XMod

A GAP4 Package

For Computing with Crossed Modules

Version 2.001

by

Murat Alp

Dumlupinar Universitesi, Fen-Edebiyat Fakultesi, Matematik Bolumu

Merkez Kampus, Kutahya, Turkey

email: malp@mail.dumlupinar.edu.tr

and

Chris Wensley

School of Informatics, University of Wales Bangor

Dean Street, Bangor, Gwynedd, LL57 1UT, U.K.

email: c.d.wensley@bangor.ac.uk

April 2002

Contents

1 Preface

The XMod package provides functions for the computation with

- crossed modules and cat1-groups;
- pre-crossed modules, pre-cat1-groups, and their Peiffer quotients;
- morphisms of these structures;
- their derivations and sections; and
- the actor crossed square of a crossed module.

It is loaded with the command

```
gap> RequirePackage( "xmod" );
```

XMod was originally implemented in 1997 using the GAP 3 language. The first two parts have been converted to GAP 4. The parts covering derivations; sections and actors will follow in due course. Many of the function names have been changed during the conversion, for example `ConjugationXMod` has become `XModByNormalSubgroup`.

1 ▶ `InfoXMod` V

In order that the user has some control of the verbosity of the XMod package's functions, an *InfoClass* `InfoXMod` is provided (see Chapter 7.4 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 levels of `1`, `2` and `3`, e.g.

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

1.1 Authors

XMod has been developed by

Murat Alp
Dumlupinar Universitesi, Fen-Edebiyat Fakultesi, Matematik Bolumu,
Merkez Kampus, Kutahya, Turkey.
e-mail: `malp@mail.dumlupinar.edu.tr`

Chris Wensley
School of Informatics, University of Wales Bangor,
Dean Street, Bangor, Gwynedd, LL57 1UT, U.K.
e-mail: `c.d.wensley@bangor.ac.uk`

Please send bug reports, suggestions and other comments to the second of these e-mail addresses.

Additional information can be found on the **Computational Discrete Algebra** web site at

`http://www.informatics.bangor.ac.uk/public/math/research/hdda/`

2 2d-objects

2.1 Introduction

The XMod package provides functions for computation with finite **crossed modules** and **cat1-groups** and their morphisms.

Crossed modules and cat1-groups are special types of **2-dimensional groups** [B82] and are implemented as `2dObjects` having a `Source` and a `Range`.

The package divides into four parts, three of which have so far been converted from GAP 3 to GAP 4.

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 deals 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 fourth part deals with actor crossed modules and actor cat1-groups. These are the automorphism objects in the appropriate categories. For the actor crossed module $\mathrm{Act}(\mathcal{X})$ 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 $\mathrm{Act}(\mathcal{X})$ whose kernel is the centre of $\mathcal{X}$.

The package may be obtained as a compressed file by ftp from one of the sites with a GAP 4 archive.

The following constructions are new in this version of the package. Firstly, sub-2d-object functions have been included. Secondly, functions for pre-crossed modules, the Peiffer subgroup of a pre-crossed module, and the associated crossed modules, have been added. The source and range groups in these constructions are no longer required to be permutation groups.

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) and **crossed squares** and the equivalent **cat2-groups**, structures which arise as 3-dimensional groups. Examples of these are implicitly included in the fourth part, namely inclusions of normal sub-crossed modules, and the inner morphism from a crossed module to its actor.

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 crossed resolutions package XRes [M01], and further work on group groupoids is planned.

2.2 Crossed modules

The term crossed module was introduced by J. H. C. Whitehead in [Whi48], [Whi49]. In [Lod82], Loday reformulated the notion of a crossed module as a cat1-group. Norrie [Nor90], [Nor87] and Gilbert [G90] have studied derivations, automorphisms of crossed modules and the actor of a crossed module, while Ellis [E84] 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 [AW96] 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.

