XMod

Crossed modules and cat1-groups in GAP

Version 2.43

11/11/2015

Murat Alp
Alper Odabaş
Enver Önder Uslu
Chris Wensley

Murat Alp  Email: muratalp@nigde.edu.tr
Address: Nigde Üniversitesi,
         Fen Edebiyat Fakültesi, Matematik Bölümü
         Nigde, 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
Enver Önder Uslu Email: enveruslu@ogu.edu.tr
Homepage: http://fef.ogu.edu.tr/matbil/enveruslu/
Address: Department of Mathematics and Computer Science,
         Osmangazi University, Eskişehir, Turkey
Chris Wensley  Email: c.d.wensley@bangor.ac.uk
Homepage: http://pages.bangor.ac.uk/~mas023/
Address: School of Computer Science, Bangor University,
         Dean Street, Bangor, Gwynedd, LL57 1UT, U.K.
Abstract

The XMod package provides functions for computation with

- finite crossed modules of groups and cat1-groups, and morphisms of these structures;
- finite pre-crossed modules, pre-cat1-groups, and their Peiffer quotients;
- isoclinism classes of groups and crossed modules;
- derivations of crossed modules and sections of cat1-groups;
- crossed squares and their morphisms, including the actor crossed square of a crossed module;
- crossed modules of finite groupoids (experimental version).

XMod was originally implemented in 1997 using the GAP3 language, when the first author was studying for a Ph.D. [Alp97] in Bangor.

In April 2002 the first and third parts were converted to GAP4, the pre-structures were added, and version 2.001 was released. The final two parts, covering derivations, sections and actors, were included in the January 2004 release 2.002 for GAP 4.4.

In October 2015 functions for computing isoclinism classes of crossed modules, written by Alper Odabaş and Enver Uslu, were added. These are contained in Chapter 4, and are described in detail in the paper [EIU16].

The current version is 2.43, released 11th November 2015 for GAP 4.7.

Bug reports, suggestions and comments are, of course, welcome. Please contact the last author at c.d.wensley@bangor.ac.uk or submit an issue at http://github.com/gap-packages/xmod/issues/.

Copyright

© 1997-2015 Chris Wensley et al. XMod is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

Acknowledgements

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

The first author wishes to acknowledge support from Dumlupinar University and the Turkish government.
Contents

1 Introduction 5

2 2d-groups : crossed modules and cat1-groups 8
  2.1 Constructions for crossed modules .......................... 8
  2.2 Pre-crossed modules ............................................. 11
  2.3 Cat1-groups and pre-cat1-groups ............................... 13
  2.4 Selection of a small cat1-group ............................... 15
  2.5 More functions for crossed modules and cat1-groups ....... 17

3 2d-mappings 18
  3.1 Morphisms of 2d-groups ........................................ 18
  3.2 Morphisms of pre-crossed modules ............................ 18
  3.3 Morphisms of pre-cat1-groups ................................. 20
  3.4 Operations on morphisms ...................................... 21

4 Isoclinism of groups and crossed modules 23
  4.1 More operations for crossed modules .......................... 23
  4.2 Isoclinism for groups .......................................... 29
  4.3 Isoclinism for crossed modules ............................... 30

5 Derivations and Sections 32
  5.1 Whitehead Multiplication ..................................... 32
  5.2 Whitehead Groups and Monoids ............................... 34

6 Actors of 2d-groups 37
  6.1 Actor of a crossed module .................................... 37

7 Induced constructions 41
  7.1 Induced crossed modules ....................................... 41

8 Crossed squares and their morphisms 44
  8.1 Constructions for crossed squares ............................ 45
  8.2 Morphisms of crossed squares ................................ 48

9 Crossed modules of groupoids 50
  9.1 Constructions for crossed modules of groupoids ............ 50
Chapter 1

Introduction

The XMod package provides functions for computation with

- finite crossed modules of groups and cat1-groups, and morphisms of these structures;
- finite pre-crossed modules, pre-cat1-groups, and their Peiffer quotients;
- derivations of crossed modules and sections of cat1-groups;
- isoclinism of groups and crossed modules;
- the actor crossed square of a crossed module;
- crossed squares and their morphisms (experimental version);
- crossed modules of groupoids (experimental version).

It is loaded with the command

\[
\text{gap> LoadPackage( "xmod" );}
\]

The term crossed module was introduced by J. H. C. Whitehead in [Whi48], [Whi49]. Loday, in [Lod82], reformulated the notion of a crossed module as a cat1-group. Norrie [Nor90], [Nor87] and Gilbert [Gil90] have studied derivations, automorphisms of crossed modules and the actor of a crossed module, while Ellis [Ell84] has investigated higher dimensional analogues. Properties of induced crossed modules have been determined by Brown, Higgins and Wensley in [BH78], [BW95] and [BW96]. For further references see [AW00], where we discuss some of the data structures and algorithms used in this package, and also tabulate isomorphism classes of cat1-groups up to size 30.

XMod was originally implemented in 1997 using the GAP 3 language. In April 2002 the first and third parts were converted to GAP 4, the pre-structures were added, and version 2.001 was released. The final two parts, covering derivations, sections and actors, were included in the January 2004 release 2.002 for GAP 4.4. Many of the function names have been changed during the conversion, for example ConjugationXMod has become XModByNormalSubgroup. For a list of name changes see the file names.pdf in the doc directory.

In October 2015 Alper Odabaş and Enver Uslu were added to the list of package authors. Their functions for computing isoclinism classes of groups and crossed modules are contained in Chapter 4, and are described in detail in their paper [EIU16].
The current version is 2.43 for GAP 4.7, released on 11th November 2015.

The package may be obtained as a compressed tar file `xmod-2.43.tar.gz` by ftp from one of the following sites:

- any GAP archive, e.g. [http://www.gap-system.org/Packages/packages.html](http://www.gap-system.org/Packages/packages.html);
- the Bangor site: [http://www.maths.bangor.ac.uk/chda/gap4/xmod/xmod243.html](http://www.maths.bangor.ac.uk/chda/gap4/xmod/xmod243.html);
- the package GitHub repository: [https://github.com/gap-packages/xmod](https://github.com/gap-packages/xmod).

Crossed modules and cat1-groups are special types of 2-dimensional groups [Bro82], [BHS11], and are implemented as 2dDomains and 2dGroups having a Source and a Range.

The package divides into eight parts. The first part is concerned with the standard constructions for pre-crossed modules and crossed modules; together with direct products; normal sub-crossed modules; and quotients. Operations for constructing pre-cat1-groups and cat1-groups, and for converting between cat1-groups and crossed modules, are also included.

The second part is concerned with morphisms of (pre-)crossed modules and (pre-)cat1-groups, together with standard operations for morphisms, such as composition, image and kernel.

The third part is the most recent part of the package, introduced in October 2015. Additional operations and properties for crossed modules are included in Section 4.1. Then, in 4.2 and 4.3 there are functions for isoclinism of groups and crossed modules.

The fourth part is concerned with the equivalent notions of derivation for a crossed module and section for a cat1-group, and the monoids which they form under the Whitehead multiplication.

The fifth part deals with actor crossed modules and actor cat1-groups. For the actor crossed module $\text{Act}(\mathcal{X})$ of a crossed module $\mathcal{X}$ we require representations for the Whitehead group of regular derivations of $\mathcal{X}$ and for the group of automorphisms of $\mathcal{X}$. The construction also provides an inner morphism from $\mathcal{X}$ to $\text{Act}(\mathcal{X})$ whose kernel is the centre of $\mathcal{X}$.

The sixth part, which remains under development, contains functions to compute induced crossed modules.

Since version 2.007 there are experimental functions for crossed squares and their morphisms, structures which arise as 3-dimensional groups. Examples of these are inclusions of normal sub-crossed modules, and the inner morphism from a crossed module to its actor.

The eighth part has some experimental functions for crossed modules of groupoids, interacting with the package Gpd. Much more work on this is needed.

Future plans include the implementation of group-graphs which will provide examples of pre-crossed modules (their implementation will require interaction with graph-theoretic functions in GAP 4). There are also plans to implement cat2-groups, and conversion between these and crossed squares.

The equivalent categories XMod (crossed modules) and Cat1 (cat1-groups) are also equivalent to GpGpd, the subcategory of group objects in the category Gpd of groupoids. Finite groupoids have been implemented in Emma Moore’s package Gpd [Moo01] for groupoids and crossed resolutions.

In order that the user has some control of the verbosity of the XMod package’s functions, an InfoClass InfoXMod is provided (see Chapter ref:Info Functions in the GAP Reference Manual for a description of the Info mechanism). By default, the InfoLevel of InfoXMod is 0; progressively more information is supplied by raising the InfoLevel to 1, 2 and 3.

```
gap> SetInfoLevel( InfoXMod, 1); #sets the InfoXMod level to 1
```

Once the package is loaded, the manual doc/manual.pdf can be found in the documentation folder. The html versions, with or without MathJax, should be rebuilt as follows:

```
 gap> ReadPackage( "xmod", "makedocrel.g" );
```

It is possible to check that the package has been installed correctly by running the test files:

```
 gap> ReadPackage( "xmod", "tst/testall.g" );
#I Testing .../pkg/xmod/tst/gp2obj.tst
...
```

Additional information can be found on the Computational Higher-dimensional Discrete Algebra website at http://pages.bangor.ac.uk/~mas023/chda/.
Chapter 2

2d-groups : crossed modules and cat1-groups

2.1 Constructions for crossed modules

A crossed module (of groups) $\mathcal{X} = (\partial : S \to R)$ consists of a group homomorphism $\partial$, called the \textit{boundary} of $\mathcal{X}$, with \textit{source} $S$ and \textit{range} $R$. The group $R$ acts on itself by conjugation, and on $S$ by an action $\alpha : R \to \text{Aut}(S)$ such that, for all $s, s_1, s_2 \in S$ and $r \in R$,

$$\text{XMod 1} : \partial(s') = r^{-1}(\partial s)r = (\partial s)'r,$$

$$\text{XMod 2} : s_1^{\partial s_2} = s_2^{-1}s_1s_2 = s_1s_2.$$

When only the first of these axioms is satisfied, the resulting structure is a \textit{pre-crossed module} (see section 2.2). (Much of the literature on crossed modules uses left actions, but we have chosen to use right actions in this package since that is the standard choice for group actions in GAP.)

The kernel of $\partial$ is abelian.

There are a variety of constructors for crossed modules:

2.1.1 XMod

- $\text{XMod}(\text{args})$
- $\text{XModByBoundaryAndAction}(\text{bdy}, \text{act})$
- $\text{XModByTrivialAction}(\text{bdy})$
- $\text{XModByNormalSubgroup}(G, N)$
- $\text{XModByCentralExtension}(\text{bdy})$
- $\text{XModByAutomorphismGroup}(\text{grp})$
- $\text{XModByInnerAutomorphismGroup}(\text{grp})$
- $\text{XModByGroupOfAutomorphisms}(G, A)$
- $\text{XModByAbelianModule}(\text{abmod})$
- $\text{DirectProduct}(X1, X2)$

The global function $\text{XMod}$ implements one of the following standard constructions:

- A \textit{trivial action crossed module} ($\partial : S \to R$) has $s^r = s$ for all $s \in S$, $r \in R$, the source is abelian and the image lies in the centre of the range.
A conjugation crossed module is the inclusion of a normal subgroup $S \trianglelefteq R$, where $R$ acts on $S$ by conjugation.

A central extension crossed module has as boundary a surjection $\partial : S \to R$, with central kernel, where $r \in R$ acts on $S$ by conjugation with $\partial^{-1}r$.

An automorphism crossed module has as range a subgroup $R$ of the automorphism group $\text{Aut}(S)$ of $S$ which contains the inner automorphism group of $S$. The boundary maps $s \in S$ to the inner automorphism of $S$ by $s$.

A crossed abelian module has an abelian module as source and the zero map as boundary.

The direct product $X_1 \times X_2$ of two crossed modules has source $S_1 \times S_2$, range $R_1 \times R_2$ and boundary $\partial_1 \times \partial_2$, with $R_1, R_2$ acting trivially on $S_2, S_1$ respectively.

2.1.2 Source

- Source($X_0$)
- Range($X_0$)
- Boundary($X_0$)
- AutoGroup($X_0$)
- XModAction($X_0$)

The following attributes are used in the construction of a crossed module $X_0$.

- Source($X_0$) and Range($X_0$) are the source $S$ and range $R$ of $\partial$, the boundary Boundary($X_0$);
- AutoGroup($X_0$) is a group of automorphisms of $S$;
- XModAction($X_0$) is a homomorphism from $R$ to AutoGroup($X_0$).

2.1.3 Size

- Size($X_0$)
- Name($X_0$)
- IdGroup($X_0$)
- ExternalSetXMod($X_0$)

More familiar attributes are Name, Size and IdGroup. The name is formed by concatenating the names of the source and range (if these exist). Size and IdGroup return two-element lists.

The ExternalSetXMod" for a crossed module is the source group considered as a G-set of the range group using the crossed module action.

The Display function is used to print details of 2d-groups.

In the simple example below, $X_1$ is an automorphism crossed module, using a cyclic group of size five. The Print statements at the end list the GAP representations, properties and attributes of $X_1$.

```gap
gap> c5 := Group( (5,6,7,8,9) );;
gap> SetName( c5, "c5" );
gap> X1 := XModByAutomorphismGroup( c5 );
```
2.1.4 \textbf{IsXMod}

\begin{verbatim}
\textbf{IsXMod}(X0) \hspace{1cm} \textbf{(property)}
\textbf{IsPreXMod}(X0) \hspace{1cm} \textbf{(property)}
\textbf{IsPerm2dGroup}(X0) \hspace{1cm} \textbf{(property)}
\textbf{IsPc2dGroup}(X0) \hspace{1cm} \textbf{(property)}
\textbf{IsFp2dGroup}(X0) \hspace{1cm} \textbf{(property)}
\end{verbatim}

The underlying category structures for the objects constructed in this chapter follow the sequence \textbf{Is2dDomain}; \textbf{Is2dMagma}; \textbf{Is2dMagmaWithOne}; \textbf{Is2dMagmaWithInverses}, mirroring the situation for (one-dimensional) groups. From these we construct \textbf{Is2dSemigroup}, \textbf{Is2dMonoid} and \textbf{Is2dGroup}.

A structure which has \textbf{IsPerm2dGroup} is a precrossed module or a pre-cat1-group (see section 2.3) whose source and range are both permutation groups. The properties \textbf{IsPc2dGroup}, \textbf{IsFp2dGroup} are defined similarly. We see in the previous example that \textbf{X1} has \textbf{IsPreXMod}, \textbf{IsXMod} and \textbf{IsPerm2dGroup}. There are also properties corresponding to the various construction methods listed in section 2.1: \textbf{IsTrivialAction2dGroup}; \textbf{IsNormalSubgroup2dGroup}; \textbf{IsCentralExtension2dGroup}; \textbf{IsAutomorphismGroup2dGroup}; \textbf{IsAbelianModule2dGroup}. 

