Computing with Finite Matrix Groups

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

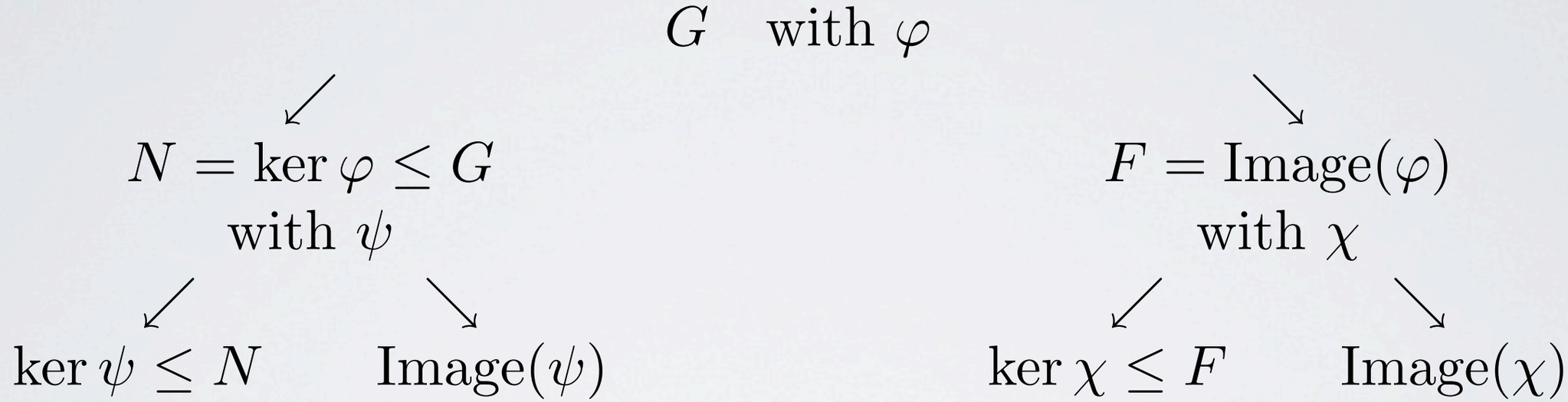
State of the Art

There are practical methods (GAP and Magma) for computing

Task	Permutation	Matrix
Order, Membership	Stabilizer Chain	Matrix Group Recognition
Composition Structure		
Homorphisms		
Centralizers, Normalizers	Backtrack	Want !
Conjugating Elements		
Classes of Elements	PcGroups	Extend Solvable Radical Method
Subgroups	Solvable Radical / Trivial Fitting	
Isomorphism Test		

Step1: Matrix Group Recognition

Matrix Group Recognition finds actions and thus obtains a composition tree



At each node, we can evaluate the homomorphism (by acting on the objects of the underlying decomposition) and have generators for the kernel.

Leafs

Each leaf of the tree is a simple group.

We know its type and have an isomorphism to a natural representation. (Sssume we know *everything* about the simple groups.)

The tree thus represents a composition series of G.

We know the subgroups in this series and for each subgroup the homomorphism on its simple quotient.

Step 2: Radical and its Quotient

To use the solvable radical method, we need to find $R = \mathrm{Rad}(G) \triangleleft G$, the largest solvable normal subgroup, and an effective homomorphism ϱ: $G \to G/R$.

$\mathrm{Soc}(G/R)$ is direct product of simple nonabelian groups and (up to isomorphism) $G/R \leqq \mathrm{Aut}(\mathrm{Soc}(G/R))$.

So ϱ should be the action of G on this socle. But the socle factors are spread over the composition series.

Reconstructing the Socle Action

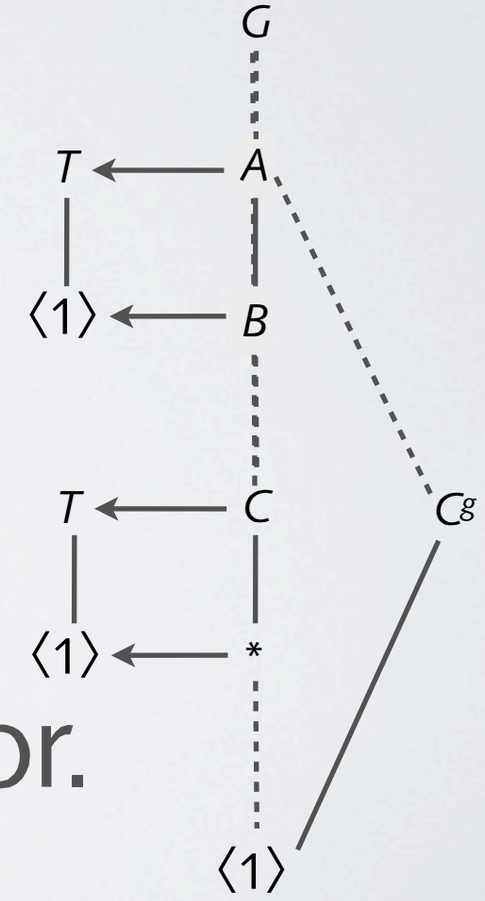
Let C be a subgroup in the composition series, $C \to T$ simple nonabelian quotient in series. If C is deepest in series, elements of C represent a **single** factor of this socle.