A crossed module $\mathcal{X} = (\partial : S \to R)$ consists of a group homomorphism ∂, called the **boundary** of $\mathcal{X}$, with **source** S and **range** R, together with an action $\alpha : R \to \mathrm{Aut}(S)$ satisfying, for all $s, s_1, s_2 \in S$ and $r \in R$,

$$\begin{array}{lcl} \textbf{XMod 1} & : & \partial(s^r) \;=\; r^{-1}(\partial s)r \;, \\ \textbf{XMod 2} & : & s_1^{\partial s_2} \;=\; s_2^{-1} s_1 s_2 \;. \end{array}$$

The kernel of ∂ is abelian.

There are a variety of constructors for crossed modules:

1 ▶ `XMod(` *args* `)` F
▶ `XModByBoundaryAndAction(` *bdy, act* `)` O
▶ `XModByTrivialAction(` *bdy* `)` O
▶ `XModByNormalSubgroup(` *G, N* `)` O
▶ `XModByCentralExtension(` *bdy* `)` O
▶ `XModByAutomorphismGroup(` *grp* `)` O
▶ `XModByInnerAutomorphismGroup(` *grp* `)` O
▶ `XModByGroupOfAutomorphisms(` *G, A* `)` O
▶ `XModByRModule(` *abgrp* `)` O

Here are the standard constructions which these implement:

- A **conjugation crossed module** is an 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 $\mathrm{Aut}(S)$ 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 **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.
- An **R-Module crossed module** has an R-module as source and the zero map as boundary.
- The direct product $\mathcal{X}_1 \times \mathcal{X}_2$ 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.

In this implementation the attributes used in the construction of a crossed module *X0* are:

- `Source(X)` and `Range(X)`, the source S and range R of $\partial =$ `Boundary(X)`;
- `XModAction(X)`, a homomorphism from R to `AutoGroup(X)`, a group of automorphisms of S:

2 ▶ `Source( X0 )` A
▶ `Range( X0 )` A
▶ `Boundary( X0 )` A
▶ `AutoGroup( X0 )` A
▶ `XModAction( X0 )` A

More familiar attributes are `Size` and `Name`, formed by concatenating the names of the source and range. An `Enumerator` function has not yet been implemented.

3 ▶ `Size( X0 )` A
▶ `Name( X0 )` A

Here is a simple example of an automorphism crossed module, the holomorph of the cyclic group of size five.

```
gap> c5 := Group( (1,2,3,4,5) );;
gap> SetName( c5, "c5" );
gap> X1 := XModByAutomorphismGroup( c5 );
[c5 -> PermAut(c5)]
gap> Display( X1 );
Crossed module [c5 -> PermAut(c5)] :-
: Source group c5 has generators:
  [ (1,2,3,4,5) ]
: Range group PermAut(c5) has generators:
  [ (1,2,4,3) ]
: Boundary homomorphism maps source generators to:
  [ () ]
: Action homomorphism maps range generators to automorphisms:
  (1,2,4,3) --> { source gens --> [ (1,3,5,2,4) ] }
  This automorphism generates the group of automorphisms.
gap> Size( X1 );
[ 5, 4 ]
```

4 ▶ `SubXMod( X0, src, rng )` O
▶ `IdentitySubXMod( X0 )` A
▶ `NormalSubXMods( X0 )` A
▶ `DirectProduct( X1, X2 )` O

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

- N, M are normal subgroups of S, R respectively,
- δ is the restriction of ∂,
- $n^r \in N$ for all $n \in N$, $r \in R$,
- $s^{-1}\, s^m \in N$ for all $m \in M$, $s \in S$.

These conditions ensure that $M \ltimes N$ is normal in the semidirect product $R \ltimes S$.

