Computational Group Theory With GAP

Alexander Hulpke
Department of Mathematics
Colorado State University
Fort Collins, CO, 80523, USA
http://hulpke.com

ICTS, November ‘16

Computational Group Theory With GAP

Alexander Hulpke
Department of Mathematics
Colorado State University
Fort Collins, CO, 80523, USA
http://hulpke.com

ICTS, November ‘16

Computational Group Theory With GAP

Alexander Hulpke
Department of Mathematics
Colorado State University
Fort Collins, CO, 80523, USA
http://hulpke.com

ICTS, November ‘16

Some Basics

- Thank you for the opportunity to speak!
- Slightly jetlagged - apologies.
- Lecture series on Computational Group Theory (CGT) and CGT system GAP. Not a reading from the manual.
- I will endeavor to put slides on the web after each talk. (also: Handbook *Abstract Algebra in GAP*, Lecture notes *Computational Group Theory*)
- If you don't understand me, please interrupt me!
- Ask Questions! This is not a long straight highway, but a tourist route with many viewpoints.

Coming Attractions

- About GAP and Computational Group Theory
- Orbits and Stabilizers
- Permutation Groups
- Finitely presented groups
- Matrix Groups
- Higher Level Calculations
- *Standard* Constructions in GAP

Goals Of The Course

- What can you expect from CGT software, in particular GAP?
- How to pose questions so a system can answer effectively.
- Some ideas about underlying algorithms; performance/memory use.
- How to interface with GAP: Writing Code, Transferring Data, Calling other software.

1

The Name Of The Game

What Is CGT?

Study of Algorithms for solving group theoretic problems.

- **Theoretical:** possibility, complexity. connections to theoretical CS, *P* versus *NP* (Graph Isomorphism).
- **Practical:** Concrete calculations for experiments, study of examples, dealing with smaller cases.
- Computers crave **finiteness**.
- While there are software engineering aspects, stay within mathematics.

What Is CGT?

Study of Algorithms for solving group theoretic problems.

- **Theoretical:** possibility, complexity. connections to oretical CS, *P* versus *NP* (Graph Isomorphism).

Here

- **ractical:** Concrete calculations for experiments, study of examples, dealing with smaller cases.
- Computers crave **finiteness**.
- While there are software engineering aspects, stay within mathematics.

What Is CGT?

Study of Algorithms for solving group theoretic problems.

- **Theoretical:** possibility, complexity. connections to theoretical CS, *P* versus *NP* (Graph Isomorphism).
- **Practical:** Concrete calculations for experiments, study of examples, dealing with smaller cases.
- Computers crave **finiteness**.
- While there are software engineering aspects, stay within mathematics.

Uses Of CGT

Concrete Calculations in concrete groups. Uses:

- Study examples, lead to conjectures
- Search through (group) libraries to find counterexample
- Base cases or borderline cases
- Applications of Group Theory
- There is no generic "*solve my problem*" function! Need to know about algorithms to see what is feasible.

GAP

- Open Source software for Computational Group theory. (Other: Magma, a bit in Maple)
- www.gap-system.org
- Installation help in first practical.
- Source on github: **gap-system/gap**.
- Unix origins, but working Windows port.
- First releases around 1990, currently GAP4.8.5 (Sep '16)
- Both to try implementations, and as calculation tool.
- Over 2000 citations in refereed papers.
- gap-forum mailing list, math.stackexchange tag **gap**