### Example

\begin{verbatim}
[c5 -> PAut(c5)]
gap> Display( X1 );
Crossed module [c5 -> PAut(c5)] :-
: Source group c5 has generators:
  [ (5,6,7,8,9) ]
: Range group PAut(c5) has generators:
  [ (1,2,3,4) ]
: Boundary homomorphism maps source generators to:
  [ () ]
: Action homomorphism maps range generators to automorphisms:
  (1,2,3,4) --> { source gens --> [ (5,7,9,6,8) ] }
This automorphism generates the group of automorphisms.

gap> Size( X1 ); IdGroup( X1 );
[ 5, 4 ]
[ [ 5, 1 ], [ 4, 1 ] ]
gap> ext := ExternalSetXMod( X1 );
<set:[ (), (5,6,7,8,9), (5,7,9,6,8), (5,8,6,9,7), (5,9,8,7,6) ]>
gap> Orbits( ext );
[ [ () ], [ (5,6,7,8,9), (5,7,9,6,8), (5,9,8,7,6), (5,8,6,9,7) ] ]
gap> RepresentationsOfObject( X1 );
[ "IsComponentObjectRep", "IsAttributeStoringRep", "IsPreXModObj" ]
gap> KnownPropertiesOfObject( X1 );
gap> KnownAttributesOfObject( X1 );
\end{verbatim}
2.1.5 SubXMod

> SubXMod(X0, src, rng)  
> TrivialSubXMod(X0)  
> NormalSubXMods(X0)

With the standard crossed module constructors listed above as building blocks, sub-crossed modules, normal sub-crossed modules \( N \triangleleft \mathcal{X} \), and also quotients \( \mathcal{X}/N \) may be constructed. A sub-crossed module \( \mathcal{X} = (\delta : N \to M) \) is normal in \( \mathcal{Y} = (\partial : S \to R) \) if

- \( N, M \) are normal subgroups of \( S, R \) respectively,
- \( \delta \) is the restriction of \( \partial \),
- \( n^r \in N \) for all \( n \in N, r \in R \),
- \( (s^{-1})^m s \in N \) for all \( m \in M, s \in S \).

These conditions ensure that \( M \rtimes N \) is normal in the semidirect product \( R \rtimes S \). (Note that \( \langle s, m \rangle = (s^{-1})^m s \) is a displacement: see Displacement (4.1.3)).

A method for \texttt{IsNormal} for crossed modules is provided. See section 4.1 for quotient crossed modules and natural homomorphisms.

The five normal subcrossed modules of \( X_4 \) found in the following example are \([\text{id}, \text{id}]\), \([k_4, k_4]\), \([k_4, a_4]\), \([a_4, a_4]\) and \( X_4 \) itself.

```
> s4 := SymmetricGroup( IsPermGroup, 4 );
Sym( [ 1 .. 4 ] )
> a4 := Subgroup( s4, [ (1,2,3), (2,3,4) ] );
> k4 := Subgroup( a4, [ (1,2)(3,4), (1,3)(2,4) ] );
> SetName(s4,"s4"); SetName(a4,"a4"); SetName(k4,"k4");
> X4 := XModByNormalSubgroup( s4, a4 );
[a4->s4]
> Y4 := SubXMod( X4, k4, a4 );
[k4->a4]
> IsNormal(X4,Y4);
true
> NX4 := NormalSubXMods( X4 );
> Length( NX4 );
5
```

2.2 Pre-crossed modules

2.2.1 PreXModByBoundaryAndAction

> PreXModByBoundaryAndAction(bdy, act)  
> SubPreXMod(X0, src, rng)

If axiom \texttt{XMod 2} is not satisfied, the corresponding structure is known as a pre-crossed module.
Example

\begin{verbatim}
gap> d1 := (11,12,13,14,15,16,17,18);; d2 := (12,18)(13,17)(14,16);;
gap> d16 := Group( d1, d2 );;
gap> sk4 := Subgroup( d16, [ d1^4, d2 ] );;
gap> SetName( d16, "d16" ); SetName( sk4, "sk4" );
gap> bdy16 := GroupHomomorphismByImages( d16, sk4, [d1,d2], [d1^4,d2] );;
gap> h1 := GroupHomomorphismByImages( d16, d16, [d1,d2], [d1^5,d2] );;
gap> h2 := GroupHomomorphismByImages( d16, d16, [d1,d2], [d1,d2^4*d2] );;
gap> aut16 := Group( [ h1, h2 ] );;
gap> act16 := GroupHomomorphismByImages( sk4, aut16, [d1^4,d2], [h1,h2] );;
gap> P16 := PreXModByBoundaryAndAction( bdy16, act16 );
[ d16->sk4 ]
gap> IsXMod(P16);
false
\end{verbatim}

2.2.2 PeifferSubgroup

\begin{verbatim}
\text{PeifferSubgroup}(X0) \hfill (attribute)
\text{XModByPeifferQuotient}(prexmod) \hfill (attribute)

The Peiffer subgroup of a pre-crossed module \( P \) of \( S \) is the subgroup of \( \ker(\partial) \) generated by Peiffer commutators
\[
[s_1, s_2] = (s_1^{-1})\partial s_2 s_1 s_2 = (\partial s_2 s_1)[s_1, s_2].
\]
Then \( \mathcal{P} = (0 : P \to \{1_R\}) \) is a normal sub-pre-crossed module of \( \mathcal{X} \) and \( \mathcal{X} / \mathcal{P} = (\partial : S/P \to R) \) is a crossed module.

In the following example the Peiffer subgroup is cyclic of size 4.
\end{verbatim}

Example

\begin{verbatim}
gap> P := PeifferSubgroup( P16 );
Group( [ (11,15)(12,16)(13,17)(14,18), (11,17,15,13)(12,18,16,14) ] )
gap> X16 := XModByPeifferQuotient( P16 );
[ D16/P->sk4 ]
gap> Display( X16 );
Crossed module [D16/P->sk4] :-
: Source group has generators:
 [ f1, f2 ]
 : Range group has generators:
 [ (11,15)(12,16)(13,17)(14,18), (12,18)(13,17)(14,16) ]
 : Boundary homomorphism maps source generators to:
 [ (12,18)(13,17)(14,16), (11,15)(12,16)(13,17)(14,18) ]
The automorphism group is trivial
gap> iso16 := IsomorphismPermGroup( Source( X16 ) );;
gap> S16 := Image( iso16 );
Group([ (1,2), (3,4) ])
\end{verbatim}
2.3 Cat1-groups and pre-cat1-groups

2.3.1 Source

- Source(C) (attribute)
- Range(C) (attribute)
- TailMap(C) (attribute)
- HeadMap(C) (attribute)
- RangeEmbedding(C) (attribute)
- KernelEmbedding(C) (attribute)
- Boundary(C) (attribute)
- Name(C) (attribute)
- Size(C) (attribute)

These are the attributes of a cat1-group C in this implementation.

In [Lod82], Loday reformulated the notion of a crossed module as a cat1-group, namely a group G with a pair of homomorphisms \( t, h : G \to G \) having a common image \( R \) and satisfying certain axioms. We find it convenient to define a cat1-group \( C = (e, t, h : G \to R) \) as having source group \( G \), range group \( R \), and three homomorphisms: two surjections \( t, h : G \to R \) and an embedding \( e : R \to G \) satisfying:

- **Cat 1**: \( t \circ e = h \circ e = \text{id}_R \),
- **Cat 2**: \[ \ker t, \ker h \] = \( \{1_G\} \).

It follows that \( t \circ e \circ h = h \circ e \circ t = t, t \circ e \circ t = t, h \circ e \circ h = h \).

The maps \( t, h \) are often referred to as the **source** and **target**, but we choose to call them the **tail** and **head** of \( C \), because **source** is the GAP term for the domain of a function. The **RangeEmbedding** is the embedding of \( R \) in \( G \), the **KernelEmbedding** is the inclusion of the kernel of \( t \) in \( G \), and the **Boundary** is the restriction of \( h \) to the kernel of \( t \).

2.3.2 Cat1

- Cat1(args) (attribute)
- PreCat1ByTailHeadEmbedding(t, h, e) (attribute)
- PreCat1ByEndomorphisms(t, h) (attribute)
- PreCat1ByNormalSubgroup(G, N) (attribute)
- Cat1ByPeifferQuotient(P) (attribute)
- Reverse(C0) (attribute)

These are some of the constructors for pre-cat1-groups and cat1-groups. The following listing shows an example of a cat1-group of pc-groups of size [28,12].

```
gap> G2 := SmallGroup( 288, 956 ); SetName( G2, "G2" );
gap> <pc group of size 288 with 7 generators>
gap> d12 := DihedralGroup( 12 ); SetName( d12, "d12" );
gap> <pc group of size 12 with 3 generators>
gap> a1 := d12.1;; a2 := d12.2;; a3 := d12.3;; one := One( d12 );;
gap> gensG2 := GeneratorsOfGroup( G2 );;
gap> t2 := GroupHomomorphismByImages( G2, d12, gensG2, 
    [ one, a1*a3, a2*a3, one, one, a3, one ] );;
```
2.3.3 Cat1OfXMod

The category of crossed modules is equivalent to the category of cat1-groups, and the functors between these two categories may be described as follows. Starting with the crossed module \( \mathcal{X} = (\partial : S \to R) \), the group \( G \) is defined as the semidirect product \( G = R \rtimes S \) using the action from \( \mathcal{X} \), with multiplication rule

\[
(r_1, s_1)(r_2, s_2) = (r_1 r_2, s_1 h(r_2) s_2).
\]

The structural morphisms are given by

\[
t(r, s) = r, \quad h(r, s) = r(\partial s), \quad er = (r, 1).
\]

On the other hand, starting with a cat1-group \( \mathcal{C} = (e: t, h : G \to R) \), we define \( S = \ker t \), the range \( R \) is unchanged, and \( \partial = h|_S \). The action of \( R \) on \( S \) is conjugation in \( G \) via the embedding of \( R \) in \( G \).
```
Example

```}

```

2.4 Selection of a small cat1-group

The \texttt{Cat1} function may also be used to select a cat1-group from a data file. All cat1-structures on
groups of size up to 70 (ordered according to the \texttt{GAP} 4 numbering of small groups) are stored in a list
in file \texttt{cat1data.g}. Global variables \texttt{CAT1\_LIST\_MAX\_SIZE} := 70 and \texttt{CAT1\_LIST\_CLASS\_SIZES}
are also stored. The data is read into the list \texttt{CAT1\_LIST} only when this function is called.

2.4.1 \texttt{Cat1Select}

\begin{verbatim}
> Cat1Select(size, gpnum, num)  (attribute)

The function \texttt{Cat1Select} may be used in three ways. \texttt{Cat1Select(size)} returns the names
of the groups with this size, while \texttt{Cat1Select(size, gpnum)} prints a list of cat1-structures for
this chosen group. \texttt{Cat1Select(size, gpnum, num)} returns the chosen cat1-group.

The example below is the first case in which $t \neq h$ and the associated conjugation crossed module
is given by the normal subgroup $c3$ of $s3$.

```
Using small generating set \([ f_1, f_2, f_2*f_3 ]\) for source of homs.

\[\{\text{range gens}, \text{tail genimages}, \text{head genimages}\}\] :

1. \([ f_1, f_1, \text{<identity> of } \ldots, \text{<identity> of } \ldots ]\)
2. \([ f_1, f_3, f_1, \text{<identity> of } \ldots, f_3 ]\)
3. \([ f_1, f_3, f_1, \text{<identity> of } \ldots, f_3, f_3^2 ]\)
4. \([ f_1, f_2, f_2*f_3, \text{tail = head = identity mapping} ]\)

\(4\)

\text{gap}\>## select the third of these cat1-structures
\text{gap}\>\text{C18 := Cat1( 18, 4, 3 );}
\text{gap}\>[\text{[(C3 x C3) : C2=>Group( [ f_1, \text{<identity> of } \ldots, f_3 ] )]]}
\text{gap}\>## convert from a pc-cat1-group to a permutation cat1-group
\text{gap}\>\text{iso18 := IsomorphismPermObject( C18 );;}
\text{gap}\>\text{PC18 := Image( iso18 );;}
\text{gap}\>\text{Display( PC18 );}
\text{Cat1-group :-}

: Source group has generators:
\[(2,3)(5,6), (4,5,6), (1,2,3)\]

: Range group has generators:
\[(2,3), (), (1,2,3)\]

: tail homomorphism maps source generators to:
\[(2,3), (), (1,2,3)\]

: head homomorphism maps source generators to:
\[(2,3), (1,3,2), (1,2,3)\]

: range embedding maps range generators to:
\[(2,3)(5,6), (), (1,2,3)\]

: kernel has generators:
\[(4,5,6)\]

: boundary homomorphism maps generators of kernel to:
\[(1,3,2)\]

: kernel embedding maps generators of kernel to:
\[(4,5,6)\]

\text{gap}\>convert the result to the associated permutation crossed module
\text{gap}\>\text{X18 := XModByCat1( PC18 );;}
\text{gap}\>\text{Display( X18 );}
\text{Crossed module:-}

: Source group has generators:
\[(4,5,6)\]

: Range group has generators:
\[(2,3), (), (1,2,3)\]

: Boundary homomorphism maps range generators to:
\[(1,3,2)\]

: Action homomorphism maps range generators to automorphisms:
\( (2,3) \rightarrow \{ \text{source gens} \rightarrow [4,6,5] \} \)
\(() \rightarrow \{ \text{source gens} \rightarrow [4,5,6] \} \)
\((1,2,3) \rightarrow \{ \text{source gens} \rightarrow [4,5,6] \} \)

These 3 automorphisms generate the group of automorphisms.

: associated cat1-group is \([\ldots=>\ldots]\)
2.4.2 AllCat1sBasic

\[ \text{AllCat1sBasic}(gp) \] (operation)

For a group \( G \) of size greater than 70 which is reasonably straightforward this function may be used to construct a list of all cat1-group structures on \( G \). The operation also attempts to write output to a file in the folder \texttt{xmod/lib}. (Other operations in the file \texttt{cat1data.gi} have been used to deal with the more complicated groups of size up to 70, but these are not described here.)

