

# **Algorithmen für freie Gruppen und ihre Automorphismen**

Christian Sievers  
Institut für Geometrie  
Technische Universität Braunschweig

# Ziele

GAP-Implementierung von Algorithmen für endlich erzeugte Untergruppen von freien Gruppen:

- (konstruktiver) Membership-test
- Indexberechnung
- Konjugiertheit von Untergruppen
- Normalisator
- *Durchschnitt*
- freies Erzeugendensystem
- Urbild unter Homomorphismen
- Berechnung von Inversen zu Isomorphismen

Außerdem ist eine endliche Präsentation der Automorphismengruppe bekannt, die in GAP implementiert wurde.

## Referenzen

- C. C. Sims. *Computation with Finitely Presented Groups*. Cambridge University Press, 1994.
- J.-C. Birget, S. Margolis, J. Meakin, und P. Weil. PSPACE-complete problems for subgroups of free groups and inverse finite automata. *Theoretical Computer Science*, 242:247–281, 2000.
- Roger C. Lyndon und Paul E. Schupp. *Combinatorial Group Theory*. 1977.
- Bernd Neumann. Die Automorphismengruppe der freien Gruppen. *Math. Annalen*, 107: 367–386, 1933.

# Freie Gruppen

- Elemente dargestellt durch Wörter in den Symbolen eines Alphabetes und ihren formalen Inversen
- kanonische Darstellung von Elementen als reduzierte Wörter
- Wortproblem trivial

## Algorithmen auf Wortebene

z.B. Konjugiertheit von Elementen

(gibt es zu  $u, v \in F$  ein  $w \in F$  mit  
 $u = w^{-1}vw$  ?)

Beispiel (in  $F = \langle a, b, c \rangle$ ):

$$b^{-1}ab^{-1}cab^{-1}cb$$

$$bacab^{-1}cab^{-1}a^{-1}b^{-1}$$

## Probleme

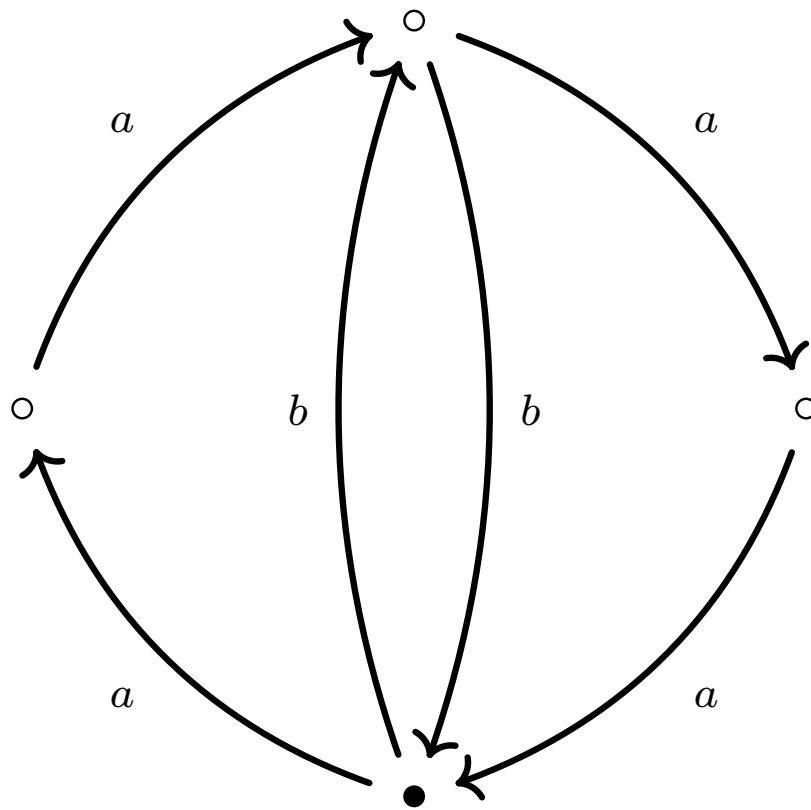
- unendliche Objekte
- eventuell unendlicher Index
- es gibt nicht einmal einen Automaten für die triviale Untergruppe der zyklischen freien Gruppe  $\langle a \rangle$

## Lösung

Es reicht, einen Automaten zu konstruieren, der für reduzierte Wörter einen Membership-test liefert. Dies ist für endlich erzeugte Untergruppen freier Gruppen immer möglich.

