FGA

Free Group Algorithms

A GAP4 Package

by

Christian Sievers

Fachbereich Mathematik und Informatik
Institut für Geometrie

TU Braunschweig

Pockelsstr. 14
D-38106 Braunschweig

January 2004

Contents

1 Introduction

1.1 Overview

This manual describes the FGA (**Free Group Algorithms**) package, a GAP package for computations with finitely generated subgroups of free groups.

This package allows you to test membership and conjugacy, and to compute free generators, the rank, the normalizer, centralizer, and index, where the groups involved are finitely generated subgroups of free groups. In addition, it provides a finite presentation for the automorphism group of a finitely generated free group following [Neu33].

See Chapter 2 for details.

Chapter 3 explains how to install and load the FGA package.

1.2 Implementation and background

The methods which are used work mainly with inverse finite automata, a variant of a concept known from theoretical computer science.

Most of these techniques are described in [Sim94].

In [BMMW00], the connection between finitely generated subgroups of free groups and inverse finite automata is used to transfer results about the space complexity of problems concerning inverse finite automata to analogous results about finitely generated subgroups of free groups.

Some word oriented algorithms in the FGA package use basic facts about free groups that can for example be found in [LS77].

The theoretical background for this implementation is explained in [Sie03].

2 Functionality of the FGA package

This chapter describes methods available from the FGA package.

In the following, let f be a free group created by `FreeGroup(`n`)`, and let u, *u1* and *u2* be finitely generated subgroups of f created by `Group` or `Subgroup`, or computed from some other subgroup of f. Let *elm* be an element of f.

For example:

```
gap> f := FreeGroup( 2 );
<free group on the generators [ f1, f2 ]>
gap> u := Group( f.1^2, f.2^2, f.1*f.2 );
Group([ f1^2, f2^2 ])
gap> u1 := Subgroup( u, [f.1^2, f.1^4*f.2^6] );
Group([ f1^2, f1^4*f2^6 ])
gap> elm := f.1;
f1
gap> u2 := Normalizer( u, elm );
Group([ f1^2 ])
```

2.1 New operations for free groups

These new operations are available for finitely generated subgroups of free groups:

1 ▶ `FreeGeneratorsOfGroup(` u `)` A

returns a list of free generators of the finitely generated free group u.

The elements in this list will form an N-reduced set. In addition to being a free (and thus minimal) generating set for u, this means that whenever *v1*, *v2* and *v3* are elements or inverses of elements of this list, then

- $v1\,v2 \neq 1$ implies $|v1\,v2| \geq \max(|v1|, |v2|)$, and
- $v1\,v2 \neq 1$ and $v2\,v3 \neq 1$ implies $|v1\,v2\,v3| > |v1| - |v2| + |v3|$

hold, where $|\cdot|$ denotes the word length.

2 ▶ `RankOfFreeGroup(` u `)` A
▶ `Rank(` u `)` O

returns the rank of the finitely generated free group u.

2.2 Method installations

This section list operations that are already known to GAP. FGA installs new methods for them so that they can also be used with free groups.

1 ▶ `Normalizer( u1, u2 )` O
▶ `Normalizer( u, elm )` O

The first variant returns the normalizer of the finitely generated subgroup *u2* in *u1*.

The second variant returns the normalizer of $\langle elm \rangle$ in the finitely generated subgroup u (see 37.10.1 in the Reference Manual).

2 ▶ `RepresentativeAction( u, d, e )` O
▶ `IsConjugate( u, d, e )` O

`RepresentativeAction` returns an element $r \in u$, where u is a finitely generated subgroup of a free group, such that $d^r = e$, or fail, if no such r exists. d and e may be elements or subgroups of u.

`IsConjugate` returns a boolean indicating whether such an element r exists.

3 ▶ `Centralizer( u, u2 )` O
▶ `Centralizer( u, elm )` O

returns the centralizer of *u2* or *elm* in the finitely generated subgroup u of a free group.

4 ▶ `Index( u1, u2 )` O
▶ `IndexNC( u1, u2 )` O

return the index of *u2* in *u1*, where *u1* and *u2* are finitely generated subgroups of a free group. The first variant returns fail if *u2* is not a subgroup of *u1*, the second may return anything in this case.

2.3 Automorphism groups of free groups

The FGA package knows how to compute automorphism groups of free groups. This sections repeats the GAP standard methods to obtain them.

1 ▶ `AutomorphismGroup( u )` A

returns the automorphism group of the finitely generated subgroup u of a free group.

Only a few methods will work with this group. But there is way to obtain an isomorphic finitely presented group:

2 ▶ `IsomorphismFpGroup( group )` A

returns an isomorphism of *group* to a finitely presented group. For automorphism groups of free groups, the FGA package implements the presentation of [Neu33]. That finitely presented group itself can then be obtained with the command `Range`.

Here is an example:

```
gap> f := FreeGroup( 2 );
<free group on the generators [ f1, f2 ]>
gap> a := AutomorphismGroup( f );
<group of size infinity with 3 generators>
gap> iso := IsomorphismFpGroup( a );
[ [ f1, f2 ] -> [ f1^-1, f2 ], [ f1, f2 ] -> [ f2, f1 ],
  [ f1, f2 ] -> [ f1*f2, f2 ] ] -> [ O, P, U ]
gap> Range( iso );
<fp group on the generators [ O, P, U ]>
```

3 Installing and loading the FGA package

3.1 Installing the FGA package

The installation of the FGA package follows standard GAP rules. So the standard method is to unzoo the package into the `pkg` directory of your GAP distribution. This will create an `fga` subdirectory.

For other non-standard options please see Chapter 74.1 in the GAP Reference Manual.

3.2 Loading the FGA package

To use the FGA Package you have to request it explicitly. This is done by calling `LoadPackage` like this:

```
gap> LoadPackage("fga");
-------------------------------------------------------------------------------
Loading  FGA 1.0 (Free Group Algorithms)
by Christian Sievers (c.sievers@tu-bs.de).
-------------------------------------------------------------------------------
true
```

The `LoadPackage` command is described in Section 74.2.1 in the GAP Reference Manual.

Bibliography

[BMMW00] J.-C. Birget, S. Margolis, J. Meakin, and P. Weil. PSPACE-complete problems for subgroups of free groups and inverse finite automata. *Theoretical Computer Science*, 242:247–281, 2000.

[LS77] Roger C. Lyndon and Paul E. Schupp. *Combinatorial Group Theory*. 1977.

[Neu33] Bernd Neumann. Die Automorphismengruppe der freien Gruppen. *Math. Annalen*, 107:367–386, 1933.

[Sie03] Christian Sievers. Algorithmen für freie Gruppen. Diplomarbeit, TU Braunschweig, 2003.

[Sim94] C. C. Sims. *Computation with Finitely Presented Groups*. Cambridge University Press, 1994.

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".