LOOPS

Version 0.9992
March 20, 2005

Computing with quasigroups and loops in GAP

Gábor P. Nagy

Department of Mathematics
University of Szeged
e-mail: nagyg@math.u-szeged.hu

Petr Vojtěchovský

Department of Mathematics
University of Denver
e-mail: petr@math.du.edu

Contents

Chapter 1

Introduction

LOOPS is a package for GAP4 whose purpose is to:

(a) provide researchers in nonassociative algebra with a powerful computational tool concerning finite loops and quasigroups,

(b) extend GAP toward the realm of nonassociative structures.

1.1 Installation

We assume that you have GAP v. 4.4 or newer installed on your computer. Download the LOOPS package from the distribution website

`http://www.math.du.edu/loops`

Unpack the downloaded file into the `pkg` subfolder of your GAP folder. After this step, there should be a subfolder `loops` in your `pkg` folder. The package LOOPS can then be loaded to GAP anytime by calling `LoadPackage("loops")`.

If you wish to load LOOPS automatically while starting GAP, proceed as follows:

(i) open the file `PackageInfo.g` in the `loops` folder,

(ii) edit the value of `Autoload` from `Autoload:=false` to `Autoload:=true`. (The parameter `Autoload` is located near the end of the file `PackageInfo.g`).

1.1.1 Brief description of LOOPS files

Table 1.1 summarizes all relevant files forming the LOOPS package. Some technical files (e.g., pictures used in the documentation) are not mentioned. You probably don't need any of this information unless you want to modify LOOPS.

1.2 Documentation

The documentation is available is several formats: LaTeX, pdf, dvi, postscript, html, and as an online help in GAP. All these formats have been obtained directly from the master LaTeX documentation file. Consequently, the different formats differ only in their appearance, not in contents.

The documentation can be found in the `doc` folder of the LOOPS package and also at the LOOPS distribution website.

The online GAP help is available upon installing LOOPS, and can be accessed in the usual way, i.e., upon typing `?command`, GAP displays the section of the LOOPS manual containing information about `command`.

Table 1.1: Brief description of files forming LOOPS.

folder	file	description
pkg	README.loops	installation and usage instructions
pkg/loops	init.g	declaration of LOOPS methods
	PackageInfo.g	loading info for GAP 4.4
	read.g	implementation of LOOPS methods
pkg/loops/data	interesting.tbl	library of interesting loops
	leftbol.tbl	library of left Bol loops
	moufang.tbl	library of Moufang loops
	moufang_discriminators.tbl	data for isomorphisms of Moufang loops
	paige.tbl	library of Paige loops
	steiner.tbl	library of Steiner loops
pkg/loops/doc	loops_manual.dvi	dvi version of the documentation
	loops_manual.html	web version of the documentation
	loops_manual.pdf	pdf version of the documentation
	loops_manual.ps	postscript version of the documentation
	loops_manual.tex	LaTeXsource for the documentation
	manual.six	contextual help for GAP
pkg/loops/etc		utilities for archive and doc generation
pkg/loops/gap	banner.g	banner of LOOPS
	examples.gd	methods for loop libraries
	loop_iso.gd	methods for isomorphisms of loops
	moufang_modifications.gd	methods for Moufang modifications
	quasigrp.gd	core methods of LOOPS
	triality.gd	methods for triality of Moufang loops
	examples.gi	methods for loop libraries
	loop_iso.gi	methods for isomorphisms of loops
	moufang_modifications.gi	methods for Moufang modifications
	quasigrp.gi	core methods of LOOPS
	triality.gi	methods for triality of Moufang loops
pkg/loops/tst	auto.tst	test automorphism groups
	lib.tst	test all libraries except Moufang
	mouflib.tst	test library of Moufang loops
	nilpot.tst	test nilpotency
	quasigrp.tst	test core functions
	testall.g	batch file for all tests

1.3 Test files

Test files conforming to the GAP standards are provided for LOOPS. They can be found in the folder `loops/tst` and run in the usual way.

The file `testall.tst` runs all tests for LOOPS with the exception of the test `mouflib.tst`. The test `mouflib.tst` builds all Moufang loops contained in the libraries of LOOPS, and runs for about 20 minutes.

1.4 Mathematical background

We assume that you are familiar with the theory of quasigroups and loops, for instance with the textbook of Bruck [1] or Pflugfelder [9]. Nevertheless, we did include definitions and results in this manual in order to unify the terminology and improve the intelligibility of the text.

1.4.1 Quasigroups and loops

A set with one binary operation (denoted $\cdot$ here) is called *groupoid* or *magma*, the latter name being used in GAP. Associative groupoids are known as *semigroups*.

An element 1 of a groupoid G is a *neutral element* or an *identity element* if $1 \cdot x = x \cdot 1 = x$ for every x in G. Semigroup with a neutral element is a *monoid*.

Let G be a groupoid with neutral element 1. Then an element y is called a *two-sided inverse* of x in G if $x \cdot y = y \cdot x = 1$. A monoid in which every element has a two-sided inverse is called a *group*.

Groups can be reached in another way from groupoids, namely through quasigroups and loops.

A *quasigroup* Q is a groupoid such that the equation $x \cdot y = z$ has a unique solution in Q whenever two of the three elements x, y, z of Q are specified. Note that multiplication tables of finite quasigroups are precisely *Latin squares*, i.e., a square arrays with symbols arranged so that each symbol occurs in each row and in each column exactly once. A *loop* L is a quasigroup with a neutral element.

Groups are clearly loops, and one can show easily that an associative quasigroup is a group. Hence the theory of quasigroups and loops is in a sense complementary to the theory of semigroups and monoids.

1.4.2 Translations

Given an element x of a quasigroup Q we can associative two permutations of Q with it: the *left translation* $L_x : Q \to Q$ defined by $y \mapsto x \cdot y$, and the *right translation* $R_x : Q \to Q$ defined by $y \mapsto y \cdot x$.

Although it is possible to compose two right (left) translations, the resulting permutation is not necessarily a right (left) translation. The set $\mathcal{L}(Q) = \{L_x; x \in Q\}$ is called the *left section* of Q, and, similarly, $\mathcal{R}(Q) = \{R_x; x \in Q\}$ is the *right section* of Q.

Let S_Q be the symmetric group on Q. Then $\mathrm{LMlt}(Q)$, the subgroup of S_Q generated by $\mathcal{L}(Q)$, is called the *left multiplication group* of Q. Similarly, $\mathrm{RMlt}(Q) = \langle \mathcal{R}(Q) \rangle$ is the *right multiplication group* of Q. The smallest group containing both $\mathrm{LMlt}(Q)$ and $\mathrm{RMlt}(Q)$ is called the *multiplication group* of Q and is denoted by $\mathrm{Mlt}(Q)$.

1.4.3 Homomorphisms and homotopisms

Let K, H be two quasigroups. Then a map $f : K \to H$ is a *homomorphism* if $f(x) \cdot f(y) = f(x \cdot y)$ for every x, $y \in K$. If f is also a bijection, we speak of an *isomorphism*, and the two quasigroups are called *isomorphic*.

The ordered triple (α, β, γ) of maps α, β, $\gamma : K \to H$ is a *homotopism* if $\alpha(x) \cdot \beta(y) = \gamma(x \cdot y)$ for every x, $y \in K$. If the three maps are bijections, (α, β, γ) is a *isotopism*, and the two quasigroups are *isotopic*.

Isotopic groups are necessarily isomorphic, but this is certainly not true for nonassociative quasigroups or loops. In fact, every quasigroup is isotopic to a loop, as we shall see.

