \((\theta, \varphi) : \mathcal{X} = (\partial : S \to R) \to \mathcal{X}' = (\partial' : S' \to R')\) be a crossed module homomorphism. Then the crossed module 
\[ \text{ker}(\theta, \varphi) = (\partial : \text{ker}\theta \to \text{ker}\varphi) \]
is called the kernel of \((\theta, \varphi)\). Also, \(\text{ker}(\theta, \varphi)\) is an ideal of \(\mathcal{X}\).

While constructing program complex, the command names we used are chosen from well-known GAP commands for avoiding confusion. For this, the implementation \texttt{Kernel} is used to find the kernel of a crossed module homomorphism. The following settings are for GAP to know whether or not this implementation belongs to package.

```gap
InstallOtherMethod( Kernel, "generic method for 2d-mappings", true, [ Is2dAlgMapping ], 0,
```

The kernel of the homomorphism given below is found by using the implementation \texttt{Kernel}.

```gap
gap> PM:=Kernel(f);
[Algebra( GF(2), [], <zero> of ... )->Algebra( GF(2), [], <zero> of ... )]
gap> IsXModAlg(PM);
true
gap> IsSubXModAlg(CM,PM);
true
```

**Definition 4** Let \((\theta, \varphi) : \mathcal{X} = (\partial : S \to R) \to \mathcal{X}' = (\partial' : S' \to R')\) be a crossed module homomorphism. Then the crossed module 
\[ \text{Im}(\theta, \varphi) = (\partial' : \text{Im}\theta \to \text{Im}\varphi) \]
is called the image of \((\theta, \varphi)\). \(\text{Im}(\theta, \varphi)\) is a subcrossed module of \((S', R', \partial')\).

In GAP, image of a crossed module homomorphism can be obtained by the command \texttt{Image2dAlgMapping}. The implementation \texttt{Sub2dAlgObject} is effectively used for finding the kernel and image crossed modules induced from a given crossed modules homomorphism.

**Definition 5** Let \((\theta, \varphi)\) be a homomorphism of crossed modules. If the homomorphisms \(\theta\) and \(\varphi\) are injective (surjective) then \((\theta, \varphi)\) is called injective (surjective).
The implementations SourceHom and RangeHom are used for finding the crossed module morphism pairs. With these implementations, the functions given below give rise to investigate the properties of crossed module morphisms.

\begin{verbatim}
IsInjective(m)  (operation)
IsSurjective(m)  (operation)
IsBijective(m)  (operation)
\end{verbatim}

\begin{verbatim}
gap> theta:=SourceHom(f);
IdentityMapping( <two-sided ideal in <algebra-with-one of dimension 6 over GF(2)>, (dimension 5)> )
gap> phi:=RangeHom(f);
IdentityMapping( <algebra-with-one of dimension 6 over GF(2)> )
gap> IsInjective(f);  true
gap> IsSurjective(f);  true
gap> IsBijective(f);  true
\end{verbatim}

3  Cat\textsuperscript{1}- Algebras

Algebraic structures which are equivalent to crossed modules of algebras, namely;

- cat\textsuperscript{1}-algebras. (Ellis, [9])
- simplicial algebras with Moore complex of length 1 (Z. Arvasi and T. Porter, [4])
- algebra-algebroids (Gaffar Musa’s Ph.D. thesis, [12])

In this section cat\textsuperscript{1}-algebras and their morphisms have been implemented in GAP.

The notion of cat\textsuperscript{1}-groups was defined as an algebraic model of 2-types by Loday in [11]. Then, Ellis defined the cat\textsuperscript{1}-algebras in [9].

