The conjugacy problem and its variations

Bettina Eick

September 24, 2018

1 The problem

Let G be a group given by a pc presentation and let $g, h \in G$. The conjugacy problem then asks whether there exists $x \in G$ with $g^x = h$ and, if so, then to determine one such x. The centralizer problem asks to determine a generating set for $C_G(g) = \{y \in G \mid g^y = g\}$. This is a very prominent problem:

(a) It is one of the fundamental problems for all types of groups (together with intersection and normalizer / conjugacy of subgroups).
(b) It has applications in cryptography (where one looks for groups with fast multiplication and slow multiple conjugacy problem).
(c) Its solution is typical for algorithms for polycyclic groups (uses induction and linear algebra in the induction step).

2 The induction

Let $G = N_1 \geq N_2 \geq \ldots \geq N_l \geq N_{l+1} = \{1\}$ be a normal series of G whose factors are either elementary abelian or free abelian.

1 Theorem: *Such a normal series exists and can be computed readily.*

Proof: Compute the derived series and refine this as desired. Subgroups in a polycyclic group can be described by polycyclic sequences which are induced with respect to the defining presentation. •

Suppose by induction that we have determined $x \in G$ with $g^x \equiv h \bmod N_l$ and $C = \{y \in G \mid g^y \equiv g \bmod N_l\}$. The aim is to determine the same information modulo $N_{l+1} = \{1\}$. Concentrate on centralizer problem (conjugacy is dual). Denote $N = N_l$ to shorten notation.
(Picture) $G - C - N - 1$ with subgroup $C_G(g)$.

2 Lemma: $C_G(g) = C_C(g)$.

Proof: This follows as $C_G(g) \leq C$. •

The definition of C implies that for each $c \in C$ there exists $n_c \in N$ with $g^c = gn_c$. Write

$$\delta : C \to N : c \mapsto n_c.$$

3 Lemma: *δ is a derivation with $ker(\delta) = \{c \in C \mid n_c = 1\} = C_C(g) = C_G(g)$.*

Proof: $\delta(1) = 1$ is obvious. Consider $\delta(ab)$ and note that $g^{ab} = (g^a)^b = (gn_a)^b = g^b n_a^b = gn_b n_a^b$. As N is abelian, it follows that $\delta(ab) = \delta(a)^b \delta(b)$. •

Let $\varphi : C \to Aut(N)$ denote the natural action by conjugation of C on N. We now assume that $N = \cong \mathbb{Z}^d$ or $N \cong \mathbb{F}_p^d$ for some prime p. Write $R = \mathbb{Z}$ or $R = \mathbb{F}_p$ depending on the case. Then $Aut(N) = GL(d, R)$ and $N = R^d$. We define

$$\rho : C \to GL(d+1, R) : c \mapsto \begin{pmatrix} \varphi(c) & 0 \\ \delta(c) & 1 \end{pmatrix}.$$

4 Theorem: *ρ is a group homomorphism and $C_C(g) = Stab_{C,\rho}(v)$ for $v = (0, \ldots, 0, 1)$.*

Proof: ρ is an affine action combined from a linear action and a derivation. It is well-known that this yields a group homomorphism. Further, note that $Stab_{C,\rho}(v) = ker(\delta) = C_C(g)$. •

Hence we have now translated the computation of a centralizer to the construction of generators for a stabilizer of a vector under a linear action. Dual to that a solution of the conjugacy problem translates to solving a corresponding orbit problem. There are two cases:

(1) $R = \mathbb{F}_p$. Then $GL(d+1, R)$ is finite and orbits and stabilizers can be computed with well-known methods. The acting group is polycyclic and this implies that there are particularly efficient methods for this purpose available.
(2) $R = \mathbb{Z}$. Then $GL(d+1, R)$ is infinite!