Let $(K, \cdot)$, $(K, \circ)$ be two quasigroups defined on the same set K. Then an isotopism $(\alpha, \beta, \mathrm{id_K})$ is called a *principal isotopism*. An important class of principal isotopisms is obtained as follows:

Let $(K, \cdot)$ be a quasigroup, and let f, g be elements of K. Define a new operation $\circ$ on K by

$$x \circ y = R_g^{-1}(x) \cdot L_f^{-1}(y),$$

where R_g, L_f are translations. Then $(K, \circ)$ is a quasigroup isotopic to $(K, \cdot)$, in fact a loop with neutral element $f \cdot g$. We call $(K, \circ)$ a *principal loop isotope* of $(K, \cdot)$.

1.5 How the package works

The package consists of three complementary components: the core algorithms for quasigroup theoretical notions (see Chapter 2), some specific algorithms, mostly for Moufang loops (see Chapter 3), and the library of small loops (see Chapter 4).

Although we do not explain the algorithms in detail here, we describe the overarching ideas so that the user should be able to anticipate the capabilities and behavior of the computation.

1.5.1 Representing quasigroups in LOOPS

Since the permutation representation in the usual sense is impossible for nonassociative structures, and since the theory of nonassociative presentations is not well understood, we had to resort to multiplication tables to represent quasigroups in GAP. In order to save storage space, we sometimes use one multiplication table to represent several quasigroups (for instance when a quasigroup is a subquasigroup of another quasigroup).

Consequently, *the package is intended primarily for quasigroups and loops of small order* (up to 1000, say).

1.5.2 Calculating with quasigroups in LOOPS

Although the quasigroups are ultimately represented by multiplication tables, the algorithms are efficient because nearly all calculations are delegated to groups. The connection between quasigroup and groups is facilitated via the above-mentioned translations, and we illustrate it with a few examples:

1) This example shows how properties of quasigroups can be translated into properties of translations in a straightforward way.

   Let Q be a quasigroup. We ask if Q is associative. We can either test if $(xy)z = x(yz)$ for every x, y, $z \in Q$, or we can ask if $L_{xy} = L_x L_y$ for every x, $y \in Q$. Note that since L_{xy}, L_x, L_y are elements of a permutation group, we do not have to refer directly to the multiplication table once the left translations of Q are known.

2) This example shows how properties of loops can be translated into properties of translations in a way that requires some theory.

   A left Bol loop is a loop satisfying $x(y(xz)) = (x(yx))z$. We claim (without proof) that a loop L is left Bol if and only if $L_x L_y L_x$ is a left translation for every x, $y \in L$.

3) This example shows that many properties of loops become purely group-theoretical once they are expresses in terms of translations.

   A loop is simple if it has no nontrivial congruences. Then it is easy to see that a loop is simple if and only if its multiplication group $\mathrm{Mlt}(L)$ is a primitive permutation group.

The main idea of the package is therefore: (i) calculate the translations and the associated permutation groups when they are needed, (ii) store them as attributes, (iii) use them in algorithms as often as possible.

1.5.3 Magmas, quasigroups, loops and groups in GAP

Whether an object is considered a quasigroup or a loop is a matter of declaration in LOOPS. A declared loop is considered to be a quasigroup, however, a declared quasigroup is *not* considered to be a loop, even if it accidentally possesses a neutral element. It is possible to convert a quasigroup Q (with or without a neutral element) to a loop using

- `AsLoop( Q )  F`

The category of quasigroups (cf. `IsQuasigroup`) is declared in `LOOPS` so that it is contained in the category of magmas (cf. `IsMagma`). All standard `GAP` command for magmas are therefore available for quasigroups and loops, too.

Although groups are quasigroups mathematically, they are not treated as quasigroups in `LOOPS`. If you wish to apply methods of `LOOPS` to groups, apply one of the conversions

- `AsQuasigroup( G )  F`

- `AsLoop( G )  F`

to the group G. These conversions fail when G is infinite and will exhaust all available memory when G is huge. For more information on conversions, see subsection 2.2.4.

1.6 Feedback

We welcome all comments and suggestions on `LOOPS`, especially those concerning the future development of the package. You can contact us by e-mail.

Chapter 2

Core methods

2.1 Naming, viewing and printing objects in LOOPS

GAP displays information about objects in two modes: `View` (default, short) and `Print` (longer). Moreover, when the name of an object is set, it is always shown, no matter which display mode is used.

2.1.1 Named quasigroups and loops

Only loops contained in the libraries of LOOPS are named. For instance, the loop obtained via `MoufangLoop( 32, 4 )`—the 4th Moufang loop of order 32—is named `<Moufang loop 32/4>`.

2.1.2 View mode

When Q is a quasigroup of order n, it is displayed as `<quasigroup of order n>` in LOOPS. Similarly, a loop of order n appears as `<loop of order n>`.

The displayed information for a loop L is enhanced when it is known that L has certain additional properties. At this point, we support:

`<associative loop ...>`,
`<extra loop ...>`,
`<Moufang loop ...>`,
`<C loop ...>`,
`<left Bol loop ...>`,
`<right Bol loop ...>`,
`<LC loop ...>`,
`<RC loop ...>`,
`<alternative loop ...>`,
`<left alternative loop ...>`,
`<right alternative loop ...>`,
`<flexible loop ...>`.

The corresponding mathematical definitions and an example can be found in subsection 2.10.4.

It is possible for a loop to have several of the above properties. In such a case, we display the first property on the list that is satisfied. For instance, a left alternative flexible loop will appear as `<left alternative loop ...>`.

By default, the mth element of a quasigroup appears as `qm`, the mth element of a loop appears as `lm`, and the neutral element of a loop is denoted by `l1`. However, it is possible to change the names of elements of a quasigroup Q or loop L to *name*`m` with

- `SetQuasigroupElmName( Q, name )` F

- `SetLoopElmName( L, name )` F

Also see the example in Section 2.8.

2.1.3 Print mode

Elements of quasigroups and loops appear in the same way in both `View` and `Print` modes.

For quasigroups and loops in the `Print` mode, we display the multiplication table (if it is known), or we display the elements. See Section 2.4 for another way in which multiplication tables can be displayed.

In the following example, L is a loop with two elements.

```
gap> L;
<loop of order 2>
gap> Print( L );
<loop with multiplication table
[ [  1,  2 ],
  [  2,  1 ] ]
>
gap> Elements( L );
[ l1, l2 ]
gap> SetLoopElmName( L, "loop_element" );; Elements( L );
[ loop_element1, loop_element2 ]
```

2.2 Creating quasigroups and loops

As mentioned above, quasigroups and loops are represented by multiplication tables, which we also refer to as *Cayley tables*. When Q is a quasigroup of order n, the associated Cayley table is an $n \times n$ array with symbols 1, ..., n such that $\mathtt{qi} \cdot \mathtt{qj} = \mathtt{qk}$ if and only if the entry in row i and column j is k. Similarly for loops.

The Cayley table can be entered manually, or read off from a file.

2.2.1 Testing multiplication tables

The following synonymous operations test if a multiplication table is a multiplication table of a quasigroup, i.e. a Latin squares with symbols 1, ..., n.

- `IsQuasigroupTable( L )` A

- `IsQuasigroupCayleyTable( L )` A

A Latin square on 1, ..., n is said to be *normalized* if the first column and the first row read 1, ..., n. Cayley table of a loop is therefore just another name for a normalized Latin square. The following operations test if L is a normalized Latin square:

- `IsLoopTable( L )` A

- `IsLoopCayleyTable( L )` A

A Latin square can be normalized by permuting its columns so that the first row reads 1, …, n, and then permuting its rows so that the first column reads 1, …, n. Note that it matters whether rows or columns are permuted first (see subsection 2.2.4 for more). This normalization is achieved in LOOPS with

- `NormalizedQuasigroupTable( L )` F

We would like to call the attention to the fact that the package GUAVA also has some operations dealing with Latin squares (in particular, the function `IsLatinSquare` is defined in GUAVA).

2.2.2 Creating quasigroups and loops manually

When L is a Latin square on 1, …, n, the corresponding quasigroup is obtained with

- `QuasigroupByCayleyTable( L )` F

- `QuasigroupByCayleyTable( L, <name> )` F

If `<name>` is given, then the output of the quasigroup elements will have the form `<name>1`, `<name>2`, ...