Further System Details

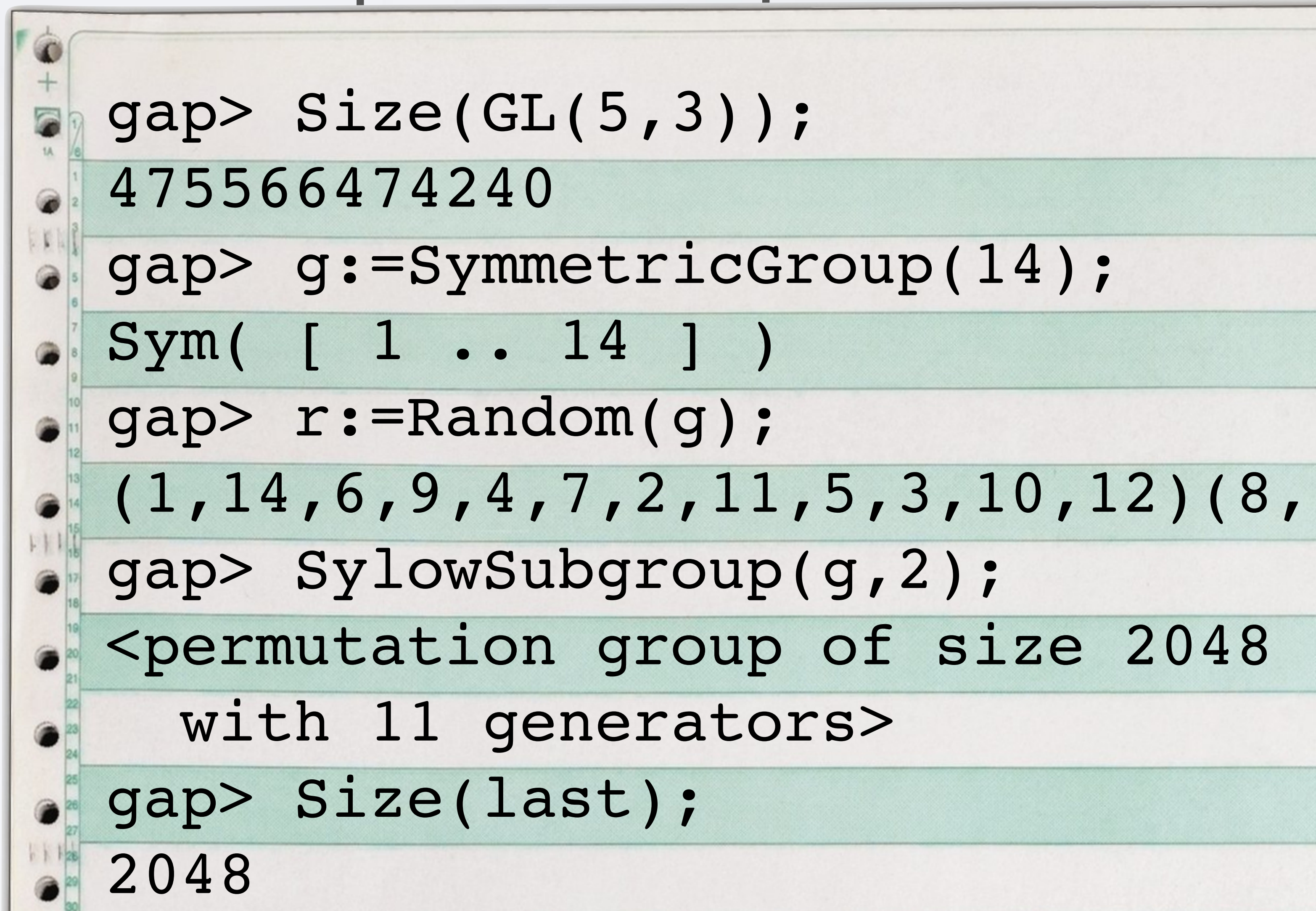
- Interpreted language. Kernel provides language and time-critical functions: 10/90% rule. Most written in library, GAP language.
- Can compile to C framework but only to load to GAP session. Don't overestimate success.
- You can call GAP from Sage, but I have never used it. There also is a way to call GAP as library in C, but I have no idea how that works.
- Substantial library of contributed *Packages*. Might need explicit loading: `LoadPackage("pkgname");`

Assumptions

- You know undergraduate college mathematics, in particular abstract algebra.
- You have done basic programming (in whatever language), know how to create text files, etc.
- Groups act on the right: Permutations, row vectors, etc.
- Pointers to manual, not attempt to give manual definitions.
- **Bootstrapping:** So we can do examples, some commands as black-box before explanations

