LINS

provides an algorithm for computing the normal subgroups of a finitely presented group up to some given index bound.

0.9

15 March 2024

Friedrich Rober

Friedrich Rober
Email: friedrich.rober@rwth-aachen.de

Contents

Chapter 1

Introduction

This chapter serves as an introduction of the package LINS.

1.1 Overview

The package LINS provides an algorithm for computing the normal subgroups of a finitely presented group up to some given index bound.

Moreover it provides an interface for searching in the normal subgroup lattice of a finitely presented group. For example, one can use this interface to search for l normal subgroups of index n.

The algorithm is based on work of David Firth [Fir05]. He implemented that algorithm in the algebra software MAGMA. That implementation in MAGMA has been revised and rewritten to a great extent by Derek Holt.

The current implementation in GAP uses a table of groups that was computed by the code in `createTables.gi`.

1.2 Examples

In this section we present example sessions which demonstrate how to use the main high-level functions provided by LINS.

1.2.1 Example : all normal subgroups up to index n

We compute all normal subgroups in D_{50}, the dihedral group of size 50.

Example

```
gap> G := DihedralGroup(50);
<pc group of size 50 with 3 generators>
gap> L := LowIndexNormalSubs(G, 50);;
gap> IsoTypes := List(L, StructureDescription);
[ "D50", "C25", "C5", "1" ]
```

1.2.2 Example : all normal subgroups of index n

We compute all normal subgroups of index $5^2 = 25$ in C_5^4, the direct product of 4 copies of the cyclic group of order 5:

Example

```
gap> G := ElementaryAbelianGroup(5^4);
<pc group of size 625 with 4 generators>
gap> L := LowIndexNormalSubs(G, 5 ^ 2 : allSubgroups := false);;
gap> IsoTypes := Collected(List(L, StructureDescription));
[ [ "C5 x C5", 806 ] ]
```

1.3 Main Functions

In this section, we include all the main high-level functions provided to the user. For advanced search methods in the lattice of normal subgroups, take a look at Chapter 2.

1.3.1 LowIndexNormalSubs

▷ `LowIndexNormalSubs(`*`G, n: allSubgroups := true`*`)` (operation)

Returns a list of all normal subgroups of *G* with index at most *n*. If the option *allSubgroups* is set to `false`, then only the normal subgroups of *G* with index equal to *n* are returned.

The generic method uses `IsomorphismFpGroup` (**Reference: IsomorphismFpGroup**) to transform *G* into an fp-group and then calls some variant of the low-level function `LowIndexNormalSubgroupsSearch` (2.3.1).

Note that a similar operation `LowIndexNormalSubgroups` (**polycyclic: LowIndexNormalSubgroups**) exists in the package polycyclic. Due to technical incompabilities, those operations could not be unified.

Chapter 2

LINS Interface

This chapter is intended for advanced users. It explains the provided search methods and the interface to the search graph structure `LinsGraph`.

2.1 LINS Graph

All search methods in LINS return a `LinsGraph` encoding a partial normal subgroup lattice of a finitely presented group G. A `LinsGraph` is a graph, where each node is a `LinsNode` that contains a normal subgroup H of G and pointers to the minimal G-normal super/sub-groups of H, i.e. its neighbours in the graph. The directed edges of the graph are therefore encoded directly into the nodes.

2.1.1 List (for a lins graph)

▷ `List(`*gr*`)` (method)

Returns a list of all `LinsNodes` in the graph *gr*.

The nodes are sorted by index in increasing order, e.g. the root node is at the first position. In order to get a list containing only the normal subgroups that the search graph attempted to find, use `ComputedNormalSubgroups` (2.1.2).

2.1.2 ComputedNormalSubgroups

▷ `ComputedNormalSubgroups(`*gr*`)` (attribute)

Returns the normal subgroups that the search graph attempted to find.

If the `ComputedNormalSubgroups` component of the graph is not set, this defaults to a call of `List` (2.1.1).

