

Abstracts

Desargues: A Finite Geometry Package

John Bamberg

I will speak about a finite geometry package for GAP, which is a collaborative effort of Anton Betten, Jan De Beule, Maska Law, Max Neunhöffer, Michael Pauley, Sven Reichard, and myself. At the moment, the package chiefly deals with finite projective and polar spaces, but is sufficiently well structured and versatile so that it can be readily extended. I will give a short review of what finite geometry is and what problems can be assisted by the use of computers, and give a brief summary of the functionality of “Desargues”.

Bad Programming in GAP

Thomas Breuer

The GAP system provides an increasing variety of functions and objects. Does it happen that inefficient usage (and extensions) of the system are caused by the availability of these tools? And is it possible that using GAP might cause a worse understanding of the underlying theory?

Nonassociative Algebras

Willem de Graaf

Some details of a new implementation in GAP of nonassociative algebras will be discussed. The basic ingredient is a tree-like structure that encodes a monomial in such an algebra. Applications to finitely presented Lie rings (e.g., those that satisfy an Engel condition) and alternative algebras will be discussed.

Investigating p -Groups by Coclass with GAP

Heiko Dietrich and Dörte Feichtenschlager (joint with Bettina Eick)

We show how the computer algebra system GAP can be applied in the investigation of p -groups by coclass, especially for constructing parts of coclass trees. We outline various experimental results which underpin many of the more recent results in coclass theory and lead to new conjectures, e.g. concerning cohomology rings for 2-groups, and Schur multipliers and automorphism groups of 5-groups of maximal class.

Package Organisation

Andreas Distler

Every user is invited to contribute to the GAP system. The most common form of contribution is a package which can easily be used with the main system by other users. For their convenience the GAP group offers to deposit contributions on the official web

pages. In addition a package can undergo a refereeing process to increase the recognition of the work and emphasise its scientific relevance. There are currently 24 deposited and 44 accepted packages.

A package has to fulfil certain standards regarding its format and appearance. Some aspects are more regulated to fit in well with the main part of GAP, like the documentation, others are up to the authors, like the copyright of their package.

One of the reasons for the standardisation is to make the redistribution of all packages a manageable task. The process to update new versions of packages and the according web pages is partially automated. Nevertheless it includes human intervention as well as communication with the authors.

In this session I shall provide information about the procedure of an update, which parts of a package have what importance for it. I will clarify what distinguishes it from a refereeing process, which steps are part of the update and which are not. Hopefully this talk will give a basis for discussing possible improvements and wishes from the authors.

A Perturbation Lemma of CTC Wall

Graham Ellis

I'll begin by explaining a perturbation technique of CTC Wall and how a prototype GAP implementation has been used to reprove some interesting calculations in group cohomology. The main aim of the talk will be to explain why a more general implementation would enhance the potential of Singular, Cocoa and Polymake software for computing free resolutions of modules.

A Nilpotent Quotient Algorithm for L -presented Groups

René Hartung (joint with Bettina Eick)

We describe a nilpotent quotient algorithm for finitely L -presented groups as implemented in the new NQL package. That is an algorithm which takes as input a finitely L -presented group G and a positive integer c . It computes a polycyclic presentation for the lower central series quotient $G/\gamma_{c+1}(G)$ of G . As finite presentations can be considered as a special type of finite L -presentations, our algorithm also applies to finitely presented groups. It coincides with the nilpotent quotient algorithm by Nickel in this special case. The talk also includes a brief introduction to L -presented groups and exhibits some well known examples such as the Grigorchuk Group. These and other examples arise from groups acting on rooted trees and have connections to many branches of mathematics, e.g. Generalized Burnside Problem, random walks on graphs, classification of finite-rank Lie-Algebras, etc.

Coding Theory and GUAVA

David Joyner

This talk will explain roughly what GUAVA can do, using a few detailed examples from coding theory as a guide. Group-theoretical connections will be stressed.

An Overview of SAGE

David Joyner

This will discuss very briefly what SAGE can do and how it does it. Connections with GAP will be given.

Computing in a Class of Infinite Permutation Groups

Stefan Kohl

For computing in finite permutation groups, many efficient algorithms are known, and there are not many recent developments in this area.

Computing in infinite permutation groups in general is not feasible, as arbitrary permutations of infinite sets cannot be described by finite amounts of data.

This talk presents a class of permutation groups over the integers, which are accessible to computational methods.

Computational investigations of these groups have led to the discovery of new infinite simple groups.