Let $A$ and $R$ be $k$-algebras,
$s, t : A \rightarrow R$ be surjections and $e : R \rightarrow A$ is inclusion. If the conditions,

\[ \textbf{Cat1Alg1} : \ se = id_R = te \quad \textbf{Cat1Alg2} : \ (\ker s)(\ker t) = \{0_A\} \]

satisfied, then the algebraic system $C := (e; t, s : A \rightarrow R)$ is called a cat$^1$-algebra. The system which satisfy the condition Cat1Alg1 is called a precat$^1$-algebra. The homomorphisms $s, t$ and $e$ are called as initial, terminal and embedding homomorphisms, respectively. The implementations,

\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{Cat1Alg}(s,t,e) \hspace{1cm} \text{(function)} \\
\textbf{PreCat1AlgByTailHeadEmbedding}(s,t,e) \hspace{1cm} \text{(operation)} \\
\textbf{PreCat1AlgByEndomorphisms}(s,t) \hspace{1cm} \text{(operation)} \\
\textbf{PreCat1AlgObj}(C) \hspace{1cm} \text{(operation)} \\
\textbf{PreCat1Alg}(C) \hspace{1cm} \text{(operation)} \\
\textbf{IsIdentityCat1Alg}(C) \hspace{1cm} \text{(operation)} \\
\textbf{IsCat1Alg}(C) \hspace{1cm} \text{(operation)} \\
\textbf{IsPreCat1Alg}(C) \hspace{1cm} \text{(operation)} \\
\hline
\end{tabular}
\end{center}

are used for construction of precat$^1$ and cat$^1$-algebra structures. The function \textbf{Cat1Alg} selects the operation from the above implementations up to user's input. The operation \textbf{PreCat1AlgObj} is used for preserving the implementations,

\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{Source}(C) \hspace{1cm} \text{(attribute)} \\
\textbf{Range}(C) \hspace{1cm} \text{(attribute)} \\
\textbf{Tail}(C) \hspace{1cm} \text{(attribute)} \\
\textbf{Head}(C) \hspace{1cm} \text{(attribute)} \\
\textbf{Kernel}(C) \hspace{1cm} \text{(attribute)} \\
\textbf{Boundary}(C) \hspace{1cm} \text{(attribute)} \\
\hline
\end{tabular}
\end{center}

of the cat$^1$-algebra structure. For example

\begin{verbatim}
gap> H:=GF(4); GF(2^2) gap> k4:=Group([[1,2],[3,4]]); Group([[1,2],[3,4]]) gap> R:=GroupRing(H,k4); <algebra-with-one over GF(2^2), with 2 generators> gap> Size(R); 256 gap> gR:=GeneratorsOfAlgebra(R); [ (Z(2)^0)^1*(1,2), (Z(2)^0)^1*(3,4) ] gap> f:=AlgebraHomomorphismByImages(R,R,gR,gR); \end{verbatim}
In general, the three morphisms used for constructing the cat$^1$-algebra structure do not satisfy the conditions. A list of finite dimensional group is obtained for using the program complex more effectively.

By using the groups and Galois fields, we obtain the list of the cat$^1$-algebras of group algebras and insert the list in to the program complex. The GAP users can easily obtain finite cat$^1$-algebras by using this list. For this, we constructed the function Cat1AlgSelect and link it to the function Cat1Alg. There are four basic parameters for using the function Cat1AlgSelect.

\text{Cat1AlgSelect}(gf, size, gpnum, num) (operation)

- $gf$ : dimension of the Galois field,
- $size$ : dimension of the group that will used,
- $gpnum$ : Order of the group with same dimension in the list.
• **num**: order of the cat\(^1\)-algebra constructed up to the given first three values.

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

```gap
gap> C:=Cat1AlgSelect(4,6,2,2);
[GF(2^2)_c6 -> GF(2^2)_triv]
gap> Size(C);[4096, 4]
gap> Display(C);
Pre-cat1-algebra [GF(2^2)_c6=>GF(2^2)_triv] :-
: 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)*() ]
: 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)*(1,2,3)(4,5), (Z(2)^0)*()+(Z(2)^0)*(1,3,2),
    (Z(2)^0)*()+(Z(2)^0)*(4,5), (Z(2)^0)*()+(Z(2)^0)*(1,2,3),
    (Z(2)^0)*()+(Z(2)^0)*(1,3,2)(4,5) ]
: boundary homomorphism maps generators of kernel to:
  [ <zero> of ..., <zero> of ..., <zero> of ..., <zero> of ...
    , <zero> of ... ]
: kernel embedding maps generators of kernel to:
  [ (Z(2)^0)*()+(Z(2)^0)*(1,2,3)(4,5), (Z(2)^0)*()+(Z(2)^0)*(1,3,2),
    (Z(2)^0)*()+(Z(2)^0)*(4,5), (Z(2)^0)*()+(Z(2)^0)*(1,2,3),
    (Z(2)^0)*()+(Z(2)^0)*(1,3,2)(4,5) ]
```

We give additional missions to the function **Cat1AlgSelect** to give more information about the list which was inserted to the program complex. At this vein, the function **Cat1AlgSelect** works without inputting all four parameters given above. In the usage of single parameter, last three parameters assumed to be zero and obtained information about the largest dimensions of groups that can be used for the Galois field with given dimension.