When L is normalized, the corresponding loop is returned by

- `LoopByCayleyTable( L )` F

2.2.3 Creating quasigroups and loops from a file

Typing a large multiplication table manually is tedious and error-prone. We have therefore included a universal algorithm in LOOPS that reads multiplication tables of quasigroups from a file. Instead of writing a separate algorithm for each common format, our algorithm relies on the user to provide a bit of information about the input file. Here is an outline of the algorithm:

```
Input: filename F, string D
Step 1: Read the entire content of F into a string S.
Step 2: Replace all end-of-line characters in S by spaces.
Step 3: Replace by spaces all characters of S that appear in D.
Step 4: Split S into maximal substrings without spaces, called chunks.
Step 5: Recognize distinct chunks. Let $n$ be the number of distinct chunks.
Step 6: If the number of chunks is not $n^2$, report error.
Step 7: Construct the multiplication table by assigning numerical values 1, …,
$n$ to chunks, depending on their position among distinct chunks.
```

The following examples clarify the algorithm and document its versatility:

input file	string D	resulting mult. table	comments
0 1 2 1 2 0 2 0 1	””	1 2 3 2 3 1 3 1 2	Data does not have to be arranged into an array of any kind.
red green green red	””	1 2 2 1	Chunks can be any strings.
[[0, 1], [1, 0]]	”[,]”	1 2 2 1	A typical table produced by `GAP` is easily parsed by deleting brackets and commas.
x&y&z\hline y&z&x\hline z&x&y	”&\\einlh”	1 2 3 2 3 1 3 1 2	A typical TeX table with rows separated by lines. We must use ”\\” in D because ”\\” represents the string ”\” in `GAP`.
I am as mad as I say I am	”day”	1 2 3 2 3 1 3 1 2	Just for fun.

And here are the needed `LOOPS` commands:

- `QuasigroupFromFile( F, D )` F

- `LoopFromFile( F, D )` F

2.2.4 Conversions

We provide conversion operations that convert between magmas, quasigroups, loops and groups, provided such conversions are possible.

If M is a declared magma that happens to be a quasigroup, the corresponding quasigroup is returned via

- `AsQuasigroup( M )` F

If M is a magma that happens to be a quasigroup, the operation

- `AsLoop( M )` F

returns a loop L as follows:

If M possesses a neutral element e and f is the first element of M, then L is an isomorphic copy of M via the transposition (e, f).

If M does not posses a neutral element, then L is returned as

```
PrincipalLoopIsotope( M, M.1, M.1 ),
```

where

- `PrincipalLoopIsotope( Q, f, g )` F

is the principal isotope of Q using elements f, g of Q, as explained in Subsection 1.4.3.

Of course, one can obtain a loop from M in different ways, for instance by normalizing the Cayley table of M. The following example shows that these three approaches can yield different results in general:

```
gap> A := [[2,1,5,3,4],[1,2,3,4,5],[5,3,4,1,2],[3,4,2,5,1],[4,5,1,2,3]];;
gap> Q := QuasigroupByCayleyTable( A );;
gap> L := AsLoop( Q );;
gap> C := LoopByCayleyTable( NormalizedQuasigroupTable( CayleyTable(L) ) );;
gap> P := PrincipalLoopIsotope( Q, Elements( Q )[ 2 ], Elements( Q )[ 2 ] );
gap> CayleyTable( L );
[ [ 1, 2, 3, 4, 5 ], [ 2, 1, 5, 3, 4 ], [ 3, 5, 4, 2, 1 ],
  [ 4, 3, 1, 5, 2 ], [ 5, 4, 2, 1, 3 ] ]
gap> CayleyTable( C );
[ [ 1, 2, 3, 4, 5 ], [ 2, 1, 4, 5, 3 ], [ 3, 5, 1, 2, 4 ],
  [ 4, 3, 5, 1, 2 ], [ 5, 4, 2, 3, 1 ] ]
gap> CayleyTable( P );
[ [ 1, 2, 3, 4, 5 ], [ 2, 1, 4, 5, 3 ], [ 3, 4, 5, 2, 1 ],
  [ 4, 5, 1, 3, 2 ], [ 5, 3, 2, 1, 4 ] ]
```

Finally, when M is a declared magma that happens to be a group, then the corresponding group is returned by

- `AsGroup( M )` F

Note that the conversions work in both directions, not just toward more special structures. Thus, if G is a declared group, then `AsLoop( G )` returns the corresponding loop, for instance.

2.2.5 Products of loops

Let $L1$, ..., Ln be a list consisting of loops and groups, where $n \geq 1$. Then

- `DirectProduct( L1, ..., Ln )` F

returns the direct product of $L1$, ..., Ln.

If there are only groups on the list, a group is returned, otherwise a loop is returned. If $n = 1$, $L1$ is returned.

2.2.6 Opposite quasigroups and loops

When Q is a quasigroup with multiplication $\cdot$, the *opposite quasigroup* of Q is a quasigroup with the same underlying set as Q and with multiplication $*$ defined by $x * y = y \cdot x$.

Since the quasigroup-theoretical concepts are often oriented (cf. left Bol loops versus right Bol loops), it is useful to have access to the opposite quasigroup of Q:

- `Opposite( Q )` F

2.3 GAP categories

One can test if an element x belong to a quasigroup or to a loop, or if a given object Q is a quasigroup or a loop:

- `IsQuasigroupElement( x )` category

- `IsLoopElement( x )` category

- `IsQuasigroup( Q )` category

- `IsLoop( Q )` category

2.4 Basic attributes

The list of elements of a quasigroup Q is obtained by the usual command

- `Elements( Q )`   A

The Cayley table of a quasigroup Q is returned with

- `CayleyTable( Q )`   A

One can use `Display( CayleyTable( Q ) )` for pretty matrix-style output of small Cayley tables.

The neutral element of a loop L is obtained via

- `One( L )`   A

If you want to know if a quasigroup Q has a neutral element, you can find out with the standard function for magmas

- `MultiplicativeNeutralElement( Q )`   A

The size of a quasigroup Q is calculated by

- `Size( Q )`   A

When L is a power associative loop (i.e., the orders of elements are well-defined in L), the *exponent* of L is the smallest positive integer divided by orders of all elements of L. The following attribute calculates the exponent without testing for power associativity:

- `Exponent( L )`   A

```
gap> Q := QuasigroupByCayleyTable( [ [ 1, 2 ], [ 2, 1 ] ] );
<quasigroup of order 2>
gap> [ IsQuasigroup( Q ); IsLoop( Q ); Size( Q ); Elements( Q ); ]
[ true, false, 2, [ q1, q2 ] ]
gap> IsQuasigroupElement( Elements( Q )[ 2 ] );
true
gap> CayleyTable( Q );
[ [ 1, 2 ], [ 2, 1 ] ]
```

2.5 Basic arithmetic operations

Each quasigroup element in GAP knows which quasigroup it belongs to. It is therefore possible to perform arithmetic operations with quasigroup elements without referring to the quasigroup. All elements involved in the calculation must belong to the same quasigroup.

2.5.1 Multiplication

Two elements x, y of the same quasigroup are multiplied by $x*y$ in GAP. Since multiplication of elements is ambiguous in the nonassociative case, we always multiply element from left to right (i.e., $x*y*z$ means $(x*y)*z$)). Of course, one can specify association by parentheses.

2.5.2 Division

Universal algebraists introduce two additional operations for quasigroups. Namely the *left division* $x\backslash y$ satisfying $x\cdot(x\backslash y)=y$, and the *right division* x/y satisfying $x/y\cdot y=x$. These two operations can be found in LOOPS as:

- `LeftDivision( x, y )`   O

- `RightDivision( x, y )` O

When Q is a quasigroup, $x \in Q$ and S is a list of elements of Q, then

- `LeftDivision( S, x )` O
- `LeftDivision( x, S )` O
- `RightDivision( S, x )` O
- `RightDivision( x, S )` O

returns the list of elements obtained by performing the respective division of S by x, or of x by S, using one element of S at a time.