Van Luyen Le has a more efficient algorithm, extending the data up to groups of size 171, which is expected to appear in a future release of \texttt{HAP}.

\begin{verbatim}
gap> gp := SmallGroup( 102, 2 );
gap> StructureDescription( gp );
"C3 x D34"
gap> all := AllCat1sBasic( gp );
#I Edit last line of .../xmod/lib/nn.kk.out to end with ] ] ] ]
[ [Group( [ f1, f2, f3 ] )=]Group( [ f1, <identity> of ... , <identity> of ... ] ), [Group( [ f1, f2, f3 ] )=]Group( [ f1, f2, <identity> of ... ] )],
[Group( [ f1, f2, f3 ] )=]Group( [ f1, <identity> of ... , f3 ] ),
[Group( [ f1, f2, f3 ] )=]Group( [ f1, f2, f3 ] )]
\end{verbatim}

2.5 More functions for crossed modules and cat1-groups

Chapter 4 contains functions for quotient crossed modules; centre of a crossed module; commutator and derived subcrossed modules; etc.

Here we mention two functions for groups which have been extended to the two-dimensional case.

2.5.1 IdGroup

\[ \text{IdGroup}(2dgroup) \] (operation)
\[ \text{StructureDescription}(2dgroup) \] (operation)

These functions return two-element lists formed by applying the function to the source and range of the 2d-group.

\begin{verbatim}
gap> IdGroup( X16 );
[ [ 8, 3 ], [ 16, 7 ] ]
gap> StructureDescription( C2 );
[ "(S3 x d24) : C2", "d12" ]
\end{verbatim}
Chapter 3

2d-mappings

3.1  Morphisms of 2d-groups

This chapter describes morphisms of (pre-)crossed modules and (pre-)cat1-groups.

3.1.1  Source

▷ Source(map)  (attribute)
▷ Range(map)   (attribute)
▷ SourceHom(map) (attribute)
▷ RangeHom(map) (attribute)

Morphisms of 2d-groups are implemented as 2d-mappings. These have a pair of 2d-groups as source and range, together with two group homomorphisms mapping between corresponding source and range groups. These functions return fail when invalid data is supplied.

3.2  Morphisms of pre-crossed modules

3.2.1  IsXModMorphism

▷ IsXModMorphism(map) (property)
▷ IsPreXModMorphism(map) (property)

A morphism between two pre-crossed modules \( \mathcal{X}_1 = (\partial_1 : S_1 \to R_1) \) and \( \mathcal{X}_2 = (\partial_2 : S_2 \to R_2) \) is a pair \((\sigma, \rho)\), where \(\sigma : S_1 \to S_2\) and \(\rho : R_1 \to R_2\) commute with the two boundary maps and are morphisms for the two actions:

\[
\partial_2 \circ \sigma = \rho \circ \partial_1, \quad \sigma(s') = (\sigma s)\rho'.
\]

Thus \(\sigma\) is the SourceHom and \(\rho\) is the RangeHom. When \(\mathcal{X}_1 = \mathcal{X}_2\) and \(\sigma, \rho\) are automorphisms then \((\sigma, \rho)\) is an automorphism of \(\mathcal{X}_1\). The group of automorphisms is denoted by \(\text{Aut}(\mathcal{X}_1)\).

3.2.2  IsInjective

▷ IsInjective(map)  (property)
▷ IsSurjective(map) (property)
The usual properties of mappings are easily checked. It is usually sufficient to verify that both the SourceHom and the RangeHom have the required property.

### 3.2.3 XModMorphism

- XModMorphism(args)
- XModMorphismByHoms(X1, X2, sigma, rho)
- PreXModMorphism(args)
- PreXModMorphismByHoms(P1, P2, sigma, rho)
- InclusionMorphism2dDomains(X1, S1)
- InnerAutomorphismXMod(X1, r)
- IdentityMapping(X1)
- IsomorphismPerm2dGroup(obj)
- IsomorphismPc2dGroup(obj)

These are the constructors for morphisms of pre-crossed and crossed modules.

In the following example we construct a simple automorphism of the crossed module X1 constructed in the previous chapter.

```gap
Example gap> sigma1 := GroupHomomorphismByImages( c5, c5, [ (5,6,7,8,9) ]
[ (5,9,8,7,6) ] );
gap> rho1 := IdentityMapping( Range( X1 ) );
IdentityMapping( PAut(c5) )
gap> mor1 := XModMorphism( X1, X1, sigma1, rho1 );
[ [c5->PAut(c5)) => [c5->PAut(c5))]
gap> Display( mor1 );
Morphism of crossed modules :-
: Source = [c5->PAut(c5))] with generating sets:
[ (5,6,7,8,9) ]
[ (1,2,3,4) ]
: Range = Source
: Source Homomorphism maps source generators to:
[ (5,9,8,7,6) ]
: Range Homomorphism maps range generators to:
[ (1,2,3,4) ]
gap> IsAutomorphism2dDomain( mor1 );
true
gap> Order( mor1 );
2
gap> RepresentationsOfObject( mor1 );
[ "IsComponentObjectRep", "IsAttributeStoringRep", "Is2dMappingRep" ]
gap> KnownPropertiesOfObject( mor1 );
[ "CanEasilyCompareElements", "CanEasilySortElements", "IsTotal",
 "IsSingleValued", "IsInjective", "IsSurjective", "RespectsMultiplication",]```
3.3 Morphisms of pre-cat1-groups

A morphism of pre-cat1-groups from $C_1 = (e_1; t_1, h_1 : G_1 \to R_1)$ to $C_2 = (e_2; t_2, h_2 : G_2 \to R_2)$ is a pair $(\gamma, \rho)$ where $\gamma : G_1 \to G_2$ and $\rho : R_1 \to R_2$ are homomorphisms satisfying

$$h_2 \circ \gamma = \rho \circ h_1, \quad t_2 \circ \gamma = \rho \circ t_1, \quad e_2 \circ \rho = \gamma \circ e_1.$$ 

3.3.1 IsCat1Morphism

- IsCat1Morphism(map)
- IsPreCat1Morphism(map)
- Cat1Morphism(args)
- Cat1MorphismByHoms(C1, C2, gamma, rho)
- PreCat1Morphism(args)
- PreCat1MorphismByHoms(P1, P2, gamma, rho)
- InclusionMorphism2dDomains(C1, S1)
- InnerAutomorphismCat1(C1, r)
- IdentityMapping(C1)
- SmallerDegreePerm2dDomain(obj)

The global function IsomorphismPermObject calls IsomorphismPerm2dGroup, which constructs a morphism whose SourceHom and RangeHom are calculated using IsomorphismPermGroup on the source and range. Similarly SmallerDegreePermutationRepresentation is used on the two groups to obtain SmallerDegreePerm2dDomain. Names are assigned automatically.

Example

gap> iso2 := IsomorphismPerm2dGroup( C2 );
[[G2=>d12] => [..]]
gap> Display( iso2 );
Morphism of cat1-groups :-
  : Source = [G2=>d12] with generating sets:
    [ f1, f2, f3, f4, f5, f6, f7 ]
    [ f1, f2, f3 ]
  : Range = P[G2=>d12] with generating sets:
      ( 2, 3)( 5,10)( 9,16)(11,18)(17,23)(19,25)(24,27),
      ( 4, 5, 7,10)( 6, 9,12,16)( 8,11,14,18)(13,17,20,23)(15,19,22,25)
        (21,24,26,27), ( 4, 6, 7,12)( 5, 9,10,16)( 8,13,14,20)(11,17,18,23)
        (15,21,22,26)(19,24,25,27), ( 4, 7)( 5,10)( 6,12)( 8,14)( 9,16)(11,18)
        (13,20)(15,22)(17,23)(19,25)(21,26)(24,27), ( 1, 2, 3),
      ( 4, 8,15)( 5,11,19)( 6,13,21)( 7,14,22)( 9,17,24)(10,18,26)(12,20,26)
        (16,23,27) ]
    [ (2,6)(3,5), (1,2,3,4,5,6), (1,3,5)(2,4,6) ]
3.4 Operations on morphisms

3.4.1 CompositionMorphism

Composition of morphisms (written \( \langle \text{map1} \rangle \ast \langle \text{map2} \rangle \)) when maps act on the right calls the CompositionMorphism function for maps (acting on the left), applied to the appropriate type of 2d-mapping.

Example

```gap
gap> H2 := Subgroup(G2,[G2.3,G2.4,G2.6,G2.7]); SetName( H2, "H2" ); Group([f3, f4, f6, f7 ])
gap> c6 := Subgroup( d12, [b,c] ); SetName( c6, "c6" ); Group([f2, f3 ])
gap> SC2 := Sub2dGroup( C2, H2, c6 ); [H2]=>c6
gap> IsCat1( SC2 ); true
gap> inc2 := InclusionMorphism2dDomains( C2, SC2 ); [[H2]=>c6] => [G2=>d12]
gap> CompositionMorphism( iso2, inc ); [[H2]=>c6] => P[G2=>d12]
```

3.4.2 Kernel

Kernel(map) (operation)
Kernel2dMapping(map) (attribute)

The kernel of a morphism of crossed modules is a normal subcrossed module whose groups are the kernels of the source and target homomorphisms. The inclusion of the kernel is a standard example of a crossed square, but these have not yet been implemented.

Example

```gap
gap> c2 := Group( (19,20) );
```
gap> X0 := XModByNormalSubgroup( c2, c2 ); SetName( X0, "X0" );
[Group([ (19,20) ]])->Group([ (19,20) ]) ]
gap> SX2 := Source( X2 );;
gap> genSX2 := GeneratorsOfGroup( SX2 );
[ f1, f4, f5, f7 ]
gap> sigma0 := GroupHomomorphismByImages(SX2,c2,genSX2,[(19,20),(),(),()]);
[ f1, f4, f5, f7 ] -> [ (19,20), (), (), () ]
gap> rho0 := GroupHomomorphismByImages(d12,c2,[a1,a2,a3],[(19,20),(),()]);
[ f1, f2, f3 ] -> [ (19,20), (), () ]
gap> mor0 := XModMorphism( X2, X0, sigma0, rho0 );;
gap> K0 := Kernel( mor0 );
[Group([ <identity> of ..., f4, f5, f7 ] )->Group([ <identity> of ..., f2, f3 ] )]
Chapter 4

Isoclinism of groups and crossed modules

This chapter describes some functions written by Alper Odabaş and Enver Uslu, and reported in their paper [EIU16]. Section 4.1 contains some additional basic functions for crossed modules, constructing quotients, centres, centralizers and normalizers. In Sections 4.2 and 4.3 there are functions dealing specifically with isoclinism for groups and for crossed modules. Since these functions represent a recent addition to the package (as of November 2015), the function names are liable to change in future versions.

4.1 More operations for crossed modules

4.1.1 FactorXMod

\[ \mathcal{X}_1 \rightarrow \mathcal{X}_2 \]

When \( \mathcal{X}_2 = (\partial_2 : S_2 \rightarrow R_2) \) is a normal subcrossed module of \( \mathcal{X}_1 = (\partial_1 : S_1 \rightarrow R_1) \), then the quotient crossed module is \((\partial : S_2/S_1 \rightarrow R_2/R_1)\) with the induced boundary and action maps.

Example

```gap
gap> d24 := DihedralGroup(24);; SetName( d24, "d24" );
gap> X24 := XModByAutomorphismGroup( d24 );;
gap> Size(X24);
[ 24, 48 ]
gap> nsx := NormalSubXMods( X24 );;
gap> ids := List( nsx, n -> IdGroup(n) );;
gap> pos1 := Position( ids, [ [4,1], [8,3] ] );;
gap> Xn1 := nsx[pos1];
gap> Size( Xn1 );
[ 4, 8 ]
gap> natn := NaturalMorphismByNormalSubXMod( X24, Xn1 );
[ [d24->PAut(d24) ] => [..] ]
gap> Qn1 := FactorXMod( X24, Xn1 );
[Group( [ f1, f2 ] )->Group( [ f1, f2 ] )]
gap> Size( Qn1 );
[ 6, 6 ]
```
### 4.1.2 IntersectionSubXMods

**IntersectionSubXMods**

When \( X_1, X_2 \) are subcrossed modules of \( X_0 \), then the source and range of their intersection are the intersections of the sources and ranges of \( X_1 \) and \( X_2 \) respectively.

```gap
gap> pos2 := Position( ids, [ [24,6], [12,4] ] );
gap> Xn2 := nsx[pos2];;
group [d24->Group( [ f1*f3, f2, f5 ] )]
gap> pos3 := Position( ids, [ [12,2], [24,5] ] );
gap> Xn3 := nsx[pos3];;
group [Group( [ f2, f3, f4 ] )->Group( [ f1, f2, f4, f5 ] )]
gap> Xn23 := IntersectionSubXMods( Xn24, Xn2, Xn3 );
group [Group( [ f2, f3, f4 ] )->Group( [ f2, f5, f2^2, f2*f5, f2^2*f5 ] )]
gap> [ Size(Xn2), Size(Xn3), Size(Xn23) ];
group [ [ 24, 12 ], [ 12, 24 ], [ 12, 6 ] ]
```

### 4.1.3 Displacement

**Displacement**

Commutators may be written \([r, q] = r^{-1}q^{-1}rq = (q^{-1})'q = r^{-1}r', \) and satisfy identities

\[
[r, q]^p = [r^p, q^p], \quad [pr, q] = [p, q]'[r, q], \quad [r, pq] = [r, q][r, p]'q, \quad [r, q]^{-1} = [q, r].
\]

In a similar way, when a group \( R \) acts on a group \( S \), the *displacement* of \( s \in S \) by \( r \in R \) is defined to be \( \langle r, s \rangle := (s^{-1})'s \in S \). When \( X = (\partial : S \rightarrow R) \) is a pre-crossed module, the first crossed module axiom requires \( \partial (r, s) = [r, \partial s] \). For a given action \( \alpha \) the Displacement function may be used to calculate \( \langle r, s \rangle \). Displacements satisfy the following identities, where \( s, t \in S, p, q, r \in R \):

\[
\langle r, s \rangle^p = \langle r^p, s^p \rangle, \quad \langle qr, s \rangle = \langle q, s \rangle \langle r, s \rangle, \quad \langle r, st \rangle = \langle r, t \rangle \langle r, s \rangle, \quad \langle r, s \rangle^{-1} = \langle r^{-1}, s^{-1} \rangle.
\]

The DisplacementSubgroup of \( X \) is the subgroup \( Disp(X) \) of \( S \) generated by these displacements. The identities imply \( \langle r, s \rangle' = \langle r, s^{-1} \rangle^{-1} \langle r^{-1}, t \rangle \), so \( Disp(X) \) is normal in \( S \).

