Guarana

A Gap4 Package

(Version 0.94)

Björn Assmann

Copyright

Contents

Chapter 1

Introduction

1.1 About Guarana

In this package we demonstrate the algorithmic usefulness of the so-called Mal'cev correspondence for computations with infinite polycyclic groups; it is a correspondence that associates to every -powered nilpotent group H a unique rational nilpotent Lie algebra L_H and vice-versa. The Mal'cev correspondence was discovered by Anatoly Mal'cev in 1951 [Mal51].

1.2 Setup for computing the correspondence

Let G be a finitely generated torsion-free nilpotent group, i.e.\ a T-group. Then G can be embedded in a $\mathbb{Q}$-powered hull $\hat{G}$. The group $\hat{G}$ is a $\mathbb{Q}$-powered nilpotent group and is unique up to isomorphism. We denote the Lie algebra which corresponds to $\hat{G}$ under the Mal'cev correspondence by $L(G) = L_{\hat{G}}$. We provide an algorithm for setting up the Mal'cev correspondence between $\hat{G}$ and the Lie algebra $L(G)$. That is, if G is given by a polycyclic presentation with respect to a Mal'cev basis, then we can compute a structure constants table of $L(G)$. Furthermore for a given $g \in G$ we can compute the corresponding element in $L(G)$ and vice versa.

1.3 Collection

Every element of a polycyclically presented group has a unique normal form. An algorithm for computing this normal form is called a collection algorithm. Such an algorithm lies at the heart of most methods dealing with polycyclically presented groups. The current state of the art is collection from the left [Geb02][LGS90][VL90] }. This package contains a new collection algorithm for polycyclically presented groups, which we call Mal'cev collection [AL07]. Mal'cev collection is in some cases dramatically faster than collection from the left, while using less memory.

Chapter 2

Computing the Mal'cev correspondence

2.1 The main functions

Let G be a T-group and $\hat{G}$ its $\mathbb{Q}$-powered hull. In this chapter we describe functionality for setting up the Mal'cev correspondence between $\hat{G}$ and the Lie algebra $L(G)$. The data structures needed for computations with $\hat{G}$ and $L(G)$ are stored in a so-called Mal'cev object. Computational representations of elements of $\hat{G}$, respectively $L(G)$, will be called Mal'cev group elements, respectively Mal'cev Lie elements.

2.1.1 MalcevObjectByTGroup

▷ `MalcevObjectByTGroup(N)` (function)

If `N` is a a `T`-group (i.e. a finitely generated torsion-free nilpotent group), given by a polycyclic presentation with respect to a Mal'cev basis, then this function computes the Mal'cev correspondence for `N` and stores the result in a so-called Mal'cev object. Otherwise this function returns 'fail'. In the moment this function is restricted to groups `N` of nilpotency class at most 9.

2.1.2 UnderlyingGroup

▷ `UnderlyingGroup(mo)` (function)

For a Mal'cev object `mo` this function returns the `T`-group, which was used to build `mo`.

2.1.3 UnderlyingLieAlgebra

▷ `UnderlyingLieAlgebra(mo)` (function)

For a Mal'cev object `mo` this function returns the Lie algebra, which underlies the correspondence described by `mo`.

2.1.4 Dimension

▷ `Dimension(mo)` (function)

Returns the dimension of the Lie algebra that underlies the Mal'cev object *mo*.

2.1.5 MalcevGrpElementByExponents

▷ MalcevGrpElementByExponents(*mo, exps*) (function)

For a Mal'cev object *mo* and an exponent vector *exps* with rational entries, this functions returns the Mal'cev group element, which has exponents *exps* with respect to the Mal'cev basis of the underlying group of *mo*.

2.1.6 MalcevLieElementByCoefficients

▷ MalcevLieElementByCoefficients(*mo, coeffs*) (function)

For a Mal'cev object *mo* and a coefficient vector *coeffs* with rational entries, this functions returns the Mal'cev Lie element, which has coefficients *coeffs* with respect to the basis of the underlying Lie algebra of *mo*.

2.1.7 RandomGrpElm

▷ RandomGrpElm(*mo, range*) (function)

For a Mal'cev object *mo* this function returns the output of MalcevGrpElementByExponents(*mo*, *exps*), where *exps* is an exponent vector whose entries are randomly chosen integers between -*range* and *range*.

2.1.8 RandomLieElm

▷ RandomLieElm(*mo, range*) (function)

For a Mal'cev object *mo* this function returns the output of MalcevLieElementByExponents(*mo*, *coeffs*), where *coeffs* is a coefficient vector whose entries are randomly chosen integers between -*range* and *range*.