For example,

```gap
gap> Cat1AlgSelect(11);
There are groups having orders maximum 9 for GF(11) in the list.
```
In two parameter usage, last two parameters assumed to be zero, and obtained information about the number of groups that can be used for the given dimensional Galois field.

For example,

```
gap> Cat1AlgSelect(11,8);
0 is a invalid gnum number.
where 0 < gpnum <= 5 for gpsize 8.
Usage:  Cat1Alg( GF(num), gpsize, gpnum, num );
[ "GF(11)_c8","GF(11)_c4^2","GF(11)_d8","GF(11)_q8","GF(11)_c2^3" ]
```

In three parameter usage, the last parameter assumed to be zero, and obtained information about the number of cat$^1$-algebras and their morphisms which can be constructed by Galois field with the given dimensional and group.
The user make selection about the last parameter by this way.

For example,

```
gap> Cat1AlgSelect(11,8,5);
There are 13 cat1-structures for the algebra GF(11)_c2^3.

| Range Alg, Tail Genimg, Head Genimg |
|--------------------------------------|
| GF(11)_c2^3, identity map identity map |
| GF(11)_triv, [1,1,1,1], [1,1,1,1] |
| GF(11)_c2, [1,1,1,2], [1,1,1,2] |
| GF(11)_c2, [1,1,2,2], [1,1,2,2] |
| GF(11)_c2, [1,2,2,2], [1,2,2,2] |
| GF(11)_c2, [1,2,1,2], [1,2,1,2] |
| GF(11)_c2, [1,2,1,2], [1,1,1,2] |
| GF(11)_c2, [1,2,2,2], [1,1,2,2] |
| GF(11)_c2, [1,2,2,2], [1,1,1,2] |
| GF(11)_k4, [1,3,2,3], [1,3,2,3] |
| GF(11)_k4, [1,3,2,3], [1,1,2,3] |
| GF(11)_k4, [1,3,2,3], [1,1,2,3] |
|-----------------------------|

Usage:  Cat1Alg( GF(num), gpsize, gpnum, num );
Algebra has generators [ (Z(11)^0)*(), (Z(11)^0)*(1,2), (Z(11)^0)*(3,4), (Z(11)^0)*(5,6) ]
```
In the four parameter usage, if the $n$th parameter, $n = 2, 3, 4$ is not compatible with the list, then the output constructed up to the $n - 1$ parameter usage.

**Definition 6** Let $\mathcal{C} = (e; t, s : A \to R)$ be a cat$^1$-algebra. $A'$ and $R'$ are subalgebras of $A$ and $R$, respectively. If the restriction morphisms

$$t' = t|_{A'} : A' \to R', \quad s' = s|_{A'} : A' \to R', \quad e' = e|_{R'} : R' \to A'$$

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

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

The implementations

| SubCat1Alg(arg) | (operation) |
| SubPreCat1Alg(arg) | (operation) |
| IsSubCat1Alg(C,S) | (operation) |
| IsSubPreCat1Alg(C,S) | (operation) |

are used for constructing subcat$^1$-algebras of a given cat$^1$-algebra.

gap> C:=Cat1AlgSelect(2,6,2,4);
[GF(2)_c6 -> GF(2)_c3]
gap> Display(C);