```gap
gap> pos4 := Position( ids, [ [6,2], [24,14] ] );;
gap> Xn4 := nsx[pos4];;
group [Sn4 := Source(Xn4)];
group [Rn4 := Range(Xn4)];
gap> r := Rn4.1;; s := Sn4.1;;
gap> d := Displacement( XModAction(Xn4), r, s );
group [f4]
gap> bn4 := Boundary( Xn4 );;
group [Image( bn4, d ) = Comm( r, Image( bn4, s ) )];
group [true]
group [DisplacementSubgroup( Xn4 );]
group [Group([ f4 ])]
```
4.1.4 CommutatorSubXMod

- CommutatorSubXMod(X, X1, X2)
- CrossActionSubgroup(X, X1, X2)

When $X_1 = (N \to Q), X_2 = (M \to P)$ are two normal subcrossed modules of $X = (\partial : S \to R)$, the displacements $\langle p, n \rangle$ and $\langle q, m \rangle$ all map by $\partial$ into $[Q, P]$. These displacements form a normal subgroup of $S$, called the CrossActionSubgroup. The CommutatorSubXMod $[X_1, X_2]$ has this subgroup as source and $[P, Q]$ as range, and is normal in $X$.

Example:

```gap
gap> Cn23 := CommutatorSubXMod( X24, Xn2, Xn3 );
gap> Size(Cn23);
[ 12, 6 ]
gap> Xn23 = Cn23;
true
gap> Q24 := CentralQuotient( d24 )
gap> DXn4 := DerivedSubXMod( Xn4 );
```

4.1.5 DerivedSubXMod

- DerivedSubXMod(X0)

The DerivedSubXMod of $X$ is the normal subcrossed module $[\mathcal{X'}, \mathcal{X}] = (\partial' : \text{Disp}(\mathcal{X'}) \to [R, R])$ where $\partial'$ is the restriction of $\partial$ (see page 66 of Norrie’s thesis [Nor87]).

Example:

```gap
gap> DXn4 := DerivedSubXMod( Xn4 );
```

4.1.6 FixedPointSubgroupXMod

- FixedPointSubgroupXMod(X0, T, Q)
- StabilizerSubgroupXMod(X0, T, Q)

The FixedPointSubgroupXMod for $X = (\partial : S \to R)$ is the subgroup Fix($\mathcal{X'}, T, Q$) of elements $t \in T \subseteq S$ individually fixed under the action of $Q \leq R$.

The StabilizerSubgroupXMod for $X = (\partial : S \to R)$ is the subgroup Stab($\mathcal{X'}, T, Q$) of $Q \leq R$ whose elements act trivially on the whole of $T \subseteq S$ (see page 19 of Norrie’s thesis [Nor87]).

Example:

```gap
gap> fix := FixedPointSubgroupXMod( Xn4, Sn4, Rn4 );
gap> stab := StabilizerSubgroupXMod( Xn4, Sn4, Rn4 );
```
4.1.7 CentreXMod

CentreXMod(X₀)

Centralizer(X, Y)

Normalizer(X, Y)

The centre \(Z(\mathcal{X})\) of \(\mathcal{X} = (\partial : S \to R)\) has as source the fixed point subgroup \(\text{Fix}(\mathcal{X}, S, R)\). The range is the intersection of the centre \(Z(R)\) with the stabilizer subgroup.

When \(\mathcal{Y} = (T \to Q)\) is a subcrossed module of \(\mathcal{X} = (\partial : S \to R)\), the centralizer \(C_{\mathcal{X}}(\mathcal{Y})\) of \(\mathcal{Y}\) has as source the fixed point subgroup \(\text{Fix}(\mathcal{X}, S, Q)\). The range is the intersection of the centralizer \(C_R(Q)\) with \(\text{Stab}(\mathcal{X}, T, R)\).

The normalizer \(N_{\mathcal{X}}(\mathcal{Y})\) of \(\mathcal{Y}\) has as source the subgroup of \(S\) consisting of the displacements \(\langle s, q \rangle\) which lie in \(S\).

Example

```gap
gap> ZXn4 := CentreXMod( Xn4 );
gap> IdGroup( ZXn4 );
[ [ 2, 1 ], [ 4, 2 ] ]
gap> CDXn4 := Centralizer( Xn4, DXn4 );
gap> IdGroup( CDXn4 );
[ [ 2, 1 ], [ 3, 1 ] ]
gap> NDXn4 := Normalizer( Xn4, DXn4 );
[Group( <identity> of ... )->Group( [ f5, f2*f3 ] )]
gap> IdGroup( NDXn4 );
[ [ 1, 1 ], [ 12, 5 ] ]
```

4.1.8 CentralQuotient

CentralQuotient(G)

The CentralQuotient of a group \(G\) is the crossed module \((G \to G/Z(G))\) with the natural homomorphism as the boundary map. This is a special case of \(\text{XModByCentralExtension}\) (see 2.1).

Similarly, the central quotient of a crossed module \(\mathcal{X}\) is the crossed square \((\mathcal{X} \Rightarrow \mathcal{X}/Z(\mathcal{X}))\) (see section 8.1).

Example

```gap
gap> Q24 := CentralQuotient( d24 );
[d24->Group( [ f1, f2, f3 ] )]
gap> Size( Q24 );
[ 24, 12 ]
```

4.1.9 IsAbelian2dGroup

IsAbelian2dGroup(X₀)

IsAspherical2dGroup(X₀)

IsSimplyConnected2dGroup(X₀)
A crossed module is \textit{abelian} if it equal to its centre. This is the case when the range group is abelian and the action is trivial.

A crossed module is \textit{aspherical} if the boundary has trivial kernel.

A crossed module is \textit{simply connected} if the boundary has trivial cokernel.

A crossed module is \textit{faithful} if the action is faithful.

\begin{verbatim}
gap> [ IsAbelian2dGroup(Xn4), IsAbelian2dGroup(X24) ];
[ false, false ]
gap> pos7 := Position( ids, [ [3,1], [6,1] ] );;
gap> [ IsAspherical2dGroup(nsx[pos7]), IsAspherical2dGroup(X24) ];
[ true, false ]
gap> [ IsSimplyConnected2dGroup(Xn4), IsSimplyConnected2dGroup(X24) ];
[ true, true ]
gap> [ IsFaithful2dGroup(Xn4), IsFaithful2dGroup(X24) ];
[ false, true ]
\end{verbatim}

4.1.10 LowerCentralSeriesOfXMod

Let \( Y \) be a subcrossed module of \( X \). A \textit{series of length} \( n \) from \( X \) to \( Y \) has the form
\[
X = X_0 \triangleright X_1 \triangleright \cdots \triangleright X_i \triangleright \cdots \triangleright X_n = Y \quad (1 \leq i \leq n).
\]

The quotients \( F_i = X_i / X_{i-1} \) are the factors of the series.

A factor \( F_i \) is \textit{central} if \( X_{i-1} \triangleleft X \) and \( F_i \) is a subcrossed module of the centre of \( X / X_{i-1} \).

A series is \textit{central} if all its factors are central.

\( X \) is \textit{soluble} if it has a series all of whose factors are abelian.

\( X \) is \textit{nilpotent} if it has a series all of whose factors are central factors of \( X \).

The \textit{lower central series} of \( X \) is the sequence (see [Nor87], p.77):
\[
X = \Gamma_1(X) \triangleright \Gamma_2(X) \triangleright \cdots \quad \text{where} \quad \Gamma_j(X) = [\Gamma_{j-1}(X), X].
\]

If \( X \) is nilpotent, then its lower central series is its most rapidly descending central series.

The least integer \( c \) such that \( \Gamma_{c+1}(X) \) is the trivial crossed module is the \textit{nilpotency class} of \( X \).

\begin{verbatim}
gap> LowerCentralSeries(X24);
[ [d24->PAut(d24)], [Group( [ f2 ] )->Group( [ f2, f5 ] )],
]
gap> IsNilpotent2dGroup(X24);
false

gap> NilpotencyClass2dGroup(X24);
0
\end{verbatim}
4.1.11 \textbf{AllXMods}

\texttt{AllXMods(args)}

The global function \texttt{AllXMods} may be called in three ways: as \texttt{AllXMods(S,R)} to compute all crossed modules with chosen source and range groups; as \texttt{AllXMods([n,m])} to compute all crossed modules with a given size; or as \texttt{AllXMods(ord)} to compute all crossed modules whose associated \texttt{cat1}-groups have a given size \texttt{ord}.

In the example we see that there are 4 crossed modules \((C_6 \rightarrow S_3)\); forming a subset of the 17 crossed modules with size \([6,6]\); and that these form a subset of the 205 crossed modules whose \texttt{cat1}-group has size 36. There are 40 precrossed modules with size \([6,6]\).

\begin{verbatim}
gap> xc6s3 := AllXMods( SmallGroup(6,2), SmallGroup(6,1) );;
gap> Length( xc6s3 );
4
gap> x66 := AllXMods( [6,6] );;
gap> Length( x66 );
17
gap> x36 := AllXMods( 36 );;
gap> Length( x36 );
205
gap> size36 := List( x36, x -> [ Size(Source(x)), Size(Range(x)) ] );
gap> Collected( size36 );
[ [ [ 1, 36 ], 14 ], [ [ 2, 18 ], 7 ], [ [ 3, 12 ], 21 ], [ [ 4, 9 ], 14 ],
  [ [ 6, 6 ], 17 ], [ [ 9, 4 ], 102 ], [ [ 12, 3 ], 8 ], [ [ 18, 2 ], 18 ],
  [ [ 36, 1 ], 4 ] ]
\end{verbatim}

4.1.12 \textbf{IsomorphismXMods}

\texttt{IsomorphismXMods(X1, X2)}

The function \texttt{IsomorphismXMods} computes an isomorphism \(\mu : \mathcal{X}_1 \rightarrow \mathcal{X}_2\), provided one exists, or else returns \texttt{fail}. In the example below we see that the 17 crossed modules of size \([6,6]\) in \texttt{x66} (see the previous subsection) fall into 9 isomorphism classes.

The function \texttt{AllXModsUpToIsomorphism} takes a list of crossed modules and partitions them into isomorphism classes.

\begin{verbatim}
Example

\texttt{gap> IsomorphismXMods( x66[1], x66[2] );
[[Group( [ f1, f2 ] )->Group( [ f1, f2 ] )] => [Group( [ f1, f2 ] )->Group( [ f1, f2 ] )]]
\texttt{gap> iso66 := AllXModsUpToIsomorphism( x66 ); Length( iso66 );
9}
\end{verbatim}
4.2 Isoclinism for groups

4.2.1 Isoclinism

Let $G, H$ be groups with central quotients $Q(G)$ and $Q(H)$ and derived subgroups $[G, G]$ and $[H, H]$ respectively. Let $c_G : G/Z(G) \times G/Z(G) \to [G, G]$ and $c_H : H/Z(H) \times H/Z(H) \to [H, H]$ be the two commutator maps. An isoclinism $G \sim H$ is a pair of isomorphisms $(\eta, \xi)$ where $\eta : G/Z(G) \to H/Z(H)$ and $\xi : [G, G] \to [H, H]$ such that $c_G \circ \xi = (\eta \times \eta) \circ c_H$. Isoclinism is an equivalence relation, and all abelian groups are isoclinic to the trivial group.

Example

```
gap> G := SmallGroup( 64, 6 );; StructureDescription( G );
"(C8 x C4) : C2"
gap> QG := CentralQuotient( G );; IdGroup( QG );
[ [ 64, 6 ], [ 8, 3 ] ]
gap> H := SmallGroup( 32, 41 );; StructureDescription( H );
"C2 x Q16"
gap> QH := CentralQuotient( H );; IdGroup( QH );
[ [ 32, 41 ], [ 8, 3 ] ]
gap> Isoclinism( G, H );
[ [ f1, f2, f3 ] -> [ f1, f2*f3, f3 ], [ f3, f5 ] -> [ f4*f5, f5 ] ]
gap> K := SmallGroup( 32, 43 );; StructureDescription( K );
"(C2 x D8) : C2"
gap> QK := CentralQuotient( K );; IdGroup( QK );
[ [ 32, 43 ], [ 16, 11 ] ]
gap> AreIsoclinicDomains( G, K );
false
```

4.2.2 IsStemDomain

A group $G$ is a stem group if $Z(G) \leq [G, G]$. Every group is isoclinic to a stem group, but distinct stem groups may be isoclinic. For example, groups $D_8, Q_8$ are two isoclinic stem groups.

The function `IsoclinicStemDomain` returns a stem group isoclinic to $G$.

The function `AllStemGroupIds` returns the `IdGroup` list of the stem groups of a specified size, while `AllStemGroupFamilies` splits this list into isoclinism classes.

Example

```
gap> DerivedSubgroup(G);
Group([ [ f3, f5 ] ])
gap> IsStemDomain( G );
false
```
4.2.3 MiddleLength

▷ MiddleLength(G) (attribute)
▷ RankXMod(X0) (attribute)

Let $G$ be a finite group. Then $\log_2 |[G,G]/(Z(G) \cap [G,G])|$ is called the middle length of $G$. Also $\log_2 |Z(G) \cap [G,G]| + \log_2 |G/Z(G)|$ is called the rank of $G$.

Example

```
gap> MiddleLength(G); 1.
gap> RankXMod(X1);  [ 2.32193, 2. ]
```

4.3 Isoclinism for crossed modules

4.3.1 Isoclinism

▷ Isoclinism(X0, Y0) (operation)
▷ AreIsoclinicDomains(X0, Y0) (operation)

Let $\mathcal{X}, \mathcal{Y}$ be crossed modules with central quotients $Q(\mathcal{X})$ and $Q(\mathcal{Y})$, and derived subcrossed modules $[\mathcal{X}, \mathcal{X}]$ and $[\mathcal{Y}, \mathcal{Y}]$ respectively. Let $c_X : Q(\mathcal{X}) \times Q(\mathcal{X}) \to [\mathcal{X}, \mathcal{X}]$ and $c_Y : Q(\mathcal{Y}) \times Q(\mathcal{Y}) \to [\mathcal{Y}, \mathcal{Y}]$ be the two commutator maps. An isoclinism $\mathcal{X} \sim \mathcal{Y}$ is a pair of bijective morphisms $(\eta, \xi)$ where $\eta : Q(\mathcal{X}) \to Q(\mathcal{Y})$ and $\xi : [\mathcal{X}, \mathcal{X}] \to [\mathcal{Y}, \mathcal{Y}]$ such that $c_X \ast \xi = (\eta \times \eta) \ast c_Y$. Isoclinism is an equivalence relation, and all abelian crossed modules are isoclinic to the trivial crossed module.

Example

```
gap> C8 := Cat1(16,8,1);;
gap> X8 := XMod(C8); IdGroup( X8 );
[ [ 8, 1 ], [ 2, 1 ] ]
gap> C9 := Cat1(32,9,1);
```
4.3.2 IsStemDomain

A crossed module $\mathcal{X}$ is a stem crossed module if $Z(\mathcal{X}) \leq [\mathcal{X}, \mathcal{X}]$. Every crossed module is isoclinic to a stem crossed module, but distinct stem crossed modules may be isoclinic.

A method for IsoclinicStemDomain has yet to be implemented.

