Polycyclic Groups (PART 4)

G fp group

$G / P_k(G)$

$G \twoheadrightarrow G/P_1(G)$

epimorphism

$H = G/P_k(G) = F/R$

$G \twoheadrightarrow H$

$K = G/P_{k+1}(G)$

$H^* = F/[F, R]R^p$

will be adding 'tail' to presentation ie adding generators w/ defining rels.

How do we compute K?

Thm: K is an image of H^*.

algorithmically, that means only need to add a few rels.

Define

$$\begin{aligned} \tau : G &\to K \\ g &\mapsto \theta(g) \cdot u_g \end{aligned}$$

$$\begin{tikzcd} 1 \arrow[r] & P_k(G)/P_{k+1}(G) \arrow[r] & K \arrow[r] & H \arrow[r] & 1 \\ & & G \arrow[u, "\tau"] \arrow[ur, "\theta"'] & & \end{tikzcd}$$

- $u_g \in P_k(G)/P_{k+1}(G)$ b/c short exact seq
- u_g is central of order p

∘ relators of $G \xrightarrow{\tau} 1$

$$r \mapsto \underset{\overset{\shortparallel}{1}}{\theta(r)} \cdot w_r = \underset{\overset{\shortparallel}{1}}{w_r}$$

We have to eliminate the new generator:

Beginning generators : $a_1, \dots, a_n$
new gens : $a_{n+1}, \dots, a_{n+t}$

Denote $M = \langle a_{n+1}, \dots, a_{n+t} \rangle$
$N := \operatorname{Ker}(K \to H)$.

<u>Thm</u> If we collect the set of words in $a_1, \dots, a_n$ in the pc-pres of H^* using the consistency rules. We obtain a set $S \subseteq M$.
Further, if we evaluate the relations of G via θ in the pc-pres of H^*, we get a set of words $T \leq M$.
Then,
$$N = M / \langle S \cup T \rangle .$$

ex: $p = 2$.

$$G = \langle b_1, b_2, b_3 \mid b_1 b_2 = b_3,\ b_2 b_3 = 1,\ b_3 b_1 = b_2 \rangle$$

$$H = G / P_1(G) = \langle a_1, a_2 \mid a_1^2 = a_2^2 = [a_2, a_1] = 1 \rangle$$

$$G \longrightarrow G/P_1(G): \quad \begin{aligned} b_1 &\mapsto a_1 \\ b_2 &\mapsto a_2 \\ b_3 &\mapsto a_1 a_2 . \end{aligned}$$

2-cover of H

$$H^* = pc\langle a_1, \dots, a_5 \mid [a_2, a_1] = a_3,\ a_1^2 = a_4,\ a_2^2 = a_5 \rangle$$

Look at im of rels in G:

$$\begin{array}{ccc} b_1 b_2 = b & b_2 b_3 = b_1 & b_3 b_1 = b_2 \\ \downarrow & \downarrow (\dots) & \downarrow (\dots) \\ a_1 a_2 = a_1 a_2 & a_5 = a_3 & a_4 = a_3 . \end{array}$$

a pc-pres of $K = G/P_2(G)$

$$pc\langle a_1, a_2, a_3 \mid [a_2, a_1] = a_3,\ a_1^2 = a_3,\ a_2^2 = a_3 \rangle$$

$\cong Q_8$ (quaternions)

$$\begin{aligned} b_1 &\mapsto a_1 \\ b_2 &\mapsto a_2 \\ b_3 &\mapsto a_1 a_2 \end{aligned} \qquad G \longrightarrow G/P_2(G)$$

Repeat for $H = G/P_2(G)$. It turns out:

$$G/P_3(G) = Q_8 .$$

```
# p-quotients; basic functions in GAP

F := FreeGroup("a", "b", "c");
H := F / ParseRelators(F,"ab=c,bc=a,ca=b");
quot := PQuotient(H, 2);
epi:=EpimorphismQuotientSystem(quot);
StructureDescription(Image(epi));
```

GAP Demo:

can specify bound in case there are too many quotient

```
quot := PQuotient(G,2);
epi := EpimorphismQuotientSystem(quot);
```

OR `epi := EpimorphismPGroup(G,2);` in 1 operation, but less good.

