Computers in Group theory

An introduction to

GAP

Alexander Hulpke
Lehrstuhl D für Mathematik
RWTH Aachen

Joint work with:

Hans-Ulrich Besche
Thomas Bischops
Thomas Breuer
Frank Celler
Bettina Eick
Volkmar Felsch
Markus Fuchs
Ansgar Kaup
Johannes Meier
Jürgen Mnich
Werner Nickel
Alice Niemeyer
Joachim Neubüser
Goetz Pfeiffer
Udo Polis
Benedikt Rothe
Michael Scherner
Martin Schönert
Akos Serres
Heiko Theißen
Alexander Wegner
...

Language

Pascal-like language, Lisp-like list management

Rationals	`1, 5/7, -1/3`
Cyclotomics	`E(7)^2+3*E(7)`
Finite Fields	`Z(5)^4, Z(2^8)^15`
Permutations	`(1,2,3,4,5)`
Lists	`[1,E(3),(1,2,3),["abc",'a']], [[1,2],[3,4]],` Sets
Records	`rec(a:=1,b:=5), Group((1,2,3)), x^3+2*x+1`

Objects are always in canonical form

Ways to get Groups

Automorphisms	(Matrices, Permutations, Galois groups)
Presentations	$\langle a, b \mid a^3 = b^2 = a^b a = 1 \rangle$

"Typical" (?) Questions

What are the maximal subgroups of

$$\langle a, b \mid aba^{-2}bab^{-1} = b^{-1}a^3b^{-1}a^{-2} = 1 \rangle?$$

How does the permutation character of Fi23 on the cosets of $3_+^{1+8}.2_-^{1+6}.3_+^{1+2}.2S_4$ *decompose into irreducibles?*
What are the minimal transitive groups of degree 24?
Does the character table determine the isomorphism type?
List the groups of order 720.

Often we need theory to reduce the problem for the computer.

Basic Algorithms

Linear Algebra — Triangulization, Basis

Enumeration — "Labyrinth search", Combinations. Hard!

Orbit/Stabilizer

```
OrbitStabilizer:=function(G,p)
local orb,stab,o,g,im,t;
  orb:=[p];
  stab:=TrivialSubgroup(G);
  t:=[];t[p]:=G.identity;
  for o in orb do
    for g in G.generators do
      im:=o^g;
      if not im in orb do
        Add(orb,im);
        t[im]:=t[o]*g;
      else
        stab:=Closure(stab,t[o]*g/t[im]);
      fi;
    od;
  od;
  return [orb,stab];
end;
```

Symbolic expressions

Library lookup — Use existing classifications

Provided functions

- Order (Size)
- Intersection, Span
- Change of Representation
- (Characteristic) Subgroups and series
- Homomorphisms

GAP automatically chooses suitable methods depending on the representation.

Finitely presented groups

Todd-Coxeter	• Index • Permutation representation • Subgroup Presentation: Modified TC / Reduced Reidemeister-Schreier • Presentation for permutation group
Low Index	
Simplification	Tietze transformations
Quotient Algorithms	• Abelian quotient (Elementary Divisors) • p-Quotient • Nilpotent Quotient • Solvable Quotient
G-Quotient	