```gap
IsStemDomain(X8);  # true
IsStemDomain(X9);  # false
```

4.3.3 MiddleLength

Description required here.

```gap
MiddleLength(X24);  # [ 2.58496, 2.58496 ]
```
Chapter 5

Derivations and Sections

5.1 Whitehead Multiplication

5.1.1 IsDerivation

\[ \text{IsDerivation}(\text{map}) \] (property)

\[ \text{IsSection}(\text{map}) \] (property)

\[ \text{IsUp2dMapping}(\text{map}) \] (property)

The Whitehead monoid \( \text{Der}(X) \) of \( X \) was defined in [Whi48] to be the monoid of all derivations from \( R \) to \( S \), that is the set of all maps \( \chi : R \to S \), with Whitehead multiplication \( \star \) (on the right)
satisfying:

\[ \text{Der 1: } \chi(qr) = (\chi q)\chi(r), \quad \text{Der 2: } (\chi_1 \star \chi_2)(r) = (\chi_2 r)(\chi_1 r)(\chi_2 \partial \chi_1 r). \]

The zero map is the identity for this composition. Invertible elements in the monoid are called regular. The Whitehead group of \( X \) is the group of regular derivations in \( \text{Der}(X) \). In the next chapter the actor of \( X \) is defined as a crossed module whose source and range are permutation representations of the Whitehead group and the automorphism group of \( X \).

The monoid of sections of \( C = (e; t, h : G \to R) \) is the set of group homomorphisms \( \xi : R \to G \), with Whitehead multiplication \( \star \) (on the right) satisfying:

\[ \text{Sect 1: } t \circ \xi = \text{id}_R, \quad \text{Sect 2: } (\xi_1 \star \xi_2)(r) = (\xi_1 r)(eh\xi_1 r)^{-1}(\xi_2 h\xi_1 r) = (\xi_2 h\xi_1 r)(eh\xi_1 r)^{-1}(\xi_1 r). \]

The embedding \( e \) is the identity for this composition, and \( h(\xi_1 \star \xi_2) = (h\xi_1)(h\xi_2) \). A section is regular when \( h\xi \) is an automorphism, and the group of regular sections is isomorphic to the Whitehead group.

If \( \varepsilon \) denotes the inclusion of \( S = \text{kert} \) in \( G \) then \( \partial = h\varepsilon : S \to R \) and

\[ \xi r = (er)(e\chi r), \quad \text{which equals } (r, \chi r) \in R \times S, \]
determines a section \( \xi \) of \( C \) in terms of the corresponding derivation \( \chi \) of \( X \), and conversely.

5.1.2 DerivationByImages

\[ \text{DerivationByImages}(X_0, \text{ims}) \] (operation)

32
Derivations are stored like group homomorphisms by specifying the images of a generating set. Images of the remaining elements may then be obtained using axiom Der 1. The function IsDerivation is automatically called to check that this procedure is well-defined.

In the following example a cat1-group C3 and the associated crossed module X3 are constructed, where X3 is isomorphic to the inclusion of the normal cyclic group c3 in the symmetric group s3.

```gap
gap> g18 := Group( (1,2,3), (4,5,6), (2,3)(5,6) );;
gap> SetName( g18, "g18" );
gap> gen18 := GeneratorsOfGroup( g18 );;
gap> g1 := gen18[1];; g2 := gen18[2];; g3 := gen18[3];;
gap> s3 := Subgroup( g18, gen18{[2..3]} );;
gap> SetName( s3, "s3" );;
gap> t := GroupHomomorphismByImages( g18, s3, gen18, [g2,g2,g3] );;
gap> h := GroupHomomorphismByImages( g18, s3, gen18, [(),g2,g3] );;
gap> e := GroupHomomorphismByImages( s3, g18, [g2,g3], [g2,g3] );;
gap> C3 := Cat1( t, h, e );
gap> C3 := Cat1( t, h, e );
gap> Kernel(t), "c3" );;
gap> X3 := XModOfCat1( C3 );;
gap> Display( X3 );
Crossed module [c3->s3] :-
  : Source group has generators:
    [ (1,2,3)(4,6,5) ]
  : Range group has generators:
    [ (4,5,6), (2,3)(5,6) ]
  : Boundary homomorphism maps source generators to:
    [ (4,6,5) ]
  : Action homomorphism maps range generators to automorphisms:
    (4,5,6) --> { source gens --> [ (1,2,3)(4,6,5) ] }
    (2,3)(5,6) --> { source gens --> [ (1,3,2)(4,5,6) ] }
  : associated cat1-group is [g18=>s3]

```

5.1.3 SectionByImages

Sections are group homomorphisms, so do not need a special representation. Operations SectionByDerivation and DerivationBySection convert derivations to sections, and vice-versa, calling Cat1OfXMod and XModOfCat1 automatically.

Two strategies for calculating derivations and sections are implemented, see [AW00]. The default method for AllDerivations is to search for all possible sets of images using a backtracking proce-
dure, and when all the derivations are found it is not known which are regular. In early versions of this package, the default method for AllSections( <C> ) was to compute all endomorphisms on the range group $R$ of $C$ as possibilities for the composite $h\xi$. A backtrack method then found possible images for such a section. In the current version the derivations of the associated crossed module are calculated, and these are all converted to sections using SectionByDerivation.

```
Example

gap> xi := SectionByDerivation( chi );
SectionByImages( s3, g18, [ (4,5,6), (2,3)(5,6) ], [ (1,2,3), (1,2)(4,6) ] )
```

5.2 Whitehead Groups and Monoids

5.2.1 RegularDerivations

There are two functions to determine the elements of the Whitehead group and the Whitehead monoid of a crossed module, namely RegularDerivations and AllDerivations. (The functions RegularSections and AllSections perform corresponding tasks for a cat1-group.)

Using our example $X3$ we find that there are just nine derivations. (In fact, six of them regular, and the associated group is isomorphic to $s3$.)

```
Example

gap> all3 := AllDerivations( X3 );;
gap> imall3 := ImagesList( all3 );;
gap> PrintListOneItemPerLine( imall3 );
[ [ (), () ],
  [ (), (1,3,2)(4,5,6) ],
  [ (), (1,2,3)(4,6,5) ],
  [ (1,3,2)(4,5,6), () ],
  [ (1,3,2)(4,5,6), (1,3,2)(4,5,6) ],
  [ (1,2,3)(4,6,5), () ],
  [ (1,2,3)(4,6,5), (1,3,2)(4,5,6) ],
  [ (1,2,3)(4,6,5), (1,2,3)(4,6,5) ] ]
gap> KnownAttributesOfObject( all3 );
[ "Object2d", "ImagesList", "AllOrRegular", "ImagesTable" ]
gap> PrintListOneItemPerLine( ImagesTable( all3 ) );
[ [ 1, 1, 1, 1, 1, 1 ],
  [ 1, 1, 1, 3, 3, 3 ],
  [ 1, 1, 1, 2, 2, 2 ],
  [ 1, 3, 2, 1, 3, 2 ] ]
```
5.2.2 CompositeDerivation

- CompositeDerivation(chi1, chi2)
- UpImagePositions(chi)
- UpGeneratorImages(chi)
- CompositeSection(xi1, xi2)

The Whitehead product $\chi_1 \ast \chi_2$ is implemented as CompositeDerivation(<chi1>,<chi2>). The composite of two sections is just the composite of homomorphisms.

Example

```gap
gap> reg3 := RegularDerivations( X3 );;
gap> imder3 := ImagesList( reg3 );;
gap> chi4 := DerivationByImages( X3, imder3[4] );
DerivationByImages( s3, c3, [ (4,5,6), (2,3)(5,6) ], [ (1,3,2)(4,5,6), () ] )
gap> chi5 := DerivationByImages( X3, imder3[5] );
DerivationByImages( s3, c3, [ (4,5,6), (2,3)(5,6) ], [ (1,3,2)(4,5,6), (1,3,2)(4,5,6) ] )
gap> im4 := UpImagePositions( chi4 );
[ 1, 3, 2, 1, 3, 2 ]
gap> im5 := UpImagePositions( chi5 );
[ 1, 3, 2, 3, 2, 1 ]
gap> chi45 := chi4 * chi5;
DerivationByImages( s3, c3, [ (4,5,6), (2,3)(5,6) ], [ (), (1,3,2)(4,5,6) ] )
gap> im45 := UpImagePositions( chi45 );
[ 1, 1, 1, 3, 3, 3 ]
gap> pos := Position( imder3, UpGeneratorImages( chi45 ) );
2
```

5.2.3 WhiteheadGroupTable

- WhiteheadGroupTable(X0)
- WhiteheadMonoidTable(X0)
- WhiteheadPermGroup(X0)
- WhiteheadTransMonoid(X0)

Multiplication tables for the Whitehead group or monoid enable the construction of permutation or transformation representations.

Example

```gap
gap> wgt3 := WhiteheadGroupTable( X3 );;
```
gap> PrintListOneItemPerLine( wgt3 );
[ [ 1, 2, 3, 4, 5, 6 ],
  [ 2, 3, 1, 5, 6, 4 ],
  [ 3, 1, 2, 6, 4, 5 ],
  [ 4, 6, 5, 1, 3, 2 ],
  [ 5, 4, 6, 2, 1, 3 ],
  [ 6, 5, 4, 3, 2, 1 ]
]

gap> wpg3 := WhiteheadPermGroup( X3 );
Group([ (1,2,3)(4,5,6), (1,4)(2,6)(3,5) ])

gap> wmt3 := WhiteheadMonoidTable( X3 );;

gap> PrintListOneItemPerLine( wmt3 );
[ [ 1, 2, 3, 4, 5, 6, 7, 8, 9 ],
  [ 2, 3, 1, 5, 6, 4, 8, 9, 7 ],
  [ 3, 1, 2, 6, 4, 5, 9, 7, 8 ],
  [ 4, 6, 5, 1, 3, 2, 7, 9, 8 ],
  [ 5, 4, 6, 2, 1, 3, 8, 7, 9 ],
  [ 6, 5, 4, 3, 2, 1, 9, 8, 7 ],
  [ 7, 7, 7, 7, 7, 7, 7, 7, 7 ],
  [ 8, 8, 8, 8, 8, 8, 8, 8, 8 ],
  [ 9, 9, 9, 9, 9, 9, 9, 9, 9 ]
]

gap> wtm3 := WhiteheadTransMonoid( X3 );
<transformation monoid on 9 pts with 3 generators>

gap> GeneratorsOfMonoid( wtm3 );
[ Transformation([ 2, 3, 1, 5, 6, 4, 8, 9, 7 ]),
  Transformation([ 4, 6, 5, 1, 3, 2, 7, 9, 8 ]),
  Transformation([ 7, 7, 7, 7, 7, 7, 7, 7, 7 ]) ]
Chapter 6

Actors of 2d-groups

6.1 Actor of a crossed module

The *actor* of $\mathcal{X}$ is a crossed module $(\Delta : W(\mathcal{X}) \to \text{Aut}(\mathcal{X}'))$ which was shown by Lue and Norrie, in [Nor87] and [Nor90] to give the automorphism object of a crossed module $\mathcal{X}$. In this implementation, the source of the actor is a permutation representation $W$ of the Whitehead group of regular derivations, and the range of the actor is a permutation representation $A$ of the automorphism group $\text{Aut}(\mathcal{X})$ of $\mathcal{X}$.

6.1.1 AutomorphismPermGroup

$$\Delta \text{AutomorphismPermGroup}(\text{xmod})$$
$$\Delta \text{GeneratingAutomorphisms}(\text{xmod})$$
$$\Delta \text{PermAutomorphismAsXModMorphism}(\text{xmod}, \text{perm})$$

The automorphisms $(\sigma, \rho$) of $\mathcal{X}$ form a group $\text{Aut}(\mathcal{X})$ of crossed module isomorphisms. The function $\text{AutomorphismPermGroup}$ finds a set of $\text{GeneratingAutomorphisms}$ for $\text{Aut}(\mathcal{X})$, and then constructs a permutation representation of this group, which is used as the range of the actor crossed module of $\mathcal{X}$. The individual automorphisms can be constructed from the permutation group using the function $\text{PermAutomorphismAsXModMorphism}$.

Example