We also support / in place of `RightDivision`. (But not \ in place of `LeftDivision`.)

2.5.3 Powers and inverses

Powers of elements are not well-defined in quasigroups, since bracketing can matter even for a single element. We say that the quasigroup Q is *monoassociative*, if for any $x \in Q$, the submagma generated by x is associative. Similarly, the loop L is said to be *power associative*, if for any $x \in L$, the subloop generated by x is associative.

For magmas and positive integer exponents, `GAP` defines the powers in the following way: $x^1 = x$, $x^{2k} = (x^k) \cdot (x^k)$ and $x^{2k+1} = (x^{2k}) \cdot x$ for positive integer k. One can easily see that this returns x^k in $\log_2(k)$ steps. For `LOOPS`, we decided to keep this method, hoping that everybody will use it with care for quasigroups.

Let x be an element of a loop L with neutral element 1. Then the *left inverse* x' of x is the unique element of L satisfying $x'x = 1$. Similarly, the *right inverse* x'' satisfies $xx'' = 1$. If $x' = x''$, we call $x^{-1} = x' = x''$ the *inverse* of x.

- `LeftInverse( x )` O
- `RightInverse( x )` O
- `Inverse( x )` O

The following examples illustrates the usage of arithmetic operations. `MoufangLoop` will be explained in Chapter 4. In this example, `M.i` coincides with `Elements( M )[ i ]`.

```
gap> M := MoufangLoop( 12, 1 );; x := M.2;
l2
gap> [ x * M.3, x^2, x^(-1), Inverse( x ) ];
[ l4, l1, l2, l2 ]
gap> One( M ) = LeftDivision( x, x );
true
```

2.5.4 Associators and commutators

Let Q be a quasigroup and x, y, $z \in Q$. Then the *associator* of x, y, z is the unique element u such that $(xy)z = (x(yz))u$. The *commutator* of x, y is the unique element v such that $xy = (yx)v$.

- `Associator( x, y, z )` O
- `Commutator( x, y )` O

2.6 Generators

The following two attributes are synonyms of `GeneratorsOfMagma`.

- `GeneratorsOfQuasigroup( Q )` A

- `GeneratorsOfLoop( L )` A

As usual in `GAP`, one can refer to the ith generator of a quasigroup Q by `Q.i`. Note that it is not necessarily the case that `Q.i = Elements( Q )[ i ]`, since the set of generators can be a proper subset of the elements.

It is easy to prove that a loop of order n can be generated by a subset containing at most $\log_2 n$ elements. Such a set is returned via

- `SmallGeneratingSet( L )` A

2.7 Permutations associated with loops

2.7.1 Sections

The following two attributes calculate the left and right section of a quasigroup Q:

- `LeftSection( Q )` A

- `RightSection( Q )` A

Given an element x of a quasigroup Q, the left and right translations of Q by x are obtained by

- `LeftTranslation( Q, x )` F

- `RightTranslation( Q, x )` F

2.7.2 Multiplication groups

The left multiplication group, right multiplication group and the multiplication group of a quasigroup Q is calculated as follows:

- `LeftMultiplicationGroup( Q )` A

- `RightMultiplicationGroup( Q )` A

- `MultiplicationGroup( Q )` A

The relative versions of multiplication groups are implemented only for loops. When S is a subloop of a loop L, the following functions return the relative multiplication groups:

- `RelativeLeftMultiplicationGroup( L, S )` F

- `RelativeRightMultiplicationGroup( L, S )` F

- `RelativeMultiplicationGroup( L, S )` F

2.7.3 Inner mapping group

The *inner mapping group* of a loop L is the stabilizer of the unit element in $\mathrm{Mlt}(L)$. The elements of this stabilizer are called *inner maps* of L. The inner mapping group of a loop L is obtained by:

- `InnerMappingGroup( L )`   A

```
gap> M := MoufangLoop( 12, 1 );
<Moufang loop 12/1>
gap> LeftSection( M )[ 2 ];
(1,2)(3,4)(5,6)(7,8)(9,12)(10,11)
gap> Mlt := MultiplicationGroup( M ); Inn := InnerMappingGroup( M );
<permutation group of size 2592 with 23 generators>
Group([ (4,6)(7,11), (7,11)(8,10), (2,6,4)(7,9,11), (3,5)(9,11), (8,12,10) ])
gap> Size( Inn );
216
```

2.8 Subquasigroups and subloops

Let Q be a quasigroup and S a subquasigroup of Q. Since the multiplication in S coincides with the multiplication in Q, it is reasonable not to store the multiplication table of S. However, the quasigroup S then must know that it is a subquasigroup of Q. In order to facilitate this relationship, we introduce the attribute

- `Parent( Q )`   A

for quasigroups. When Q is *not* created as a subquasigroup of another quasigroup, the attribute `Parent( Q )` is set to Q. When Q is created as a subquasigroup of a quasigroup H, we set `Parent( Q ) := Parent( H )`. Thus, in effect, `Parent( Q )` is the largest quasigroup from which Q was created.

Given a collection C of elements of a quasigroup Q, the function

- `PosInParent( C )`   F

returns the list of positions of the elements of C among the elements of `Parent( Q )`. The multiplication in Q can therefore be easily reconstructed from `Parent( Q )` via `PosInParent`.

When S is a subset of a quasigroup Q (loop L), the subquasigroup of Q (subloop of L) is returned via:

- `Subquasigroup( Q, S )`   F

- `Subloop( L, S )`   F

Finally, the following two functions test if a quasigroup (loop) S is a subquasigroup (subloop) of a quasigroup Q (loop L). Note that the functions return false when S and Q (L) do not have the same parent.

- `IsSubquasigroup( Q, S )`   F

- `IsSubloop( L, S )`   F

The following example illustrates the main features of the subquasigroup construction. Note how the Cayley table of the subquasigroup is created only upon explicit demand. Also note that changing the names of elements of a suquasigroup (subloop) automatically changes the names of the elements of the parent subquasigroup (subloop). This is because the elements are shared.

```
gap> M := MoufangLoop( 12, 1 );; S := Subloop( M, [ M.5 ] );
```

```
<loop of order 3>
gap> [ Parent( S ) = M, Elements( S ), PosInParent( S ) ];
[ true, [ l1, l3, l5], [ 1, 3, 5 ] ]
gap> HasCayleyTable( S );
false
gap> SetLoopElmName( S, "s" );; Elements( S ); Elements( M );
[ s1, s3, s5 ]
[ s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12 ]
gap> CayleyTable( S );
[ [ 1, 2, 3 ], [ 2, 3, 1 ], [ 3, 1, 2 ] ]
gap> [ HasCayleyTable( S ), Parent( S ) = M ];
[ true, true]
gap> L := LoopByCayleyTable( CayleyTable( S ) );
<loop of order 3>
gap> [ Parent( L ) = L, IsSubloop( M, S ), IsSubloop( M, L ) ];
[ true, true, false ]
```

2.9 Nucleus, commutant, center

Let Q be a quasigroup. The *left nucleus* $N_\lambda(Q)$ of Q is the set $\{x \in Q;\ x(yz) = (xy)z$ for every $y, z \in Q\}$. One defines similarly the *middle nucleus* $N_\mu(Q)$ and the *right nucleus* $N_\rho(Q)$. Then the *nucleus* $N(Q)$ of Q is equal to $N_\lambda(Q) \cap N_\mu(Q) \cap N_\rho(Q)$. These nuclei are calculated in `LOOPS` as follows:

- `LeftNucleus( Q )` A

- `MiddleNucleus( Q )` A

- `RightNucleus( Q )` A

- `Nuc( Q )` A

The word `Nucleus` is reserved in the core `GAP` system for a function in two variables, therefore we use the abbreviation `Nuc`, which is also common in the literature. However, we support these synonyms of `Nuc`:

- `NucleusOfLoop( Q )` A

- `NucleusOfQuasigroup( Q )` A

Since all nuclei are subquasigroups of Q, they are returned as subquasigroups (or subloops). The *commutant* $C(Q)$ of Q is the set $\{x \in Q;\ xy = yx$ for every $y \in Q\}$. It is also known under the name *Moufang center*. It is obtained via

- `Commutant( Q )` A

The center $Z(Q)$ is defined as $C(Q) \cap N(Q)$, and it is obtained via

- `Center( Q )` A

Finally, the *associator subloop* of a loop L is the subloop of L generated by all associators of L. (Note that some authors define the associator subloop as the smallest normal subloop A of L such that L/A is associative. The two definitions are not equivalent in general.)