2.1.3 LinsRoot

▷ `LinsRoot(`*gr*`)` (attribute)

Returns the root node of the graph.

If the search was started in the finitely presented group G, this will return the `LinsNode` that contains G.

2.1.4 IndexBound

▷ `IndexBound(gr)` (attribute)

Returns the index bound for the search in *gr*.

2.1.5 LinsOptions

▷ `LinsOptions(gr)` (attribute)

Returns the search options of the graph *gr*.

2.1.6 IsomorphismFpGroup (for a lins graph)

▷ `IsomorphismFpGroup(gr)` (attribute)

Returns the isomorphism from the original group of the search onto the fp-group contained in the root.

2.2 LINS Node

A `LinsNode` is a part of the search graph structure `LinsGraph` (see 2.1). As such, all methods are with respect to the search graph, where the node is contained in.

2.2.1 Grp (for a lins node)

▷ `Grp(rH)` (method)

Returns the group contained in the node.

2.2.2 Index (for a lins node)

▷ `Index(rH)` (method)

Let G be the group contained in the root node and H be the G-normal subgroup contained in *rH*. Returns the index $[G : H]$.

2.2.3 LinsNodeMinimalSupergroups

▷ `LinsNodeMinimalSupergroups(rH)` (attribute)

Let G be the group contained in the root node and H be the G-normal subgroup contained in *rH*. Returns a list of all `LinsNodes` containing minimal G-normal supergroups of H.

2.2.4 LinsNodeMinimalSubgroups

▷ `LinsNodeMinimalSubgroups(rH)` (attribute)

Let G be the group contained in the root node and H be the G-normal subgroup contained in `rH`. Returns a list of all `LinsNodes` containing minimal G-normal subgroups of H.

2.2.5 LinsNodeSupergroups

▷ `LinsNodeSupergroups(rH)` (operation)

Let G be the group contained in the root node and H be the G-normal subgroup contained in `rH`. Returns a list of all `LinsNodes` containing G-normal supergroups of H.

2.2.6 LinsNodeSubgroups

▷ `LinsNodeSubgroups(rH)` (operation)

Let G be the group contained in the root node and H be the G-normal subgroup contained in `rH`. Returns a list of all `LinsNodes` containing G-normal subgroups of H.

2.3 LINS Search Functions

2.3.1 LowIndexNormalSubgroupsSearch

▷ `LowIndexNormalSubgroupsSearch(G, n[, opts])` (function)

Given a finitely presented group `G` and some index bound `n`, this will start a search in the normal subgroup lattice of `G` up to index `n`.

The optional argument `opts` must be a record containing valid search options (see 3.1).

If the optional argument `opts` is not given, the search will be started with the default options, i.e. it will terminate once all normal subgroups of `G` with index at most `n` are found.

It is possible to call the function with a group `G` that is not an fp-group. The group will be automatically replaced with an fp-group (see `IsomorphismFpGroup` (2.1.6)). **Returns:** `LinsGraph` encoding a partial normal subgroup lattice of `G`

2.3.2 LowIndexNormalSubgroupsSearchForAll

▷ `LowIndexNormalSubgroupsSearchForAll(G, n)` (function)

Given a finitely presented group `G` and some index bound `n`, this will compute all normal subgroups of `G` with index at most `n`.

This is a synonym for calling `LowIndexNormalSubgroupsSearch` (2.3.1) without any options.

It is possible to call the function with a group `G` that is not an fp-group. The group will be automatically replaced with an fp-group (see `IsomorphismFpGroup` (2.1.6)). **Returns:** `LinsGraph` encoding a partial normal subgroup lattice of `G`

2.3.3 LowIndexNormalSubgroupsSearchForIndex

▷ `LowIndexNormalSubgroupsSearchForIndex(G, n, l)` (function)

Given a finitely presented group `G`, some index `n` and `l` being a positive integer or `infinity`, this will attempt to find `l` normal subgroups of `G` with index `n`.

In particular, if `l` is `infinity`, all normal subgroups of `G` with index `n` will be computed.

