corefreesub

A GAP Package for calculating the core-free subgroups and their faithful transitive permutation representations

0.3

7 August 2023

Claudio Alexandre Piedade

Manuel Delgado

Claudio Alexandre Piedade
Email: claudio.piedade@fc.up.pt
Homepage: https://www.fc.up.pt/pessoas/claudio.piedade/

Manuel Delgado
Email: mdelgado@fc.up.pt
Homepage: https://cmup.fc.up.pt/cmup/mdelgado/

Copyright

Acknowledgements

The authours wish to thank all the comments, suggestions and issue reporting from users and developers of GAP, both past and future. Both authors were partially supported by CMUP, member of LASI, which is financed by Portuguese national funds through FCT – Fundação para a Ciência e a Tecnologia, I.P., under the project with references UIDB/00144/2020 and UIDP/00144/2020.

Contents

Chapter 1

Introduction

The corefreesub package was created to calculate core-free subgroups of a group, their indexes, and faithful transitive permutation representations.

A core-free subgroup of a group `G` is a subgroup `H` such that

$$\cap_{g\in G} H = \{id_G\}.$$

These subgroups are important since the action of `G` on the cosets of `H` is both transitive and faithful. Hence, this gives us a faithful transitive permutation representation of `G` with degree `n`, where `n` is the index of `H` in `G`.

There are many articles studying faithful permutation representation of groups, such as [Joh71], [EP88], [Sau14] and [EH16]. However the restriction on transitive actions is more recent and there are fewer studies like [FP20],[FP21a],[FP21b] and [FP22].

During C.A. Piedade's PhD thesis, he studied many of these faithful transitive permutation representations of automorphism groups of abstract regular polytopes and hypertopes. It was also during this period that this author noticed the abcense of functions/methods in GAP to compute core-free subgroups of a group. As a consequence, he created many functions to help in his research, resulting in many of the functions and methods implemented in this package.

One of the important tools for studying faithful transitive permutation representations is by using *faithful transitive permutation representation graphs*, which are *Schreier coset graphs*. A *Schreier coset graph* is a graph associated with a group `G`, its generators and a subgroup `H` of `G`. The vertices of the graph are the right cosets of `H` and there is a directed edge (Hx, Hy) with label g if g is a generator of `G` and $Hxg = Hy$. When g is an involution, the two directed edges (Hx, Hy) and (Hy, Hx) are replaced by a single undirected edge $\{Hx, Hy\}$ with label g.

In the corefreesub package, this can achieved by creating graphs as DOT files and using an adaptation of the visualization package developed by M. Delgado et al. [Del17] [DLM05], which can be found on Chapter 4.

1.1 Instalation

To install this package, you can simply copy the folder of corefreesub and its contents into your */pkg* folder inside your GAP instalation folder. This should work for Windows, Ubuntu and MacOS. If you are using GAP.app on MacOS, the corefreesub folder should be copied into your user Library/Preferences/GAP/pkg folder.

This package was tested with GAP version greater or equal to 4.11.

1.2 Testing your instalation

To test your instalation, you can run the function *CF_TESTALL()*. This function will run two sets of tests, one dependent on the documentation of the corefreesub package and another with assertions with groups with bigger size.

If the test runs with no issue, the output should look something similar to the following:

Example

```
gap> CF_TESTALL();
Running list 1 . . .
gap>
```

This tests will also produce two pictures that are supposed to be outputed and open in the user system. If the tests run with no error but they do not output any of the graphs, then it may mean the user might not be able to use this functionality. If so, please report an issue on CoreFreeSub GitHub Issues. Moreover, the Architecture part of the test might vary depending on the Operating System the user might be running.

Chapter 2

Obtaining Core-Free Subgroups

2.1 Core-Free Subgroups

A core-free subgroup is a subgroup in which its (normal) core is trivial.

2.1.1 IsCoreFree

▷ `IsCoreFree(G, H)` (function)

Returns: a boolean

Given a group G and one of its subgroups H, it returns whether H is core-free in G.