- `AssociatorSubloop( L )` A

2.10 Testing properties

The reader should be aware that although loops are quasigroups, it is often the case in the literature that a property named P can differ for quasigroups and loops; for instance, a Steiner loop is not necessarily a Steiner quasigroup. To avoid such ambivalences, we often include the noun `Loop` or `Quasigroup` as part of the name of the property; e.g. `IsSteinerQuasigroup` versus `IsSteinerLoop`. On the other hand, some properties coincide for quasigroups and loops and we therefore do not include `Loop`, `Quasigroup` as part of the name of the property; e.g. `IsCommutative`.

2.10.1 Associativity, commutativity and generalizations

The following properties test if a quasigroup Q is associative and commutative.

- `IsAssociative( Q )  P`

- `IsCommutative( Q )  P`

A loop L is said to be *power associative* (resp. *diassociative*) if every monogenic subloop of L (resp. every 2-generated subloop of L) is a group.

- `IsPowerAssociative( L )  P`

- `IsDiassociative( L )  P`

2.10.2 Inverse properties

A loop L has the *left inverse property* if $x'(xy) = y$ for every x, $y \in L$, where x' is the left inverse of x. Dually, L has the *right inverse property* if $(yx)x'' = y$ for every x, $y \in L$, where x'' is the right inverse of x. If L has both the left and right inverse property, it has the *inverse property*. We say that L has *two-sided inverses* if $x' = x''$ for every $x \in L$.

- `HasLeftInverseProperty( L )  P`

- `HasRightInverseProperty( L )  P`

- `HasInverseProperty( L )  P`

- `HasTwosidedInverses( L )  P`

A loop L with two-sided inverses has the *automorphic inverse property* if $(xy)^{-1} = x^{-1}y^{-1}$ for every x, $y \in L$. Similarly, it has the *antiautomorphic inverse property* if $(xy)^{-1} = y^{-1}x^{-1}$.

- `HasAutomorphicInverseProperty( L )  P`

- `HasAntiautomorphicInverseProperty( L )  P`

There are many additional inverse properties but we decided against including them at this stage.

2.10.3 Properties of quasigroups

A quasigroup Q is *semisymmetric* if $(xy)x = y$ for every x, $y \in Q$. Equivalently, Q is semisymmetric if $x(yx) = y$ for every x, $y \in Q$. A semisymmetric commutative quasigroup is known as *totally symmetric*. Totally symmetric quasigroups are precisely quasigroups satisfying $xy = x \setminus y = x/y$.

- `IsSemisymmetric( Q )  P`

- `IsTotallySymmetric( Q )   P`

A quasigroup Q is *idempotent* if $x^2 = x$ for every $x \in Q$. Idempotent totally symmetric quasigroups are known as *Steiner quasigroups*. A quasigroup Q is *unipotent* if $x^2 = y^2$ for every x, $y \in Q$.

- `IsIdempotent( Q )   P`

- `IsSteinerQuasigroup( Q )   P`

- `IsUnipotent( Q )   P`

A quasigroup is *left distributive* if it satisfies $x(yz) = (xy)(xz)$. Similarly, it is *right distributive* if it satisfies $(xy)z = (xz)(yz)$. A *distributive quasigroup* is a quasigroup that is both left and right distributive. A quasigroup is called *entropic* or *medial* if it satisfies $(xy)(zw) = (xz)(yw)$.

- `IsLeftDistributive( Q )   P`

- `IsRightDistributive( Q )   P`

- `IsDistributive( Q )   P`

- `IsEntropic( Q )   P`

- `IsMedial( Q )   P`

TO be compatible with GAP's terminology, we also support the synonyms

- `IsLDistributive( Q )   P`

- `IsRDistributive( Q )   P`

for `IsLeftDistributive` and `IsRightDistributive`, respectively.

2.10.4 Loops of Bol-Moufang type and related properties

Following [5] and [10], a variety of loops is said to be of *Bol-Moufang type* if it is defined by a single *identity of Bol-Moufang type*, i.e., by an identity that: (i) contains the same 3 variables on both sides, (ii) exactly one of the variables occurs twice on both sides, (iii) the variables occur in the same order on both sides.

It is proved in [10] that there are 13 varieties of nonassociative loops of Bol-Moufang type, as summarized in Figure 2.1.

Note that although some of the defining identities are not of Bol-Moufang type, they are equivalent to a Bol-Moufang identity. Moreover, some varieties in the Figure are defined by several, equivalent identities of Bol-Moufang type.

There are several varieties related to loops of Bol-Moufang type. A loop is said to be: *alternative* if it is both left and right alternative; *nuclear square loop* if it is left, middle and right nuclear square.

Here are the corresponding `GAP` commands (argument L indicates that the property applies only to loops, argument Q indicates that the property applies to quasigroups):

- `IsExtraLoop( L )   P`

- `IsMoufangLoop( L )   P`

- `IsCLoop( L )   P`

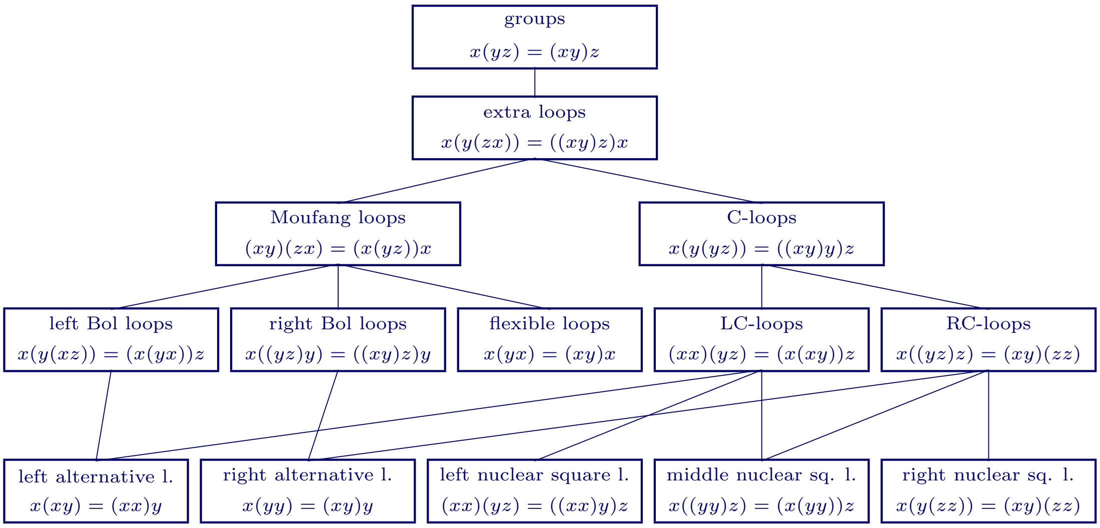

Figure 2.1: Varieties of loops of Bol-Moufang type.