2.1.9 Log

▷ Log(*g*) (function)

For Mal'cev group element *g* this function returns the corresponding Mal'cev Lie element.

2.1.10 Exp

▷ Exp(*x*) (function)

For Mal'cev Lie element *x* this function returns the corresponding Mal'cev group element.

2.1.11 *

▷ *(*g*, *h*) (function)

Returns the product of Mal'cev group elements.

2.1.12 Comm

▷ Comm(*x*, *y*) (function)

If *x*,*y* are Mal'cev group elements, then this function returns the group theoretic commutator of *x* and *y*. If *x*,*y* are Mal'cev Lie elements, then this function returns the Lie commutator of *x* and *y*.

2.1.13 MalcevSymbolicGrpElementByExponents

▷ MalcevSymbolicGrpElementByExponents(*mo*, *exps*) (function)

For a Mal'cev object *mo* and an exponent vector *exps* with rational indeterminates as entries, this functions returns the Mal'cev group element, which has exponents *exps* with respect to the Mal'cev basis of the underlying group of *mo*.

2.1.14 MalcevLieElementByCoefficients

▷ MalcevLieElementByCoefficients(*mo*, *coeffs*) (function)

For a Mal'cev object *mo* and a coefficient vector *coeffs* with rational indeterminates as entries, this functions returns the Mal'cev Lie element, which has coefficients *coeffs* with respect to the basis of the underlying Lie algebra of *mo*.

2.2 An example application

Example

```
gap> n := 2;
2
gap> F := FreeGroup( n );
<free group on the generators [ f1, f2 ]>
gap> c := 3;
3
gap> N := NilpotentQuotient( F, c );
Pcp-group with orders [ 0, 0, 0, 0, 0 ]

gap> mo := MalcevObjectByTGroup( N );
<<Malcev object of dimension 5>>
gap> dim := Dimension( mo );
5
gap> UnderlyingGroup( mo );
Pcp-group with orders [ 0, 0, 0, 0, 0 ]
gap> UnderlyingLieAlgebra( mo );
<Lie algebra of dimension 5 over Rationals>
```

```
gap> g := MalcevGrpElementByExponents( mo, [1,1,0,2,-1/2] );
[ 1, 1, 0, 2, -1/2 ]
gap> x := MalcevLieElementByCoefficients( mo, [1/2, 2, -1, 3, 5 ] );
[ 1/2, 2, -1, 3, 5 ]

gap> h := RandomGrpElm( mo );
[ 5, -3, 0, -2, 8 ]
gap> y := RandomLieElm( mo );
[ 3, 9, 5, 5, 2 ]

gap> z := Log( g );
[ 1, 1, -1/2, 7/3, -1/3 ]
gap> Exp( z ) = g;
true
gap> k := Exp( y );
[ 3, 9, 37/2, 77/4, 395/4 ]
gap> Log( k ) = y;
true

gap> g*h;
[ 6, -2, 5, 10, -15/2 ]
gap> Comm(g,h);
[ 0, 0, 8, 10, -18 ]
gap> Comm(x,y);
[ 0, 0, 3/2, -25/4, -79/4 ]

gap> indets := List( List( [1..dim], i->Concatenation( "a_", String(i) ) ),
>                    x->Indeterminate( Rationals, x : new ) );
[ a_1, a_2, a_3, a_4, a_5 ]
gap> g_sym := MalcevSymbolicGrpElementByExponents( mo, indets );
[ a_1, a_2, a_3, a_4, a_5 ]
gap> x_sym := Log( g_sym );
[ a_1, a_2, -1/2*a_1*a_2+a_3, 1/12*a_1^2*a_2+1/4*a_1*a_2-1/2*a_1*a_3+a_4,
-1/12*a_1*a_2^2+1/4*a_1*a_2-1/2*a_2*a_3+a_5 ]
gap> g_sym * g;
[ a_1+1, a_2+1, a_2+a_3, a_3+a_4+2, 1/2*a_2^2+1/2*a_2+a_3+a_5-1/2 ]
```