2.3 Pre-crossed modules

1 ▶ `PeifferSubgroup(` *X0* `)` A
▶ `PreXModByBoundaryAndAction(` *bdy, act* `)` O
▶ `PreXModByCentralExtension(` *bdy* `)` O
▶ `SubPreXMod(` *X0, src, rng* `)` O
▶ `XModByPeifferQuotient(` *pre-xmod* `)` A

When axiom **XMod 2** is **not** satisfied, the corresponding structure is known as a **pre-crossed module**. In this case the **Peiffer subgroup** of P of S is the subgroup of $\ker(\partial)$ generated by **Peiffer commutators**

$$\langle s_1, s_2 \rangle \quad = \quad (s_1^{-1})^{\partial s_2}\, s_2^{-1}\, s_1\, s_2 \ .$$

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.

2 ▶ `IsPermXMod(` *X0* `)` P
▶ `IsPcPreXMod(` *X0* `)` P

When both source and range groups are of the same type, corresponding properties are assigned to the crossed module.

In the following example the Peiffer subgroup ios cyclic of size 4.

```
gap> d16 := Group( (1,2,3,4,5,6,7,8), (2,8)(3,7)(4,6) );;
gap> SetName( d16, "d16" );
gap> gend16 := GeneratorsOfGroup( d16 );;
gap> sk4 := Subgroup( d16, [ (1,5)(3,7)(2,6)(4,8), (2,8)(3,7)(4,6) ] );;
gap> gensk4 := GeneratorsOfGroup( sk4 );;
gap> SetName( sk4, "sk4" );
gap> f16 := GroupHomomorphismByImages( d16, sk4, gend16, gensk4 );;
gap> P16 := PreXModByCentralExtension( f16 );;
gap> Display(P16);
Pre-crossed module [d16 -> sk4] :-
: Source group d16 has generators:
  [ (1,2,3,4,5,6,7,8), (2,8)(3,7)(4,6) ]
: Range group has generators:
  [ (1,5)(2,6)(3,7)(4,8), (2,8)(3,7)(4,6) ]
: Boundary homomorphism maps source generators to:
  [ (1,5)(2,6)(3,7)(4,8), (2,8)(3,7)(4,6) ]
: Action homomorphism maps range generators to automorphisms:
  (1,5)(2,6)(3,7)(4,8) --> { source gens -->
[ (1,2,3,4,5,6,7,8), (2,8)(3,7)(4,6) ] }
  (2,8)(3,7)(4,6) --> { source gens --> [ (1,8,7,6,5,4,3,2), (2,8)(3,7)(4,6) ] }
  These 2 automorphisms generate the group of automorphisms.

gap> P := PeifferSubgroup( P16 );
Group([ (1,7,5,3)(2,8,6,4) ])
gap> X16 := XModByPeifferQuotient( P16 );;
gap> Display( X16 );
Crossed module [? -> sk4] :-
: Source group has generators:
  [ f1, f2 ]
: Range group has generators:
  [ (1,5)(2,6)(3,7)(4,8), (2,8)(3,7)(4,6) ]
```

```
: Boundary homomorphism maps source generators to:
  [ (2,8)(3,7)(4,6), (1,5)(2,6)(3,7)(4,8) ]
  The automorphism group is trivial
gap> iso16 := IsomorphismPermGroup( Source( X16 ) );;
gap> S16 := Image( iso16 );
Group([ (1,3)(2,4), (1,2)(3,4) ])
```

2.4 Cat1-groups and pre-cat1-groups

In this implementation a cat1-group $\mathcal{C}$ has attributes:

1 ▶ `Source( C )` A
▶ `Range( C )` A
▶ `Tail( C )` A
▶ `Head( C )` A
▶ `RangeEmbedding( C )` A
▶ `KernelEmbedding( C )` A
▶ `Boundary( C )` A
▶ `Name( C )` A
▶ `Size( C )` A

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 $\mathcal{C} = (e; t, h : G \to R)$ 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:

$$\begin{array}{lcl} \textbf{Cat 1} & : & te \;=\; he \;=\; \mathrm{id}_R \;, \\ \textbf{Cat 2} & : & [\ker t, \ker h] \;=\; \{1_G\} \;. \end{array}$$

It follows that $teh = h$, $het = t$, $tet = t$, $heh = h$.