Perhaps the nicest of them is the group $CT(\mathbb{Z})$, which is generated by the set of all *class transpositions* – given disjoint residue classes $r_1(m_1)$ and $r_2(m_2)$ of \mathbb{Z} , the corresponding *class transposition* is the permutation which interchanges $r_1 + km_1$ and $r_2 + km_2$ for each integer k and which fixes all other points.

The corresponding algorithms and methods are implemented in the GAP package RCWA (‘Residue-Class-Wise Affine Groups’), which is available at <http://www.gap-system.org/Packages/rcwa.html>.

Symbolic Computation Software Composability Protocol in GAP

Alexander Konovalov (joint with Steve Linton)

Modern symbolic computations increasingly require infrastructure for combining capabilities available in several different systems and implementing parallel algorithms. Moreover, there are growing numbers of symbolic computation resources, such as databases or specialized software which could be made available as Web services or Grid services accessible over the Internet.

The EU Framework VI project RII3-CT-2005-026133 **SCIENCE** – Symbolic Computation Infrastructure in Europe (<http://www.symbolic-computation.org/>) addresses these needs of the symbolic computation user community.

In the direction of the software composability, on the first step we designed the *Symbolic Computation Software Composability Protocol (SCSCP)* [1] by which a computer algebra system (CAS) may offer services and a client may employ them. We envisage clients for this protocol including:

- A Web server which passes on the same services as Web services using SOAP/HTTP protocols to a variety of possible clients;
- Grid middleware;
- Another instance of the same CAS (in a parallel computing context);

- Another CAS running on the same computer system or remotely.

All messages in the protocol are represented as OpenMath objects [2], using the new Content Dictionary `cascall1` developed for this purpose [3].

The protocol determines the format of the remote procedure calls and returned results, including options to specify, for example, runtime and memory requirements, and other directives. Moreover, as well as transmission of actual mathematical objects, we support transmission of references, which can be used in subsequent requests. For example, we envisage the possibility to construct and manipulate large mathematical objects on remote grids, while there is no need to send anything over the Internet apart from the properties the user is interested in.

We already have a prototype implementation of the Symbolic Computation Software Composability Protocol for the computational algebra system GAP in the form of the GAP package, and in the nearest future we expect its implementations for KANT/KASH, Maple and MuPAD systems. This will constitute a basis for the next stages of the project, when we will extend the support of the OpenMath functionality in all participating systems and develop tools for deploying Web and Grid services based on these systems.

- [1] A. Konovalov, S. Linton. *Symbolic Computation Software Composability Protocol Specification*. CIRCA preprint 2007/5, University of St Andrews (<http://www-circa.mcs.st-and.ac.uk/pre-prints.html>).
- [2] *OpenMath* (<http://www.openmath.org>).
- [3] D. Roozmond. *OpenMath Content Dictionary cascall1*. (<http://www.win.tue.nl/SCIENCE/cds/cascall1.html>).

Exception and Error Handling

Steve Linton

Documenting GAP Code with GAPDoc

Frank Lübeck

I will shortly explain the idea and mention some features of the GAPDoc package for writing GAP documentation. Of course, during the workshop I may explain more details if there is interest, and I will answer more individual questions.

Enumerating Big Orbits and an Application: B acting on the Cosets of Fi_{23}

Jürgen Müller (joint with Max Neunhöffer and Robert A. Wilson)

We describe a novel technique to handle big permutation domains for large groups. It is applied to the multiplicity-free action of the sporadic simple Baby Monster group on the cosets of its maximal subgroup Fi_{23} to determine the character table of the associated endomorphism ring.

The GAP Package LOOPS – Computing with Quasigroups and Loops in GAP

Gábor P. Nagy (University of Szeged, HU, and University of Würzburg) (joint with Petr Vojtěchovský (University of Denver, USA))

The set Q endowed with a binary operation “ $*$ ” is a *quasigroup* if the equation $x * y = z$ has a unique solution whenever two of the three unknowns are given. *Loops* are quasigroups with a unit element. The maps $L_a, R_a : Q \rightarrow Q$, $L_a(x) = a * x$, $R_a(x) = x * a$ are the *left* and *right multiplication* maps of Q . The respective groups generated by multiplication maps are the *left*, *right* and *full multiplication groups* of the quasigroup Q . Loops with certain forms of weak associativity are extensively studied in different branches of mathematics.

In the GAP Workshop 2007, I will present our GAP package LOOPS. The implementation of the package started in 2002. Until now, we have many basic operation and attributes for loops and partially for quasigroups; most part of these are generalizations of the respective notions from group theory, *e.g.* `Center`, `DerivedSubloop`, `IsomorphismLoops`, etc. We represent loops and quasigroup by their Cayley table. The methods make heavy use of the multiplication groups. The present implementation can efficiently handle loops up to order 1000. The package also contains libraries of small loops.