Cat1-algebra [GF(2)_c6=>GF(2)_c3] :-
: 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)*(1,2,3) ]
: head homomorphism maps source generators to:
  [ (Z(2)^0)*(), (Z(2)^0)*(1,2,3) ]
: range embedding maps range generators to:
  [ (Z(2)^0)*(), (Z(2)^0)*(1,2,3) ]
: 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)*(1,2,3)(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) ]

gap> A:=Source(C);
GF(2)_c6
gap> B:=Range(C);
GF(2)_c3
gap> eA:=Elements(A);;
gap> eB:=Elements(B);;
gap> AA:=Subalgebra(A,[eA[1],eA[2],eA[3]]);
<algebra over GF(2), with 3 generators>
gap> A=AA;
false
gap> BB:=Subalgebra(B,[eB[1],eB[2]]);
<algebra over GF(2), with 2 generators>
gap> BB=B;
false
gap> CC:=SubCat1Alg(C,AA,BB);
[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> IsSubCat1Alg(C,CC);
true
gap> Display(CC);
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:
3.1 Cat¹- Algebra Morphism

Let \( C = (e; t, s : A \rightarrow R) \), \( C' = (e'; t', s' : A' \rightarrow R') \) be cat¹-algebras and \( \phi : A \rightarrow A' \) and \( \varphi : R \rightarrow R' \) be algebra homomorphisms. If the diagram

\[
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
A \\
\phi \\
\downarrow \\
R \\
\varphi \\
\downarrow \\
R'
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
A' \\
\phi \\
\downarrow \\
R' \\
\varphi \\
\downarrow \\
R'
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\]

commutes, (i.e \( t' \phi = \varphi t \), \( s' \phi = \varphi s \), \( e' \varphi = \varphi e \)), then the pair \((\phi, \varphi)\) is called a cat¹-algebra morphism. The following implementations are used for constructing cat¹-algebra morphisms. Details of the implementations can be found in [13].

\[
\text{Cat1AlgMorphism(arg)} \quad \text{(function)}
\]
\[
\text{IdentityMapping(C)} \quad \text{(operation)}
\]
\[
\text{PreCat1AlgMorphismByHoms(f,g)} \quad \text{(operation)}
\]
\[
\text{Cat1AlgMorphismByHoms(f,g)} \quad \text{(operation)}
\]
\[
\text{IsPreCat1AlgMorphism(m)} \quad \text{(operation)}
\]
\[
\text{IsCat1AlgMorphism(m)} \quad \text{(operation)}
\]

The following implementations are contain the properties for constructing the cat¹-algebra morphism \( m \).

\[
\text{Source(m)} \quad \text{(attribute)}
\]
\[
\text{IsTotal(m)} \quad \text{(attribute)}
\]
\[
\text{IsSingleValued(m)} \quad \text{(attribute)}
\]
\[
\text{Range(m)} \quad \text{(attribute)}
\]
\[
\text{Name(m)} \quad \text{(attribute)}
\]
\[
\text{Boundary(m)} \quad \text{(attribute)}
\]

The implementations written for crossed module morphism are reimplemented for cat¹-algebra morphisms as follows;

\[
gap> C1:=Cat1Alg(2,1,1,1);
\]
\[
[GF(2)_triv -> GF(2)_triv]
\]
\[
gap> Display(C1);
\]
Cat1-algebra \([\text{GF}(2)_\text{triv} \Rightarrow \text{GF}(2)_\text{triv}]\) :-
: source algebra has generators:
\[[ \text{Z}(2)^0)*() , \text{Z}(2)^0)*() \]
: range algebra has generators:
\[[ \text{Z}(2)^0)*() , \text{Z}(2)^0)*() \]
: tail homomorphism maps source generators to:
\[[ \text{Z}(2)^0)*() , \text{Z}(2)^0)*() \]
: head homomorphism maps source generators to:
\[[ \text{Z}(2)^0)*() , \text{Z}(2)^0)*() \]
: range embedding maps range generators to:
\[[ \text{Z}(2)^0)*() , \text{Z}(2)^0)*() \]
: the kernel is trivial.