The maps t, h are often referred to as the **source** and **target**, but we choose to call them the **tail** and **head** of $\mathcal{C}$, 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.

Here are some constructors for pre-cat1-groups and cat1-groups:

2 ▶ `Cat1( args )` F
▶ `PreCat1ByTailHeadEmbedding( t, h, e )` O
▶ `PreCat1ByEndomorphisms( t, h )` O
▶ `PreCat1ByNormalSubgroup( G, N )` O
▶ `PreCat1OfPreXMod( P )` A

The following listing shows an example of a cat1-group of pc-groups:

```
gap> s3 := SymmetricGroup(IsPcGroup,3);;   SetName(s3,"s3");
gap> gens3 := GeneratorsOfGroup(s3);
[ f1, f2 ]
gap> c4 := CyclicGroup(4);;  SetName( c4, "c4" );
gap> s3c4 := DirectProduct( s3, c4 );;  SetName( s3c4, "s3c4" );
gap> gens3c4 := GeneratorsOfGroup( s3c4 );
[ f1, f2, f3, f4 ]
gap> a := gens3[1];;  b := gens3[2];;  one := One(s3);;
gap> t2 := GroupHomomorphismByImages( s3c4, s3, gens3c4, [a,b,one,one] );
[ f1, f2, f3, f4 ] -> [ f1, f2, <identity> of ..., <identity> of ... ]
```

```
gap> e2 := Embedding( s3c4, 1 );
Pcgs([ f1, f2 ]) -> [ f1, f2 ]
gap> C2 := Cat1( t2, t2, e2 );
[s3c4=>s3]
gap> Display( C2 );
Cat1-group [s3c4=>s3] :-
: source group has generators:
  [ f1, f2, f3, f4 ]
:  range group has generators:
  [ f1, f2 ]
: tail homomorphism maps source generators to:
  [ f1, f2, <identity> of ..., <identity> of ... ]
: head homomorphism maps source generators to:
  [ f1, f2, <identity> of ..., <identity> of ... ]
: range embedding maps range generators to:
  [ f1, f2 ]
: kernel has generators:
  [ f3, f4 ]
: boundary homomorphism maps generators of kernel to:
  [ <identity> of ..., <identity> of ... ]
: kernel embedding maps generators of kernel to:
  [ f3, f4 ]
gap> IsPcCat1( C2 );
true
gap> Size( C2 );
[ 24, 6 ]
```

3 ▶ `Cat1OfXMod( X0 )` A

▶ `XModOfCat1( C )` A

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 $G = R \ltimes S$ using the action from $\mathcal{X}$, with multiplication rule

$$(r_1, s_1)\,(r_2, s_2) \;=\; (r_1\, r_2,\; s_1^{r_2}\, s_2)\ .$$

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 remains unchanged, and $\partial = h\,|_S$. The action of R on S is conjugation in S via the embedding of R in G.

```
gap> X2 := XModOfCat1( C2 );
[Group( [ f3, f4 ] )->s3]
gap> Display( X2 );
Crossed module [..->s3] :-
: Source group has generators:
  [ f3, f4 ]
: Range group s3 has generators:
  [ f1, f2 ]
: Boundary homomorphism maps source generators to:
  [ <identity> of ..., <identity> of ... ]
  The automorphism group is trivial
```

3 2d-mappings

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

1 ▶ `Source( map )` A
▶ `Range( map )` A
▶ `SourceHom( map )` A
▶ `RangeHom( map )` A

Morphisms of `2dObjects` are implemented as `2dMappings`. These have a pair of 2d-objects as source and range, together with two group homomorphisms mapping between corresponding source and range groups.