```
gap> APX3 := AutomorphismPermGroup( X3 );
group([ (5,7,6), (1,2)(3,4)(6,7) ])
gap> Size( APX3 );
6
gap> genX3 := GeneratingAutomorphisms( X3 );
[ [[c3->s3] => [c3->s3]], [[c3->s3] => [c3->s3]] ]
gap> e6 := Elements( APX3 )[6];
(1,2)(3,4)(5,7)
gap> m6 := PermAutomorphismAsXModMorphism( X3, e6 );
gap> Display( m6 );
Morphism of crossed modules:
: Source = [c3->s3] with generating sets:
  [ (1,2,3)(4,6,5) ]
  [ (4,5,6), (2,3)(5,6) ]
: Range = Source
```

37
6.1.2 WhiteheadXMod

\begin{itemize}
  \item $X = (\partial : S \to R)$, the initial crossed module, on the left,
  \item $\mathcal{W}(X) = (\eta : S \to W)$, the Whitehead crossed module of $X$, at the top,
  \item $\mathcal{N}(X) = (\alpha : R \to A)$, the Norrie crossed module of $X$, at the bottom,
  \item $\text{Act}(X) = (\Delta : W \to A)$, the actor crossed module of $X$, on the right, and
  \item $\mathcal{L}(X) = (\Delta \circ \eta = \alpha \circ \partial : S \to A)$, the Lue crossed module of $X$, along the top-left to bottom-right diagonal.
\end{itemize}
Boundary homomorphism maps source generators to:
\[(5,7,6)\]

Action homomorphism maps range generators to automorphisms:
\[(5,7,6) \rightarrow \{\text{source gens} \rightarrow [(1,2,3)(4,6,5)]\}\]
\[(1,2)(3,4)(6,7) \rightarrow \{\text{source gens} \rightarrow [(1,3,2)(4,5,6)]\}\]
These 2 automorphisms generate the group of automorphisms.

\[\text{gap} \] \text{NX3 := NorrieXMod( X3 );;} \\
\text{gap} \] \text{Display( NX3 );}

Crossed module Norrie\[c3->s3\] :-

Source group has generators:
\[(4,5,6), (2,3)(5,6)\]

Range group has generators:
\[(5,7,6), (1,2)(3,4)(6,7)\]

Boundary homomorphism maps source generators to:
\[(5,7,6), (1,2)(3,4)(6,7)\]

Action homomorphism maps range generators to automorphisms:
\[(5,7,6) \rightarrow \{\text{source gens} \rightarrow [(4,5,6), (2,3)(4,5)]\}\]
\[(1,2)(3,4)(6,7) \rightarrow \{\text{source gens} \rightarrow [(4,6,5), (2,3)(5,6)]\}\]
These 2 automorphisms generate the group of automorphisms.

\[\text{gap} \] \text{AX3 := ActorXMod( X3 );;} \\
\text{gap} \] \text{Display( AX3 );}

Crossed module Actor\[c3->s3\] :-

Source group has generators:
\[(1,2,3)(4,5,6), (1,4)(2,6)(3,5)\]

Range group has generators:
\[(5,7,6), (1,2)(3,4)(6,7)\]

Boundary homomorphism maps source generators to:
\[(5,6,7), (1,2)(3,4)(6,7)\]

Action homomorphism maps range generators to automorphisms:
\[(5,7,6) \rightarrow \{\text{source gens} \rightarrow [(4,5,6), (2,3)(4,5)]\}\]
\[(1,2)(3,4)(6,7) \rightarrow \{\text{source gens} \rightarrow [(4,6,5), (2,3)(5,6)]\}\]
These 2 automorphisms generate the group of automorphisms.

\[\text{gap} \] \text{IAX3 := InnerActorXMod( X3 );;} \\
\text{gap} \] \text{Display( IAX3 );}

Crossed module InnerActor\[c3->s3\] :-

Source group has generators:
\[(1,2,3)(4,5,6)\]

Range group has generators:
\[(5,6,7), (1,2)(3,4)(6,7)\]

Boundary homomorphism maps source generators to:
\[(5,7,6)\]

Action homomorphism maps range generators to automorphisms:
\[(5,7,6) \rightarrow \{\text{source gens} \rightarrow [(1,2,3)(4,5,6)]\}\]
\[(1,2)(3,4)(6,7) \rightarrow \{\text{source gens} \rightarrow [(1,3,2)(4,6,5)]\}\]
These 2 automorphisms generate the group of automorphisms.
6.1.3 \textbf{XModCentre}

\begin{itemize}
\item XModCentre(xmod)
\item InnerActorXMod(xmod)
\item InnerMorphism(xmod)
\end{itemize}

Pairs of boundaries or identity mappings provide six morphisms of crossed modules. In particular, the boundaries of $\mathcal{W}(\mathcal{X})$ and $\mathcal{N}(\mathcal{X})$ form the \textit{inner morphism} of $\mathcal{X}$, mapping source elements to principal derivations and range elements to inner automorphisms. The image of $\mathcal{X}$ under this morphism is the \textit{inner actor} of $\mathcal{X}$, while the kernel is the \textit{centre} of $\mathcal{X}$. In the example which follows, the inner morphism of $X_3=(c_3\to s_3)$, from Chapter 5, is an inclusion of crossed modules.

Note that we appear to have defined two sorts of \textit{centre} for a crossed module: XModCentre here, and CentreXMod (4.1.7) in the chapter on isoclinism. We suspect that these two definitions give the same answer, but this remains to be resolved.

\begin{verbatim}
gap> IMX3 := InnerMorphism( X3 );;
gap> Display( IMX3 );
Morphism of crossed modules :-
  : Source = [c3->s3] with generating sets:
    [ (1,2,3)(4,6,5) ]
    [ (4,5,6), (2,3)(5,6) ]
  : Range = Actor[c3->s3] with generating sets:
    [ (1,2,3)(4,5,6), (1,4)(2,6)(3,5) ]
    [ (5,7,6), (1,2)(3,4)(6,7) ]
  : Source Homomorphism maps source generators to:
    [ (1,2,3)(4,5,6) ]
  : Range Homomorphism maps range generators to:
    [ (5,6,7), (1,2)(3,4)(6,7) ]
gap> IsInjective( IMX3 );
true

gap> ZX3 := XModCentre( X3 );
[Group( () )->Group( () )]
\end{verbatim}
Chapter 7

Induced constructions

7.1 Induced crossed modules

7.1.1 InducedXMod

A morphism of crossed modules \((\sigma, \rho) : X_1 \to X_2\) factors uniquely through an induced crossed module \(\rho_\ast X_1 = (\delta : \rho_\ast S_1 \to R_2)\). Similarly, a morphism of cat1-groups factors through an induced cat1-group. Calculation of induced crossed modules of \(X\) also provides an algebraic means of determining the homotopy 2-type of homotopy pushouts of the classifying space of \(X\). For more background from algebraic topology see references in [BH78], [BW95], [BW96]. Induced crossed modules and induced cat1-groups also provide the building blocks for constructing pushouts in the categories \(XMod\) and \(Cat1\).

Data for the cases of algebraic interest is provided by a conjugation crossed module \(X = (\partial : S \to R)\) and a homomorphism \(\iota\) from \(R\) to a third group \(Q\). The output from the calculation is a crossed module \(\iota_\ast X = (\delta : \iota_\ast S \to Q)\) together with a morphism of crossed modules \(X \to \iota_\ast X\). When \(\iota\) is a surjection with kernel \(K\) then \(\iota_\ast S = [S, K]\) (see [BH78]). When \(\iota\) is an inclusion the induced crossed module may be calculated using a copower construction [BW95] or, in the case when \(R\) is normal in \(Q\), as a coproduct of crossed modules ([BW96], but not yet implemented). When \(\iota\) is neither a surjection nor an inclusion, \(\iota\) is factored as the composite of the surjection onto the image and the inclusion of the image in \(Q\), and then the composite induced crossed module is constructed. These constructions use Tietze transformation routines in the library file tietze.gi.

As a first, surjective example, we take for \(X\) the normal inclusion crossed module of \(a_4\) in \(s_4\), and for \(\iota\) the surjection from \(s_4\) to \(s_3\) with kernel \(k_4\). The induced crossed module is isomorphic to \(X_3\).

Example

```
gap> a4gens := [ (1,2), (2,3), (3,4) ];;
gap> a4 := Group( a4gens );; SetName(a4,"a4");
gap> a4gens := [ (1,2,3), (2,3,4) ];;
gap> a4 := Subgroup( a4, a4gens );; SetName( a4, "a4" );
gap> s3 := Group( (5,6),(6,7) );; SetName( s3, "s3" );
```
\begin{verbatim}
gap> epi := GroupHomomorphismByImages( s4, s3, s4gens, [(5,6),(6,7),(5,6)] );;
gap> X4 := XModByNormalSubgroup( s4, a4 );;
gap> indX4 := SurjectiveInducedXMod( X4, epi );
[a4/ker->s3]
gap> Display( indX4 );
Crossed module [a4/ker->s3] :-
: Source group a4/ker has generators:
    [ (1,3,2), (1,2,3) ]
: Range group s3 has generators:
    [ (5,6), (6,7) ]
: Boundary homomorphism maps source generators to:
    [ (5,6,7), (5,7,6) ]
: Action homomorphism maps range generators to automorphisms:
    (5,6) --> { source gens --> [ (1,2,3), (1,3,2) ] }
    (6,7) --> { source gens --> [ (1,2,3), (1,3,2) ] }
    These 2 automorphisms generate the group of automorphisms.

gap> morX4 := MorphismOfInducedXMod( indX4 );
[[a4->s4] => [a4/ker->s3]]
\end{verbatim}

For a second, injective example we take for \( X \) a conjugation crossed module.

\begin{verbatim}
Example

\begin{verbatim}
gap> d8 := Subgroup( d16, [ d1^2, d2 ] ); SetName( d8, "d8" );
Group( [ (11,13,15,17)(12,14,16,18), (12,18)(13,17)(14,16) ])
gap> c4 := Subgroup( d8, [ d1^2 ] ); SetName( c4, "c4" );
Group( [ (11,13,15,17)(12,14,16,18) ])
gap> Y16 := XModByNormalSubgroup( d16, d8 );
d8->d16
gap> Y8 := SubXMod( Y16, c4, d8 );
c4->d8
\end{verbatim}
\end{verbatim}

For a third example we take the identity mapping on s3 as boundary, and the inclusion of s3 in s4 as \( \iota \). The induced group is a general linear group GL(2,3).

\begin{verbatim}
Example

\begin{verbatim}
Example
\end{verbatim}
\end{verbatim}
# I induced group has Size: 48
i*(s3b->s3b])
gap> StructureDescription(indX3);
[ "GL(2,3)", "S4" ]

## 7.1.2 AllInducedXMods

> AllInducedXMods(Q)

This function calculates all the induced crossed modules \( \text{InducedXMod}(Q, P, M) \), where \( P \) runs over all conjugacy classes of subgroups of \( Q \) and \( M \) runs over all non-trivial subgroups of \( P \).
Chapter 8
Crossed squares and their morphisms

Crossed squares were introduced by Guin-Walery and Loday (see, for example, [BL87]) as fundamental crossed squares of commutative squares of spaces, but are also of purely algebraic interest. We denote by \([n]\) the set \(\{1, 2, \ldots, n\}\). We use the \(n = 2\) version of the definition of crossed \(n\)-cube as given by Ellis and Steiner [ES87].

A crossed square \(\mathcal{S}\) consists of the following:

- Groups \(S_J\) for each of the four subsets \(J \subseteq \{2\}\);

- a commutative diagram of group homomorphisms:
  \[\partial_1 : S_2 \to S_{\{2\}}, \quad \partial_2 : S_2 \to S_{\{1\}}, \quad \partial_1 : S_{\{1\}} \to S_0, \quad \partial_2 : S_2 \to S_0;\]

- actions of \(S_0\) on \(S_{\{1\}}, S_{\{2\}}, S_2\) which determine actions of \(S_{\{1\}}\) on \(S_2\) and \(S_{\{2\}}\) via \(\partial_1\) and actions of \(S_{\{2\}}\) on \(S_{\{1\}}\) and \(S_2\) via \(\partial_2\);

- a function \(\mathbin{\ll} : S_{\{1\}} \times S_{\{2\}} \to S_2\).

The following axioms must be satisfied for all \(l \in S_2, m, m_1, m_2 \in S_{\{1\}}, n, n_1, n_2 \in S_{\{2\}}, p \in S_0:\

- the homomorphisms \(\partial_1, \partial_2\) preserve the action of \(S_0\);

- each of

  \[\mathcal{J}_1 = (\partial_1 : S_2 \to S_{\{2\}}), \quad \mathcal{J}_2 = (\partial_2 : S_2 \to S_{\{1\}}), \quad \mathcal{J}_1 = (\partial_1 : S_{\{1\}} \to S_0), \quad \mathcal{J}_2 = (\partial_2 : S_{\{2\}} \to S_0),\]

  and the diagonal

  \[\mathcal{J}_{12} = (\partial_{12} : \partial_1 \partial_2 = \partial_2 \partial_1 : S_2 \to S_0)\]

  are crossed modules (with actions via \(S_0\));

- \(\mathbin{\ll}\) is a crossed pairing:
  \[\begin{align*}
  & (m_1 m_2 \mathbin{\ll} n) = (m_1 \mathbin{\ll} n)^{m_2} (m_2 \mathbin{\ll} n), \\
  & (m \mathbin{\ll} n_1 n_2) = (m \mathbin{\ll} n_2) (m \mathbin{\ll} n_1)^{m_2}, \\
  & (m \mathbin{\ll} n)^p = (m^p \mathbin{\ll} n^p); \\
  & \partial_1 (m \mathbin{\ll} n) = (n^{-1})^m n \quad \text{and} \quad \partial_2 (m \mathbin{\ll} n) = m^{-1} m^n,
  \end{align*}\]
• \((m \boxtimes \partial_1 l) = (l^{-1})^m l\) and \((\partial_2 l \boxtimes n) = l^{-1} l^n\).

Note that the actions of \(S\{1\}\) on \(S\{2\}\) and \(S\{2\}\) on \(S\{1\}\) via \(S\{0\}\) are compatible since
\[
m_1(n^m) = m_1 \partial_2(n^m) = m_1^{m^{-1}(\partial_2 n)m} = ((m_1^{m^{-1}} n^m)^m).
\]

### 8.1 Constructions for crossed squares

Analogously to the data structure used for crossed modules, crossed squares are implemented as 3d-groups. When time allows, cat2-groups will also be implemented, with conversion between the two types of structure. Some standard constructions of crossed squares are listed below. At present, a limited number of constructions are implemented. Morphisms of crossed squares have also been implemented, though there is a lot still to do.

#### 8.1.1 CrossedSquare

- \texttt{CrossedSquare(args)}
- \texttt{CrossedSquareByNormalSubgroups(P, N, M, L)}
- \texttt{ActorCrossedSquare(X0)}
- \texttt{Transpose3dGroup(S0)}
- \texttt{Name(S0)}

Here are some standard examples of crossed squares.

- If \(M, N\) are normal subgroups of a group \(P\), and \(L = M \cap N\), then the four inclusions, \(L \to N\), \(L \to M\), \(M \to P\), \(N \to P\), together with the actions of \(P\) on \(M\), \(N\) and \(L\) given by conjugation, form a crossed square with crossed pairing

\[
\boxtimes : M \times N \to M \cap N, \ (m, n) \mapsto [m, n] = m^{-1} n^{-1} mn = (n^{-1})^m n = m^{-1} m^n.
\]

This construction is implemented as \texttt{CrossedSquareByNormalSubgroups(P, N, M, L)};.

- The actor \(\mathcal{X}(\mathcal{X}_0)\) of a crossed module \(\mathcal{X}_0\) has been described in Chapter 5. The crossed pairing is given by

\[
\boxtimes : R \times W \to S, \ (r, \chi) \mapsto \chi r.
\]

This is implemented as \texttt{ActorCrossedSquare(X0)};.

- The transpose of \(\mathcal{X}\) is the crossed square \(\tilde{\mathcal{X}}\) obtained by interchanging \(S\{1\}\) with \(S\{2\}\), \(\partial_1\) with \(\partial_2\), and \(\partial_1\) with \(\partial_2\). The crossed pairing is given by

\[
\tilde{\boxtimes} : S\{2\} \times S\{1\} \to S\{2\}, \ (n, m) \mapsto n \tilde{\boxtimes} m := (m \boxtimes n)^{-1}.
\]

Example

\[
\begin{array}{l}
gap> c := (11,12,13,14,15,16);
gap> d := (12,16)(13,15);
gap> cd := c*d;;
gap> dl2 := Group([c, d]);;
gap> s3a := Subgroup(dl2, [c^2, d]);;
gap> s3b := Subgroup(dl2, [c^2, cd]);;
\end{array}
\]
8.1.2 CentralQuotient

The central quotient of a crossed module $\mathcal{X} = (\partial : S \rightarrow R)$ is the crossed square with:

- the left crossed module is $\mathcal{X}$;
- the right crossed module is the quotient $\mathcal{X}/\mathbb{Z}(\mathcal{X})$ (see CentreXMod (4.1.7));
- the top and bottom homomorphisms are the natural homomorphisms onto the quotient groups;
- the crossed pairing $\triangleright : (R \times F) \rightarrow S$, where $F = \text{Fix}(\mathcal{X},S,R)$, is the displacement element $\triangleright(r,Fs) = (r,s) = (s^{-1})'s$ (see Displacement (4.1.3) and section 4.3).

This is the special case of an intended function CrossedSquareByCentralExtension which has not yet been implemented. In the example $X_7 \leq X_{24}$, constructed in section 4.1.

