Cubic Braid Groups

Talk on the Sage Days 94 in Saragossa

Cubic Braid Groups

Talk on the Sage Days 94 in Saragossa

following the introduction given on CoCalc

Welcome to the Cubic Braid Group page on CoCalc

This page contains a new class declaration to be used with sage. It deals with certain factor groups of the Artin braid group. This class is not integrated into the sage library, right now. You can test it here on CoCalc (if you are signed in to your own acount) or on your own computer (if you have got sage installed on it). Everyone is invited to help to improve this class. If you like to get full access to this project (as a collaborator) please contact "s.oehms@web.de".

To learn more about this new class you may follow one of the following links:

Introduction to the Cubic Braid Groups using jupyter notebook (.ipynb)

Introduction to the Cubic Braid Groups using sage worksheet (.sagews)

Download the introduction to the Cubic Braid Groups as PDF

Installation instruction:

Installation of the Cubic Braid Group class

Download

Go to the "Files"-Menue an click on the Icon on the right of "cbg.tgz" (cloud with download arrow) !

Overview

Contents

Overview

Contents

Introduction

This module is devoted to factor groups of the Artin braid groups, such that the images s_i of the braid generators have order three:

$$s_i^3 = 1$$

In general these groups have firstly been investigated by Coxeter, H.S.M in: "Factor groups of the braid groups, Proceedings of the Fourth Candian Mathematical Congress (Vancover 1957), pp. 95-122".

Coxeter showed, that these groups are finite as long as the number of strands is less than 6 and infinite elsewise. More explicitely the factor group on three strand braids is isomorphic to $SL(2,3)$, on four strand braids to $GU(3,2)$ and on five strand braids to $Sp(4,3) \times C_3$. Coxeter realized these groups as subgroups of unitary groups with respect to a certain hermitian form over the complex numbers (in fact over $\mathbb{Q}$ adjoined with a primitive 12-th root of unity).

Introduction

This module is devoted to factor groups of the Artin braid groups, such that the images s_i of the braid generators have order three:

$$s_i^3 = 1$$

In general these groups have firstly been investigated by Coxeter, H.S.M in: "Factor groups of the braid groups, Proceedings of the Fourth Candian Mathematical Congress (Vancover 1957), pp. 95-122".

Coxeter showed, that these groups are finite as long as the number of strands is less than 6 and infinite elsewise. More explicitely the factor group on three strand braids is isomorphic to $SL(2,3)$, on four strand braids to $GU(3,2)$ and on five strand braids to $Sp(4,3) \times C_3$. Coxeter realized these groups as subgroups of unitary groups with respect to a certain hermitian form over the complex numbers (in fact over $\mathbb{Q}$ adjoined with a primitive 12-th root of unity).

Introduction

This module is devoted to factor groups of the Artin braid groups, such that the images s_i of the braid generators have order three:

$$s_i^3 = 1$$

In general these groups have firstly been investigated by Coxeter, H.S.M in: "Factor groups of the braid groups, Proceedings of the Fourth Candian Mathematical Congress (Vancover 1957), pp. 95-122".

Coxeter showed, that these groups are finite as long as the number of strands is less than 6 and infinite elsewise. More explicitely the factor group on three strand braids is isomorphic to $SL(2,3)$, on four strand braids to $GU(3,2)$ and on five strand braids to $Sp(4,3) \times C_3$. Coxeter realized these groups as subgroups of unitary groups with respect to a certain hermitian form over the complex numbers (in fact over $\mathbb{Q}$ adjoined with a primitive 12-th root of unity).

Artin Braid Groups

```
sage: B3 = BraidGroup(3)
sage: braid5_2 = B3((-1, -1, -1, -2, 1, -2))
sage: braid5_2.plot()
```