Case study 1

A. Cavicchioli (Modena): Topological problem yields groups

$$G_1 = \langle a, b \mid aba^{-2}bab^{-1} = (b^{-1}a^3b^{-1}a^{-3})^q = 1 \rangle$$

```
gap> InfoFpGroup1:=Print;; InfoFpGroup2:=Print;;
gap> f:=FreeGroup(2); a:=f.1;; b:=f.2;;
Group( f.1, f.2 )
gap> g:=f/[a*b*a^-2*b*a*b^-1,(b^-1*a^3*b^-1*a^-3)^1*a];
Group( f.1, f.2 )
gap> u:=Subgroup(g,[g.1]);
Subgroup( Group( f.1, f.2 ), [ f.1 ] )
gap> Index(g,u);
#I  CosetTableFpGroup called:
#I      defined deleted alive   maximal
#I 13 1 12 13
12
gap> op:=OperationCosetsFpGroup(g,u);
Group( ( 2, 3, 5, 7, 9)( 4, 6, 8,10,11),
( 1, 2, 4, 6, 5)( 7, 9,11,12, 8) )
gap> Size(op);
60
gap> Index(g,TrivialSubgroup(g));
#I  CosetTableFpGroup called:
#I      defined deleted alive   maximal
#I 133 13 120 133
120
gap> AbelianInvariants(CommutatorFactorGroup(g));
[  ]
gap> g:=f/[a*b*a^-2*b*a*b^-1,(b^-1*a^3*b^-1*a^-3)^2*a];
Group( f.1, f.2 )
gap> Index(g,Subgroup(g,[g.1]));
```

```
#I  CosetTableFpGroup called:
#I      defined deleted alive   maximal
#I 999 1 998 999
   . . .
#I 63927 73 63854 63887
Error, the coset enumeration has defined more than
64000 cosets:
type 'return;' if you want to continue with a new limit
of 128000 cosets, type 'quit;' if you want to quit the
coset enumeration, type 'maxlimit := 0; return;' in
order to continue without a limit, in
CosetTableFpGroup( G, H ) called from
G.operations.Index( G, U ) called from
Index( g, Subgroup( g, [ g.1 ] ) ) called from
main loop
brk> quit;
gap> AbelianInvariants(CommutatorFactorGroup(g));
[  ]
gap> u:=LowIndexSubgroupsFpGroup(g,TrivialSubgroup(g),15);;
#I  LowIndexSubgroupsFpGroup called
#I   class 1 of index 1 and length 1
   . . .
#I   class 25 of index 11 and length 11
#I  LowIndexSubgroupsFpGroup returns 25 classes
gap> time;
2259011
gap> u:=Filtered(u,i->Index(g,i)>1);;
gap> op:=List(u,i->OperationCosetsFpGroup(g,i));;
gap> id:=List([1..24],i->[Index(g,u[i]),Size(op[i]),
>                      TransitiveIdentification(op[i])]);;
gap> Collected(id);
```

```
[ [ [ 11, 660, 5 ], 2 ], [ [ 11, 19958400, 7 ], 8 ],
  [ [ 12, 660, 179 ], 1 ], [ [ 12, 95040, 295 ], 4 ],
  [ [ 12, 239500800, 300 ], 3 ],
  [ [ 13, 3113510400, 8 ], 6 ] ]
gap> [Factorial(11)/2,Factorial(12)/2,Factorial(13)/2];
[ 19958400, 239500800, 3113510400 ]
gap> d:=DirectProduct(op);;
gap> Sum(id,i->i[1]);
284
gap> img:=Subgroup(d,[Product([1..24],
>                              i->d.generators[2*i-1]),
>                     Product([1..24],
>                              i->d.generators[2*i])]);;
gap> List(g.relators,i->MappedWord(i,g.generators,
>                                    img.generators));
[ (), () ]
gap> Size(img);
18783884078113708380328017564655096104009422379395561\
19661996443156145275345435097975617726355751505731109\
984424427520000000000000000000000000000000000000
gap> Order(img,img.1);
11
gap> List(u,i->AbelianInvariants(CommutatorFactorGroup(i)));
[ [ 2, 6 ], [ 2, 6 ], [ 2, 6 ], [ 2, 6 ], [ 2, 6 ],
  [ 2, 6 ], [ 2, 6 ], [ 2, 6 ], [ 2, 6 ], [ 2, 6 ],
  [ 2, 0, 0 ], [ 3, 18 ], [ 3, 18 ], [ 2, 2, 8 ],
  [ 2, 2, 8 ], [ 2, 2, 8 ], [ 2, 2, 8 ],
  [ 2, 2, 8 ], [ 143 ], [ 143 ], [ 143 ], [ 143 ],
  [ 143 ], [ 143 ] ]
```

Permutation groups

Orbit/Stabilizer	
Blocks	
Schreier-Sims	Tree search through the group to get elements with certain properties. Much work is needed to cut off branches.
Homomorphisms	• Operation Homomorphisms • Generator images

The Schreier-Sims algorithm is of exponential complexity. Using a Random Schreier Sims, one can get probabilistic algorithms of almost linear complexity.

Ag-Groups

Special type of presentation, adapted to subnormal series. Allows utilization of elementary abelian factors. Problems are reduced to linear algebra.

Character tables

Characters are given as lists of their values on classes. They are collected as the list `irreducibles` in the character table record. These records are identified by a `name` component. Functions exist to

- Compute class fusions;
- Induce,
- Restrict,
- Decompose,

- Reduce Characters;
- Compute Kernels, indicators etc.

Table libraries provide extended versions of the ATLAS and the Modular Atlas.