3.1 Morphisms of pre-crossed modules

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 (σ, ρ), 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:

$$\partial_2\sigma = \rho\partial_1, \qquad \sigma(s^r) = (\sigma s)^{\rho r}.$$

Thus σ is the `SourceHom` and ρ is the `RangeHom`.

When $\mathcal{X}_1 = \mathcal{X}_2$ and σ, ρ are automorphisms then (σ, ρ) is an automorphism of $\mathcal{X}_1$. The group of automorphisms is denoted by $\mathrm{Aut}(\mathcal{X}_1)$.

The usual properties of mappings may be checked:

1 ▶ `IsInjective( map )` P
▶ `IsSurjective( map )` P
▶ `IsSingleValued( map )` P
▶ `IsTotal( map )` P
▶ `IsBijective( map )` P
▶ `IsEndomorphism( map )` P
▶ `IsAutomorphism( map )` P

Constructors for morphisms of pre-crossed modules include:

2 ▶ `PreXModMorphism( args )` F
▶ `XModMorphism( args )` F
▶ `PreXModMorphismByHoms( P1, P2, sigma, rho )` O
▶ `XModMorphismByHoms( X1, X2, sigma, rho )` O
▶ `InclusionMorphism( X1, S1 )` O
▶ `InnerAutomorphism( X1, r )` O
▶ `IdentityMapping( X1 )` A
▶ `IsomorphismPermGroup( obj )` A

In the following example we construct a simple automorphism of the crossed module $\mathcal{X}_1$ constructed in the previous chapter.

```
gap> sigma1 := GroupHomomorphismByImages( c5, c5, [ (1,2,3,4,5) ]
        [ (1,5,4,3,2) ] );;
gap> rho1 := IdentityMapping( Range( X1 ) );
IdentityMapping( PermAut(c5) )
gap> mor1 := XModMorphism( X1, X1, sigma1, rho1 );
[[c5->PermAut(c5)] => [c5->PermAut(c5)]]
gap> Display( mor1 );
Morphism of crossed modules :-
: Source = [c5->PermAut(c5)] with generating sets:
  [ (1,2,3,4,5) ]
  [ (1,2,4,3) ]
: Range = Source
: Source Homomorphism maps source generators to:
  [ (1,5,4,3,2) ]
: Range Homomorphism maps range generators to:
  [ (1,2,4,3) ]
gap> IsAutomorphism( mor1 );
true
```

3.2 Morphisms of pre-cat1-groups

A morphism of pre-cat1-groups from $\mathcal{C}_1 = (e_1; t_1, h_1 : G_1 \to R_1)$ to $\mathcal{C}_2 = (e_2; t_2, h_2 : G_2 \to R_2)$ is a pair (γ, ρ) where $\gamma : G_1 \to G_2$ and $\rho : R_1 \to R_2$ are homomorphisms satisfying

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

1 ▶ `PreCat1Morphism(` *args* `)` F
▶ `Cat1Morphism(` *args* `)` F
▶ `PreCat1MorphismByHoms(` *P1, P2, gamma, rho* `)` O
▶ `Cat1MorphismByHoms(` *C1, C2, gamma, rho* `)` O
▶ `InclusionMorphism(` *C1, S1* `)` O
▶ `InnerAutomorphism(` *C1, r* `)` O
▶ `IdentityMapping(` *C1* `)` A
▶ `IsomorphismPermGroup(` *obj* `)` A