Basic System Interaction

- GAP prompts for input
- Enter command (ending in semicolon)
- GAP responds with result. (Object is returned and `View`-ed.) Double semicolon supresses output.
- Command could be assignment (and shows the value of assignment)
- Variables `last`,`last2` for prior results.



```
gap> Size(GL(5,3));
475566474240
gap> g:=SymmetricGroup(14);
Sym( [ 1 .. 14 ] )
gap> r:=Random(g);
(1,14,6,9,4,7,2,11,5,3,10,12)(8,
gap> SylowSubgroup(g,2);
<permutation group of size 2048
  with 11 generators>
gap> Size(last);
2048
```

Using Files

- Any longer input: `Read("filename")` in file with sequence of commands (e.g function definitions.)
- `?` for online manual, `??` for keyword search in manual.
- `LogTo("filename")` produces file transcript.
- `PrintTo` and `AppendTo` create specific file output.
- There is a `SaveWorkspace` function, but somewhat tricky to use under Windows. Saved workspaces do not port between versions/platforms.
- Exit with `quit;` (or CTRL-D)

Files: Names And Places

By default, GAP accesses the folder where it was started.

All other paths must be relative to this folder or global.

Under Windows this is the folder in which GAP was installed :-| and paths must be Unix style :-(

```
/cygdrive/c/Users/Alexander Hulpke/My Documents/
```

Use `DirectoryHome` to get paths to documents folder:

```
gap> DirectoryHome();
dir("/cygdrive/c/Users/Alexander Hulpke/My Documents/")
gap> f:=Filename(DirectoryHome(),"newfile.txt");
"/cygdrive/c/Users/Alexander Hulpke/My Documents/newfile.txt"
```

Caveat: Windows might be set to not display the suffix, so the file `new.g` you created in Notepad might really be `new.g.txt`.

Objects

- Integers/Rationals/Cyclotomics, Booleans (including `fail`)
- Finite field elements, Permutations
- Record, Lists. Lists are used to represent sets, vectors, matrices, etc.
- Objects built from lists and records (groups, words, …)
- Variables: Any non-reserved string. Assign with :=
- List entries are pointers, function calls are call-by-reference. It is possible to protect objects from modification.
- ALGOL (Pascal, Java, Python) family programming language.
- Interpreted, break loop allows variable inspection

Objects

- Integers/Rationals/C
- Finite field element
- Record, Lists. Lists a matrices, etc.
- Objects built from l
- Variables: Any non-
- List entries are poi ters, function calls are call-by-reference. It is possible to pro ect objects from modification.
- ALGOL (Pascal, Java, Python) family programming language.
- Interpreted, break loop allows variable inspection

GAP will give an error message and the prompt changes to `brk>`
At this point one can inspect local variables for debugging, use `Where(lev);` to see whole call stack.
Leave break loop with quit or with CTRL-D.
(Can trigger error in own functions with `Error("message");`)

Objects

- Integers/Rationals/Cyclotomics, Booleans (including `fail`)
- Finite field elements, Permutations
- Record, Lists. Lists are used to represent sets, vectors, matrices, etc.
- Objects built from lists and records (groups, words, …)
- Variables: Any non-reserved string. Assign with :=
- List entries are pointers, function calls are call-by-reference. It is possible to protect objects from modification.
- ALGOL (Pascal, Java, Python) family programming language.
- Interpreted, break loop allows variable inspection

Lists

(Almost) any collection which is not an algebraic structure is represented as a list: Sets, Vectors, Matrices,…

Lists might be stored efficiently (ranges, enumerators, bitlists, compact vectors,…) but always allow the basic operations of `Length`, element access and `Position`. If GAP can compare elements cheaply, it can be worth to sort for fast position tests.

Lists can have unbound entries and can grow arbitrarily large.

Li

it will try to be clever, but you might need to help it.