\text{gap> } C2:=\text{Cat1Alg}(2,2,1,2);
[\text{GF}(2)_c2 \Rightarrow \text{GF}(2)_\text{triv}]
\text{gap> } \text{Display}(C2);

Cat1-algebra \([\text{GF}(2)_c2 \Rightarrow \text{GF}(2)_\text{triv}]\) :-
: source algebra has generators:
\[[ \text{Z}(2)^0)*() , \text{Z}(2)^0)*(1,2) \]
: range algebra has generators:
\[[ \text{Z}(2)^0)*() , \text{Z}(2)^0)*() \]
: tail homomorphism maps source generators to:
\[[ \text{Z}(2)^0)*() , \text{Z}(2)^0)*() \]
: head homomorphism maps source generators to:
\[[ \text{Z}(2)^0)*() , \text{Z}(2)^0)*() \]
: range embedding maps range generators to:
\[[ \text{Z}(2)^0)*() , \text{Z}(2)^0)*() \]
: kernel has generators:
\[[ \text{Z}(2)^0)*() + (\text{Z}(2)^0)*(1,2) \]
: boundary homomorphism maps generators of kernel to:
\[[ \text{<zero>} \text{ of ... } \]
: kernel embedding maps generators of kernel to:
\[[ \text{Z}(2)^0)*() + (\text{Z}(2)^0)*(1,2) \]

\text{gap> } C1=C2;
false
\text{gap> } R1:=\text{Source}(C1);
\text{gap> } R2:=\text{Source}(C2);
\text{gap> } S1:=\text{Range}(C1);
\text{gap> } S2:=\text{Range}(C2);
\text{gap> } \text{gR1:=GeneratorsOfAlgebra}(R1);
\[[ \text{Z}(2)^0)*() , \text{Z}(2)^0)*() \]
\text{gap> } \text{gR2:=GeneratorsOfAlgebra}(R2);
\[[ \text{Z}(2)^0)*() , \text{Z}(2)^0)*(1,2) \]
3.2 Equivalent Categories

The categories \textbf{Cat1Alg} (cat
\textsuperscript{1}-algebra) and \textbf{XModAlg} (crossed modules) are naturally equivalent, \cite{9}. We just remained this equivalences in the following. For a given crossed module \( \partial : A \to R \), we can construct the semidirect product \( R \rtimes A \) thanks to the action of \( R \) on \( A \). If we define \( t, s : R \rtimes A \to R \) and \( e : R \to R \rtimes A \) by

\[
  s(r, a) = r \quad t(r, a) = r + \partial(a) \quad e(r) = (r, 0),
\]
respectively, then \( \mathcal{C} = (e; t, s : R \rtimes A \to R) \) is a cat\(^1\)-algebra.

Conversely, for a given cat\(^1\)-algebra \( \mathcal{C} = (e; t, s : A \to R) \), \( \partial : \ker s \to R \) is a crossed module, where the action is conjugate action and \( \partial \) is the restriction of \( t \) to \( \ker s \).

The implementations

| Function                                      | Type    |
|-----------------------------------------------|---------|
| PreCat1AlgByPreXModAlg (X)                   | operation |
| PreXModAlgByPreCat1Alg (C)                   | operation |
| XModAlgByCat1Alg (C)                         | operation |
| Cat1AlgByXModAlg (X)                         | operation |
| KernelDenkt (t)                              | operation |
| KernelDenkh (h)                              | operation |
| SDproduct (C)                                | operation |

are used for constructing a cat\(^1\)-algebra from a given crossed module and for reverse.

```gap
gap> R:=GroupRing(GF(3),CyclicGroup(2));
<algebra-with-one over GF(3), with 1 generators>
gap> I:=AugmentationIdeal(R);
<two-sided ideal in <algebra-with-one over GF(3), with 1 generators>,
   (1 generators)>
gap> CM:=XModAlgByIdeal(R,I);
[Algebra( GF(3), [ (Z(3))^0 )*<identity> of ...+(Z(3)^0)*f1 ] ) -> AlgebraWithOne( GF(3), [ (Z(3)^0)*f1 ] )]
gap> IsXModAlg(CM);
true
gap> C:=Cat1AlgByXModAlg(CM);
[AlgebraWithOne( GF(3), [ (Z(3)^0)*f1 ] ) IX Algebra( GF(3), [ (Z(3)^0)*f1 ] ) -> AlgebraWithOne( GF(3), [ (Z(3)^0)*f1 ] )]
gap> IsCat1Alg(C);
true
gap> SM:=XModAlgByCat1Alg(C);
[...->...]
gap> SM=CM;
true
```