- `IsLeftBolLoop( L )` P
- `IsRightBolLoop( L )` P
- `IsLCLoop( L )` P
- `IsRCLoop( L )` P
- `IsLeftNuclearSquareLoop( L )` P
- `IsMiddleNuclearSquareLoop( L )` P
- `IsRightNuclearSquareLoop( L )` P
- `IsNuclearSquareLoop( L )` P
- `IsFlexible( Q )` P
- `IsLeftAlternative( Q )` P
- `IsRightAlternative( Q )` P
- `IsAlternative( Q )` P

All inclusions among the varieties of loops of Bol-Moufang type are summarized in Figure 2.1, and all these inclusions are built into the LOOPS package. (See Section 2.15.)

The following trivial example shows some of the implications and the naming conventions of LOOPS at work:

```
gap> L := LoopByCayleyTable( [ [ 1, 2 ], [ 2, 1 ] ] );
<loop of order 2>
```

```
gap> IsLeftBolLoop( L );
true
gap> [ HasIsLeftAlternativeLoop( L ), IsLeftAlternativeLoop( L ) ];
[ true, true ]
gap> [ HasIsRightBolLoop( L ), IsRightBolLoop( L ) ];
[ false, true ]
gap> L;
<Moufang loop of order 2>
gap> [ IsAssociative( L ), L ];
[ true, <associative loop of order 2> ]
```

The analogous terminology for quasigroups of Bol-Moufang type is not standard yet, and hence is not supported in LOOPS.

2.10.5 Conjugacy closed loops

A loop is *left* (resp. *right*) *conjugacy closed* if its left (resp. right) translations are closed under composition. A loop that is both left and right conjugacy closed is called *conjugacy closed*. It is common to refer to these loops are LCC, RCC, CC-loops, respectively.

- `IsLCCLoop( L )` P

- `IsRCCLoop( L )` P

- `IsCCLoop( L )` P

The equivalence LCC + RCC = CC is built into the LOOPS package.

2.10.6 Additional varieties of loops

A left (resp. right) Bol loop with the automorphic inverse property is known as *left* (resp. *right*) *Bruck loop*. Bruck loops are also known as *K-loops*.

- `IsLeftBruckLoop( L )` P

- `IsLeftKLoop( L )` P

- `IsRightBruckLoop( L )` P

- `IsRightKLoop( L )` P

Steiner loop is an inverse property loop of exponent 2.

- `IsSteinerLoop( L )` P

2.11 Normality

A subloop S of a loop L is *normal* if it is invariant under all inner mappings of L. Normality is tested via:

- `IsNormal( L, S )` F

When S is a subset of a loop L or a subloop of L the *normal closure* of S in L is the smallest normal subloop of L containing S. It is obtained by

- `NormalClosure( L, S )` F

A loop L is *simple* if all normal subloops of L are trivial. The corresponding test in LOOPS is:

- IsSimple(L) P

2.12 Factor loop

When N is a normal subloop of a loop L, the factor loop L/N can be obtained directly via the command L/N, or by

- FactorLoop(L, N) F

The natural projection from L to L/N is returned as follows:

- NaturalHomomorphismByNormalSubloop(L, N) F

```
gap> M := MoufangLoop( 12, 1 );; S := Subloop( M, [ M.3 ] );
<loop of order 3>
gap> IsNormal( M, S );
true
gap> F := FactorLoop( M, S );
<loop of order 4>
gap> NaturalHomomorphismByNormalSubloop( M, S );
MappingByFunction( <loop of order 12>, <loop of order 4>,
    function( elm ) ... end )
```

2.13 Nilpotency

The definition of nilpotency and nilpotence class is the same as in group theory. The corresponding commands are:

- NilpotencyClassOfLoop(L) A

- IsNilpotent(L) P

When L is not nilpotent, NilpotencyClassOfLoop(L) returns fail.

A loop L is said to be *strongly nilpotent* if its multiplication group is nilpotent. This property is obtained by:

- IsStronglyNilpotent(L) P

2.14 Solvability

The definition of solvability, derived subloop, derived length, Frattini subloop and Frattini factor size is the same as for groups. Frattini subloop is calculated only for strongly nilpotent loops.

- IsSolvable(L) P

- DerivedSubloop(L) A

- DerivedLength(L) A

- `FrattiniSubloop( L )  A`

- `FrattinifactorSize( L )  A`

2.15 Filters built into LOOPS

Many implications among properties of loops are built directly into LOOPS. A sizeable portion of these properties are of trivial character or are based on definitions (e.g., alternative loops = left alternative loops + right alternative loops). The remaining implications are theorems.

All filters of LOOPS are summarized below (using the GAP convention that the property on the left is implied by the property (properties) on the right).

```
( IsExtraLoop,  IsAssociative and IsLoop )
( IsDiassociative, IsAssociative and IsLoop )
( HasInverseProperty, HasRightInverseProperty and IsCommutative )
( HasInverseProperty, HasLeftInverseProperty and IsCommutative )
( IsMoufangLoop, IsRightBolLoop and IsCommutative )
( IsMoufangLoop, IsLeftBolLoop and IsCommutative )
( IsMoufangLoop, IsRightBruckLoop and IsCommutative )
( IsMoufangLoop, IsLeftBruckLoop and IsCommutative )
( IsRightNuclearSquareLoop, IsLeftNuclearSquareLoop and IsCommutative )
( IsLeftNuclearSquareLoop, IsRightNuclearSquareLoop and IsCommutative )
( HasAutomorphicInverseProperty, HasAntiautomorphicInverseProperty and IsCommutative )
( HasAntiautomorphicInverseProperty, HasAutomorphicInverseProperty and IsCommutative )
( IsAlternative, IsLeftAlternative and IsCommutative )
( IsAlternative, IsRightAlternative and IsCommutative )
( HasTwosidedInverses, IsPowerAssociative )
( IsPowerAssociative, IsDiassociative )
( IsAlternative, IsDiassociative )
( IsFlexible, IsDiassociative )
( HasLeftInverseProperty, HasInverseProperty )
( HasRightInverseProperty, HasInverseProperty )
( HasTwosidedInverses, HasInverseProperty )
( HasInverseProperty, HasLeftInverseProperty and HasRightInverseProperty )
( IsMoufangLoop, IsExtraLoop )
( IsNuclearSquareLoop, IsExtraLoop )
( IsCLoop, IsExtraLoop )
( IsExtraLoop, IsMoufangLoop and IsLeftNuclearSquareLoop )
( IsExtraLoop, IsMoufangLoop and IsMiddleNuclearSquareLoop )
( IsExtraLoop, IsMoufangLoop and IsRightNuclearSquareLoop )
( IsLeftBolLoop, IsMoufangLoop )
( IsRightBolLoop, IsMoufangLoop )
( IsFlexible, IsMoufangLoop )
( IsDiassociative, IsMoufangLoop )
( IsMoufangLoop, IsLeftBolLoop and IsRightBolLoop )
( IsLCLoop, IsCLoop )
( IsRCLoop, IsCLoop )
( IsCLoop, IsLCLoop and IsRCLoop )
( IsLeftAlternative, IsLeftBolLoop )
( HasTwosidedInverses, IsLeftBolLoop )
( IsRightAlternative, IsRightBolLoop )
( HasTwosidedInverses, IsRightBolLoop )
( IsLeftAlternative, IsLCLoop )
( IsLeftNuclearSquareLoop, IsLCLoop )
( IsMiddleNuclearSquareLoop, IsLCLoop )
( IsPowerAssociative, IsLCLoop )
( IsRightAlternative, IsRCLoop )
( IsRightNuclearSquareLoop, IsRCLoop )
( IsMiddleNuclearSquareLoop, IsRCLoop )
( IsPowerAssociative, IsRCLoop )
( IsLeftNuclearSquareLoop, IsNuclearSquareLoop )
( IsRightNuclearSquareLoop, IsNuclearSquareLoop )
( IsMiddleNuclearSquareLoop, IsNuclearSquareLoop )
( IsNuclearSquareLoop, IsLeftNuclearSquareLoop and IsRightNuclearSquareLoop and IsMiddleNuclearSquareLoop )
( IsLeftAlternative, IsAlternative )
( IsRightAlternative, IsAlternative )
( IsAlternative, IsLeftAlternative and IsRightAlternative )
( IsLCCLoop, IsCCLoop )
( IsRCCLoop, IsCCLoop )
( IsCCLoop, IsLCCLoop and IsRCCLoop )
( HasAutomorphicInverseProperty, IsLeftBruckLoop )
( IsLeftBolLoop, IsLeftBruckLoop )
( IsLeftBruckLoop, IsLeftBolLoop and HasAutomorphicInverseProperty )
( HasAutomorphicInverseProperty, IsRightBruckLoop )
( IsRightBolLoop, IsRightBruckLoop )
( IsRightBruckLoop, IsRightBolLoop and HasAutomorphicInverseProperty )
( IsCommutative, IsSteinerLoop )
( HasInverseProperty, IsSteinerLoop )
```

