Extensions of PcGroups by rewriting systems

Jack Schmidt

University of Kentucky

2007-09-13

A package for computing extensions of pc-groups by groups given by a confluent rewriting system is in preparation. The GAP implementation was initially dificult, but the new linear algebra proposal and implementation by Max Neunhoeffer greatly assisted. The GAP implementation also allowed a novel use of collectors to mimic linear combinations of collectors. Current implementation bottlenecks are in the rewriting process, but could benefit from those with expertise in collectors. Future work will involve providing fundamental algorithms for groups in this data representation.

Problem

- For a finite group G and fixed prime p, let $F_i(G)$ be the ith term of the lower central series of $O_p(G)$. Define $G_i = G/F_i(G)$.

- Given a finite group G with $G_{k-1} \not\cong G$ and $G_k \cong G$, and given G-modules V, consider all groups H with $H_k \cong G$ and $F_i(H)/F_{i+1}(H) \in V$ for all $i \geq k$.

- Produce algorithm for given p, G, V that takes H_i as input, and outputs all H_{i+1}. Run in time and space polynomial in i.

Solution

- Old solutions by Eick, Holt cannot work (only soluble, or only $i = k$)
- Old solutions are the right way, just use wrong datatype
- New datatype makes output length polynomial in i, so algorithm is possible
- Brief sketch of cohomology:
    - $\theta : G \times G \to N$ finishes extension
    - Not all $\theta \in N^{G \times G}$ work, kernel of linear operator
    - For p-groups, only need $\theta(g, g^{p-1})$ and $\theta(g^{-1}h^{-1}, gh)$
    - Can rewrite all products in p-group using this

Overview

- Rewriting systems are important to CGT
- Difficulties in implementation:
  1. Linear Algebra - method selection
  2. Rewriting systems - poor support
  3. Collectors - badly adapted to this case
- Success in implementation:
  1. CVEC fixes most linear algebra concerns
  2. One collector can easily replace millions
- Future work:
  1. RWS cleanup in GAP
  2. Permutations given by RWS

Rewriting systems

- Formalize simplification based on linear ordering of words
- Can explicitly refine normal series, like pcgs
- Can be expanded to give coset representatives: membership test for subgroups, permutation rep
- Expensive to find
- Can be very large
- Multiplication can be expensive (even in PcGroups)

Quick formal definition

- For X a monoid generating set for a group G, form free monoid X^*
- Define well-ordering on X^* with $x \leq y$ iff $axb \leq ayb$ for $a, b, x, y \in X^*$
- Given $g \in G$, define $g^* \in X^*$ as smallest element which evaluates to g
- $\mathit{Rules}(G, X, \leq) = \{(x^*g^*y^*, (xgy)^*) : g \in G, x, y \in X, (xg)^* = x^*g^*, (gy)^* = g^*y^*, (xgy)^* \neq x^*g^*y^*\}$
- Given rule (y, x) then $x \leq y$, and one should replace ayb with axb. Well-ordering $\Rightarrow$ stops.

Rewriting systems for extensions

- Given a finite presentation of an extension, how do you multiply elements?
$$(q,k)\cdot(q',k') = (q'',k'')$$
- Easy to find a quotient part, but one quickly flounders finding the kernel part, since multiple words q_1, q_2 represent same quotient element, but $(q_1, 1) \neq (q_2, 1)$.
- Rewriting systems form a systematic answer, each rule of quotient gets an element of the kernel, a **tail**
- Ensures $(q,k)^* = q^* k^*$

Finding extensions

- Given a rewriting system for G/N and a basis for G-module N, the only missing information for an extension is a vector in $N^{Rules(G/N)}$
- Not all vectors work, but there is a systematic **confluence test**
- Test ends up being a nullspace calculation for an $|Rules(G/N)| \times |Overlaps(G/N)|$ matrix
- Grows as n^5 where n is composition length of solvable radical of G/N, polynomial but large

Implementations

- First, Initial implementation in C for $GF(2)$-modules
- Then, GAP implementation with some kernel hacking
- Now, GAP implementation with CVEC and primitive RWS
- Future, GAP implementation with CVEC and full RWS support
- For PcGroups, already implemented by B. Eick in lib/twocohom.g*

Main components

- Rewriting engine for original quotient with rule-reporting

- Arithmetic engine for iterated kernel (*p*-core) including automorphisms induced by quotient

- Linear algebra to find nullspaces

Software problems

- C code had barbaric p-group collectors, very slow
- GAP p-group collectors ill-suited to large elementary abelian subgroup, but no general solution
- GAP is slow to evaluate automorphisms on groups like this
- GAP code had barbaric linear algebra; cannot use method selection, kernel methods tend to segfault
- GAP RWS group code mostly buggy, no separation of RWS from PCGroup, no specification of rules, etc.

Math problems

- Not all groups have small rewriting systems
- Many coclass trees are “bushy” and while single depth-first path is fast, full breadth-first is exponential
- Not all groups have small modules, so even initial linear algebra can be hard
- More or less all modules contribute cohomology

Software success

- CVEC is wonderful, though GAP LA still needs rewrite (type conversion is awful and needed too often)
- Calculating the large matrix is effectively doing arithmetic in a basis of proposed extensions – just pretend each is an extension and use a collector
- Even better, take their subdirect product, just one collector with large elementary abelian subgroup (and write special purpose code to set it up)

Math success

- Able to explore such interesting groups as $SL(2,7).7^4.7^5.7^6.7^7.7^6.7^5.7^4.7^3$ and $A_7.2^6.2^{14}.2^{14}.2^{14}.2^{14}.2^{14}.2^{14}$
- Found "barren" branch with no twigs in $SL(2,p)$ mod p corresponding to $SL(2,\hat{\mathbb{Z}}_p)$
- Found first cohomology in constant time
- Found some second cohomology in constant time

Future software

- Need consistently written linear algebra methods
- Need workable support for rewriting system groups
- Need basic algorithms for groups given as extensions of pc-groups by rewriting systems
- Explore coset rewriting as a type of permutation group
- Use PcGroup extensions and “stable elements” to count dimension, then stop constructing nullspace when dimension agrees

Future math

- Prove permutation representation cannot work (example where degree increases too quickly, but not a “straw-man” like dihedral groups)

- Use local control (Glauberman, Holt, et al.) to allow even smaller collectors

- Use inflation-restriction sequence to give quicker bounds on dimension and possibly for direct gain

- Generalize from $SL(2, p)$ to $SL(2, p^n)$ and thus from $SL(2, \hat{\mathbb{Z}}_p)$ to $SL(2, R)$ for other complete DVRs R

Summary

- Rewriting systems are useful for extensions
- Important to improve linear algebra and collection in GAP
- RWS in GAP needs rewriting