Example

```
gap> G := SymmetricGroup(4);; H := Subgroup(G, [(1,3)(2,4)]);;
gap> Core(G,H);
Group(())
gap> IsCoreFree(G,H);
true
gap> H := Subgroup(G, [(1,4)(2,3), (1,3)(2,4)]);;
gap> IsCoreFree(G,H);
false
gap> Core(G,H);# H is a normal subgroup of G, hence it does not have a trivial core
Group([ (1,4)(2,3), (1,3)(2,4) ])
```

2.1.2 CoreFreeConjugacyClassesSubgroups

▷ `CoreFreeConjugacyClassesSubgroups(G)` (function)

Returns: a list

Returns a list of all conjugacy classes of core-free subgroups of G

Example

```
gap> G := SymmetricGroup(4);; dh := DihedralGroup(10);;
gap> CoreFreeConjugacyClassesSubgroups(G);
[ Group( () )^G, Group( [ (1,3)(2,4) ] )^G, Group( [ (3,4) ] )^G,
Group( [ (2,4,3) ] )^G, Group( [ (3,4), (1,2)(3,4) ] )^G,
Group( [ (1,3,2,4), (1,2)(3,4) ] )^G, Group( [ (3,4), (2,4,3) ] )^G ]
gap> CoreFreeConjugacyClassesSubgroups(dh);
[ Group( <identity> of ... )^G, Group( [ f1 ] )^G ]
```

2.1.3 AllCoreFreeSubgroups

▷ `AllCoreFreeSubgroups(G)` (function)

Returns: a list

Returns a list of all core-free subgroups of `G`

Example

```
gap> G := SymmetricGroup(4);; dh := DihedralGroup(10);;
gap> AllCoreFreeSubgroups(G);
[ Group(()), Group([ (1,3)(2,4) ]), Group([ (1,4)(2,3) ]), Group([ (1,2)(3,4) ]),
 Group([ (3,4) ]), Group([ (2,4) ]), Group([ (2,3) ]), Group([ (1,4) ]),
 Group([ (1,3) ]), Group([ (1,2) ]), Group([ (2,4,3) ]), Group([ (1,3,2) ]),
 Group([ (1,3,4) ]), Group([ (1,4,2) ]), Group([ (3,4), (1,2)(3,4) ]),
 Group([ (2,4), (1,3)(2,4) ]), Group([ (2,3), (1,4)(2,3) ]),
 Group([ (1,3,2,4), (1,2)(3,4) ]), Group([ (1,2,3,4), (1,3)(2,4) ]),
 Group([ (1,2,4,3), (1,4)(2,3) ]), Group([ (3,4), (2,4,3) ]),
 Group([ (1,3), (1,3,2) ]), Group([ (1,3), (1,3,4) ]), Group([ (1,4), (1,4,2) ])
]
gap> AllCoreFreeSubgroups(dh);
[ Group([  ]), Group([ f1 ]), Group([ f1*f2^2 ]), Group([ f1*f2^4 ]),
Group([ f1*f2 ]), Group([ f1*f2^3 ]) ]
```

2.2 Degrees of Core-Free subgroups

2.2.1 CoreFreeDegrees

▷ `CoreFreeDegrees(G)` (function)

Returns: a list

Returns a list of all possible degrees of faithful transitive permutation representations of `G`. The degrees of a faithful transitive permutation representation of `G` are the index of its core-free subgroups.

Example

```
gap> G := SymmetricGroup(4);; dh := DihedralGroup(10);;
gap> CoreFreeDegrees(G);
[ 24, 12, 8, 6, 4 ]
gap> CoreFreeDegrees(dh);
[10, 5]
```

Chapter 3

Faithful Transitive Permutation Representations

The action of a group G on the coset space of a subgroup gives us a transitive permutation representation of the group. Whenever the subgroup is core-free, we have that the action of G on the coset space of the subgroup will be faithful. Moreover, the stabilizer of a point on a faithful transitive permutation representation of G will always be a core-free subgroup.