3 Computing stabilizers

Eick & Ostheimer (2003) exhibit the first (and so far only) practical algorithm to compute orbits and stabilizers for the action of polycyclic groups as subgroups of $GL(d+1, \mathbb{Z})$. The algorithm is quite complex and makes use of the structure of polycyclic subgroups of $GL(d+1, \mathbb{Z})$ as discovered by Dixon (1985).
The algorithm is based on work by Dixon ().

```
finite case:
------------

G := Group(
[ [ [ 0, 1, 0, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ], [ 1, 0, 0, 0 ] ],
  [ [ 0, 1, 0, 0 ], [ 1, 0, 0, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ] ] ]);

gap> iso := IsomorphismPcpGroup(G);;
gap> H := Image(iso);
Pcp-group with orders [ 2, 3, 2, 2 ]
gap> mats := List(Cgs(H), x -> PreImagesRepresentative(iso,x));
[ [ [ 1, 0, 0, 0 ], [ 0, 0, 1, 0 ], [ 0, 1, 0, 0 ], [ 0, 0, 0, 1 ] ],
  [ [ 1, 0, 0, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ], [ 0, 1, 0, 0 ] ],
  [ [ 0, 1, 0, 0 ], [ 1, 0, 0, 0 ], [ 0, 0, 0, 1 ], [ 0, 0, 1, 0 ] ],
  [ [ 0, 0, 0, 1 ], [ 0, 0, 1, 0 ], [ 0, 1, 0, 0 ], [ 1, 0, 0, 0 ] ] ]

gap> StabilizerIntegralAction(H, mats, [1,0,0,0]);
Pcp-group with orders [ 2, 3 ]
gap> StabilizerIntegralAction(H, mats, [1,0,0,1]);
Pcp-group with orders [ 2, 2 ]


unipotent case:
---------------

gap> G := UnitriangularPcpGroup(4,0);
Pcp-group with orders [ 0, 0, 0, 0, 0, 0 ]
gap> mats := G!.mats;
[ [ [ 1, 1, 0, 0 ], [ 0, 1, 0, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ] ],
  [ [ 1, 0, 0, 0 ], [ 0, 1, 1, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ] ],
  [ [ 1, 0, 0, 0 ], [ 0, 1, 0, 0 ], [ 0, 0, 1, 1 ], [ 0, 0, 0, 1 ] ],
  [ [ 1, 0, 1, 0 ], [ 0, 1, 0, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ] ],
  [ [ 1, 0, 0, 0 ], [ 0, 1, 0, 1 ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ] ],
  [ [ 1, 0, 0, 1 ], [ 0, 1, 0, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1 ] ] ]
gap> Cgs(G);
[ g1, g2, g3, g4, g5, g6 ]

gap> StabilizerIntegralAction(G, mats, [1,0,0,0]);
Pcp-group with orders [ 0, 0, 0 ]
gap> Cgs(last);
[ g2, g3, g5 ]
gap> StabilizerIntegralAction(G, mats, [1,1,1,1]);
Pcp-group with orders [ 0, 0, 0 ]
gap> Cgs(last);
```

```
[ g2*g4^-1, g3*g6^-1, g5*g6^-1 ]


field case:
-----------

gap> f := x^3 - 7*x + 1;
x^3-7*x+1
gap> Factors(f);
[ x^3-7*x+1 ]
gap> K := FieldByPolynomial(f);
<algebraic extension over the Rationals of degree 3>
gap> U := UnitGroup(K);
<group with 3 generators>
gap> b := Basis(K);
CanonicalBasis( <algebraic extension over the Rationals of degree 3> )
gap> u := GeneratorsOfGroup(U);
[ !-1, a, a^2+2*a-2 ]

gap> m2 := List(b, x-> Coefficients(b,x*u[2]));
[ [ 0, 1, 0 ], [ 0, 0, 1 ], [ -1, 7, 0 ] ]
gap> m3 := List(b, x-> Coefficients(b,x*u[3]));
[ [ -2, 2, 1 ], [ -1, 5, 2 ], [ -2, 13, 5 ] ]

G := Group(m2, m3);
gap> IsAbelian(G);
true
gap> Size(G);
infinity

gap> iso := IsomorphismPcpGroup(G);;
[ [ [ 0, 1, 0 ], [ 0, 0, 1 ], [ -1, 7, 0 ] ],
  [ [ -2, 2, 1 ], [ -1, 5, 2 ], [ -2, 13, 5 ] ] ] -> [ g1, g2 ]
gap> H := Image(iso);
Pcp-group with orders [ 0, 0 ]
gap> mats := List(Cgs(H), x -> PreImagesRepresentative(iso,x));
[ [ [ 0, 1, 0 ], [ 0, 0, 1 ], [ -1, 7, 0 ] ],
  [ [ -2, 2, 1 ], [ -1, 5, 2 ], [ -2, 13, 5 ] ] ]

gap> StabilizerIntegralAction(H, mats, [1,0,0]);
Pcp-group with orders [  ]
gap> StabilizerIntegralAction(H, mats, [1,1,0]);
Pcp-group with orders [  ]
gap> StabilizerIntegralAction(H, mats, [1,1,1]);
```

```
Pcp-group with orders [  ]


a := [[-1, 1, 8],[-5, -2, 20],[-1, 0, 5]];
b := [[-47, -24, 192],[0,1,0],[-12,-6,49]];
c := [[-23, 0, 96],[0,1,0],[-6, 0, 25]];

G := Group(a,b,c);
H := Image(IsomorphismPcpGroup(G));

gap> Cgs(StabilizerIntegralAction(H, [a,b,c], [1,0,0]));
[ g1^24*g2^-10572*g3^-17106 ]
gap> Cgs(StabilizerIntegralAction(H, [a,b,c], [0,1,0]));
[ g2, g3 ]
gap> Cgs(StabilizerIntegralAction(H, [a,b,c], [0,0,1]));
[ g1^24*g2^-12504*g3^-20232 ]
```