```
gap> pos7 := Position( ids, [ [12,2], 24,5 ] );;
gap> Xn7 := nsx[pos7];
gap> IdGroup( CentreXMod(Xn7) );
[ [ 4, 1 ], [ 4, 1 ] ]
gap> CQXn7 := CentralQuotient( Xn7 );
crossed square with:
    up = Whitehead[s3a->d12]
    left = [s3a->d12]
    down = Norrie[s3a->d12]
    right = Actor[s3a->d12]
```
8.1.3 IsCrossedSquare

- IsCrossedSquare(obj) (property)
- Is3dObject(obj) (property)
- IsPerm3dObject(obj) (property)
- IsPc3dObject(obj) (property)
- IsPfp3dObject(obj) (property)
- IsPreCrossedSquare(obj) (property)

These are the basic properties for 3d-groups, and crossed squares in particular.

8.1.4 Up2dGroup

- Up2dGroup(XS) (attribute)
- Left2dGroup(XS) (attribute)
- Down2dGroup(XS) (attribute)
- Right2dGroup(XS) (attribute)
- DiagonalAction(XS) (attribute)
- XPair(XS) (attribute)
- ImageElmXPair(XS, pair) (operation)

In this implementation the attributes used in the construction of a crossed square XS are the four crossed modules (2d-groups) on the sides of the square; the diagonal action of $P$ on $L$, and the crossed pairing.

The GAP development team have suggested that crossed pairings should be implemented as a special case of BinaryMappings – a structure which does not yet exist in GAP. As a temporary measure, crossed pairings have been implemented using Mapping2ArgumentsByFunction.
8.2 Morphisms of crossed squares

This section describes an initial implementation of morphisms of (pre-)crossed squares.

8.2.1 Source

- **Source**
- **Range**
- **Up2dMorphism**
- **Left2dMorphism**
- **Down2dMorphism**
- **Right2dMorphism**

Morphisms of 3d objects are implemented as 3d mappings. These have a pair of 3d-groups as source and range, together with four 2d-morphisms mapping between the four pairs of crossed modules on the four sides of the squares. These functions return fail when invalid data is supplied.

8.2.2 IsCrossedSquareMorphism

- **IsCrossedSquareMorphism**
- **IsPreCrossedSquareMorphism**
- **IsBijective**
- **IsEndomorphism3dObject**
- **IsAutomorphism3dObject**

A morphism \( \text{mor} \) between two pre-crossed squares \( \mathcal{X}_1 \) and \( \mathcal{X}_2 \) consists of four crossed module morphisms \( \text{Up2dMorphism}( \text{mor} ) \), mapping the \( \text{Up2dGroup} \) of \( \mathcal{X}_1 \) to that of \( \mathcal{X}_2 \), \( \text{Left2dMorphism}( \text{mor} ) \), \( \text{Down2dMorphism}( \text{mor} ) \) and \( \text{Right2dMorphism}( \text{mor} ) \). These four morphisms are required to commute with the four boundary maps and to preserve the rest of the structure. The current version of \( \text{IsCrossedSquareMorphism} \) does not perform all the required checks.

Example

```gap
gap> ad12 := GroupHomomorphismByImages( d12, d12, [c,d], [c,d^c] );;
gap> as3a := GroupHomomorphismByImages( s3a, s3a, [c^2,d], [c^2,d^c] );;
gap> as3b := GroupHomomorphismByImages( s3b, s3b, [c^2,cd], [c^2,cd^c] );;
gap> idc3 := IdentityMapping( c3 );;
gap> upconj := Up2dGroup( XSconj );;
gap> leftconj := Left2dGroup( XSconj );;
gap> downconj := Down2dGroup( XSconj );;
gap> rightconj := Right2dGroup( XSconj );;
gap> up := XModMorphismByHoms( upconj, upconj, idc3, as3b );
[[c3->s3b] => [c3->s3b]]
gap> left := XModMorphismByHoms( leftconj, leftconj, idc3, as3a );
[[c3->s3a] => [c3->s3a]]
gap> down := XModMorphismByHoms( downconj, downconj, as3a, ad12 );
[[s3a->d12] => [s3a->d12]]
gap> right := XModMorphismByHoms( rightconj, rightconj, as3b, ad12 );
[[s3b->d12] => [s3b->d12]]
gap> autoconj := CrossedSquareMorphism( XSconj, XSconj, up, left, down, right );;
gap> ord := Order( autoconj );;
```
Morphism of crossed squares :-
  Source = [c3->s3b, s3a->d12]
  Range = [c3->s3b, s3a->d12]
  order = 3
  up-left: [ [ (11,13,15)(12,14,16) ], [ (11,13,15)(12,14,16) ] ]
  up-right: [ [ (11,13,15)(12,14,16), (11,16)(12,15)(13,14) ],
               [ (11,13,15)(12,14,16), (11,12)(13,16)(14,15) ] ]
  down-left: [ [ (11,13,15)(12,14,16), (12,16)(13,15) ],
                [ (11,13,15)(12,14,16), (11,13)(14,16) ] ]
  down-right: [ [ (11,12,13,14,15,16), (12,16)(13,15) ],
                 [ (11,12,13,14,15,16), (11,13)(14,16) ] ]

gap> KnownPropertiesOfObject( autoconj );
  "IsPreCrossedSquareMorphism", "IsCrossedSquareMorphism", "IsEndomorphism3dObject" ]

gap> IsAutomorphism3dObject( autoconj );
true
Chapter 9

Crossed modules of groupoids

9.1 Constructions for crossed modules of groupoids

A typical example of a crossed module over a groupoid has as range a connected groupoid which is a direct product of a group and a complete graph, and as source a totally disconnected groupoid, with the same objects. The boundary morphism is constant on objects. For details and other references see [AW10].

9.1.1 PreXModWithObjectsObj

\[
\text{\texttt{PreXModWithObjectsObj}} \quad (\text{operation})
\]

\[
\text{\texttt{DiscreteNormalPreXModWithObjects}} \quad (\text{operation})
\]

The next stage of development of this package will be to implement constructions of crossed modules over groupoids. This will require further developments in the \texttt{Gpd} package. The following example shows what has been achieved in an earlier version, but which fails in \texttt{GAP} 4.7.

```
gap> d8 := Group( (1,2,3,4), (1,3) );;
gap> SetName( d8, "d8" );
gap> Gd8 := SinglePieceGroupoid( d8, [-9,-8,-7] );;
gap> Display( Gd8 );
single piece groupoid:
  objects: [-9, -8, -7]
  group: \texttt{Group}( \langle (1,2,3,4), (1,3) \rangle )
gap> k4 := Subgroup( d8, [ (1,2)(3,4), (1,3)(2,4) ] );;
gap> PX0 := DiscreteNormalPreXModWithObjects( Gd8, k4 );
```

```
homogeneous, discrete groupoid with:
  group: \texttt{Group}( \langle (1,2)(3,4), (1,3)(2,4) \rangle ) >
  objects: [-9, -8, -7]
#I now need to be able to test: ok := IsXMod( PM );
```

```
gap> Source(PX0);
perm homogeneous, discrete groupoid: < \texttt{Group}( \langle (1,2)(3,4), (1,3)(2,4) \rangle ), [-9, -8, -7] >
```
Chapter 10

Utility functions

By a utility function we mean a GAP function which is
• needed by other functions in this package,
• not (as far as we know) provided by the standard GAP library,
• more suitable for inclusion in the main library than in this package.

10.1 Printing Lists

10.1.1 PrintListOneItemPerLine

Example

```
gap> L := [ [1,2,3,4], true, [ (1,2), (2,3) ] ];
gap> PrintListOneItemPerLine( L );
[ [ 1, 2, 3, 4 ],
true,
[ (1,2), (2,3) ]
]
```

10.2 Inclusion and Restriction Mappings

These functions have been moved to the gpd package, but are still documented here.

10.2.1 InclusionMappingGroups

Example

```
gap> L := [ [1,2,3,4], true, [ (1,2), (2,3) ] ];
gap> PrintListOneItemPerLine( L );
[ [ 1, 2, 3, 4 ],
true,
[ (1,2), (2,3) ]
]
```

This set of utilities concerns mappings. The map incd8 is the inclusion of d8 in d16 used in Section 3.4.
Example

```gap
Print( incd8, "\n" );
[ (11,13,15,17)(12,14,16,18), (11,18)(12,17)(13,16)(14,15) ] ->
[ (11,13,15,17)(12,14,16,18), (11,18)(12,17)(13,16)(14,15) ]
gap> imd8 := Image( incd8 );;
gap> MappingToOne( c4, imd8 );
[ (11,13,15,17)(12,14,16,18) ] -> [ () ]
```

10.2.2 InnerAutomorphismByNormalSubgroup

- `InnerAutomorphismByNormalSubgroup(G, N)` (operation)
- `IsGroupOfAutomorphisms(A)` (property)

Inner automorphisms of a group `G` by the elements of a normal subgroup `N` are calculated with the first of these functions, usually with `G = N`.

Example

```gap
autd8 := AutomorphismGroup( d8 );;
innd8 := InnerAutomorphismsByNormalSubgroup( d8, d8 );;
gap> GeneratorsOfGroup( innd8 );
[ ^(1,2,3,4), ^(1,3) ]
gap> IsGroupOfAutomorphisms( innd8 );
true
```

10.3 Abelian Modules

10.3.1 AbelianModuleObject

- `AbelianModuleObject(grp, act)` (operation)
- `IsAbelianModule(obj)` (property)
- `AbelianModuleGroup(obj)` (attribute)
- `AbelianModuleAction(obj)` (attribute)

An abelian module is an abelian group together with a group action. These are used by the crossed module constructor `XModByAbelianModule`.

The resulting `Xabmod` is isomorphic to the output from `XModByAutomorphismGroup( k4 );`.

Example

