GAP: STATUS REPORT AND SOME POSSIBLE DIR

Steve Linton

Division of Computer Science, St. Andrews

July 7, 1997

Overview

1. Status report – GAP 3.4.4, handover, GAP 4 releases
2. Quick overview of GAP 4
3. Some thoughts on longer term directions

STATUS REPORT

GAP 3.4.4

- Released April from St Andrews, Last full release of GAP3 (p
  - Bug fixes
  - Better Tietze routines
  - Better matrix group handling over small fields
  - Some improvements in algorithms (interfaces unchanged)
  - HTML manual (now 1600 pages)
  - New data libraries: groups order < 1000, transitive groups
  - New share packages: autag, Chevie, CrystGAP, glissando, matrix, pcqa, specht, xmod
  - Updated share packages: grape, guava, anupq

Transfer to St Andrews

- St Andrews role has been increasing – 3.4.4, Web site, AH, W
(occasionally) work on GAP4
- Formal handover (and transfer of e-mail addresses) in the nex
- System guaranteed to remain free and open
- Secured some funding for people to work on GAP (2 jobs adv
- A move from an Aachen-based project with international involvement to an international project coordinated at St A

IE: we hope to persuade you, and others world-wide, to do a lo
We are trying to set up ways to help, and to acknowledge the
involved.

GAP

- The GAP Council is established to encourage GAP developme
us in touch with GAP users world-wide.

Steve Linton

GAP 4

- Developed in Aachen (mostly) over the last two-three years
- Many new features – system and mathematical
- Alpha test releases (weekly snapshots of our development vers
now
- Beta test release (with some documentation) in next few week
- Full release early to mid 1998

OVERVIEW OF GAP 4

What's Not Changed

- For interactive use, or simple programming, GAP 4 looks a lot
- Some commands have changed name
- Some things will be done much more efficiently

The GAP 4 Kernel

- Rebuilt from the ground up – Martin Schönert, Frank Celler
- More efficient memory manager
- Easier to extend
- 64 bit clean
- Faster function calling
- Save/load workspace
- Fast vector arithmetic over finite fields
- Streams
  - More sophisticated file handling
  - More flexible communication with other processes

The GAP Compiler

- Ferenc Rakoczi, MS, FC
- Compiles GAP to human-readable C
- C can be compiled and loaded dynamically (UNIX only) or cc kernel
- Compiled code automatically loaded when it exists
- Possible to further optimize the C by hand (care)

The GAP 4 Library

- Much extended – Aachen team, including Thomas Breuer, Be Alexander Hulpke, Heiko Theißen
- Some new features:
  - p-adic numbers
  - Infrastructure for semigroups and monoids
  - Much better handling of vector spaces, bases, and algebras
  - Multivariate polynomials and rational functions (basic oper
  - $GF(p) : p > 2^16$
  - Groups given by confluent rewriting systems
  - Smash meat-axe in the library

New Library Features Continued

- New polycyclic group code, handles infinite pc groups, bas
  Thought – Werner Nickel, Wolfgang Merkwitz
- Much improved permutation group code – HT, Akos Seres
- Lie Algebras – Willem de Graaf
- Better handling of morphisms
- Hash tables

Immutability

- Problem in GAP 3
  - Sets are stored as sorted lists
  - A list of lists or records, cannot be guaranteed to remain s
  - Algorithms for such sets are painfully slow.
- In GAP 4:
  - Lists and records can be made immutable – changing them
  - Objects (later) may claim to be immutable – promises that
    change their behaviour under = or <
  - Sets of such things stay sorted
  - Much less copying

Enumerators

- An enumerator is a virtual list
- It behaves like a list, but may be stored differently inside
- For example:
  - elements of a finite vector space
  - elements of a permutation group
  - a sparse list
- Use wherever you would use a list

Iterators

- An iterator provides a way to run through a set
- May not need to list the set in full, or even enumerate it
- Set may be infinite
- Used in `for` loops
- Any object may supply an iterator, can then use it

```
for g in G do
```

GAP 4 Library Structure

- New scheme replaces operations records
- Most functions a user calls become Operations
- An Operation is a place-holder for a whole lot of different func
Methods
- All Methods for an Operation compute the same thing, but m
different sets of inputs
- The system will call the "best" Method applicable to the inpu
selection)
- Method selection can depend on "types" of inputs, on their re
and on facts learned about them since they were created
- Easy to add new methods that work in specific circumstances

GAP

- Fairly easy to add new types of objects, new Operations, etc.
- Should be ideal for implementation of black box groups etc

Steve Linton

Objects and Types

- In GAP 3, most things are represented by records – the meth
to them are in the operations subrecord
- In GAP 4, most things are represented by Objects
- An Object stores some data (either like a list or like a record) a
- The type is partially fixed when the object is created, partially
things learnt since
- The types of the arguments determine the applicable methods
- List and record access are Operations, so an Object can prete
or a record (or both)

DIRECTIONS

Ground Rules

- Main aim of GAP is to help people do research, teach and lea
- Main focus is discrete mathematics especially algebra
- Must aim for a more distributed development effort

Some Things I think Will be Important/Useful/Intere

- Algebraic broadening – algebra, semigroups, quantum groups, combinatorics
- Links to other systems – trade capabilities, expand role as the front-end
- Library modularity – rethink the library/share package divisio
- Integrate support for parallel/distributed processing
- Split off the user interface – different UIs for different jobs

Databases and Computation with Known Group

- A favourite hobby-horse of mine – “Art and Science”
- “Computing for the post-classification world”
- System and packages contain a lot of databases
- New algorithms often depend on database type information
- Much information about known groups is not used
- For example, computing centralizers is much easier if you kno centralizer order

Computing with Known Groups – Issues 1

- How to store and organise the information
  - what to store, what to compute
  - how to avoid redundancy and error
  - how to establish provenance of data
  - groups vs generated groups
  - big variations – some databases have many small groups, o huge ones

Computing with Known Groups – Issues 2

- How to connect a concrete group with a database group
  - Constructive recognition, essentially
  - What is the user promises that the groups are isomorphic - get standard generators, match up conjugacy classes, etc
- How to use database info in all our algorithms
  - How much of the "art" can be made into "science"