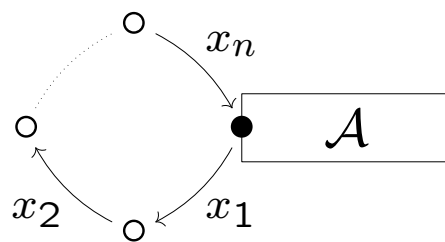
Diese Automaten ermöglichen es auch, die anderen Fragestellungen anzugehen.

# Ein Automat

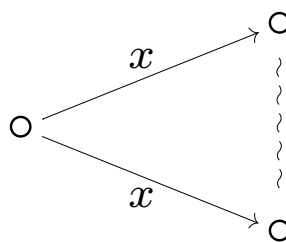


## Konstruktion eines Automaten zu einer endlich erzeugten Untergruppe

Sei  $\mathcal{A}$  ein Automat für  $G$ , dann ist ein Automat für  $\langle G, g \rangle$  mit  $g = x_1 x_2 \dots x_n$  gegeben durch



Zusammenfassen von äquivalenten Zuständen:



# Bestimmung eines freien Erzeugendensystems

nach Schreier durch Berechnung eines aufspannenden Baumes.

Durch Breitensuche erhält man sogar ein Nielsen-reduziertes Erzeugendensystem.

## Normalisator

erfordern die Bestimmung von Zuständen, die als neue Anfangs- und Endzustände genommen einen isomorphen Automaten ergeben.

## Konstruktiver Membership-test

erfolgt analog zur Reidemeister-Schreier-Prozedur durch erweiterte Automaten (schon bei der Konstruktion)

## Homomorphismen

Konstruktiver Elementtest ermöglicht die Berechnung eines Urbildes.

Bei einem Endomorphismus kann festgestellt werden, ob er surjektiv ist. Dann ist er ein Automorphismus (Hopfsche Eigenschaft), und die inverse Abbildung ist berechenbar.

# Durch Homomorphismen definierte Untergruppen

## Bild

Das Bild wird durch die Bilder der Erzeuger erzeugt.

## Kern

Ist der Kern weder trivial noch die ganze Gruppe, so ist er ein nichttrivialer Normalteiler von unendlichem Index und somit nicht endlich erzeugt.

## Fixpunkte

Die Gruppe der Fixpunkte ist endlich erzeugt und wäre deshalb durch einen Automaten repräsentierbar, aber es sind keine Methoden zu ihrer Berechnung bekannt.

```
gap> LoadPackage("FGA");;
```

```
-----  
Loading FGA 0.9 (Free Group Algorithms)  
by Christian Sievers (c.sievers@tu-bs.de).  
-----
```

```
gap> f := FreeGroup( "a", "b" );;  
gap> AssignGeneratorVariables( f );  
#I Assigned the global variables [ a, b ]
```

```
gap> g := Group( a^4, b^2, a^2*b );;
```

```
gap> b*a^2 in g;
```

```
true
```

```
gap> a^2 in g;
```

```
false
```

```
gap> FreeGeneratorsOfGroup( g );
```

```
[ a^2*b^-1, b*a^2, b^2 ]
```

```
gap> n := Normalizer( f, g );;
```

```
gap> RankOfFreeGroup( n );
```

```
2
```

```
gap> a^2 in n;
```

```
true
```

```
gap> Index( n, g );
```

```
2
```

```
gap> FreeGeneratorsOfGroup( n );
```

```
[ b, a^2 ]
```

```

gap> aut := AutomorphismGroup( n );
<group of size infinity with 3 generators>
gap> GeneratorsOfGroup( aut );
[ [ b, a^2 ] -> [ b^-1, a^2 ], [ b, a^2 ] -> [ a^2, b ],
  [ b, a^2 ] -> [ b*a^2, a^2 ] ]
gap> IsomorphismFpGroup( aut );
[ [ b, a^2 ] -> [ b^-1, a^2 ], [ b, a^2 ] -> [ a^2, b ],
  [ b, a^2 ] -> [ b*a^2, a^2 ] ] -> [ 0, P, U ]
gap> RelatorsOfFpGroup( Range( last ));
[ P^2, 0^2, 0*P*0*P*0*P*0*P, U^-1*0^-1*U^-1*0^-1*U*0*U*0,
  P*0*P*U*P*0*P*U, U*P*0*U*P*0*U*P*0 ]

```

```

gap> h := GroupHomomorphismByImages(
      f, f, [a, b], [a^2*b^2, a^2*b] );;
gap> PreImagesRepresentative( h, a^2 );
b*a^-1*b
gap> PreImagesRepresentative( h, a );
fail

```

```

gap> e := Enumerator(f);;
gap> b := Group(List(e{[1..Position(e,a^5)-1]},g->g^3));
Group(<161 generators>)
gap> Index( f, b );
27

```