Case study 2

G. Hiss (Heidelberg): Permutation character of Fi23 on the cosets of $3_+^{1+8}.2_-^{1+6}.3_+^{1+2}.2S_4$.

```
gap> D:=CharTable("Fi23");;
gap> permchar:=Sum(D.irreducibles{[1,2,6]});;
gap> permchar[1];
31671
gap> Filtered([1..98],i->D.orders[i]=3);
[ 5, 6, 7, 8 ]
gap> permchar{last};
[ 351, 324, 135, 27 ]
gap> roots:=[6];;
>    for i in [1..Length(D.classes)] do
>      if ForAny(D.powermap,j->j[i] in roots) then
>        AddSet(roots,i);
>      fi;
>    od;
gap> roots;
[ 6, 15, 17, 19, 21, 33, 34, 35, 36, 43, 45, 46, 47, 52,
 66, 67, 68, 69, 70, 71, 72, 83, 87, 93, 94 ]
gap> Sum(roots,i->1/D.centralizers[i]);
257647/1594323                # 16%
gap> Read("fischer23.grp");  # taken from a local file
gap> StabChain(Fi23,rec(size:=4089470473293004800,
```

```
> random:=100));;
gap> opdom:=PermGroupOps.MovedPoints(Fi23);;
gap> found:=false;; g:=();; h:=();;
gap> repeat
>      g:=Random(Fi23);
>      if Order(Fi23,g) mod 3 = 0 then
>        h:=g^(Order(Fi23,g)/3);
>        if Number(opdom, i->i^h=i) = 324 then
>          found:=true;
>        fi;
>      fi;
>    until found;
gap> H:=Subgroup(Fi23, [h]);;
gap> N:=Normalizer(Fi23, H);;Size(N);
3265173504
gap> orb:=Orbits(N,opdom);;List(orb,Length);
[ 11664, 19683, 324 ]
gap> P:=Operation(N,orb[1]);;
gap> MakeStabChainRandom(P);;Size(P);
3265173504
gap> A:=AgGroup(P);;Size(A);
3265173504
gap> A.name:="Fi23M7";;
gap> ds:=DerivedSeries(A);;Length(ds);
11
gap> List(ds,i->Collected(Factors(Size(i))));
[ [ [ 2, 11 ], [ 3, 13 ] ], [ [ 2, 10 ], [ 3, 13 ] ],
  [ [ 2, 10 ], [ 3, 12 ] ], [ [ 2, 8 ], [ 3, 12 ] ],
  [ [ 2, 7 ], [ 3, 12 ] ], [ [ 2, 7 ], [ 3, 10 ] ],
  [ [ 2, 7 ], [ 3, 9 ] ], [ [ 2, 1 ], [ 3, 9 ] ],
  [ [ 3, 9 ] ], [ [ 3, 1 ] ], [ [ 1, 1 ] ] ]
```

```
gap> conjcl:=ConjugacyClasses(A);;Length(conjcl);
181
gap> C:=CharTable(A);;
gap> Maximum(List(C.irreducibles,i->i[1]));
18432
gap> irrat:=Filtered(Union(C.irreducibles),i->not IsRat(i));
[ -4*E(8)-4*E(8)^3, -2*E(8)-2*E(8)^3, -E(8)-E(8)^3,
  E(8)+E(8)^3, 2*E(8)+2*E(8)^3, 4*E(8)+4*E(8)^3 ]
gap> alpha:=irrat[4];
E(8)+E(8)^3
gap> X(Rationals).name:="x";;
gap> Polynomial(Rationals,MinPol(alpha));
x^2 + 2
gap> Polynomial(Rationals,MinPol(E(8)));
x^4 + 1
gap> fus:=SubgroupFusions(C,D,rec(quick:=true));;
#I SubgroupFusions: fusion initialized
#I SubgroupFusions: consistency with power maps checked,
#I    the actual indeterminateness is
707731566368444002744845613785551034049388197773286l0
. . .
#I SubgroupFusions: 6 solutions
gap> au:=TableAutomorphisms(C,C.irreducibles);
Group( (160,161)(170,171)(172,177)(173,178)(174,179)
(175,180)(176,181), ( 72, 73, 74)( 76, 77, 78)
( 79, 80, 81)( 84, 85, 86)( 87, 88, 89)( 93, 94, 95)
( 96, 97, 98)( 99,100,101)(106,107,108)(109,110,111)
(113,114,115)(116,117,118)(122,123,124)(127,128,129)
(130,131,132)(134,135,136)(138,139,140)(142,143,144)
(147,148,149) )
gap> RepresentativesFusions(au,fus,D.automorphisms);
```

```
#I RepresentativesFusions: There is 1 orbit of length 6.
[ [ 1, 6, 7, 5, 8, 6, 19, 4, 43, 10, 16, 17, 3, 24,
. . .
      82, 30, 84, 64 ] ]
gap> fus:=last[1];;
gap> ind:=Induced(C,D,C.irreducibles{[1]},fus);
[ [ 1252451200, 12812800, 98560, 44160, 167440,
      35761, 5170, 904, 1120, 112, 320, 80, 0, 3640,
      3217, 280, 145, 472, 561, 120, 177, 528, 166,
      10, 66, 24, 30, 48, 0, 20, 12, 4, 121, 13, 49,
      13, 1, 0, 0, 0, 0, 40, 25, 28, 13, 13, 17, 4,
      10, 8, 4, 5, 8, 4, 2, 2, 0, 0, 0, 0, 0, 0, 2,
      2, 0, 1, 7, 9, 9, 1, 1, 3, 1, 0, 0, 0, 0, 0,
      0, 0, 0, 6, 1, 2, 0, 0, 1, 0, 0, 0, 0, 0, 1,
      1, 0, 0, 0, 0 ] ]
gap> MatScalarProducts(D,D.irreducibles,ind);
[ [ 1, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0, 0, 0, 1, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0,
      3, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 2, 0, 0,
      0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0 ] ]
```