Since all these operations linked to the function Cat1Alg and XModAlg, all of them can be done by using these two functions and also we can use the function Cat1Alg instead of the operation Cat1AlgSelect.

```gap
gap> C:=Cat1Alg(4,6,1,1);
[GF(2^2)_s3 -> GF(2^2)_s3]
```
gap> IsCat1Alg(C);
true
gap> Display(C);

Cat1-algebra [GF(2^2)_s3=>GF(2^2)_s3] :-
: source algebra has generators:
[ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(2,3) ]
: range algebra has generators:
[ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(2,3) ]
: tail homomorphism maps source generators to:
[ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(2,3) ]
: head homomorphism maps source generators to:
[ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(2,3) ]
: range embedding maps range generators to:
[ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(2,3) ]
: the kernel is trivial.

gap> CM:=XModAlg(C);
[Algebra( GF(2^2), [], <zero> of ... )->GF(2^2)_s3]

gap> IsXModAlg(CM);
true

gap> Display(CM);

Crossed module [..->GF(2^2)_s3] :-
: Source group has generators:
[ ]
: Range group has generators:
[ (Z(2)^0)*(), (Z(2)^0)*(1,2), (Z(2)^0)*(2,3) ]
: Boundary homomorphism maps source generators to:
[ ]

gap> CC:=Cat1Alg(CM);
[GF(2^2)_s3=>GF(2^2)_s3]

gap> CC=C;
true

Acknowledgements

This works was partially supported by TÜBİTAK (The Scientific and Technical Research Council of Turkey).

Project Number : 107T542
References

[1] **ALP M. AND C.D. WENSLEY** Crossed Modules and Cat¹-groups in GAP, *version 1 Manual for the XMOD share package for GAP3*, (1996).

[2] **ALP M. AND C.D. WENSLEY** Crossed Modules and Cat¹-groups in GAP, *version 2.23 Manual for the XMOD share package for GAP4*, (2013).

[3] **ALP M. AND C.D. WENSLEY** Enumeration of cat¹-groups of low order, *Int. J. Algebra Comput.*, 10, 407-424, (2000).

[4] **ARVASI Z. AND PORTER T.** Simplicial and Crossed Resolutions of Commutative Algebras, *Journal of Algebra*, 181, 426-448, (1996).

[5] **ARVASI Z. AND PORTER T.** Freeness Conditions for 2-Crossed Modules of Commutative Algebras, *Applied Categorical Structures*, 6, 455-471, (1998).

[6] **ARVASI Z. AND EGE U.** Annihilators, multipliers and crossed modules, *Applied Categorical Structures*, 11, 487–506 (2003).

[7] **BROWN R.** Higher Dimensional Group Téory, Low Dimensional Topology, *London Math. Soc. Lecture Note Series*, 48, 215-238, (1982).

[8] **BROWN R.** From groups to groupoids: a brief survey, *Bull. London Math. Soc.*, 19, 113-134, (1987).

[9] **ELLIS G.J.** Higher Dimensional Crossed Modules of Algebras. *J.Pure Appl. Algebra* 52, 277-282, (1988).

[10] **GAP - GROUPS, ALGORITHMS, AND PROGRAMMING, VERSION 4** Lehrstuhl D für Mathematik, RWTH Aachen Germany and School of Mathematical and Computational Sciences, *U. St. Andrews, Scotland*, (1997).

[11] **LODAYS J.L.** Spaces with finitely many non-trivial homotopy groups., *J. App. Algebra*, 24 , 179–202, (1982).

[12] **MOSA G. H.** Higher dimensional algebroids and crossed complexes. *Ph.D. Thesis*, University of Wales, Bangor (U.K.) 1991-94 (1987).

[13] **ODABAS A.** Crossed Modules of Algebras with GAP. *Ph.D. Thesis*, Osmangazi University. (2009).

[14] **PORTER T.** Some categorical results in the theory of crossed modules in commutative algebras. *J. Algebra*, 109, 415-429, (1987).
[15] Whitehead J.H.C. Combinatorial Homotopy II, Bull. American Math. Society, 55, 453-456, (1949).