Chapter 3

Mal'cev collection

Let G be an infinite polycyclic group. It is well-known that there exist a normal T-group N and a T-group C such that $H = CN$ is normal of finite index in G and H/N is free abelian of finite rank [Seg83]. In this chapter we present an effective collection method for an infinite polycyclic group which is given by a polycyclic presentation with respect to a polycyclic sequence P going through the normal series $1 \leq N \leq H \leq G$. This polycyclic sequence P must be chosen as follows. Let $(n_1, \ldots, n_l)$ be a Mal'cev basis of N and let $(c_1N, \ldots, c_kN)$ be a basis for the free abelian group CN/N. Then $(c_1, \ldots, c_k, n_1, \ldots, n_l)$ is a polycyclic sequence for $H = CN$. Further there exists $f_1, \ldots, f_j \in G$ such that $(f_1H, \ldots, f_jH)$ is a polycyclic sequence for G/H. Now we set

$$P = (f_1, \ldots, f_j, c_1, \ldots, c_k, n_1, \ldots, n_l)$$

3.1 The main functions

3.1.1 MalcevCollectorConstruction

▷ `MalcevCollectorConstruction(G, inds, C, CC, N, NN)` (function)

Returns a Mal'cev collector for the infinite polycyclically presented group G. The group G must be given with respect to a polycyclic sequence $(g_1, \ldots, g_r, c_{r+1}, \ldots, c_{r+s}, n_{r+s+1}, \ldots, n_{r+s+t})$ with the following properties:

- (a) $(n_{r+s+1}, \ldots, n_{r+s+t})$ is a Mal'cev basis for the T-group $N \leq G$,
- (b) $(c_{r+1}N, \ldots, c_{r+s}N)$ is a basis for the free-abelian group CN/N where $C \leq G$ is a T-group generated by $c_{r+1}, \ldots, c_{r+s}$,
- (c) $(g_1CN, \ldots, g_rCN)$ is a polycyclic sequence for the finite group G/CN.

The list `inds` is equal to $[[1, \ldots, r], [r+1, \ldots, r+s], [r+s+1, \ldots, r+s+t]]$. The group CC is isomorphic to C via `CC`!.bijection and given by a polycyclic presentation with respect to a Mal'cev basis starting with $c_{r+1}, \ldots, c_{r+s}$. The group NN is isomorphic to N via `NN`!.bijection. and given by a polycyclic presentation with respect to the Mal'cev basis $(n_{r+s+1}, \ldots, n_{r+s+t})$.

3.1.2 GUARANA.Tr_n_O1

▷ `GUARANA.Tr_n_O1(n)` (function)
▷ `GUARANA.Tr_n_O2(n)` (function)

for a positive integer `n` these functions construct polycyclically presented groups that can be used to test the Mal'cev collector. They return a list which can be used as input for the function MalcevCollectorConstruction. The constructed groups are isomorphic to triangular matrix groups of dimension `n` over the ring O_1, respectively O_2. The ring O_1, respectively O_2, is the maximal order of $\mathbb{Q}(\theta_i)$ where θ_1, respectively θ_2, is a zero of the polynomial $p_1(x) = x^2 - 3$, respectively $p_2(x) = x^3 - x^2 + 4$.

3.1.3 GUARANA.F_2c_Aut1

▷ `GUARANA.F_2c_Aut1(c)` (function)
▷ `GUARANA.F_3c_Aut1(c)` (function)

for a positive integer `c` these functions construct polycyclically presented groups that can be used to test the Mal'cev collector. They return a list which can be used as input for the function MalcevCollectorConstruction. These groups are constructed as follows: Let $F_{n,c}$ be the free nilpotent of class c group on n generators. An automorphism φ of the free group F_n naturally induces an automorphism $\bar{\varphi}$ of $F_{n,c}$. We use the automorphism φ_1 of F_2 which maps f_1 to f_2^{-1} and f_2 to $f_1 f_2^3$ and the automorphism φ_2 of F_3 mapping f_1 to f_2^{-1}, f_2 to f_3^{-1} and f_3 to $f_2^{-3} f_1^{-1}$ for our construction. The returned group F_2c_Aut1, respectively F_3c_Aut2, is isomorphic to the semidirect product $\langle \varphi_1 \rangle \ltimes F_{2,c}$, respectively $\langle \varphi_2 \rangle \ltimes F_{3,c}$.

3.1.4 MalcevGElementByExponents

▷ `MalcevGElementByExponents(malCol, exps)` (function)

For a Mal'cev collector `malCol` of a group G and an exponent vector `exps` with integer entries, this functions returns the group element of G, which has exponents `exps` with respect to the polycyclic sequence underlying `malCol`.

3.1.5 Random

▷ `Random(malCol, range)` (function)

For a Mal'cev collector `malCol` this function returns the output of MalcevGElementByExponents(`malCol`, `exps`), where `exps` is an exponent vector whose entries are randomly chosen integers between -`range` and `range`.

3.1.6 *

▷ `*(g, h)` (function)

Returns the product of group elements which are defined with respect to a Mal'cev collector by the the function MalcevGElementByExponents.

