The Joy of GAP Packages

Leonard H. Soicher
Queen Mary, University of London

Groups in Galway 2009

GAP

- is an internationally developed system for Computational Group Theory and related areas;
- is Open Source, and freely available from `www.gap-system.org`
- provides the GAP programming language and a library of thousands of functions written in this language;
- provides large data libraries of mathematical objects;
- is used in research and teaching for studying groups and their representations, rings, vector spaces, algebras, combinatorial structures, and more.

GAP Packages

- are structured user-contributions to GAP;
- provide many useful extensions to GAP;
- integrate smoothly with the GAP system and its help system;
- are distributed with GAP, but package authors get full credit and remain responsible for the maintenance of their packages;
- may be “deposited” and/or submitted for formal refereeing.

GAP Packages include

- GAP interfaces to other mathematical software systems (such as singular) and to standalone programs (such as ACE);
- packages for research in specialised areas of group theory and algebra (such as Polycyclic and HAP);
- databases of group-related objects (such as SmallGroups, CTblLib, and Tables of Marks);
- tools for graphics (such as XGAP) and documentation (GAPDoc);
- extensions of GAP into areas making use of groups, such as graph theory (GRAPE), coding theory (guava), and design theory (DESIGN and RDS);
- significant contributions from researchers at NUI Galway.

GAP Package refereeing

- is run by the GAP Council, an international body of mathematicians and computer scientists engaged in a broad spectrum of Computational Group Theory.

- A successfully refereed package obtains the official status of "accepted", as a mark of quality and so that package authors can obtain credit as they would for a journal publication.

- Information on structuring, writing and submitting a GAP package is available from the GAP website.

- Package submissions for refereeing, as well as informal queries, should be sent to `council@gap-system.org`

- Please also talk to me here in Galway.

The first GAP Package: GRAPE

- GRAPE is a package for computing with graphs together with groups acting on them.

- It uses a group of automorphisms associated to a graph to store that graph compactly and to speed up computations with that graph.

- GRAPE also provides a seamless interface to Brendan McKay's nauty programs for computing automorphism groups of graphs and testing graph isomorphism.

Simple GRAPE example

```
gap> LoadPackage("grape");

Loading  GRAPE 4.3
(GRaph Algorithms using PErmutation groups),
by L.H.Soicher@qmul.ac.uk.

true
gap> J:=JohnsonGraph(4,2);
rec( isGraph := true, order := 6,
  group := Group([ (1,4,6,3)(2,5), (2,4)(3,5) ]),
  schreierVector := [ -1, 2, 1, 1, 1, 1 ],
  adjacencies := [ [ 2, 3, 4, 5 ] ],
  representatives := [ 1 ],
  names := [ [ 1, 2 ], [ 1, 3 ], [ 1, 4 ],
     [ 2, 3 ], [ 2, 4 ], [ 3, 4 ] ],
  isSimple := true )
gap> Size(J.group);
24
gap> G:=AutomorphismGroup(J);
Group([ (3,4), (2,3)(4,5), (1,2)(5,6) ])
gap> Size(G);
48
gap> GlobalParameters(J);
[ [ 0, 0, 4 ], [ 1, 2, 1 ], [ 4, 0, 0 ] ]
```

```
gap> D:=BipartiteDouble(J);
rec( isGraph := true, order := 12,
  group := Group([
      (1,4,6,3)(2,5)(7,10,12,9)(8,11),
      (2,4)(3,5)(8,10)(9,11),
      (1,7)(2,8)(3,9)(4,10)(5,11)(6,12) ]),
  schreierVector := [ -1, 2, 1, 1, 1, 1, 3,
       3, 3, 3, 3, 3 ],
  adjacencies := [ [ 8, 9, 10, 11 ] ],
  representatives := [ 1 ],
  isSimple := true,
  names :=
      [ [ [ 1, 2 ], "+" ], [ [ 1, 3 ], "+" ],
      [ [ 1, 4 ], "+" ], [ [ 2, 3 ], "+" ],
      [ [ 2, 4 ], "+" ], [ [ 3, 4 ], "+" ],
      [ [ 1, 2 ], "-" ], [ [ 1, 3 ], "-" ],
      [ [ 1, 4 ], "-" ], [ [ 2, 3 ], "-" ],
      [ [ 2, 4 ], "-" ], [ [ 3, 4 ], "-" ] ] )
gap> GlobalParameters(D);
[ [ 0, 0, 4 ], [ 1, 0, 3 ], [ -1, 0, -1 ],
  [ 4, 0, 0 ] ]
```

The DESIGN Package

- is for constructing, classifying, partitioning and studying block designs;

- builds heavily on GRAPE, especially its powerful generalised clique classifier with respect to a group of automorphisms;

- applies graph automorphism group computation and graph isomorphism testing in GRAPE to provide those operations for block designs;

- is very general and flexible, allowing for the classification and study of many types of designs (see `designtheory.org`);

- is used by combinatorialists, group theorists, and statisticians.

Simple DESIGN example

```
gap> LoadPackage("design");

Loading  GRAPE 4.3
(GRaph Algorithms using PErmutation groups),
by L.H.Soicher@qmul.ac.uk.

------------------------------------------------
Loading  DESIGN 1.3 (The Design Package for GAP)
by Leonard H. Soicher
(http://www.maths.qmul.ac.uk/~leonard/).
------------------------------------------------
true
gap> Runtime(); # in milliseconds
4409
gap> D:=BlockDesigns(rec( v:=12, blockSizes:=[6],
>  tSubsetStructure:=rec(t:=5, lambdas:=[1])));;
gap> Runtime(); # in milliseconds
8276
gap> Length(D);
1
gap> AllTDesignLambdas(D[1]);
[ 132, 66, 30, 12, 4, 1 ]
gap> Size(AutomorphismGroup(D[1]));
95040
```

Why write a GAP Package?

- You can provide high quality, specialised algorithms and software for your particular area of research. [Start with a particular research focus and provide more general functionality around that.]

- You get to make full use of the GAP system, language, functions, data types, documentation system, user/developer support, international reputation, and distribution.

- Writing a GAP package provides an environment, structure and discipline for you to provide useful, well-structured, general, well-documented and tested software, very useful to you (over many years) as well as others.

- You get to contribute to the worldwide community of GAP users.

- You get full credit for your work – it can be referenced like a paper.

- It's enjoyable and very satisfying!