

**Algorithmische Gruppentheorie
und das GAP System**

Bettina Eick

Kassel, den 16. Mai 2003

- Kann man nur eine Ecke in Rubic's Cube drehen?
- Wie berechnet man die Galoisgruppe eines Polynoms?
- Was sind die möglichen Symmetriegruppen von Kristallen?
- Sind diese Graphen isomorph?

Probleme aus der algorithmischen Algebra,
insbesondere Gruppentheorie

Codierungstheorie

Was kann GAP?

- Konstruktion von (linearen) Codes
- Untersuchung von Codes
- Decodieralgorithmen
- Isomorphietest von Codes
- Automorphismengruppen von Codes

Codes liefern Darstellungen von Gruppen.

Bemerkungen:

- Isomorphismen/Automorphismen von J. Leon
- Codierungstheorie in GAP in Delft entwickelt.

Beispiel: Der Golay Code

Finde ein irreduzibles Polynom in $\mathbb{F}_2[x]$,
welches $x^{23} - 1$ teilt:

```
gap> x := Indeterminate(GF(2));; SetName(x, "x");
gap> f := x^(23) - 1;
x^23+Z(2)^0
gap> ff := Factors(f);
[ x+Z(2)^0,
  x^11+x^9+x^7+x^6+x^5+x+Z(2)^0,
  x^11+x^10+x^6+x^5+x^4+x^2+Z(2)^0 ]
gap> g := ff[2];
x^11+x^9+x^7+x^6+x^5+x+Z(2)^0
```

Konstruiere den entsprechenden zyklischen Code:

```
gap> C := GeneratorPolCode( g, 23, GF(2) );
a cyclic [23,12,1..7]3 code over GF(2)
```

Untersuche diesen Code:

```
gap> IsSelfDualCode( C );
false
gap> Redundancy(C);
11
gap> MinimumDistance(C);
7
gap> WeightDistribution(C);
[ 1, 0, 0, 0, 0, 0, 0, 253, 506, 0, 0, 1288,
  1288, 0, 0, 506, 253, 0, 0, 0, 0, 0, 0, 1 ]
```

Ist es der Golay-Code?

```
gap> CodeIsomorphism( BinaryGolayCode(), C );
(4,8,19,13)(5,11,12,22)(6,14)
          (9,10,21,23,17,16)(15,18)
```

Die Antwort ist ja und ein Isomorphismus wird angegeben!

Berechne die Automorphismengruppe von C :

```
gap> G := AutomorphismGroup(C);  
<permutation group of size 10200960>
```

Untersuche G :

```
gap> IsSimpleGroup(G);  
true  
gap> IsomorphismTypeInfoFiniteSimpleGroup(G);  
rec( series := "Spor", name := "M(23)" )
```

- Die Automorphismengruppe ist M_{23} , die Mathieugruppe auf 23 Punkten.
- Diese Gruppe ist eine der sporadischen endlichen einfachen Gruppen.

Graphentheorie

Was kann GAP?

- Konstruktion von Graphen
- Untersuchung von Graphen
- Isomorphietest von Graphen
- Automorphismengruppen von Graphen

Graphen liefern Darstellungen von Gruppen.

Bemerkungen:

- Isomorphien/Automorphismen von B. McKay
- Graphentheorie in GAP in London entwickelt von L. Soicher.

Beispiel: Der Higman-Sims Graph

- Dieser Graph hat 100 Ecken und 1100 Kanten
- Eintippen der Kanten ist zu zeitaufwendig!
- Verwende Gruppentheorie zur Reduktion:

Starte mit dem leeren Graphen auf 100 Ecken mit Gruppe G :

```
gap> G;  
<permutation group of size 887040>  
gap> HS := NullGraph( G, 100 );;
```

Füge Kanten hinzu:

```
gap> AddEdgeOrbit( HS, [1,100] );
gap> AddEdgeOrbit( HS, [100, 1] );
gap> for i in [ 3, 16, 19, 6, 21, 5, 40 ] do
    AddEdgeOrbit( HS, [i, 23] );
    AddEdgeOrbit( HS, [23, i] );
od;
```

Mit jeder Kante wird auch jedes Bild der Kante unter G hinzugefügt.

Wie sieht der Graph jetzt aus?

```
gap> Length( UndirectedEdges(HS) );
1100
```

Es ist also tatsächlich ein Graph mit 1100 Kanten.