In the second part of the talk, I would like to present some new ideas and new difficulties. Our aim is extend the package for dealing with loops of order up to 10000.

Matrix Group Recognition in GAP

Max Neunhöffer

This talk gives an overview over the current state of the implementation of matrix group recognition algorithms in GAP and possible future plans.

A matrix group $G < \text{GL}_n(q)$ (over a finite field \mathbb{F}_q), given by k generators, is called “recognised constructively” if one has determined the order $|G|$ and a procedure that decides for any given invertible matrix, whether or not it is contained in G , and if so, expresses it explicitly as product in the generators. The matrix group recognition project aims to develop an algorithm that recognises a given group constructively and for which it is proven, that its runtime is bounded from above by a polynomial in n , k and $\log(q)$.

A New Programmer’s Interface for Vectors and Matrices

Max Neunhöffer

Traditionally in GAP, a vector is a list and a matrix is a list of lists. This is convenient, but poses some difficulties. In particular, the filter `IsMatrix` and method selection for matrices in general do not work satisfactorily.

To overcome these problems, I recently proposed to wrap vectors and matrices into “proper” objects remembering their type and to define a new programmer’s interface to vectors and matrices, which is inspired by the behaviour of lists of lists but which allows to develop code that retains the wrapping around the given objects and automatically wrap objects that are created anew.

I will explain the problems and solutions in detail and hope to start a lively discussion by this talk.

The Linear Matrix Problems and the Determination of p -Groups

Olga Pyliavska

Let M be a set of matrices and F a set of linear transformations which can be applied to M . The problem to find a canonical form of M with respect to F is called a matrix problem. The simplest examples of a matrix problem are the problems to find a Gauß form or Jordan normal form of a matrix. Various classification problems from different fields of mathematics can be reduced to a matrix problem. In my talk I show how the problem of determination of p -groups up to isomorphism can be reformulated in the term of matrices.

Resolutions for Bieberbach Groups Using GAP and polymake

Marc Röder

We present a geometric way of calculating free resolutions for torsion-free crystallographic groups. It is based on calculating a fundamental domain using the geometry software *polymake* (from within GAP via an interface package).

The main advantage of this approach is that we get a resolution which has only finitely many non-trivial terms (all of which are calculated).

Also, the fundamental domain can be used to visualize the manifold associated with a given Bieberbach group.

On the Genus of a p -Group

Siddhartha Sarkar

Let G be a finite group. A number $g \geq 2$ is called an admissible genus of G if G acts on a compact Riemann Surface of genus g preserving orientation. The list of all admissible genus of G is called a genus spectrum of G . Since genus spectrum of a given group embeds in an arithmetic progression we scale it by the common difference and call it a reduced genus spectrum. The reduced genus spectrum of a given group is eventually everything. Hence a natural question is to understand the finite gap arise in this reduced spectrum. I would like to discuss the recent results on genus spectrum of p -groups of exponent p and p -groups of maximal class.

Extensions of PcGroups by RWSGroups

Jack Schmidt

A package for computing extensions of pc-groups by groups given by a confluent rewriting system is in preparation. The GAP implementation was initially difficult, but the new linear algebra proposal and implementation by Max Neunhoeffler greatly assisted. The GAP implementation also allowed a novel use of collectors to mimic linear combinations of collectors. Current implementation bottlenecks are in the rewriting process, but could benefit from those with expertise in collectors. Future work will involve providing fundamental algorithms for groups in this data representation.

LieAlgDB: A Database of Lie Algebras

Csaba Schneider

I describe the new GAP package LieAlgDB (jointly written with Willem de Graaf) that is designed to access and to work with several classifications of Lie algebras. In the mathematical literature many classifications of Lie algebras have been published. However, working with these classifications from paper is not always easy. This package aims at making a few classifications of small-dimensional Lie algebras that have appeared in recent years more accessible. In the talk I will focus on both the theoretical and the implementational details.

The CHDA Packages

Chris D. Wensley

The CHDA packages developed at Bangor are:

- **XMod** – crossed modules and cat1-groups (with Murat Alp),
- **Gpd** – groupoids, group graphs and groupoid graphs (with Emma Moore),
- **Kan** – double coset rewriting systems for fp-groups (with Anne Heyworth),
- **IdRel** – logged rewriting and identities among relators (with Anne Heyworth).

This talk describes recent and planned developments for **XMod** and **Gpd**, including an implementation of crossed modules over groupoids.