Conjugation by $g \in G$ will map C to C^g.

In chain, C^g maps to quotient A/B of same isomorphism type.

A/B represents another socle factor.

We thus can act on Soc(G/R).



Combining Actions

The G on non-abelian composition factors of one type T yields a homomorphism $\alpha: G \to (\mathrm{Aut}\ T) \wr S_n$. Image is permutation group (or matrix group). Combine to $\varrho = \alpha_1 \times \ldots \times \alpha_m$ into direct product.

This is the action of G on $\mathrm{Soc}(G/R)$. Thus $\ker \varrho = R$.

If the image is a permutation group, use existing methods for computation.

Layering the Radical

Conjecture: Solvable matrix group R usually has a short orbit on vectors or submodules.

If no large primes: $\max(12,n)\cdot(q^{(n/2)}+1)$

Submodules for R', R'',... give candidates.

Algorithm by SIMS (solvable BSGS) finds series $G \rhd R = R_0 \rhd R_1 \rhd \cdots \rhd \langle 1 \rangle$ with R_i/R_{i+1} elementary abelian, coefficients in these vector spaces (PCGS).

Step 3: Working with Subgroups

To avoid evaluating ϱ represent $U \leqq G$ by:

- An induced PCGS (think: REF for matrix) for $U \cap R$.
- Generators $u_i \in U$ s.t. $U = \langle U \cap R, u_1, u_2, \ldots \rangle$
- Images $u_i \varrho$ as elements of $G/R \leqq D$.

Element test in U then first tests in $U \varrho$.
Then divide off and test in $U \cap R$.

Analogously, for any $x \in G$ in the algorithm also maintain its image $x\varrho \in G/R$.

Step 4: Lifting

We can now proceed essentially in the same way as for permutation groups:

Assume we know the result in $G/R = G/R_0$. (E.g. by permutation group methods, if this is a permutation group.)

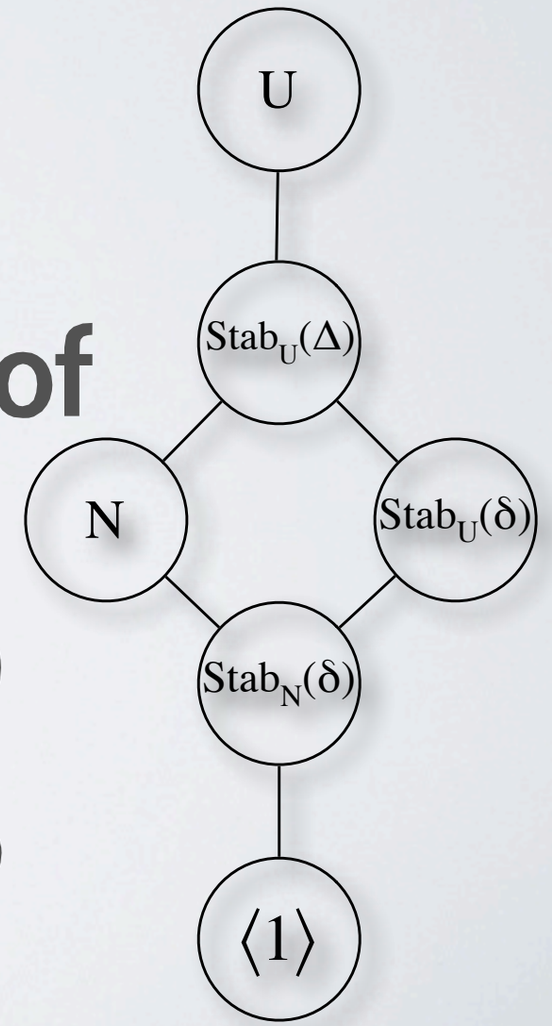
Now go repeatedly from G/R_i to G/R_{i+1} until we reach $R_k = \langle 1 \rangle$.