3.1 Obtaining Faithful Transitive Permutation Representations

3.1.1 FaithfulTransitivePermutationRepresentations (for IsGroup)

▷ `FaithfulTransitivePermutationRepresentations(`*`G[, all_ftpr]`*`)` (operation)

Returns: a list

For a finite group *G*, *FaithfulTransitivePermutationRepresentations* returns a list of a faithful transitive permutation representation of *G* for each degree. If *all_ftpr* is true, then it will return a list of all faithful transitive permutation representations.

Example

```
gap> sp := SymplecticGroup(4,2);;
gap> CoreFreeDegrees(sp);
[ 720, 360, 240, 180, 144, 120, 90, 80, 72, 60, 45, 40, 36, 30, 20, 15, 12,
10, 6 ]
gap> ftprs := FaithfulTransitivePermutationRepresentations(sp);;
gap> Size(ftprs);
19
gap> all_ftprs := FaithfulTransitivePermutationRepresentations(sp,true);;
gap> Size(all_ftprs);
54
```

3.2 Faithful Transitive Permutation Representation of Minimal Degree

To complement the already existing functions in GAP *MinimalFaithfulPermutationRepresentation* and *MinimalFaithfulPermutationDegree*, the following functions to retreive the *MinimalFaithfulTransitivePermutationRepresentation* and *MinimalFaithfulTransitivePermutationDegree*.

3.2.1 MinimalFaithfulTransitivePermutationRepresentation (for IsGroup)

▷ MinimalFaithfulTransitivePermutationRepresentation(*G[, all_minimal_ftpr]*) (operation)

Returns: an isomorphism (or a list of isomorphisms)

For a finite group *G*, *MinimalFaithfulTransitivePermutationRepresentation* returns an isomorphism of *G* into the symmetric group of minimal degree acting transitively on its domain. If *all_minimal_ftpr* is set as *true*, then it returns a list of all isomorphisms *G* into the symmetric group of minimal degree.

Example

```
gap> sp := SymplecticGroup(4,2);;
gap> min_ftpr := MinimalFaithfulTransitivePermutationRepresentation(sp);
CompositionMapping( <action epimorphism>, <action isomorphism> )
gap> min_ftpr(sp);
Group([ (1,6,4,3), (1,3)(2,4,6,5) ])
gap> min_ftprs := MinimalFaithfulTransitivePermutationRepresentation(sp,true);
[ CompositionMapping( <action epimorphism>, <action isomorphism> ),
CompositionMapping( <action epimorphism>, <action isomorphism> ) ]
gap> min_ftprs[2](sp);
Group([ (2,3,6,5), (1,3)(2,5,6,4) ])
```

3.2.2 MinimalFaithfulTransitivePermutationDegree

▷ MinimalFaithfulTransitivePermutationDegree(*G*) (function)

Returns: an integer

For a finite group *G*, *MinimalFaithfulTransitivePermutationDegree* returns the least positive integer n such that *G* is isomorphic to a subgroup of the symmetric group of degree n acting transitively on its domain.

Example

```
gap> sp := SymplecticGroup(4,2);; g:=SimpleGroup("PSL",3,5);;
gap> MinimalFaithfulTransitivePermutationDegree(sp);
6
gap> MinimalFaithfulTransitivePermutationDegree(g);
31
```

Chapter 4

Drawing the Faithful Transitive Permutation Representation Graph

4.1 Drawing functions

One of the advantages of Faithful Transitive Permutation Representation Graph are on Groups generated by involutions, such as C-groups. These graphs are very useful in the research of abstract polytopes and hypertopes, mainly called as "Schreier coset graphs" or "CPR graphs" in this area. Here we will give a function that builds this graph given a permutation group generated by involutions, a group and one of its core-free subgroups or by giving an isomorphism of the group into the symmetric group acting faithfully and transitively on its domain. To use Graphviz in order to create the image file, you need to be running GAP on a Linux Environment (Windows Subsystem for Linux is supported), with graphviz installed.