(Almost) any collection which is not an algebraic structure is represented as a list: Sets, Vectors, Matrices,…

Lists might be stored efficiently (ranges, enumerators, bitlists, compact vectors,…) but always allow the basic operations of `Length`, element access and `Position`. If GAP can compare elements cheaply, it can be worth to sort for fast position tests.

Lists can have unbound entries and can grow arbitrarily large.

Lists

(Almost) any collection which is not an algebraic structure is represented as a list: Sets, Vectors, Matrices,…

Lists might be stored efficiently (ranges, enumerators, bitlists, compact vectors,…) but always allow the basic operations of `Length`, element access and `Position`. If GAP can compare elements cheaply, it can be worth to sort for fast position tests.

Lists can have unbound entries and can grow arbitrarily large.

Lists

(Almost) any collection which is not an algebraic structure is represented as a list: Sets, Vectors, Matrices,…

Lists might be stored efficiently (ranges, enumerators, bitlists, compact vectors,…) but always allow the basic operations of `Length`, element access and `Position`. If GAP can compare elements ch

sort for fast position tests.

well, up to 10^{b-4}, where b is the “bitness" — 32 or 64.

Lists can have unbound entries and can grow arbitrarily large.

List Operations

A number of list operations, taking as argument a list and a function, allow functionality that loops over a list:

List, Filtered, Number, ForAll, ForAny, First, PositionProperty

The function might be given in λ-syntax: x->expr(x)

This allows for whole functions being written in one line:

```
  Filtered( Filtered([2..2000],n->not IsPrime(n)),
      n->ForAll( Filtered([2..n-1],b->Gcd(b,n)=1),
                          b->b^(n-1) mod n=1) );
```

returns [561, 1105, 1729]

List Operati

A number of list operations, taking
a function, allow functionality that l

List, Filtered, Number, ForAll,
PositionProperty

The function might be given in λ-s ntax: x->expr(x)

This allows for whole functions b ing written in one line:

```
  Filtered( Filtered([2..200 ],n->not IsPrime(n)),
     n->ForAll( Filtered([2 .n-1],b->Gcd(b,n)=1),
                      b->b^( -1) mod n=1) );
```

returns [561, 1105, 1729]

$=1^3+12^3$

$=9^3+10^3.$

General Language Conventions

- Library operations are named in upper case with infix upper case letters: `SylowSubgroup`.
- If the *modus operandi* is specified, it comes aft the end of the name: `GroupHomomorphismByImages`.
- Arguments are typically listed in deceasing size: group, permutation, point
- *Attributes* (e.g. `Size`) store information about objects that has been obtained in calculations. Associated are functions to test for knowledge (`HasSize`) or to store knowledge (`SetSize`).
- If possible, classes (or elements, subgroups) with `Representative`

Useful Group Functionality

Group(gens), Subgroup(G,gens), Random(G)

Centralizer(G,elm) and Normalizer(G,subgroup)
ConjugacyClasses(G) of elements, returns list of classes, each class stores Representative, Centralizer.

ConjugacyClassesSubgroups(G) returns list of classes, each class stores Representative, Normalizer.

IntermediateSubgroups subgroups $S<U<T$

NormalSubgroups(G), MaximalSubgroupClassReps(G) list of representatives. ComplementClassesRepresentatives(G,N)

IsomorphismGroups(G,H) and IdGroup(G). GQuotients(G,H) (up to kernel equality). **Beware** StructureDescription

Useful Group Functionality

Group(gens), Subgroup(G,gens), Random(G)

Centralizer(G,elm) and Normalizer(G,subgroup)
ConjugacyClasses(G) of elements, returns list of classes, each class stores Representative, Centralizer.

ConjugacyClassesSubgroups(G) returns lis ach class stores Representative, Normalizer.

soon to be less lousy!

IntermediateSubgroups subgroups $S<U<T$

NormalSubgroups(G), MaximalSubgroupClassReps(G) list of representatives. ComplementClassesRepresentatives(G,N)

IsomorphismGroups(G,H) and IdGroup(G). GQuotients(G,H) (up to kernel equality). **Beware** StructureDescription

Useful Group Functionality

Group(gens), Subgroup(G,gens), Random(G)

Centralizer(G,elm) and Normalizer(G,subgroup)
ConjugacyClasses(G) of elements, returns list of classes, each class stores Representative, Centralizer.