Chapter 3

Specific methods

3.1 Isomorphisms and automorphisms

All isomorphisms between two loops can be found with `LOOPS`. The function

- `IsomorphismLoops( L, M )   F`

returns a single isomorphism between loops L, M, if the loops are isomorphic, and it fails otherwise. If an isomorphism exists it is returned as a permutation $\pi \in S_{|L|}$, where $\pi(i) = j$ means that the ith element of L is mapped onto the jth element of M.

The function

- `AutomorphismGroup( L )   F`

returns the automorphism group of the loop L. Since two isomorphisms differ by an automorphism, all isomorphisms can be obtained by the above two functions.

3.1.1 Discriminator

In order to speed up the search for isomorphisms and automorphisms, we first calculate some loop invariants preserved under isomorphisms, and use these invariants to partition the loop into blocks of elements preserved under isomorphism. These invariants for a loop L can be obtained via

- `Discriminator( L )   F`

Since the details are technical, we will not present them here. See [11] for more.

If two loops have different discriminators, they are not isomorphic. If they have identical discriminator, they may or may not be isomorphic. The function

- `AreEqualDiscriminators( D1, D2 )   F`

returns true if the discriminators $D1$, $D2$ are equal.

Given a loop L and its discriminator D, the function

- `EfficientGenerators( L, D )   F`

returns a generating set of L that is optimized with respect to the discriminator D. Once again, the details are too technical to be presented here. The returned set of generators is usually very small. Also see `SmallGeneratingSet`.

3.2 Moufang modifications

Aleš Drápal discovered two prominent families of extensions of Moufang loops. These extensions can be used to obtain many, perhaps all, nonassociative Moufang loops of order at most 64. We call these two constructions *Moufang modifications*. The library of Moufang loops included with `LOOPS` is based on Moufang modifications. We describe the two modifications briefly here. See [4] for details.

3.2.1 Cyclic modification

Assume that L is a Moufang loop with normal subloop S such that L/S is a cyclic group of order $2m$. Let $h \in S \cap Z(L)$. Let α be a generator of L/S and write $L = \bigcup_{i \in M} \alpha^i$, where $M = \{-m+1, \dots, m\}$. Let $\sigma : \mathbb{Z} \to M$ be defined by

$$\sigma(i) = \begin{cases} 0, & i \in M, \\ 1, & i > m, \\ -1, & i < -m+1. \end{cases}$$

Introduce a new multiplication $*$ on L defined by

$$x * y = xyh^{\sigma(i+j)},$$

where $x \in \alpha^i$, $y \in \alpha^j$, $i \in M$, $j \in M$. Then $(L, *)$ is a Moufang loop, a *cyclic modification* of L.

When L, S, α, h are as above and when a is any element of α, the corresponding cyclic modification is obtained via

- `LoopByCyclicModification( L, S, a, h )` F

3.2.2 Dihedral modification

Assume that L is a Moufang loop with normal subloop S such that L/S is a dihedral group of order $4m$, with $m \geq 1$. Let M and σ be defined as in the cyclic case. Let β, $\gamma \in L/S$ be two involutions of L/S such that $\alpha = \beta\gamma$ generates a cyclic subgroup of L/S of order $2m$. Let $e \in \beta$ and $f \in \gamma$ be arbitrary. Then L can be written as a disjoint union $L = \bigcup_{i \in M}(\alpha^i \cup e\alpha^i)$, and also $L = \bigcup_{i \in M}(\alpha^i \cup \alpha^i f$. Let $G_0 = \bigcup_{i \in M} \alpha^i$, and $G_1 = L \setminus G_0$. Let $h \in S \cap N(L) \cap Z(G_0)$. Introduce a new multiplication $*$ n L defined by

$$x * y = xyh^{(-1)^r \sigma(i+j)},$$

where $x \in \alpha^i \cup e\alpha^i$, $y \in \alpha^j \cup \alpha^j f$, $i \in M$, $j \in M$, $y \in G_r$, $r \in \{0, 1\}$. Then $(L, *)$ is a Moufang loop, a *dihedral modification* of L.

When L, S, e, f and h are as above, the corresponding dihedral modification is obtained via

- `LoopByDihedralModification( L, S, e, f, h )` F

3.2.3 Loops $M(G, 2)$

In order to apply the cyclic and dihedral modification, it is beneficial to have access to a class of nonassociative Moufang loops. The following construction is due to Chein:

Let G be a group. Let $\overline{G} = \{\overline{g};\ g \in G\}$ be a set of new elements. Define multiplication $*$ on $L = G \cup \overline{G}$ by

$$g * h = gh,\ \ g * \overline{h} = \overline{hg},\ \ \overline{g} * h = \overline{gh^{-1}},\ \ \overline{g} * \overline{h} = h^{-1}g,$$

where $g, h \in G$. Then $L = M(G,2)$ is a Moufang loop that is nonassociative if and only if G is nonabelian.

The loop $M(G,2)$ can be obtained from a finite group G with

- `LoopMG2( G )  F`

in `LOOPS`.

3.3 Triality for Moufang loops

Let G be a group and σ, ρ be automorphisms of G, satisfying $\sigma^2 = \rho^3 = (\sigma\rho)^2 = 1$. We write the automorphisms of a group as exponents and $[g,\sigma]$ for $g^{-1}g^{\sigma}$. We say that the triple (G,ρ,σ) is a *group with triality* if $[g,\sigma][g,\sigma]^{\rho}[g,\sigma]^{\rho^2} = 1$ holds for all $g \in G$. It is known that one can associate a group with triality (G,ρ,σ) in a canonical way with a Moufang loop L. See [8] for more details.

For any Moufang loop L, we can calculate the triality group as a permutation group acting on $3|L|$ points. If the multiplication group of L is polycyclic, then we can also represent the triality group as a pc group. In both cases, the automorphisms σ and ρ are in the same family as the elements of G.

Given a Moufang loop L, the function

- `TrialityPermGroup( L )  F`

returns a record $[G, \rho, \sigma]$, where G is the group with triality associated with L, and ρ, σ are the corresponding triality automorphisms.

The function

- `TrialityPcGroup( L )  F`

differs from `TrialityPermGroup` only in that G is returned as a pc group.

Chapter 4

Libraries of small loops

Libraries of small loops are an integral part of `LOOPS`.

4.1 A typical library

A library named "my Library" is stored in file `data/mylibrary.tbl`, and the corresponding data structure is named `my_library_data`.

The array `my_library_data` consists of three lists: `my_library_data[ 1 ]` is a list of orders for which there is at least one loop in the library, `my_library_data[ 2 ][ k ]` is the number of loops of order `my_library_data[ 1 ][ k ]` in the library, and `my_library_data[ 3 ][ s ]` contains data necessary to produce the sth loop in the library. The format of `my_library_data[ 3 ]` depends on the particular library and is not standardized in any way.

The user can retrieve the mth loop of order n from library named "my Library" according to the template

- `MyLibraryLoop( n, m )  global function template`

It is also possible to obtain the same loop with

- `LibraryLoop( name, n, m )  F`

where `name` is the name of the library.