The function `IsomorphismPermGroup` constructs a morphism whose `SourceHom` and `RangeHom` are the `IsomorphismPermGroup`s of the source and range.

```
gap> iso := IsomorphismPermGroup( C2 );
[[..] => [..]]
gap> Display( iso );
Morphism of cat1-groups :-
: Source = [s3c4=>s3] with generating sets:
  [ f1, f2, f3, f4 ]
  [ f1, f2 ]
:  Range = [Group(
[ ( 1,13)( 2,14)( 3,15)( 4,16)( 5,21)( 6,22)( 7,23)( 8,24)( 9,17)(10,18)
    (11,19)(12,20), ( 1, 5, 9)( 2, 6,10)( 3, 7,11)( 4, 8,12)(13,17,21)
    (14,18,22)(15,19,23)(16,20,24), ( 1, 3, 2, 4)( 5, 7, 6, 8)( 9,11,10,12)
    (13,15,14,16)(17,19,18,20)(21,23,22,24),
  ( 1, 2)( 3, 4)( 5, 6)( 7, 8)( 9,10)(11,12)(13,14)(15,16)(17,18)(19,20)
    (21,22)(23,24) ] )=>Group( [ (1,4)(2,6)(3,5), (1,2,3)(4,5,6)
 ] )] with generating sets:
```

```
  [ ( 1,13)( 2,14)( 3,15)( 4,16)( 5,21)( 6,22)( 7,23)( 8,24)( 9,17)(10,18)
    (11,19)(12,20), ( 1, 5, 9)( 2, 6,10)( 3, 7,11)( 4, 8,12)(13,17,21)
    (14,18,22)(15,19,23)(16,20,24), ( 1, 3, 2, 4)( 5, 7, 6, 8)( 9,11,10,12)
    (13,15,14,16)(17,19,18,20)(21,23,22,24),
  ( 1, 2)( 3, 4)( 5, 6)( 7, 8)( 9,10)(11,12)(13,14)(15,16)(17,18)(19,20)
    (21,22)(23,24) ]
  [ (1,4)(2,6)(3,5), (1,2,3)(4,5,6) ]
: Source Homomorphism maps source generators to:
  [ ( 1,13)( 2,14)( 3,15)( 4,16)( 5,21)( 6,22)( 7,23)( 8,24)( 9,17)(10,18)
    (11,19)(12,20), ( 1, 5, 9)( 2, 6,10)( 3, 7,11)( 4, 8,12)(13,17,21)
    (14,18,22)(15,19,23)(16,20,24), ( 1, 3, 2, 4)( 5, 7, 6, 8)( 9,11,10,12)
    (13,15,14,16)(17,19,18,20)(21,23,22,24),
  ( 1, 2)( 3, 4)( 5, 6)( 7, 8)( 9,10)(11,12)(13,14)(15,16)(17,18)(19,20)
    (21,22)(23,24) ]
: Range Homomorphism maps range generators to:
  [ (1,4)(2,6)(3,5), (1,2,3)(4,5,6) ]
```

3.3 Operations on morphisms

1 ▶ `CompositionMapping(` *map2, map1* `)` O
▶ `Order(` *auto* `)` A
▶ `Kernel(` *map* `)` O
▶ `Kernel2dMapping(` *map* `)` A

Composition of morphisms, written (*map1* `*` *map2*) for maps acting of the right, calls the `Composition-Mapping` function for maps acting on the left, applied to the appropriate type of 2d-mapping.

```
gap> mor1;
[[c5->PermAut(c5)] => [c5->PermAut(c5)]]
gap> Order(mor1);
2
gap> mor23;
[[nq8->sl23] => [k4->a4]]
gap> k23 := Kernel( mor23 );
[Group( [ ( 1, 6)( 2, 3)( 4, 5)( 7, 8) ] )->Group(
[ ( 1, 6)( 2, 3)( 4, 5)( 7, 8) ] )]
gap> Display(k23);
Crossed module [..->..] :-
: Source group has generators:
  [ ( 1, 6)( 2, 3)( 4, 5)( 7, 8) ]
: Range group has generators:
  [ ( 1, 6)( 2, 3)( 4, 5)( 7, 8) ]
: Boundary homomorphism maps source generators to:
  [ (1,6)(2,3)(4,5)(7,8) ]
  The automorphism group is trivial
gap> IsNormal( Source(mor23), k23 );
true
```