Furthermore, if `l` is a positive integer and the `ComputedNormalSubgroups` of the graph has less than `l` nodes, then all normal subgroups of `G` with index `n` were computed.

It is possible to call the function with a group `G` that is not an fp-group. The group will be automatically replaced with an fp-group (see `IsomorphismFpGroup` (2.1.6)). **Returns:** `LinsGraph` encoding a partial normal subgroup lattice of `G`

2.4 Examples

In this section we present example sessions which demonstrate how to use the advanced search methods provided by LINS. For this we revise the examples from the introduction as well as include new ones.

2.4.1 Revised Example : all normal subgroups up to index *n*

We compute all normal subgroups in D_{50}, the dihedral group of size 50.

Example

```
gap> G := DihedralGroup(50);
<pc group of size 50 with 3 generators>
```

The search algorithm automatically translates the group into a finitely presented group via a call to `IsomorphismFpGroup`.
The isomorphism is stored inside the lins graph.

Example

```
gap> gr := LowIndexNormalSubgroupsSearchForAll(G, 50);
<lins graph contains 4 normal subgroups up to index 50>
gap> r := LinsRoot(gr);
<lins node of index 1>
gap> H := Grp(r);
<fp group of size 50 on the generators [ F1, F2, F3 ]>
gap> Iso := IsomorphismFpGroup(gr);
[ f1, f2, f3 ] -> [ F1, F2, F3 ]
gap> Source(Iso) = G;
true
gap> Range(Iso) = H;
true
```

In order to get all nodes from the search graph, we need to use `List`. As expected, the algorithm finds D_{50}, C_{25}, C_5 and the trivial group.

Example

```
gap> L := List(gr);
[ <lins node of index 1>, <lins node of index 2>, <lins node of index 10>,
  <lins node of index 50> ]
gap> IsoTypes := List(L, node -> StructureDescription(Grp(node)));
[ "D50", "C25", "C5", "1" ]
```

2.4.2 Revised Example : all normal subgroups of index n

We compute all normal subgroups of index $5^2 = 25$ in C_5^4, the direct product of 4 copies of the cyclic group of order 5:

Example

```
gap> G := ElementaryAbelianGroup(5^4);
<pc group of size 625 with 4 generators>
```

Again, the search algorithm automatically translates the group into a finitely presented group via a call to `IsomorphismFpGroup`.

Example

```
gap> gr := LowIndexNormalSubgroupsSearchForIndex(G, 5 ^ 2, infinity);
<lins graph contains 963 normal subgroups up to index 25>
```

Now we are not interested in all normal subgroups that the search graph considered, but only in those of index 25. Thus we need to use `ComputedNormalSubgroups`. For a prime p, and integers $d, s \in \mathbb{N}$, the number of subgroups of order p^s of an elementary abelian p-group of order p^d is exactly

$$\frac{\left(p^d-1\right)\left(p^d-p\right)\cdots\left(p^d-p^{s-1}\right)}{\left(p^s-1\right)\left(p^s-p\right)\cdots\left(p^s-p^{s-1}\right)}.$$

Thus we expect to find $\frac{(5^4-1)\cdot(5^4-5)}{(5^2-1)\cdot(5^2-5)} = 806$ normal subgroups of index 25.
Furthermore, all subgroups need to be of the isomorphism type C_5^2.

Example

```
gap> L := ComputedNormalSubgroups(gr);;
gap> IsoTypes := Collected(List(L, node -> StructureDescription(Grp(node))));
[ [ "C5 x C5", 806 ] ]
```

2.4.3 Example : a normal subgroup of index n

We compute a normal subgroup of index $3 \cdot 5 = 15$ in $C_3 \times C_3 \times C_4 \times C_5$, a direct product of cyclic groups:

Example

```
gap> G := AbelianGroup([3, 3, 4, 5]);
<pc group of size 180 with 4 generators>
gap> gr := LowIndexNormalSubgroupsSearchForIndex(G, 15, 1);
<lins graph contains 7 normal subgroups up to index 15>
```

We use `ComputedNormalSubgroups` in order to get the normal subgroup of index 15. As expected, the algorithm finds a group of the isomorphism type $C_{12} = C_3 \times C_4$.