Each step reduces to orbit calculations for an (affine) action on R_i/R_{i+1}.

Orbit/Stabilizer Algorithm

When calculating orbit/stabilizer of δ under U (this will be a basic operation)

- Calculate the orbit Δ of δ under $N = U \cap R \trianglelefteq U$ and the stabilizer $V \leqq U \cap R$ of δ. Δ is a U-block.
- Calculate the orbit and stabilizer **of** Δ under U by computing in U^{ϱ}. (Represent Δ^u by single element.)
- Correct generators of $\mathrm{Stab}_U(\Delta)$ to get $\mathrm{Stab}_U(\delta)$ as complement.



Step 5: Implementations

New interface for solvable radical code in GAP 4.7. Used by new ConjugacyClasses/Centralizer/Canonical Conjugate routine. (MECKY/NEUBUESER, SOUVIGNIER/HOLT/CANNON, H.)

(Backtrack centralizer is often faster, but canonical element is nice.)

Experimental implementation of this interface for matrix groups, using `recog` package (NEUNHOEFFER, SERESS).

Applicable to matrix groups of considerable size.

Conjugacy Classes Runtimes

Times in Seconds on a 2.6GHz MacPro.

Group	Order	deg	q	#Classes	t_{Setup}	t_{Calc}
$(GL_2(5) \wr S_3) \perp_6 (L_2(11) \wr S_3)$	19076854579200000	21	5	1235200	17	22886
$(GL_2(5) \wr S_3) \perp_2 (L_2(11) \wr S_3)$	572305637376000000	21	5	503808	22	9078
$2^{9+16}.S_8(2)$	1589728887019929600	394	2	703	455	998
$3^{1+12}.2Suz.2$	28592301550804992000	78	3	253	427	76
$5^9:(GL_3(5) \times GL_3(5))$	4324500000000000000	6	5	18464	3	361
$(6.A_5) \wr S\_5$	56422198149120000000	30	25	526473	27	2863
$3^{15}:(M_{11} \wr S_3)$	42770626907728896000	16	3	3200	14	129
$2^{1+22}.Co_2$	3548835956612136960000	1045	2	448	654	4790
$((2^2 \times 3).U_6(2)) \wr S_2$	2435952824419268689920000	54	4	77814	41	1281
$11^9:(SL_3(11) \times SL_3(11))$	5364074707031250000000000	6	11	20759	5	2819
$(5^3{:}L_3(5)) \wr S_3.$	6032677500000000000000000	75	5	12200	97	1270
$2^{44}:(M_{11} \times (2^4:(S_3 \times S_3)))$	2480816141360352083312640	15	2	10759	5	4167
$(3^{10}:(M_{11} \wr 2)) \perp_2 (J_2 \wr 2)$	270967042389167308800000000	83	3	127764	44	13729
$7^{12}:(SL_3(7) \times SP_4(7))$	21556715139427451384217600	7	7	7701	12	2670