NILPOTENT QUOTIENT ALGORITHM

G fp, lower central series:

$$G = \gamma_1(G) \geqslant \gamma_2(G) \geqslant \dots$$

$G/\gamma_k(G)$ largest nilp of step $k-1$ quotient of G.

$$G/\gamma_2(G) = G^{ab}$$

$$G \to G^{ab}$$

polycyclic

$$H = G/\gamma_k(G)$$

$$K = G/\gamma_{k+1}(G)$$

- define cover group of H.

  if $H = F/R$

  then $H^* = F/[F,R]$

  no RP!

  (can do consistency check same as before)

can do as long as want FINITE QUOTIENT

- $H^* \longrightarrow K$.

SOLVABLE QUOTIENT ALGORITHM

G fp,

$$G \geqslant G' \geqslant G'' \geqslant \dots$$

derived subgroup

$$G/G^{(a)}, \quad F/\gamma_k(F).$$

Demo

bound order in case too big

epi := EpimorphismNilpotentQuotient(G, 100);

epi := EpimorphismSolvableQuotient(G);

s := SQ(G, 1000);

solvable quotient — maximal size of grp

> produces a record of the group, w/ image, map, source.

NOTE! We cannot compute 2-covers in GAP.

```
# Different kinds of quotients of H:

epi := EpimorphismNilpotentQuotient(H);
NilpotencyClassOfGroup(Image(epi));
StructureDescription(Image(epi));

epi := EpimorphismNilpotentQuotient(FreeGroup(2),3);
# doesn't work

F:=FreeGroup(2);
epi:=EpimorphismNilpotentQuotient(F/[F.1^2,F.2^3]);
StructureDescription(Image(epi));

record := SQ(H, 1000);
StructureDescription(record.image);
epi:=EpimorphismSolvableQuotient(H, 1000);
```

PACKAGES IN GAP

can enforce an identity

- anu pq :
  - p - quotient
  - p - group gen. alg
  - std presentation of finite p-group
  - Fast isomorphism checks for p-group.

Pq(fp group : options);

OPTIONS
- Prime := p;
- Exponent := n;
- Metabelian
- Identities := [list of functions].
- ClassBound := c

PqPCover same options

Demo

ex1:

LoadPackage("anupq");

H := Klein four group (fp)

HH:= PqPCover(H: Prime := 2);

ex2: $F_2 / P_3(F)$, $p = 2$.

G:= Pq(FreeGroup(2): Prime := 2,
ClassBound := 3);

ex3: n-Engel group

$[x, \underbrace{y, \ldots, y}_{n}] = 1 \quad \forall x, y \in G.$

```
LoadPackage("anupq");

# 2 cover of C2 x C2:
F := FreeGroup("a","b");
H := F/ParseRelators(F, "[a,b]=a2=b2=1");
HH:=PqPCover(H: Prime := 2);
PrintPcpPresentation(PcGroupToPcpGroup(HH));


# 2-quotient of a free group of rank 2 of 2-class 3:

Pq(FreeGroup(2): Prime := 2, ClassBound := 3);

# 3-quotient of a certain fp-group:

F := FreeGroup("a","b");
H := F / ParseRelators(F, "a9=b9=1");
Pq(H: Prime := 3, ClassBound := 4);


# Metabelian quotient of the above fp-group:
Pq(H: Prime := 3, ClassBound := 7, Metabelian := true);
```

NILPOTENT PKG

ex:

```
LoadPackage("nq");
G := NilpotentQuotient(FreeGroup(2), 5)
```

class bound

ex:

```
H := F/R;
NilpotentQuotient(H, idgens := [F.1, F.2]);
```

ex:

```
NilpotentEngelQuotient(FreeGroup(3), 4);
```

```
# Nilpotent quotient package
LoadPackage("nq");

# Free nilpotent group of class 5 on 3 generators
NilpotentQuotient(FreeGroup(3), 5);

# Free 3-Engel group on 3 generators:
F := FreeGroup("x", "y", "a", "b", "c");
AssignGeneratorVariables(F);
G := NilpotentQuotient(F/[LeftNormedComm([x,y,y,y])]: idgens := [x, y]);
Exponent(LowerCentralSeriesOfGroup(G)[5]);

#Free 4-Engel group on 2 generators:
NilpotentEngelQuotient(FreeGroup(2), 4);

# Largest nilpotent group generated by two right 3-Engel elements:
F := FreeGroup("x", "a", "b");
AssignGeneratorVariables(F);
H := F / [LeftNormedComm([a,x,x,x]), LeftNormedComm([b,x,x,x])];
G := NilpotentQuotient(H: idgens := [x]);
NilpotencyClassOfGroup(G);
a := G.1; b:= G.2;
Order(LeftNormedComm([b,a,a,b]));
```