4 Derivations and Sections

4.1 Whitehead Multiplication

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

$$\begin{array}{lll} \textbf{Der 1} & : & \chi(qr) \;=\; (\chi q)^r\,(\chi r)\;, \\ \textbf{Der 2} & : & (\chi_1 \circ \chi_2)(r) \;=\; (\chi_2 r)(\chi_1 r)(\chi_2 \partial \chi_1 r)\;. \end{array}$$

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

The construction for cat1-groups equivalent to the derivation of a crossed module is the **section**. The monoid of sections of $\mathcal{C} = (e; t, h : G \to R)$ is the set of group homomorphisms $\xi : R \to G$, with Whitehead multiplication $\circ$, (on the **right**) satisfying:

$$\begin{array}{lll} \textbf{Sect 1} & : & t\xi \;=\; \mathrm{id}_R\;, \\ \textbf{Sect 2} & : & (\xi_1 \circ \xi_2)(r) \;=\; (\xi_1 r)(eh\xi_1 r)^{-1}(\xi_2 h\xi_1 r) \;=\; (\xi_2 h\xi_1 r)(eh\xi_1 r)^{-1}(\xi_1 r)\;. \end{array}$$

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

1 ▶ `Object2d(` *chi* `)` A
▶ `GeneratorImages(` *chi* `)` A
▶ `IsDerivation(` *chi* `)` P
▶ `DerivationByImages(` *X0*`,` *ims* `)` O

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 crossed module $\mathcal{X}_3$ is constructed, isomorphic to the inclusion of the normal cyclic group *c3* in the symmetric group *s3*.

```
gap> g18 := Group( (1,2,3), (4,5,6), (2,3)(5,6) );;
gap> SetName( g18, "g18" );
gap> gen18 := GeneratorsOfGroup( g18 );;
gap> a := gen18[1];;  b := gen18[2];;  c := gen18[3];;
gap> s3 := Subgroup( g18, gen18{[2..3]} );;
gap> SetName( s3, "s3" );;
gap> t := GroupHomomorphismByImages( g18, s3, gen18, [ b,b,c] );;
gap> h := GroupHomomorphismByImages( g18, s3, gen18, [(),b,c] );;
gap> e := GroupHomomorphismByImages( s3, g18, [b,c], [b,c] );;
gap> C3 := Cat1( t, h, e );
[g18=>s3]
```

```
gap> X3 := XModOfCat1( C3 );;
gap> Display( X3 );
Crossed module [..->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) ] }
  These 2 automorphisms generate the group of automorphisms.
: associated cat1-group is [g18=>s3]

gap> imchi := [ (1,2,3)(4,6,5), (1,2,3)(4,6,5) ];;
gap> chi := DerivationByImages( X3, imchi );
DerivationByImages( s3, Group( [ ( 1, 2, 3)( 4, 6, 5) ] ),
[ (4,5,6), (2,3)(5,6) ], [ (1,2,3)(4,6,5), (1,2,3)(4,6,5) ] )
```

2 ▶ `SectionByImages(` *C, ims* `)` O
▶ `SectionByDerivation(` *chi* `)` O
▶ `DerivationBySection(` *xi* `)` O

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 [AW96]. The default method for `AllDerivations` is to search for all possible sets of images using a backtracking procedure, and when all the derivations are found it is not known which are regular. In the GAP 3 version 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 this version the derivations of the associated crossed module are calculated, and these are all converted to sections using `SectionByDerivation`.

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

If ϵ denotes the inclusion of $S = \ker t$ in G then $\partial = h\epsilon : S \to R$ and

$$\xi r \quad = \quad (er)(e\chi r) \quad = \quad (r, \chi r)$$

determines a section ξ of $\mathcal{C}$ in terms of the corresponding derivation χ of $\mathcal{X}$, and conversely.