Weitere Eigenschaften von HS:

```
gap> VertexDegrees(HS);
```

```
[ 22 ]
```

```
gap> IsSimpleGraph(HS);
```

```
true
```

```
gap> Diameter(HS);
```

```
2
```

```
gap> Girth(HS);
```

```
4
```

Betrachte die Automorphismengruppe von HS:

```
gap> A := AutGroupGraph(HS);
<permutation group with 10 generators>
gap> B := DerivedSubgroup(A);
<permutation group with 3 generators>
gap> Size(B);
44352000
gap> Size(G);
887040
```

Die Gruppe B ist eine endliche einfache Gruppe:

```
gap> IsSimpleGroup(B);
true
gap> IsomorphismTypeInfoOfFiniteSimpleGroup(B);
rec( series := "Spor", name := "HS" )
```

Die Gruppe B heißt Higman-Sims-Gruppe.

Polyzyklische Gruppen

Was kann GAP?

- Diverse Strukturuntersuchungen
- Rechnen mit polyzyklischen Präsentationen
- Rechnen mit polyzyklischen Matrixgruppen
- Rechnen mit polyzyklischen endlich präsentierten Gruppen

GAP-Projekt von B. Assmann, B.E., W. Nickel

Definition und Beispiele

Eine Gruppe G heißt polyzyklisch, wenn sie eine Reihe $G = G_1 \triangleright G_2 \triangleright \dots \triangleright G_n \triangleright G_{n+1} = 1$ mit zyklischen Faktoren G_i/G_{i+1} hat

Beispiele:

- endlich erzeugte nilpotente Gruppen
- endliche auflösbare Gruppen
- viele kristallographische Gruppen
- viele fast-kristallographische Gruppen

Elementare Eigenschaften:

- endlich erzeugt und auflösbar
- Unter- / Faktorgruppen sind polyzyklisch
- Anzahl der unendlichen Faktoren G_i/G_{i+1} ist eine Invariante (Hirschlänge)

Polyzyklische Präsentationen

Die Reihe $G = G_1 \triangleright G_2 \triangleright \dots \triangleright G_n \triangleright G_{n+1} = 1$ liefert zwei Sequenzen

- eine polyzyklische Sequenz (g_1, \dots, g_n) definiert durch $G_i = \langle g_i, G_{i+1} \rangle$
- die relativen Ordnungen (r_1, \dots, r_n) definiert durch $r_i = [G_i : G_{i+1}] \in \mathbb{N} \cup \{\infty\}$

Eine polyzyklische Sequenz (g_1, \dots, g_n) definiert eine polyzyklische Präsentation mit Relationen:

$$\begin{aligned}
 g_i^{g_j} &= g_{j+1}^{e(i,j,j+1)} \dots g_n^{e(i,j,n)} \quad (j < i), \\
 g_i^{g_j^{-1}} &= g_{j+1}^{f(i,j,j+1)} \dots g_n^{f(i,j,n)} \quad (j < i \text{ und } r_j = \infty), \\
 g_i^{r_i} &= g_{i+1}^{\ell(i,i+1)} \dots g_n^{\ell(i,n)} \quad (r_i < \infty)
 \end{aligned}$$

Wortproblem: "Collection-Algorithmus"

Andere Darstellungen

- Problem: Polyzyklischen Präsentationen bilden spezielle Darstellungsform
- Brauchen: Umwandlung der klassischen Darstellungen in polyzyklische Präsentationen

Beispiele:

- Endlich präsentierte Gruppe \rightarrow
Polyzyklische Präsentation
(Lo 1996, Eick und Niemeyer 2001)
- Matrixgruppe über \mathbb{Z} oder \mathbb{Q} \rightarrow
Polyzyklische Präsentation
(Ostheimer 1996, Assmann und Eick 2003)

Elementare Probleme:

- Die Ordnung und die Hirschlänge von G
- Ist G abelsch, nilpotent oder überauflösbar?

Klassische Probleme:

- Zentralisatoren $C_G(g)$, und gilt $g^x = h$?
- Normalisatoren $N_G(U)$, und gilt $U^x = V$?
- Schnitte von Untergruppen $U \cap V$
- Automorphismengruppen $Aut(G)$

Weitere interessante Probleme:

- Torsionsuntergruppe $T(G)$
- Erweiterungen von G
- Untergruppen von "kleinem" Index
- $Fit(G)$, $Z(G)$ und andere Strukturuntergruppen

Implementation dazu in GAP + Kant:

- Polycyclic – (Eick und Nickel, 2000)
Algorithmen für polyzyklische Gruppen
- IPCQ – (Eick und Niemeyer, 2002)
Berechnung polyzyklischer Quotienten endlich präsentierter Gruppen
- Polenta – (Assmann 2003)
Berechnung polyzyklischer Präsentationen für Matrixgruppen über \mathbb{Q}
- ACLIB – (Dekimpe und Eick, 2001)
Bibliothek fast-kristallographischer Gruppen
- CrystCat – (Felsch und Gähler, 1999)
Bibliothek kristallographischer Gruppen

```
gap> G := AlmostCrystallographicGroup(4, 50, [1,-4,1,2]);
<matrix group of size infinity with 5 generators>
gap> iso := IsomorphismPcpGroup(G);
[ [ [ 1, -4, 1, 0, 1/2 ],
    [ 0, 0, -1, 0, 0 ],
    [ 0, 1, 0, 0, 0 ],
    [ 0, 0, 0, 1, 1/4 ],
    [ 0, 0, 0, 0, 1 ] ],
  [ [ 1, 0, -1/2, 0, 0 ],
    [ 0, 1, 0, 0, 1 ],
    [ 0, 0, 1, 0, 0 ],
    [ 0, 0, 0, 1, 0 ],
    [ 0, 0, 0, 0, 1 ] ],
  [ [ 1, 1/2, 0, 0, 0 ],
    [ 0, 1, 0, 0, 0 ],
    [ 0, 0, 1, 0, 1 ],
    [ 0, 0, 0, 1, 0 ],
    [ 0, 0, 0, 0, 1 ] ],
  [ [ 1, 0, 0, 0, 0 ],
    [ 0, 1, 0, 0, 0 ],
    [ 0, 0, 1, 0, 0 ],
    [ 0, 0, 0, 1, 1 ],
    [ 0, 0, 0, 0, 1 ] ],
  [ [ 1, 0, 0, 0, 1 ],
    [ 0, 1, 0, 0, 0 ],
    [ 0, 0, 1, 0, 0 ],
    [ 0, 0, 0, 1, 0 ],
    [ 0, 0, 0, 0, 1 ] ] ] -> [ g1, g2, g3, g4, g5 ]
```

```
gap> H := Image( iso);
Pcp-group with orders [ 4, 0, 0, 0, 0 ]

gap> IsTorsionFree(H);
true

gap> FittingSubgroup(H);
Pcp-group with orders [ 0, 0, 0, 0 ]

gap> C := Centre(H);
Pcp-group with orders [ 0, 0 ]

gap> nat := NaturalHomomorphism(H,C);
[ g1, g2, g3, g4, g5 ] -> [ g1, g2, g3, id, id ]

gap> F := Image(nat);
Pcp-group with orders [ 4, 0, 0 ]

gap> FiniteSubgroupClasses(F);
[ Pcp-group with orders [ 4 ]^G,
  Pcp-group with orders [ 4 ]^G,
  Pcp-group with orders [ 2 ]^G,
  Pcp-group with orders [ 2 ]^G,
  Pcp-group with orders [ 2 ]^G,
  Pcp-group with orders [ ]^G ]
```

```
gap> rep := List(last,
    x -> PreImage(nat, Representative(x)));
[ Pcp-group with orders [ 4, 0, 0 ],
  Pcp-group with orders [ 4, 0, 0 ],
  Pcp-group with orders [ 2, 0, 0 ],
  Pcp-group with orders [ 2, 0, 0 ],
  Pcp-group with orders [ 2, 0, 0 ],
  Pcp-group with orders [ 0, 0 ] ]

gap> List( rep, IsAbelian );
[ true, true, true, true, true, true ]

gap> List( rep, IsTorsionFree );
[ true, true, true, true, true, true ]

gap> List( rep, x -> Normalizer(H,x));
[ Pcp-group with orders [ 4, 0, 0 ],
  Pcp-group with orders [ 4, 0, 0 ],
  Pcp-group with orders [ 4, 0, 0 ],
  Pcp-group with orders [ 2, 0, 0 ],
  Pcp-group with orders [ 4, 0, 0 ],
  Pcp-group with orders [ 4, 0, 0, 0, 0 ] ]
```

```
gap> low := LowIndexNormalSubgroups(H,60);
[ Pcp-group with orders [ 0, 0, 0, 0 ],
  Pcp-group with orders [ 2, 0, 0, 0, 0 ],
  Pcp-group with orders [ 2, 0, 0, 0, 0 ],
  Pcp-group with orders [ 0, 0, 0, 0 ],
  Pcp-group with orders [ 0, 0, 0, 0 ] ]
```

```
gap> List( low, x -> Index(H,x));
[ 60, 60, 60, 60, 60 ]
```

```
gap> List( [1..60],
          x -> Length(LowIndexSubgroupClasses(H,x)));
[ 1, 3, 1, 7, 3, 5, 1, 28, 2, 11,
  1, 23, 3, 7, 3, 89, 3, 20, 1, 41,
  1, 9, 1, 112, 16, 15, 5, 47, 3, 25,
  1, 261, 1, 17, 3, 146, 3, 13, 3, 196,
  3, 25, 1, 79, 6, 15, 1, 449, 2, 74,
  3, 105, 3, 67, 3, 252, 1, 23, 1, 225 ]
```