```gap
gap> x := (6,7)(8,9);; y := (6,8)(7,9);; z := (6,9)(7,8);;
gap> k4 := Group( x, y );; SetName( k4, "k4" );
gap> s3 := Group( (1,2), (2,3) );; SetName( s3, "s3" );
gap> alpha := GroupHomomorphismByImages( k4, k4, [x,y], [y,x] );;
gap> beta := GroupHomomorphismByImages( k4, k4, [x,y], [x,z] );;
gap> aut := Group( alpha, beta );;
gap> act := GroupHomomorphismByImages( s3, aut, [(1,2),(2,3)], [alpha,beta] );
gap> abmod := AbelianModuleObject( k4, act );
```
gap> Xabmod := XModByAbelianModule( abmod );
[k4->s3]
gap> Display( Xabmod );

Crossed module [k4->s3] :-
: Source group k4 has generators:
  [ (6,7)(8,9), (6,8)(7,9) ]
: Range group s3 has generators:
  [ (1,2), (2,3) ]
: Boundary homomorphism maps source generators to:
  [ (), () ]
: Action homomorphism maps range generators to automorphisms:
  (1,2) --> { source gens --> [ (6,8)(7,9), (6,7)(8,9) ] } 
  (2,3) --> { source gens --> [ (6,7)(8,9), (6,9)(7,8) ] } 
These 2 automorphisms generate the group of automorphisms.

10.4 Distinct and Common Representatives

10.4.1 DistinctRepresentatives

The final set of utilities deal with lists of subsets of [1...n] and construct systems of distinct and common representatives using simple, non-recursive, combinatorial algorithms.

When \( L \) is a set of \( n \) subsets of [1...n] and the Hall condition is satisfied (the union of any \( k \) subsets has at least \( k \) elements), a set of distinct representatives exists.

When \( J, K \) are both lists of \( n \) sets, the function CommonRepresentatives returns two lists: the set of representatives, and a permutation of the subsets of the second list. It may also be used to provide a common transversal for sets of left and right cosets of a subgroup \( H \) of a group \( G \), although a greedy algorithm is usually quicker.

Example

gap> J := [ [1,2,3], [3,4], [3,4], [1,2,4] ];
[ [ 1, 2, 3 ], [ 3, 4 ], [ 3, 4 ], [ 1, 2, 4 ] ]
gap> DistinctRepresentatives( J );
[ 1, 3, 4, 2 ]
gap> K := [ [3,4], [1,2], [2,3], [2,3,4] ];
[ [ 3, 4 ], [ 1, 2 ], [ 2, 3 ], [ 2, 3, 4 ] ]
gap> CommonRepresentatives( J, K );
[ 3, 3, 3, 1 ], [ 1, 3, 4, 2 ]
gap> CommonTransversal( d16, c4 );
[ (), (12,18)(13,17)(14,16), (11,12,13,14,15,16,17,18), (11,12)(13,18)(14,17)(15,16) ]
Chapter 11

Development history

This chapter, which contains details of the major changes to the package as it develops, was first created in April 2002. Details of the changes from XMod 1 to XMod 2.001 are far from complete. Starting with version 2.009 the file CHANGES lists the minor changes as well as the more fundamental ones.

The inspiration for this package was the need, in the mid-1990’s, to calculate induced crossed modules (see [BW95], [BW96], [BW03]). GAP was chosen over other computational group theory systems because the code was freely available, and it was possible to modify the Tietze transformation code so as to record the images of the original generators of a presentation as words in the simplified presentation. (These modifications are now a standard part of the Tietze transformation package in GAP.)

11.1 Changes from version to version

11.1.1 Version 1 for GAP 3

The first version of XMod became an accepted package for GAP 3.4.3 in December 1996.

11.1.2 Version 2

Conversion of XMod 1 from GAP 3.4.3 to the new GAP syntax began soon after GAP 4 was released, and had a lengthy gestation. The new GAP syntax encouraged a re-naming of many of the function names. An early decision was to introduce generic categories 2dDomain for (pre-)crossed modules and (pre-)cat1-groups, and 2dMapping for the various types of morphism. In 2.009 3dDomain was used for crossed squares and cat2-groups, and 3dMapping for their morphisms. A generic name for derivations and sections is also required, and Up2dMapping is currently used.

11.1.3 Version 2.001 for GAP 4

This was the first version of XMod for GAP 4, completed in April 2002 in a rush to catch the release of GAP 4.3. Functions for actors and induced crossed modules were not included, nor many of the functions for derivations and sections, for example InnerDerivation.
11.1.4 Induced crossed modules

During the period May 20th - May 27th 2002 converted induce.g to induce.gd and induce.gi (later renamed gp2ind.gd, gp2ind.gi), at least as regards induced crossed modules. (Induced cat1-groups may be converted one day.) For details, see the file CHANGES.

11.1.5 Versions 2.002 – 2.006

Version 2.002 was prepared for the 4.4 release at the end of January 2004.
  Version 2.003 of February 28th 2004 just fixed some file protections.
  Version 2.004 of April 14th 2004 added the Cat1Select functionality of version 1 to the Cat1 function (see also version 2.007).
  Version 2.005 of April 16th 2004 moved the example files from tst/test_i.g to examples/example_i.g, and converted testmanual.g to a proper test file tst/xmod_manual.tst.
  A significant change was the conversion of the actor crossed module functions from the 3.4.4 version, including AutomorphismPermGroup for a crossed module, WhiteheadXMod, NorrieXMod, LueXMod, ActorXMod, Centre of a crossed module, InnerMorphism and InnerActorXMod.

11.1.6 Versions 2.007 – 2.010

These versions contain changes made between September 2004 and October 2007.
  • Added basic functions for crossed squares, considered as 3dObjects with crossed pairings, and their morphisms. Groups with two normal subgroups, and the actor of a crossed module, provide standard examples of crossed squares. (Cat2-groups are not yet implemented.)
  • Converted the documentation to the format of the GAPDoc package.
  • Improved AutomorphismPermGroup for crossed modules, and introduced a special method for conjugation crossed modules.
  • Substantial revision made to XModByCentralExtension, NorrieXMod, LueXMod, ActorXMod, and InclusionInducedXModByCopower.
  • Reintroduced the Cat1Select operation.
  • Version 2.010, of October 2007, was timed to coincide with the release of GAP 4.4.10, and included a change of filenames; correct file protection codes; and an improvement to AutomorphismPermGroup for crossed modules.

11.2 Versions for GAP 4.5, 4.6 and 4.7

Version 2.19, released on 9th June 2012, included the following changes:
  • The file makedocrel.g was copied, with appropriate changes, from GAPDoc, and now provides the correct way to update the documentation.
  • The first functions for crossed modules of groupoids were introduced.
  • The package webpage has moved along with the whole of the Bangor Maths website: http://www.maths.bangor.ac.uk/chda/.
  • A GNU General Public License declaration has been added.
11.2.1 AllCat1s

Version 2.21 contained major changes to the Cat1Select operation: the list CAT1_LIST of cat1-structures in the data file cat1data.g was changed from permutation groups to pc-groups, with the generators of subgroups; images of the tail map; and images of the head map being given as ExtRep0f0bj of words in the generators.

The AllCat1s function was reintroduced from the GAP3 version, and renamed as the operation AllCat1sBasic.

In version 2.25 the data in cat1data.g was extended from groups of size up to 48 to groups of size up to 70. In particular, the 267 groups of size 64 give rise to a total of 1275 cat1-groups. The authors are indebted to Van Luyen Le in Galway for pointing out a number of errors in the version of this list distributed with version 2.24 of this package.

11.2.2 Version 2.43

This latest version was released on 11th November 2015, following a change in the package website location. A Bitbucket repository for the package was also created.

11.3 What needs doing next?

- Speed up the calculation of Whitehead groups.
- Add more functions for 3dObjects and implement cat2-groups.
- Improve interaction with the package Gpd implementing the group groupoid version of a crossed module, and adding more functions for crossed modules of groupoids.
- Add interaction with IdRel (and possibly XRes and natp).
- Need InverseGeneralMapping for morphisms and more features for FpXMods, PcXMods, etc.
- Implement actions of a crossed module.
- Implement FreeXMods and an operation Isomorphism2dDomains.
- Allow the construction of a group of morphisms of crossed modules.
- Complete the conversion from Version 1 of the calculation of sections using EndoClasses.
- More crossed square constructions:
  - If $M,N$ are ordinary $P$-modules and $A$ is an arbitrary abelian group on which $P$ acts trivially, then there is a crossed square with sides
    \[ 0 : A \to N, \quad 0 : A \to M, \quad 0 : M \to P, \quad 0 : N \to P. \]
  - For a group $L$, the automorphism crossed module $\text{Act } L = (t : L \to \text{Aut } L)$ splits to form the square with $(t_1 : L \to \text{Inn } L)$ on two sides, and $(t_2 : \text{Inn } L \to \text{Aut } L)$ on the other two sides, where $t_1$ maps $L \in L$ to the inner automorphism $\beta l : L \to L, l' \to l^{-1}l'$, and $\text{Inn } L \to \text{Inn } L$ is the inclusion of $\text{Inn } L$ in $\text{Aut } L$. The actions are standard, and the crossed pairing is
    \[ \boxtimes : \text{Inn } L \times \text{Inn } L \to L, \quad (\beta l, \beta l') \mapsto [l,l'] \].
References


Index

2d-domain, 8
2d-domain with objects, 50
2d-group, 8
2d-mapping, 18
3d-group, 44
3d-mapping, 48

abelian module, 52
AbelianModuleAction, 52
AbelianModuleGroup, 52
AbelianModuleObject, 52
actor, 37
ActorCrossedSquare, 45
ActorXMod, 38
AllCat1sBasic, 17
AllDerivations, 34
AllInducedXMods, 43
AllSections, 34
AllStemGroupFamilies, 29
AllStemGroupIds, 29
AllXMods, 28
AllXModsUpToIsomorphism, 28
AreIsoclinicDomains, 29, 30
AutoGroup, 9
AutomorphismPermGroup, 37, 38
Boundary, 9, 13

Cat1, 13
cat1-group, 13
Cat1ByPeifferQuotient, 13
Cat1Morphism, 20
Cat1MorphismByHoms, 20
Cat1OfXMod, 14
Cat1Select, 15
cat2-group, 44
Centralizer, 26
Central1Quotient, 26, 46
CentreXMod, 26
CommonRepresentatives, 53
CommonTransversal, 53
CommutatorSubXMod, 25
CompositeDerivation, 35
CompositeSection, 35
CompositionMorphism, 21
CrossActionSubgroup, 25
crossed module, 8, 44
crossed module morphism, 18
crossed module of groupoids, 50
crossed module over a groupoid, 50
crossed pairing, 44
crossed square, 38
crossed square morphism, 48
CrossedSquare, 45
CrossedSquareByNormalSubgroups, 45
derivation, of crossed module, 32
DerivationByImages, 32
DerivationBySection, 33
DerivedSubXMod, 25
DiagonalAction, 47
DirectProduct, 8
DiscreteNormalPreXModWithObjects, 50
Displacement, 24
DisplacementSubgroup, 24
display a 2d-group, 9
display a 2d-mapping, 19
distinct and common representatives, 53
DistinctRepresentatives, 53
Down2dGroup, 47
Down2dMorphism, 48
ExternalSetXMod, 9
FactorXMod, 23
FixedPointSubgroupXMod, 25
GeneratingAutomorphisms, 37
HeadMap, 13
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IdentityMapping, 19, 20</td>
<td></td>
</tr>
<tr>
<td>IdGroup, 9, 17</td>
<td></td>
</tr>
<tr>
<td>ImageElmXPair, 47</td>
<td></td>
</tr>
<tr>
<td>ImagesList, 34</td>
<td></td>
</tr>
<tr>
<td>ImagesTable, 34</td>
<td></td>
</tr>
<tr>
<td>inclusion mapping, 51</td>
<td></td>
</tr>
<tr>
<td>InclusionMappingGroups, 51</td>
<td></td>
</tr>
<tr>
<td>InclusionMorphism2dDomains, 19, 20</td>
<td></td>
</tr>
<tr>
<td>induced crossed module, 41</td>
<td></td>
</tr>
<tr>
<td>InducedCat1, 41</td>
<td></td>
</tr>
<tr>
<td>InducedXMod, 41</td>
<td></td>
</tr>
<tr>
<td>InfoXMod, 6</td>
<td></td>
</tr>
<tr>
<td>InnerActorXMod, 40</td>
<td></td>
</tr>
<tr>
<td>InnerAutomorphismByNormalSubgroup, 52</td>
<td></td>
</tr>
<tr>
<td>InnerAutomorphismCat1, 20</td>
<td></td>
</tr>
<tr>
<td>InnerAutomorphismXMod, 19</td>
<td></td>
</tr>
<tr>
<td>InnerMorphism, 40</td>
<td></td>
</tr>
<tr>
<td>IntersectionSubXMods, 24</td>
<td></td>
</tr>
<tr>
<td>Is3dObject, 47</td>
<td></td>
</tr>
<tr>
<td>IsAbelian2dGroup, 26</td>
<td></td>
</tr>
<tr>
<td>IsAbelianModule, 52</td>
<td></td>
</tr>
<tr>
<td>IsAbelianModule2dGroup, 10</td>
<td></td>
</tr>
<tr>
<td>IsAspherical2dGroup, 26</td>
<td></td>
</tr>
<tr>
<td>IsAutomorphism3dObject, 48</td>
<td></td>
</tr>
<tr>
<td>IsAutomorphismGroup2dGroup, 10</td>
<td></td>
</tr>
<tr>
<td>IsBijective, 19, 48</td>
<td></td>
</tr>
<tr>
<td>IsCat1Morphism, 20</td>
<td></td>
</tr>
<tr>
<td>IsCentralExtension2dGroup, 10</td>
<td></td>
</tr>
<tr>
<td>IsCommonTransversal, 53</td>
<td></td>
</tr>
<tr>
<td>IsCrossedSquare, 47</td>
<td></td>
</tr>
<tr>
<td>IsCrossedSquareMorphism, 48</td>
<td></td>
</tr>
<tr>
<td>IsDerivation, 32</td>
<td></td>
</tr>
<tr>
<td>IsEndo2dMapping, 19</td>
<td></td>
</tr>
<tr>
<td>IsEndomorphism3dObject, 48</td>
<td></td>
</tr>
<tr>
<td>IsFaithful2dGroup, 27</td>
<td></td>
</tr>
<tr>
<td>IsFp2dGroup, 10</td>
<td></td>
</tr>
<tr>
<td>IsFp3dObject, 47</td>
<td></td>
</tr>
<tr>
<td>IsGroupOfAutomorphisms, 52</td>
<td></td>
</tr>
<tr>
<td>IsInducedXMod, 41</td>
<td></td>
</tr>
<tr>
<td>IsInjective, 18</td>
<td></td>
</tr>
<tr>
<td>IsNilpotent2dGroup, 27</td>
<td></td>
</tr>
<tr>
<td>IsNormal for crossed modules, 10</td>
<td></td>
</tr>
<tr>
<td>IsNormalSubgroup2dGroup, 10</td>
<td></td>
</tr>
<tr>
<td>IsoclinicStemDomain, 29, 31</td>
<td></td>
</tr>
<tr>
<td>Isoclinism, 29, 30</td>
<td></td>
</tr>
<tr>
<td>IsomorphismPc2dGroup, 19</td>
<td></td>
</tr>
<tr>
<td>IsomorphismPerm2dGroup, 19</td>
<td></td>
</tr>
<tr>
<td>IsomorphismXMods, 28</td>
<td></td>
</tr>
<tr>
<td>IsPc2dGroup, 10</td>
<td></td>
</tr>
<tr>
<td>IsPc3dObject, 47</td>
<td></td>
</tr>
<tr>
<td>IsPerm2dGroup, 10</td>
<td></td>
</tr>
<tr>
<td>IsPerm3dObject, 47</td>
<td></td>
</tr>
<tr>
<td>IsPreCat1Morphism, 20</td>
<td></td>
</tr>
<tr>
<td>IsPreCrossedSquare, 47</td>
<td></td>
</tr>
<tr>
<td>IsPreCrossedSquareMorphism, 48</td>
<td></td>
</tr>
<tr>
<td>IsPreXMod, 10</td>
<td></td>
</tr>
<tr>
<td>IsPreXModMorphism, 18</td>
<td></td>
</tr>
<tr>
<td>IsSection, 32</td>
<td></td>
</tr>
<tr>
<td>IsSimplyConnected2dGroup, 26</td>
<td></td>
</tr>
<tr>
<td>IsSingleValued, 19</td>
<td></td>
</tr>
<tr>
<td>IsStemDomain, 29, 31</td>
<td></td>
</tr>
<tr>
<td>IsSurjective, 18</td>
<td></td>
</tr>
<tr>
<td>IsTotal, 19</td>
<td></td>
</tr>
<tr>
<td>IsTrivialAction2dGroup, 10</td>
<td></td>
</tr>
<tr>
<td>IsUp2dMapping, 32</td>
<td></td>
</tr>
<tr>
<td>IsXMod, 10</td>
<td></td>
</tr>
<tr>
<td>IsXModMorphism, 18</td>
<td></td>
</tr>
<tr>
<td>Kernel, 21</td>
<td></td>
</tr>
<tr>
<td>Kernel2dMapping, 21</td>
<td></td>
</tr>
<tr>
<td>KernelEmbedding, 13</td>
<td></td>
</tr>
<tr>
<td>Left2dGroup, 47</td>
<td></td>
</tr>
<tr>
<td>Left2dMorphism, 48</td>
<td></td>
</tr>
<tr>
<td>License, 2</td>
<td></td>
</tr>
<tr>
<td>LowerCentralSeriesOfXMod, 27</td>
<td></td>
</tr>
<tr>
<td>LueXMod, 38</td>
<td></td>
</tr>
<tr>
<td>MappingToOne, 51</td>
<td></td>
</tr>
<tr>
<td>MiddleLength, 30, 31</td>
<td></td>
</tr>
<tr>
<td>morphism, 18</td>
<td></td>
</tr>
<tr>
<td>morphism of 2d-group, 18</td>
<td></td>
</tr>
<tr>
<td>morphism of 3d-group, 48</td>
<td></td>
</tr>
<tr>
<td>Morphism0fInducedXMod, 41</td>
<td></td>
</tr>
<tr>
<td>Name, 9, 13, 45</td>
<td></td>
</tr>
<tr>
<td>NaturalMorphismByNormalSubXMod, 23</td>
<td></td>
</tr>
<tr>
<td>NilpotencyClass2dGroup, 27</td>
<td></td>
</tr>
<tr>
<td>Normalizer, 26</td>
<td></td>
</tr>
<tr>
<td>NormalSubXMods, 11</td>
<td></td>
</tr>
<tr>
<td>NorrieXMod, 38</td>
<td></td>
</tr>
<tr>
<td>operations on morphisms, 21</td>
<td></td>
</tr>
<tr>
<td>order of a 2d-automorphism, 19</td>
<td></td>
</tr>
<tr>
<td>Peiffer subgroup, 12</td>
<td></td>
</tr>
</tbody>
</table>