Artin Braid Groups

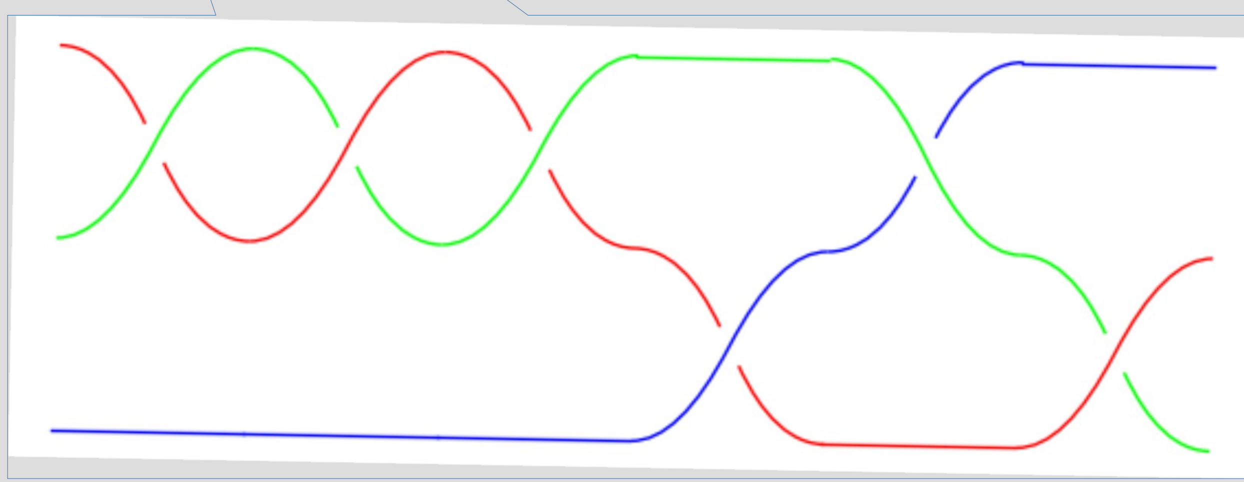
```
sage: B3 = BraidGroup(3)
sage: braid5_2 = B3((-1, -1, -1, -2, 1, -2))
sage: braid5_2.plot()
```

Artin Braid Groups

```
sage: B3 = BraidGroup(3)
sage: braid5_2 = B3((-1, -1, -1, -2, 1, -2))
sage: braid5_2.plot()
```

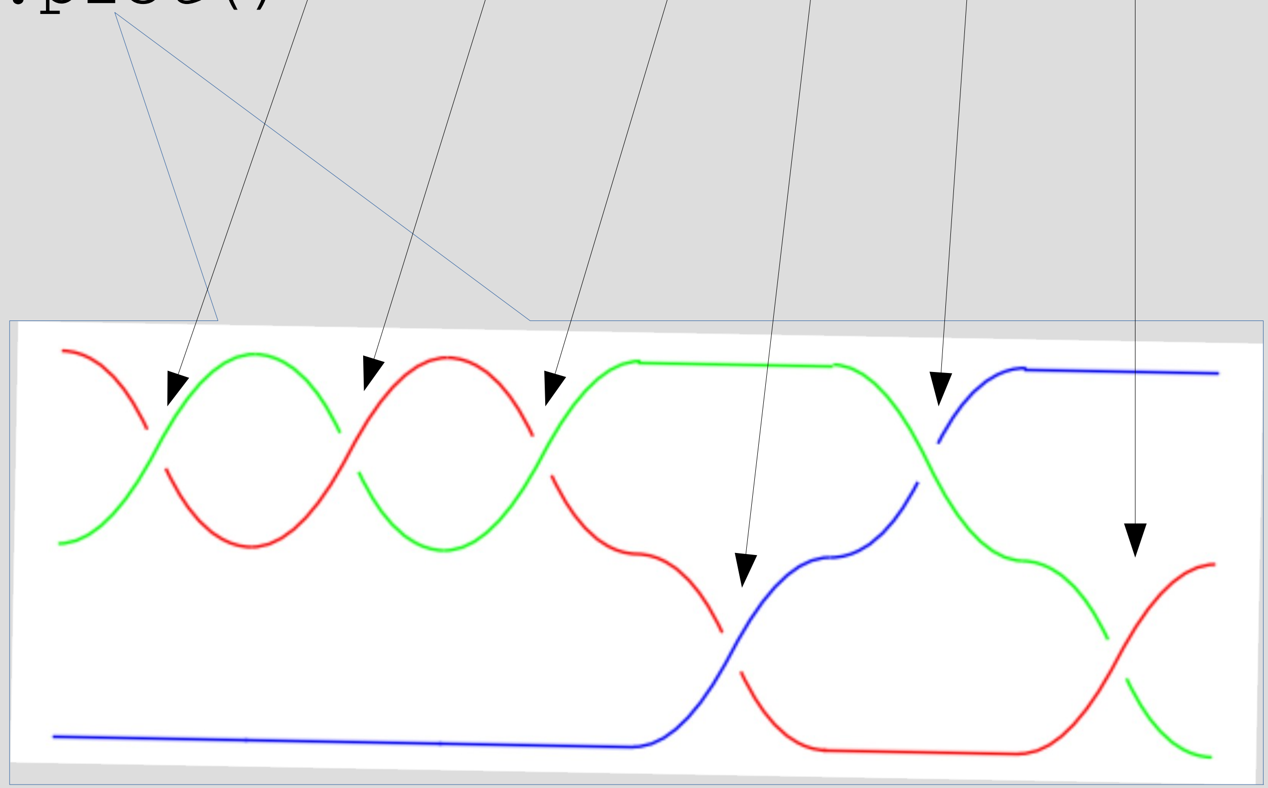
Artin Braid Groups

```
sage: B3 = BraidGroup(3)
sage: braid5_2 = B3((-1, -1, -1, -2, 1, -2))
sage: braid5_2.plot()
```