ConjugacyClassesSubgroups(G) returns list of classes, each class stores Representative, Normalizer.

IntermediateSubgroups subgroups $S<U<T$

NormalSubgroups(G), MaximalSubgroupClassReps(G) list of representatives. ComplementClassesRepresentatives(G,N)

IsomorphismGroups(G,H) and IdGroup(G). GQuotients(G,H) (up to kernel equality). **Beware** StructureDescription

Basic Example

Suppose we want to find how many subgroups of the Mathieu group M_{11} are isomorphic to A_5.

Not necessarily the fastest/best way, but to show some functionality.

```
gap> g:=Group((1,2,3,4,5,6,7,8,9,10,11),(3,7,11,8)
(4,10,5,6));;
gap> s5:=SymmetricGroup(5);
Sym( [ 1 .. 5 ] )
gap> a5:=DerivedSubgroup(s5);
Alt( [ 1 .. 5 ] )
gap> u:=List(ConjugacyClassesSubgroups(g),Representative);;
gap> Length(u);
39
gap> u:=Filtered(u,x->Size(x)=Size(a5));;Length(u);
2
gap> List(u,IsSimpleGroup);
[ true, true ]
gap> IsomorphismGroups(u[1],a5);
[ (3,5)(4,6)(7,9)(10,11), (1,2,3)(4,9,8)(5,10,11) ] ->
[ (2,4)(3,5), (1,2,3) ]
gap> RepresentativeAction(SymmetricGroup(11),u[1],u[2]);
fail
gap> Sum(u,x->Index(g,Normalizer(g,x)));
198
```

GAP Is A Good Bureaucrat

It will only results that are correct, subject to

- Program Errors, Computer Errors
- Publication-level research (including CFSG!)
- Also (verification) true if random methods are used.

It will aim to answer the question posed, if it can at all.

Even if that means enumerating the elements of E_8, exceeds all memory / run time available, and brings a multi-user machine to its knees.

Error, if limit (initially 2GB, then doubles) is exceeded.

GAP Is A Good Bureaucrat

- Can change limit by a startup-option (batch file on Windows)
- Can continue with `return;` then limit is doubled each time

Think about whether what GAP does is sensible (missing information?)

Is there a chance for the calculation to complete?

It will aim to answer the question posed, if it can at all.

Even if that means enumerating the elements of E_8, exceeds all memory / run time available, and brings a multi-user machine to its knees.

Error, if limit (initially 2GB, then doubles) is exceeded.

Trust is Good, Control is Better

GAP will not automatically trust all user input but perform sanity checks (element in group, mapping on generators is a homomorphism,…) and trigger errors, return `fail`.

In some cases such checks can take substantial time, in particular if many are performed in a loop.

Certain operations have a `NC` (no check) variant that skips the tests and can be used in situations when the context makes it clear the parameters will be valid.

Dually: `Assertions` in code. Error if condition is not met. Turn on by `SetAssertionLevel(`*nr*`)` (default: 0)

Method Selection

GAP tries to select the “best” method (of several) for a particular operation (say `NormalSubgroups`) based on:

- What kind of object (Category – group of permutations)
- How is it stored (Representation – sparse polynomial)
- What is known (Attributes – solvable of order 1234)

It basically tries to select the most specialized method that still applies, assuming it is the best.

You can sometimes help GAP by explicitly setting information, in particular `Size` for large groups.

Das sollt Ihr mir nicht zweimal sagen!
Ich denke mir, wie viel es nützt
Denn, was man schwarz auf weiß besitzt,
Kann man getrost nach Hause tragen.
J.W.V.GOETHE, Faust, 1. Akt

This, Sir, a second time you need not say!
Your counsel I appreciate quite;
What we possess in black and white,
We can in peace and comfort bear away.

Slides, Lecture Notes at

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