4.2 Whitehead Groups and Monoids

1 ▶ `RegularDerivations(` *X0* `)` A
▶ `AllDerivations(` *X0* `)` A
▶ `RegularSections(` *C0* `)` A
▶ `AllSections(` *C0* `)` A
▶ `ImagesList(` *obj* `)` A
▶ `ImagesTable(` *obj* `)` A

There are two functions to determine the elements of the Whitehead group and the Whitehead monoid of $\mathcal{X}_0$, namely `RegularDerivations` and `AllDerivations`. (The functions `RegularSections` and `AllSections` perform corresponding tasks for a cat1-group.)

Using our example $\mathcal{X}_3$ we find that there are just nine derivations, six of them regular, and the associated group is isomorphic to *s3*.

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

2 ► `CompositeDerivation(` *chi1*`,` *chi2* `)` O
► `ImagePositions(` *chi* `)` A

The Whitehead multiplication is implemented by the function `CompositeDerivation(` *chi1*`,` *chi2* `);` which has the same result as *chi2* `*` *chi1*, where composition is on the left and multiplication on the right.

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

3 ▶ `WhiteheadGroupTable( X0 )` A
▶ `WhiteheadMonoidTable( X0 )` A
▶ `WhiteheadPermGroup( X0 )` A
▶ `WhiteheadTransMonoid( X0 )` A

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

```
gap> wgt3 := WhiteheadGroupTable( X3 );; Display( 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), (1,3,2)(4,6,5), (1,4)(2,6)(3,5), (1,5)(2,4)(3,6),
  (1,6)(2,5)(3,4) ])
gap> wtm3 := WhiteheadTransMonoid( X3 );
<monoid with 9 generators>
```

Bibliography

[A97] M. Alp, *GAP, crossed modules, cat1-groups: applications of computational group theory,* PhD Thesis, University of Wales, Bangor, 1997.

[AW96] M. Alp and C. D. Wensley, Enumeration of cat1-groups of low order, *Int. J. Algebra and Computation*, 10:407–424, 2000.

[B82] R. Brown, Higher-dimensional group theory, in Low-dimensional topology, London Math. Soc. Lecture Note Series 48, ed. R. Brown and T. L. Thickstun, Cambridge University Press, 1982, pp. 215-238.

[BH78] R. Brown and P. J. Higgins, On the connection between the second relative homotopy group and some related spaces, *Proc. London Math. Soc.* 36:193–212, 1978.

[BW95] R. Brown and C. D. Wensley, On finite induced crossed modules, and the homotopy 2-type of mapping cones, *Theory and Applications of Categories* 1:54–71, 1995.

[BW96] R. Brown and C. D. Wensley, On the computation of induced crossed modules, *Theory and Applications of Categories* 2:3–16, 1996.

[E84] G. Ellis, *Crossed modules and their higher dimensional analogues*, PhD thesis, University of Wales, Bangor, 1984.

[G90] N. D. Gilbert, Derivations, automorphisms and crossed modules, *Comm. in Algebra* 18:2703–2734, 1990.

[Lod82] J. L. Loday. Spaces with finitely many non-trivial homotopy groups, *J. App. Algebra*, 24:179–202, 1982.

[M01] E. J. Moore, *Graphs of Groups: Word Computations and Free Crossed Resolutions,* PhD Thesis, University of Wales, Bangor, 2001.

[Nor87] K. J. Norrie. *Crossed modules and analogues of group theorems*, PhD thesis, King's College, University of London, 1987.

[Nor90] K. J. Norrie. Actions and automorphisms of crossed modules, *Bull. Soc. Math. France*, 118:129–146, 1990.

[Whi48] J. H. C. Whitehead. On operators in relative homotopy groups, *Ann. of Math.*, 49:610–640, 1948.

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

Index

This index covers only this manual. A page number in *italics* refers to a whole section which is devoted to the indexed subject. Keywords are sorted with case and spaces ignored, e.g., "`PermutationCharacter`" comes before "permutation group".