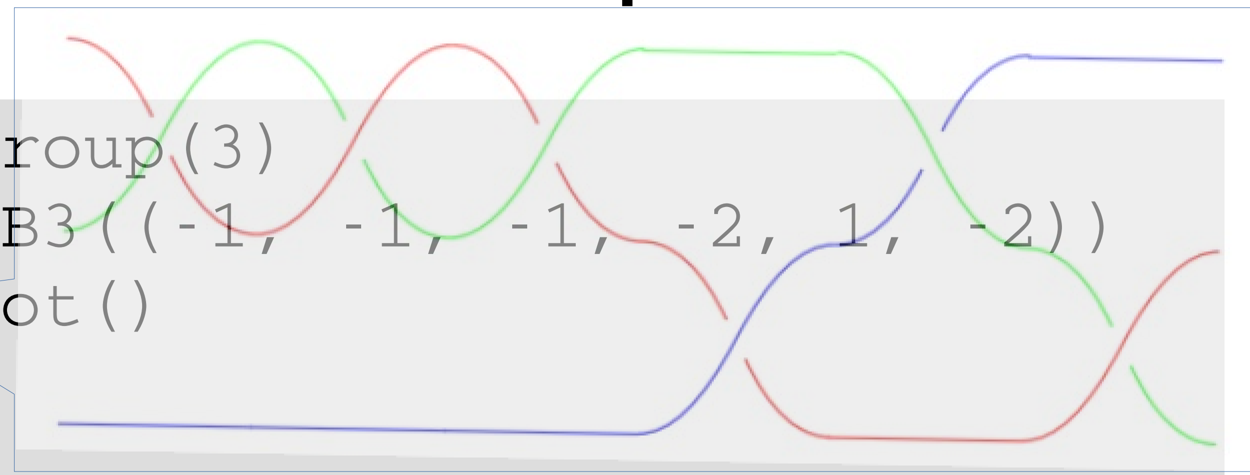
Artin Braid Groups

```
sage: B3 = BraidGroup(3)
sage: braid5_2 = B3((-1, -1, -1, -2, 1, -2))
sage: braid5_2.plot()
```



Artin Braid Groups

```
sage: B3 = BraidGroup(3)
sage: braid5_2 = B3((-1, -1, -1, -2, 1, -2))
sage: braid5_2.plot()
```