For example, when the library is called "left Bol", the corresponding data file is called `data/leftbol.tbl`, the corresponding data structure is named `left_bol_data`, and the mth left Bol loop of order n is obtained via

- `LeftBolLoop( n, m )  F`

or via

- `LibraryLoop( "left Bol", n, m )  F`

We are now going to describe the individual libraries in detail. A brief information about the library named `name` can also be obtained in `LOOPS` with

- `DisplayLibraryInfo( name )  F`

4.2 Left Bol loops

The library named "left Bol" contains all 6 nonassociative left Bol loops of order 8. Following the general pattern, the mth nonassociative left Bol loop of order n is obtained by

- `LeftBolLoop( n, m )` F

We intend to enlarge this library significantly in future versions of `LOOPS`, when the classification of small Bol loops is completed.

4.3 Small Moufang loops

The library named "Moufang" contains all nonassociative Moufang loops of order less than 64, and additional 4262 nonassociative Moufang loops of order 64. It is possible that there are no other nonassociative Moufang loops of order 64 than those contained in the library.

The mth nonassociative Moufang loop of order n is obtained by

- `MoufangLoop( n, m )` F

For $n \leq 63$, our catalog numbers coincide with those of Goodaire et al. [6].

The extent of the library is summarized below:

order	12	16	20	24	28	32	36	40	42	44	48	52	54	56	60	64
loops in the libary	1	5	1	5	1	71	4	5	1	1	51	1	2	4	5	4262

The *octonion loop* of order 16 (i.e., the multiplication loop of the ± basis elements in the 8-dimensional standard real octonion algebra) is `MoufangLoop( 16, 3 )`.

4.3.1 Search for additional Moufang loops

Since we would like to know if there are additional nonassociative Moufang loops of order 64, we have implemented the function

- `IsomorphismTypeOfMoufangLoop( L )` F

If L is a Moufang loop cataloged in `LOOPS` as the mth Moufang loop of order n, the function returns `[[n,m],p]`, where p is a permutation of `[1..n]` that is an isomorphism from L to the cataloged copy of L. If $n = 64$ and L is Moufang loop not cataloged in `LOOPS`, the user is prompted to contact the authors of `LOOPS`.

In order to speed up the function `IsomorphismTypeOfMoufangLoop`, we have precalculated and stored in the data file `data/moufang_discriminators.tbl` the discriminators of all Moufang loops in the library. The file is rather large (850 KB) and took about 20 minutes to precalculate. You can delete the file if you won't use `IsomorphismTypeOfMoufangLoop`.

```
gap> D := DirectProduct( MoufangLoop( 16, 2 ), CyclicGroup( 2 ) );
<loop of order 32>
gap> IsomorphismTypeOfMoufangLoop( D );
[ [ 32, 2 ], (2,3,12,20,11,29,23,13,30,31,28,27,22,15,32,18,10,19,16,24,14,
25,21,8,7,6,9,17,5) ]
gap> A := AutomorphismGroup( D ); Size( A );
<permutation group with 34 generators>
3072
```

4.4 Steiner loops

Here is how the libary "Steiner" is described within `LOOPS`:

```
gap> DisplayLibraryInfo( "Steiner" );
The library contains all nonassociative Steiner loops of order less or equal
to 16. It also contains the associative Steiner loops of order 4 and 8.
------
Extent of the library:
   1 loop of order 4
   1 loop of order 8
   1 loop of order 10
   2 loops of order 14
   80 loops of order 16
true
```

The mth Steiner loop of order n is obtained by

- `SteinerLoop( n, m )` F

Our catalog numbers coincide with those of Colbourn and Rosa [3].

4.5 Paige loops

Paige loops are nonassociative finite simple Moufang loops. By [7], there is precisely one Paige loop for every finite field $GF(q)$.

The library named "Paige" contains the smallest nonassociative simple Moufang loop

- `PaigeLoop( 2 )` F

4.6 Interesting loops

The library named "interesting" contains some loops that are illustrative for the theory of loops. At this point, the library contains a nonassociative loop of order 5, a nonassociative nilpotent loop of order 6, a nonMoufang left Bol loop of order 16, and the loop of sedenions of order 32 (sedenions generalize octonions).

The loops are obtained with

- `InterestingLoop( n, m )` F

Chapter 5

Plans for future versions

We hope that the LOOPS package will become a standard computational tool in quasigroup theory and loop theory, and we therefore anticipate some interest among researchers in expanding the package. In this chapter, we present several possible directions in which this future expansion could lead. Since we will base our decision on your feedback, please let us know what you would like to see implemented in LOOPS.

5.1 Alternative representations of quasigroups and loops in GAP

(The word ”representation” does not have the usual mathematical meaning in this section.) Direct products, semidirect products and many other constructions of loops can be represented in a more space-efficient way than by Cayley tables. Large Paige loops, generalizations of octonions and other loops can be represented nicely. None of these representations is currently implemented in LOOPS.

Presentations of some loops within their varieties are known and perhaps should be found in LOOPS.

5.2 Better support for quasigroups

This package is concerned primarily with loops. Although some functions are kept on a level general enough for quasigroups, many are not. Only a few quasigroup-theoretical properties are testable in LOOPS at this point. The operations `LeftDivision` and `RightDivision` are awkward to work with.

5.3 More homomorphisms and homotopisms

The general concept of homomorphisms of quasigroup is obvious. Beside

`NaturalHomomorphismByNormalSubloop`, `PrincipalLoopIsotope`,

other homomorphisms and homotopisms should be defined for loops and quasigroups.

5.4 Expanded libraries

More Bol loops should be cataloged. Interesting loops should be gathered in a more systematic way. Loops could be cataloged not only up to isomorphism but also up to isotopism.

We know how to create the library of all conjugacy closed loops of order p^2. Any interest?

5.5 Bits and pieces

We would like to see the following features in a future version of `LOOPS`: M_k laws, cross inverse property (CIP), weak inverse property (WIP), homotopisms.

Bibliography

[1] R. Hubert Bruck, A Survey of Binary Systems, third printing, corrected, *Ergebnisse der Mathematik und ihrer Grenzgebiete*, *Neue Folge* **20**, Springer-Verlag, 1971.

[2] O. Chein, H. O. Pflugfelder, J. D. H. Smith (editors), Quasigroups and Loops: Theory and Applications, *Sigma Series in Pure Mathematics* **8**, Heldermann Verlag Berlin, 1990.

[3] Charles J. Colbourn and Alexander Rosa, Triple systems, *Oxford Mathematical Monographs*, The Clarendon Press, Oxford University Press, New York, 1999.

[4] Aleš Drápal and Petr Vojtěchovský, *Moufang loops that share associator and three quarters of their multiplication tables*, to appear in Rocky Mountain Journal of Mathematics.

[5] Ferenc Fenyves, *Extra loops II, On loops with identities of Bol-Moufang type*, Publ. Math. Debrecen **16**(1969), 187–192.

[6] Edgar G. Goodaire, Sean May and Maitreyi Raman, The Moufang loops of order less than 64, Commack, NY: Nova Science Publishers, 1999.

[7] M. Liebeck, *The classification of finite simple Moufang loops*, Math. Proc. Cambridge Philos. Soc. **102** (1987), 33–47.

[8] Gábor P. Nagy and Petr Vojtěchovský, *Octonions, simple Moufang loops and triality*, Quasigroups and Related Systems **10** (2003), 65–94.

[9] Hala O. Pflugfelder, Quasigroups and Loops: Introduction, *Sigma Series in Pure Mathematics* **7**, Heldermann Verlag Berlin, 1990.

[10] J. D. Phillips and Petr Vojtěchovský, *Varieties of loops of Bol-Moufang type*, submitted.

[11] Petr Vojtěchovský, *Toward the classification of Moufang loops of order* 64, to appear in European Journal of Combinatorics.

Index