GAP – System development for Computational Algebra

Alexander Hulpke

Department of Mathematics

Colorado State University

Fort Collins, CO, 80523

hulpke@math.colostate.edu

http://www.math.colostate.edu/~hulpke

Parts

1. What is GAP (`http://www.gap-system.org`)
2. Development model and -lessons
3. How can one get involved

Needless to say this are all my personal opinions and I don't neccesarily speak for every contributor to the system.

What is GAP

GAP is a *free*, *open* system for computational discrete mathematics, in particular group theory.

free GAP is can be downloaded from `www.gap-system.org`.

free GAP (as of version 4.3) is released under the GPL.

open Complete Sources available.

open Mechanism for third-party contributions.

GAP runs on (almost) every platform.

We estimate to have 1000-2000 users world-wide.

Technical support is offered at `gap-trouble@dcs.st-and.ac.uk`, a discussion list exists at `gap-forum.dcs.st-and.ac.uk`.

Features

Comparatively small C-kernel provides (modeled on MAPLE)

- Basic data types
- Programming language (also called GAP), Memory managment
- User environment

Library (written in the GAP language) that provides most of the functionality.

Data library (the largest part in terms of disk space).

Packages. Many of them contributed from outside the GAP developer group.

Setup is well-suited for algebra/combinatorics.

Further features

- An extensible type system that reflects:
  - Mathematical identity (*matrix group over GF(9)*)
  - Internal representation (*Univariate polynomial given by coefficiens*)
  - Acquired knowledge (*found the group is solvable nonabelian*)

  Methods can be selected automatically based on the types of all arguments.
- Convenient list routines and memory managment.
- Fast routines for compact matrices/vectors over finite fields.
- Many "standard" routines (factorizations, combinatorics, …)

Differences to Maple &c.

No analysis/calculus functionality (would be lots of work to add).

Very rudimentary floating point routines.

There is no support for general “terms" or “expressions".

No “unbound" variables.

All expressions are evaluated and are usually in a “normal form".

Comparatively little emphasis on graphical user interface.

Design philosophy

GAP and the design and implementation of new algorithms is part of our research as mathematicians.

- The code is open and free.
- Principal algorithms are state-of-the-art.
- The system is able to tackle research-level problems.
- Time critical routines are in the kernel and thus fast.
- The interpreted language provides a convenient programming environment.
- We do not provide complete functionality for *every perceivable* task but tools with which these tasks can be done.
- The system does not try to be “intelligent”

Interfacing

It is impossible to have one system do every task.

There are very specialized/optimized programs for particular tasks.

Thus there is need to transfer information between different systems

It seems tempting to design a *"grand scheme"* for this, but general problems will hold up work.

It seems better to start with more simple interfaces that are actually usable.

Very specialized systems often anyhow need a specialized interface.

Producing a rudimentary (no documentation) interface typically takes just a few hours if the function calls are clear.

Development Model

GAP started as a master's project of 4 students. It grew initially with further masters and PhD theses.

(To some extent this mimics on the setup of a German university.)

We now have about 10-15 developers (varying degree of involvement).

Most are now roughly at the start of an academic career (Assistant Professor level).

We hope that in time we will be able again to get more students involved.

Funding is as usual for academic research.

Taking on tasks is basically voluntary.

Packages give possibility for contributions.

Keeping it together

Developers are by now spread over the Western hemisphere.

Communication is mainly via email. However people know each other and meet in smaller groups if there is opportunity.

We started to slightly formalize peoples areas of work/responsibility.

CVS used for code coordination.

Required setup for developers is given by standard Unix. (No special development environment.)

Advantages

System adapted well to academic environment.

Contributors are knowledgeable in mathematics.

Students can contribute to research-level project and their contribution gets used.

It is possible to write code without knowing the intricacies of C or dealing with memory managment.

Authors are still around to maintain the system.

Ability to develop a professional system without charging money for it:

It Works

Problems

There is desirable functionality that is *tedious* or *difficult* to implement well, but for which implementation does not qualify as “research" in mathematics.

(Matrix arithmetic over finite fields. I/O routines. Arbitrary precision floats. More generally: published “standard algorithms".)

It is often hard to find people who can implement such routines,are interested in doing it, and will be available in 3 years if maintainance is needed.

Funding agencies are not eager to support “infrastructure" work.

System might be too sophisticated for some casual users.

Other issues

Some problems are hard to solve via email.

"Periodic" patterns of requests/available time.

Academia does not provide a "program design" background.

- Programming Style of various people.
- Maintainability vs. Features/Speed.
- We found a few times that we made bad decisions that were hard to reverse.

Do not underestimate the amount of work it takes to get a usable system.

Contributions

Packages are external contributions to GAP that provide new functionality.

(One can also contribute to the core but often a package requires less coordination.)

It is possible to

- Add kernel functionality
- Interface with external binaries
- Autoload seamlessly

We have set up a system for *refereeing* such contributions. The hope is to get eventually similar credit as for a paper.

There are currently about 15 refereed and 10 unrefereed packages.

Contributions are often very research oriented:

- “State-of-the-art" functionality.
- Functionality might be in a limited area.

You get contributions in what the contributors are interested in, not necessarily what you would need most.

Who uses GAP

- The Developers
- Researchers in (Computational) group theory
- Researchers in other disciplines that use group theory (Mathematics, Physics, Chemistry, Computer Science)
- Amateurs
- Students (Some textbooks now involve GAP exercises)

The "average" user is not a computer expert.

Support Tasks

Work that makes the program usable to other people is at least as much as getting the algorithms working.

Maintainance

- Correcting bugs.
- Adding/Changing functionality for user tasks

Wrapping new releases and releasing bugfixes

User Support

- Documentation
- “Helpline” email.

Frequent Requests

Naïve functionality for teaching or small tasks.

RTFM

- At least 60% of all questions could have been answered by consulting existing documentation.
- But what if the documentation is bad or nonexistent?

Windows look-and-feel

We have a text-based interface (using `cygwin`).

(The users who most want a better interface would have difficulties installing `rxvt` or similar. But I have been unable to install X11 under `cygwin`.)

Most developers do not use Windows

Easier Installation

Brute-force calculations Users try to apply algorithms in hopeless situations.

(Often mathematical insight or knowledge of the algorithm gives feasible shortcuts.)

Simple programming Inertia stops some users from trying to write small routines.

We currently maintain

- CVS server
- FTP server
- Web server
- Mailing list

Why not use Sourceforge/Program XYZ

Some of the setup is “old" (dates back to about 1990).

“Current" technique has switched several times.

Demand for particular “bespoke" features.

Gain/Cost of switch often is initially very small.

Changing a setup can be notable work.

What happens if support for a service suddenly vanishes?

What would be needed to extend the user community

- More “How-To” documentation.
- Better Windows interface
- Better Floating Point Arithmetic.
- Extended functionality