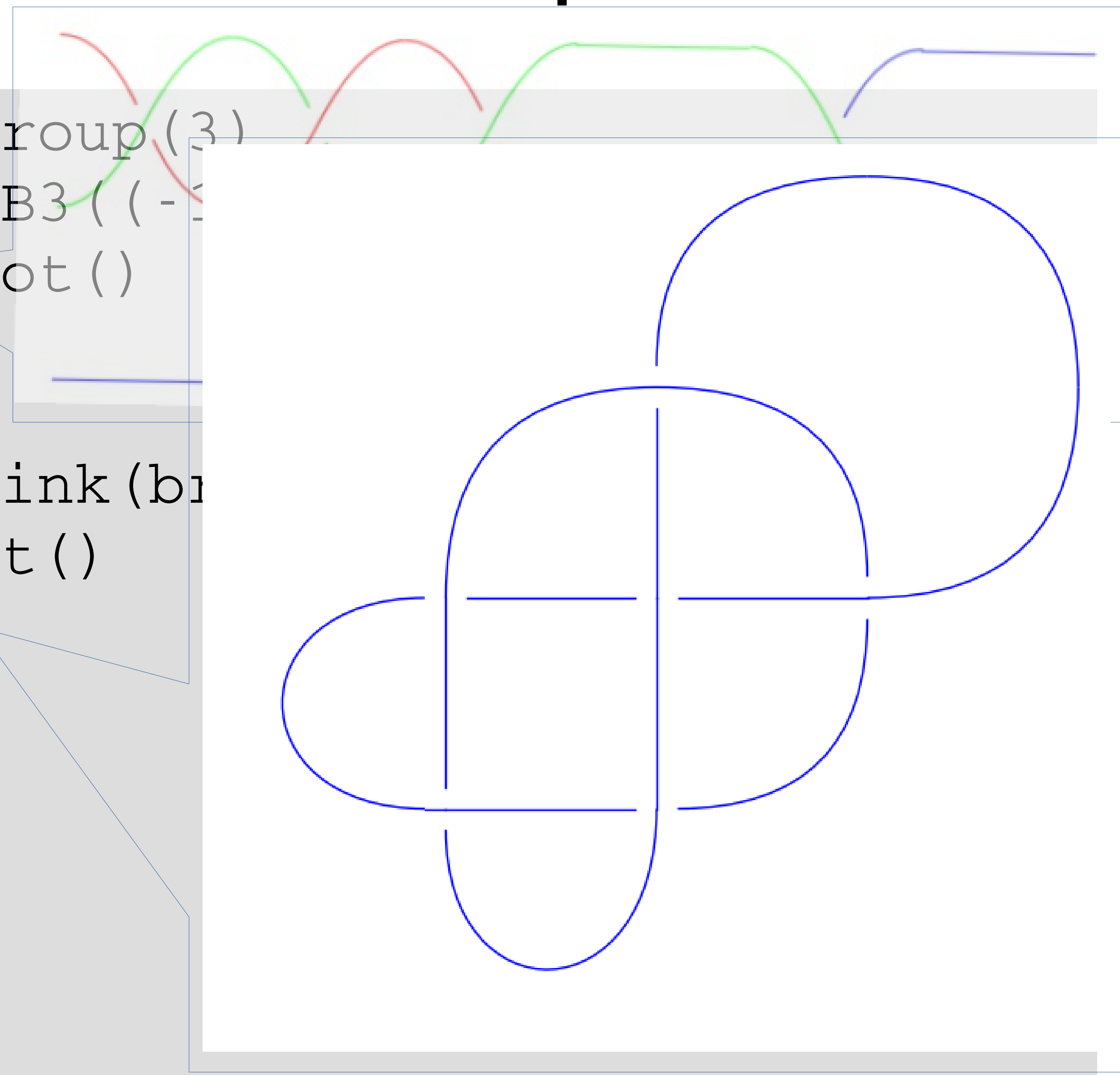


```
sage: knot5_2 = Link(braid5_2)
sage: knot5_2.plot()
```

Artin Braid Groups

```
sage: B3 = BraidGroup(3)
sage: braid5_2 = B3((-1
sage: braid5_2.plot()


sage: knot5_2 = Link(br
sage: knot5_2.plot()
```



Introduction

This module is devoted to factor groups of the Artin braid groups, such that the images s_i of the braid generators have order three:

$$s_i^3 = 1$$

In general these groups have firstly been investigated by Coxeter, H.S.M in: "Factor groups of the braid groups, Proceedings of the Fourth Candian Mathematical Congress (Vancover 1957), pp. 95-122".

Coxeter showed, that these groups are finite as long as the number of strands is less than 6 and infinite elsewise. More explicitely the factor group on three strand braids is isomorphic to $SL(2,3)$, on four strand braids to $GU(3,2)$ and on five strand braids to $Sp(4,3) \times C_3$. Coxeter realized these groups as subgroups of unitary groups with respect to a certain hermitian form over the complex numbers (in fact over $\mathbb{Q}$ adjoined with a primitive 12-th root of unity).

Introduction

Exponent 2
→ symmetric group

This module is devoted to factor groups of the Artin braid groups, such that the images s_i of the braid generators have order three:

$$s_i^3 = 1$$

In general these groups have firstly been investigated by Coxeter, H.S.M in: "Factor groups of the braid groups, Proceedings of the Fourth Candian Mathematical Congress (Vancover 1957), pp. 95-122".

Coxeter showed, that these groups are finite as long as the number of strands is less than 6 and infinite elsewise. More explicitely the factor group on three strand braids is isomorphic to $SL(2,3)$, on four strand braids to $GU(3,2)$ and on five strand braids to $Sp(4,3) \times C_3$. Coxeter realized these groups as subgroups of unitary groups with respect to a certain hermitian form over the complex numbers (in fact over $\mathbb{Q}$ adjoined with a primitive 12-th root of unity).

Introduction

This module is devoted to factor groups of the Artin braid groups, such that the images s_i of the braid generators have order three:

$$s_i^3 = 1$$

In general these groups have firstly been investigated by Coxeter, H.S.M in: "Factor groups of the braid groups, Proceedings of the Fourth Candian Mathematical Congress (Vancover 1957), pp. 95-122".

Coxeter showed, that these groups are finite as long as the number of strands is less than 6 and infinite elsewise. More explicitely the factor group on three strand braids is isomorphic to $SL(2,3)$, on four strand braids to $GU(3,2)$ and on five strand braids to $Sp(4,3) \times C_3$. Coxeter realized these groups as subgroups of unitary groups with respect to a certain hermitian form over the complex numbers (in fact over $\mathbb{Q}$ adjoined with a primitive 12-th root of unity).