Example

```
gap> L := ComputedNormalSubgroups(gr);
[ <lins node of index 15> ]
gap> IsoTypes := List(L, node -> StructureDescription(Grp(node)));
[ "C12" ]
```

Chapter 3

LINS Search

This chapter is for advanced users and those interested in a brief introduction to the mathematical background of LINS.

(NOTE: THE INTERNAL OPTIONS BELOW MIGHT CHANGE DURING FURTHER DEVELOPMENT!)

3.1 LINS Search Options

The method `LowIndexNormalSubgroupsSearch` allows an optional argument *`opts`* which must be a record and modifies the search for the execution of only this single command.

The following components of *`opts`* are supported.

`DoSetParent`
: `true` to set parent of every subgroup to the group *`G`* that is contained in the root of the search graph. It also sets the property `IsNormalInParent` to `true` for every subgroup.

  This should be only disabled for debugging or testing purposes.

`InitGraph`
: a function that takes as an argument a `LinsGraph` *`gr`*. It can popullate the record object *`gr`* via calls like `gr!.NAME = VALUE` in order to initialize certain components that might be needed by other functions later on (see option `DoTerminate`).

`DoCut`
: a function that takes as arguments a `LinsGraph` *`gr`* and a `LinsNode` *`rH`*.

  Returns `true` if subgroups under *`rH`* should not be computed, i.e. the branch will be cut under the node *`rH`*.

`DoTerminate`
: a function that takes as arguments a `LinsGraph` *`gr`* and two `LinsNodes` *`rH`* and *`rK`*. We are currently computing the subgroups under *`rH`* and found the normal subgroup *`rK`*.

  This function may write data to `gr!.ComputedNormalSubgroups‘`. Make sure to initalize this via the option `InitGraph` (for example to an empty list), since it is not bound by default.

  Returns `true` if the search can be terminated.

`UseLIS`

`false` to use the new procedure with `GQuotient` calls in `LINS_FindTQuotients`.

`true` to use the old procedure with a `LowIndexSubgroupsFpGroup` call in `LINS_FindTQuotients`.

`FilterTQuotients`

a function that takes as an argument a `LinsGraph` *gr* and a list of *targets* following the specifications of `LINS_FindTQuotients`.

The value of `targets` in a call via `LowIndexNormalSubgroupsSearch` depends on the value of the option `UseLIS`.

If `UseLIS` is false, we have `targets = LINS_TargetsQuotient`.

If `UseLIS` is true, we have `targets = LINS_TargetsQuotientUseLIS`.

Returns a sublist from *targets*, that will be used by `LINS_FindTQuotients`.

`DoIntersection`

a function that takes as an argument a `LinsGraph` *gr*, two `LinsNodes` *rH* and *rK*, and a positive integer *i*. Let G be the group that is contained in the root of the search graph.

Returns `true` if the intersection U of the groups in *rH* and *rK* with index $[G:U] = i$ should be computed.

`DoPQuotient`

a function that takes as an argument a `LinsGraph` *gr*, a `LinsNode` *rH* and a prime *p*.

Returns `true` if p-quotients under *rH* should be computed for the prime *p*.

`DoPModule`

a function that takes as an argument a `LinsGraph` *gr*, a `LinsNode` *rH*, a prime *p* and a positive integer *i*. Let G be the group that is contained in the root of the search graph.

Returns `true` if the normal subgroup K of index $[G:K] = i$ with elementary abelian p-quotient in *rH* should be computed.

Chapter 4

License

LINS is free software you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. For details, see the file LICENSE distributed as part of this package or see the FSF's own site.

References

[Fir05] David Firth. *An Algorithm to Find Normal Subgroups of a Finitely Presented Group, up to a Given Finite Index*. Ph.d. thesis, University of Warwick, 2005. 3

Index