4.1.1 DotFTPRGraph (for IsPermGroup)

▷ DotFTPRGraph(*G*) (operation)
▷ DotFTPRGraph(*G[, generators_name]*) (operation)
▷ DotFTPRGraph(*map*) (operation)
▷ DotFTPRGraph(*map[, generators_name]*) (operation)
▷ DotFTPRGraph(*H, K*) (operation)
▷ DotFTPRGraph(*H, K[, generators_name]*) (operation)

Returns: a graph written in dot

Given a transitive permutation group *G*, a faithful transitive permutation representation of a group *map* or a group *H* and one of its core-free subgroups *K*, the function will output the permutation representation graph written in the language of a Dot file. If given a list of the name of the generators *generators_name*, these will be given to the label of their action on the graph. Otherwise, the labels will be *r0, r1, r2, ...* for the generators *G.1, G.2, G.3, ...*

Example

```
gap> G:= SymmetricGroup(4);;H:= Subgroup(G,[(1,2)]);;K:= Subgroup(G,[(1,2,3)]);;
gap> DotFTPRGraph(G);
"digraph {\n1 -> 2 [label = r1];\n 2 -> 3 [label = r1];\n 3 -> 4 [labe\
l = r1];\n 4 -> 1 [label = r1];\n 1 -> 2 [label = r2,dir=none];\n }\n"
gap> DotFTPRGraph(G,H);
"digraph {\n1 -> 10 [label = r1];\n 2 -> 12 [label = r1];\n 3 -> 11 [l\
```

```
abel = r1];\n 4 -> 8 [label = r1];\n 5 -> 9 [label = r1];\n 6 -> 7 [la\
bel = r1];\n 7 -> 3 [label = r1];\n 8 -> 2 [label = r1];\n 9 -> 1 [lab\
el = r1];\n 10 -> 5 [label = r1];\n 11 -> 6 [label = r1];\n 12 -> 4 [l\
abel = r1];\n 2 -> 3 [label = r2,dir=none];\n 4 -> 7 [label = r2,dir=n\
one];\n 5 -> 8 [label = r2,dir=none];\n 6 -> 9 [label = r2,dir=none];\
\n 10 -> 11 [label = r2,dir=none];\n }\n"
gap> Print(DotFTPRGraph(FactorCosetAction(G,K),["A","B"]));
digraph {
1 -> 3 [label = A];
2 -> 4 [label = A];
3 -> 5 [label = A];
4 -> 6 [label = A];
5 -> 8 [label = A];
6 -> 7 [label = A];
7 -> 2 [label = A];
8 -> 1 [label = A];
1 -> 2 [label = B,dir=none];
3 -> 5 [label = B,dir=none];
4 -> 6 [label = B,dir=none];
7 -> 8 [label = B,dir=none];
}
```

4.1.2 DrawFTPRGraph

▷ DrawFTPRGraph(*arg*) (function)

Returns: an image of the faithful transitive permutation representation graph

This global function takes as input the following arguments:

- *arg* := *dotstring*[, *rec*]
- *arg* := *G*[, *rec*]
- *arg* := *map*[,*rec*]
- *arg* := *H*,*K*[,*rec*]

Given a string of a graph in dot *dotstring*, this function will output and show an image of the graph. Alternatively, a transitive permutation group *G*, a faithful transitive permutation representation of a group *map* or a group *H* and one of its core-free subgroups *K*, can be given. This will use *DotFTPRGraph* to calculate the *dotstring*. Moreover, extra parameters can be given as a form of a record *rec*. The set of parameters that can be given inside a record can be found below, with information regarding their effect:

- *layout* - (a string) the engine that is used to calculate the layout of the vertices and edges of graph to output in the dot image (not used for TeX output). The supported layouts are "dot", "neato", "twopi", "circo", "fdp", "sfdp", "patchwork", "osage". By default "neato" is used.
- *directory* - (a string) the name of the folder where the dot and image files are created. By default, a temporary folder of GAP is used.
- *path* - (a string) the path where the directory will be created. If the directory is not specified, a folder "tmp.viz" will be created at the determined path. If no path is given, the default path is "~/". If no path nor directory is given, it will be saved in a temporary path of GAP.