3.1.7 GUARANA.AverageRuntimeCollec

▷ GUARANA.AverageRuntimeCollec(*malCol, ranges, no*) (function)

For a Mal'cev collector *malCol*, a list of positive integers *ranges* and a positive integer *no* this function computes for each number *r* in *ranges* the average runtime of *no* multiplications of two random elements of *malCol* of range *r*, as generated by Random(*malCol*, *r*).

3.2 An example application

Example

```
gap> ll := GUARANA.Tr_n_O1( 3 );
[ Pcp-group with orders [ 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
  [ [ 1 .. 3 ], [ 4 .. 6 ], [ 7 .. 12 ] ],
  Pcp-group with orders [ 0, 0, 0, 0, 0, 0 ],
  Pcp-group with orders [ 0, 0, 0, 0, 0, 0 ],
  Pcp-group with orders [ 0, 0, 0 ], Pcp-group with orders [ 0, 0, 0 ] ]
gap> malCol := MalcevCollectorConstruction( ll );
<<Malcev collector>>
  F : [ 2, 2, 2 ]
  C : <<Malcev object of dimension 3>>
  N : <<Malcev object of dimension 6>>

gap> exps_g := [ 1, 1, 1, -3, -2, 1, -2, -1, 0, 3, -1,3 ];
[ 1, 1, 1, -3, -2, 1, -2, -1, 0, 3, -1, 3 ]
gap> exps_h := [ 1, 0, 1, -1, 0, 2, 0, 4, -1, 5, 9,-5 ];
[ 1, 0, 1, -1, 0, 2, 0, 4, -1, 5, 9, -5 ]
gap> g := MalcevGElementByExponents( malCol, exps_g );
[ 1, 1, 1, -3, -2, 1, -2, -1, 0, 3, -1, 3 ]
gap> h := MalcevGElementByExponents( malCol, exps_h );
[ 1, 0, 1, -1, 0, 2, 0, 4, -1, 5, 9, -5 ]

gap> k := g*h;
[ 0, 1, 0, -4, -2, 3, -7, 0, -37, -16, -352, -212 ]

gap> Random( malCol, 10 );
[ 0, 0, 1, 9, 5, 5, 2, -2, 7, -10, 7, -6 ]
```

Chapter 4

Installation

4.1 Installing this package

The Guarana package is part of the standard distribution of GAP and so normally there should be no need to install it separately. If by any chance it is not part of your GAP distribution, then the standard method is to unpack the package into the 'pkg' directory of your GAP distribution. This will create a 'guarana' subdirectory. For other non-standard options please see Chapter (**Reference: Installing a GAP Package**) of the GAP Reference Manual. Note that the GAP-Packages Polycyclic and Polenta are needed for this package. Normally they should be contained in your distribution. If not, they can be obtained at http://www.gap-system.org/Packages/packages.html

4.2 Loading the Guarana package

If the Guarana Package is not already loaded then you have to request it explicitly. This can be done by 'LoadPackage("guarana")'. The 'LoadPackage' command is described in Section (**Reference: LoadPackage**) in the GAP Reference Manual.

4.3 Running the test suite

Once the package is installed, it is possible to check the correct installation by running the test suite of the package.

Example

```
gap> Read(Filename(DirectoriesPackageLibrary("guarana","tst")[1],"testall.g"));
```

Note that this now disables running of the tests in tst/guarana2.tst, since these require alnuth which in turn requires kash/kant. If your system has these prerequisites in place you can manually run those test examples as follows.

Example

```
gap> ReadTest(Filename(DirectoriesPackageLibrary("guarana","tst")[1],"guarana2.tst"));
```

For more details on Test Files see Section (**Reference: Test Files**) of the GAP Reference Manual. If the test suite runs into an error, then please send a message to 'jjm@mcs.st-andrews.ac.uk' including the error message.

References

[AL07] B. Assmann and S. Linton. Using the Mal'cev correspondence for collection in polycyclic groups. *Journal of Algebra*, 2007. 4

[Geb02] V. Gebhardt. Efficient collection in infinite polycyclic groups. *Journal of Symbolic Computation*, 34(3):213–228, 2002. 4

[LGS90] C. R. Leedham-Green and L. H. Soicher. Collection from the left and other strategies. *Journal of Symbolic Computation*, 9:665 – 675, 1990. 4

[Mal51] A. J. Mal'cev. On certain classes of infinite soluble groups. *Mat. Sb.*, 28:567 – 588, 1951. 4

[Seg83] D. Segal. *Polycyclic Groups*. Cambridge University Press, Cambridge, 1983. 9

[VL90] M. Vaughan-Lee. Collection from the left. *Journal of Symbolic Computation*, 9:725–733, 1990. 4

Index