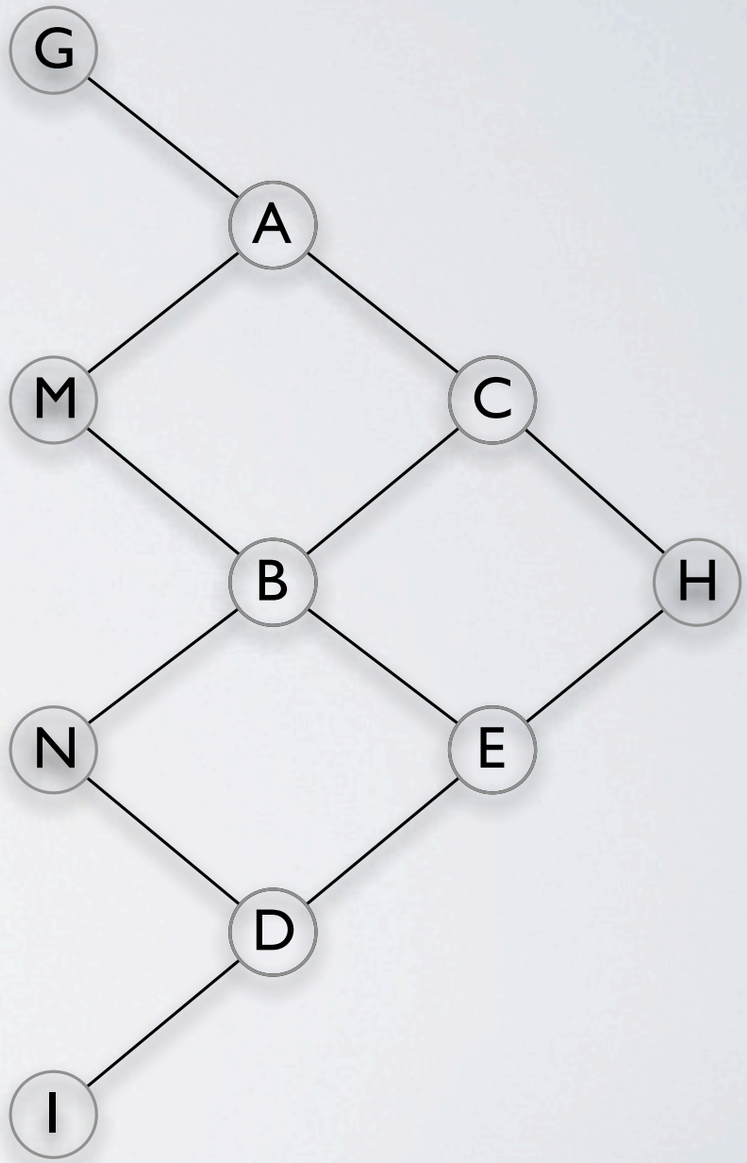
Subgroup Normalizers

Experimental implementation of Subgroup Normalizer (following GLASBY/SLATTERY method for PcGroups.)

To normalize H in G, when $M,N \triangleleft G$, normalize closures/intersections in order *A,B,C,...*

Up step: Stabilize complement (act on cohomology).

Down Step: Stabilize Subspace (Scarily long orbits, centralizer and induced automorphisms.)



Constructive Existence Proof of Implementation

```
gap> g:=AtlasSubgroup("J4",IsMatrixGroup,1);  #2^11.M24
<matrix group of size 501397585920 with 2 generators>
gap> ff:=FittingFreeLiftSetup(g);;          # very fast
gap> u:=Subgroup(g,[g.1,g.1^g.2]);;
gap> Size(u);
22
gap> NormalizerViaRadical(g,u);
<matrix group of size 220 with 4 generators>

gap> u:=HallViaRadical(g,[3]);       # Joint work with Eick
[ <matrix group of size 27 with 3 generators> ]
gap> n:=NormalizerViaRadical(g,u[1]);time;
<matrix group of size 432 with 7 generators>
6583
```

Somewhat larger example

```
gap> g;  # Max in BM,  dimension 813 over GF(2), 4ms/prod.
2^(2+10+20).(M22:2 x S3)
gap> Size(g);
2285884674l463040
gap> ff:=FittingFreeLiftSetup(g);; # ~10 minutes
#I  Used Base Points[ ...] Lengths [ 3, 2, 32768, 32, 4096 ]
gap> ff.radical;
<matrix group of size 25769803776 with 34 generators>
gap> ff.pcisom;
Pcgs([ <an immutable 813x813 matrix over GF2>, [...]
  <an immutable 813x813 matrix over GF2> ]) -> Pcgs(
[ f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15,
  f16, f17, f18, f19, f20, f21, f22, f23, f24, f25, f26, f27, f28,
  f29, f30, f31, f32, f33, f34 ])
gap> u:=HallViaRadical(g,[3]);time;
[ <matrix group of size 27 with 3 generators> ]
36697
```

...continued

```
gap> n:=NormalizerViaRadical(g,u[1]);
#I  Radsize= 9 index 1
#I  abelian factor 2: 25769803776->12884901888 central:true
#I  down
#I  module of dimension 1 subspace 0
#I  up 0: on 2 cobounds:1
#I  abelian factor 3: 12884901888->4294967296 central:false
#I  down
#I  module of dimension 1 subspace 1
#I  abelian factor 4: 4294967296->4096 central:false
#I  down
#I  module of dimension 2 subspace 0
#I  module of dimension 18 subspace 0
#I  module of dimension 20 subspace 0
#I  up 2:3 on 1048576 cobounds:1048576
#I  up 0: on 1048576 cobounds:1
```

...continued more

```
#I  abelian factor 5: 4096->4 central:false
#I  down
#I  module of dimension 1 subspace 0
#I  module of dimension 2 subspace 0
#I  module of dimension 10 subspace 0
#I  up 2:3 on 1024 cobounds:1
#I  up 0: on 1024 cobounds:1
#I  abelian factor 6: 4->1 central:false
#I  down
#I  module of dimension 2 subspace 0
#I  up 2:3 on 4 cobounds:4
#I  up 0: on 4 cobounds:1
<matrix group of size 3456 with 9 generators>
gap> time;   # about 50 minutes
2828467
gap> Size(g)/Size(last);
6614249635840
```