- *file* - (a string) the name of the dot and image files created. By default, the name will be "vizpicture".
- *filetype* - (a string) the image file type that will be created. By default, the filetype will be "pdf".
- *viewer* - (a string) the name of the visualizer used to open the image. The supported ones are "evince","xpdf","xdg-open","okular", "gv", "open" (for the different System Architectures).
- *tikz* - (a boolean) if true, then the function will produce a TeX file, compile it to pdf and open.
- *viewtexfile* - (a boolean) if true, then the function will produce a TeX file and return the text of the Tex file (but it will not compile and open any pdf from the TeX file).

Example

```
gap> G:= SymmetricGroup(4);;H:= Subgroup(G,[(1,2)]);;K:= Subgroup(G,[(1,2,3)]);;
gap> DrawFTPRGraph(G);
gap> texfile := DrawFTPRGraph(G,H,rec(viewtexfile := true));;
gap> Print(texfile{[1..152]});
\documentclass{article}
\usepackage[x11names, svgnames, rgb]{xcolor}
\usepackage[utf8]{inputenc}
\usepackage{tikz}
\usetikzlibrary{snakes,arrows,shapes}
gap> SetInfoLevel(InfoDrawFTPR,2);
gap> DrawFTPRGraph(FactorCosetAction(G,K),rec(directory:="myfolder",layout:="fdp"));
#I  The directory used is: ~/myfolder/
```

4.1.3 TeXFTPRGraph

▷ TeXFTPRGraph(*arg*) (function)

Returns: an image of the faithful transitive permutation representation graph

The same as *DrawFTPRGraph* with the parameter *viewtexfile := true*.

4.1.4 DrawTeXFTPRGraph

▷ DrawTeXFTPRGraph(*arg*) (function)

Returns: an image of the faithful transitive permutation representation graph

The same as *DrawFTPRGraph* with the parameter *tikz := true*.

4.2 Information Level of Drawing Functions

We can set the amount of verbosity of the functions "DrawFTPRGraph", "TeXFTPRGraph" and "DrawTeXFTPRGraph", which can be controlled by the *InfoDrawFTPR* variable. As of right now, there are only two levels of the *InfoDrawFTPR* and, by default, the level is set as 1. To change to level 2, you can do the following:

Example

```
gap> SetInfoLevel(InfoDrawFTPR,2);
```

Particularly, *InfoDrawFTPR* in level 2 will give information regarding the location in which the files are being created and processed.

References

[Del17] Manuel Delgado. IntPic : a GAP package for drawing integers, 9, 2017. 4

[DLM05] Manuel Delgado, S Linton, and J Morais. automata: a GAP package for finite automata, Version 1.05, 2005, 2005. 4

[EH16] David Easdown and Michael Hendriksen. Minimal permutation representations of semidirect products of groups. *Journal of Group Theory*, 19(6):1017–1048, November 2016. 4

[EP88] David Easdown and Cheryl E. Praeger. On minimal faithful permutation representations of finite groups. *Bulletin of the Australian Mathematical Society*, 38(2):207–220, October 1988. 4

[FP20] M. E. Fernandes and C. A. Piedade. Faithful permutation representations of toroidal regular maps. *Journal of Algebraic Combinatorics*, 52(3):317–337, 2020. 4

[FP21a] M. E. Fernandes and C. A. Piedade. Correction to "faithful permutation representations of toroidal regular maps". *Journal of Algebraic Combinatorics*, 54:733–738, 2021. 4

[FP21b] M. E. Fernandes and C. A. Piedade. The degrees of toroidal regular proper hypermaps. *The Art of Discrete and Applied Mathematics*, 4(3):P3–13, 2021. 4

[FP22] M. E. Fernandes and C. A. Piedade. The degrees of regular polytopes of type [4, 4, 4]. *SIAM Journal on Discrete Mathematics*, 36(2):1143–1155, 2022. 4

[Joh71] D. L. Johnson. Minimal Permutation Representations of Finite Groups. *American Journal of Mathematics*, 93(4):857, October 1971. 4

[Sau14] Neil Saunders. Minimal faithful permutation degrees for irreducible Coxeter groups and binary polyhedral groups. *Journal of Group Theory*, 17(5):805–832, September 2014. 4

Index