Why are cubic braid groups interesting?

Why are cubic braid groups interesting?

The HOMFLY-PT Polynomial

The *HOMFLY-PT polynomial* $H(L)(a,z)$ (see [HOMFLY] and [PT]) of a knot or link L is defined by the skein relation

$$aH\left(\nwarrow\!\!\!\!\nearrow\right) - a^{-1}H\left(\nwarrow\!\!\!\!\nearrow\right) = zH\left()\,(\right)$$

and by the initial condition $H(\bigcirc)$=1.

Why are cubic braid groups interesting?

The HOMFLY-PT Polynomial

The *HOMFLY-PT polynomial* $H(L)(a, z)$ (see [HOMFLY] and [PT]) of a knot or link L is defined by the skein relation

$$aH\left(\times\right) - a^{-1}H\left(\times\right) = zH\left(\,)\,(\,\right)$$

and by the initial condition $H(\bigcirc)=1$.

quadratic skein relation

Why are cubic braid groups interesting?

The HOMFLY-PT Polynomial

The *HOMFLY-PT polynomial* $H(L)(a,z)$ (see [HOMFLY] and [PT]) of a knot or link L is defined by the skein relation

$$aH\left(\times\right) - a^{-1}H\left(\times\right) = zH\left()\,(\right)$$

and by the initial condition $H(\bigcirc)$=1.

The Kauffman Polynomial

The *Kauffman polynomial* $F(K)(a,z)$ (see [Kauffman]) of a knot or link K is $a^{-w(K)}L(K)$ where $w(L)$ is Computed?) and where $L(K)$ is the regular isotopy invariant defined by the skein relations

$$L(s_+) = aL(s), \qquad L(s_-) = a^{-1}L(s)$$

(here s is a strand and $s_\pm$ is the same strand with a $\pm$ kink added) and

$$L(\times) + L(\times) = z\left(L()\,() + L(\asymp)\right)$$

and by the initial condition $L(U) = 1$ where U is the unknot $\bigcirc$.

Why are cubic braid groups interesting?

The HOMFLY-PT Polynomial

The *HOMFLY-PT polynomial* $H(L)(a,z)$ (see [HOMFLY] and [PT]) of a knot or link L is defined by the skein relation

$$aH\left(\nwarrow\!\!\!\nearrow\right) - a^{-1}H\left(\nwarrow\!\!\!\nearrow\right) = zH\left(\,\rangle\langle\,\right)$$

and by the initial condition $H(\bigcirc)$=1.

cubic skein relation

The Kauffman Polynomial

The *Kauffman polynomial* $F(K)(a,z)$ (see [Kauffman]) of a knot or link K is $a^{-w(K}$ … here $w(L)$ is Computed?) and where $L(K)$ is the regular isotopy invariant defined by the skein rel…

$$L(s_+) = aL(s), \qquad L(s_-) = a^{-1}L(s)$$

(here s is a strand and $s_\pm$ is the same strand with a $\pm$ kink added) and

$$L(\times) + L(\times) = z\left(L(\,\rangle\langle\,) + L(\asymp)\right)$$

and by the initial condition $L(U) = 1$ where U is the unknot $\bigcirc$.

Why are cubic braid groups interesting?

Homfly-PT
invariant

Why are cubic braid groups interesting?

Homfly-PT invariant

Markov trace on Iwahori Hecke algebra

Why are cubic braid groups interesting?

Homfly-PT invariant	Markov trace on Iwahori Hecke algebra	deformation of group algebra of symmetric group

Why are cubic braid groups interesting?

Homfly-PT invariant

Markov trace on Iwahori Hecke algebra

deformation of group algebra of symmetric group

Kauffman invariant

Why are cubic braid groups interesting?

Homfly-PT invariant	Markov trace on Iwahori Hecke algebra	deformation of group algebra of symmetric group
Kauffman invariant	Markov trace on cubic Hecke algebra	

Why are cubic braid groups interesting?

Homfly-PT invariant	Markov trace on Iwahori Hecke algebra	deformation of group algebra of symmetric group
Kauffman invariant	Markov trace on cubic Hecke algebra	deformation of group algebra of cubic braid group

Why are cubic braid groups interesting?

Homfly-PT invariant	Markov trace on Iwahori Hecke algebra	deformation of group algebra of symmetric group
Kauffman invariant	Markov trace on cubic Hecke algebra	deformation of group algebra of cubic braid group

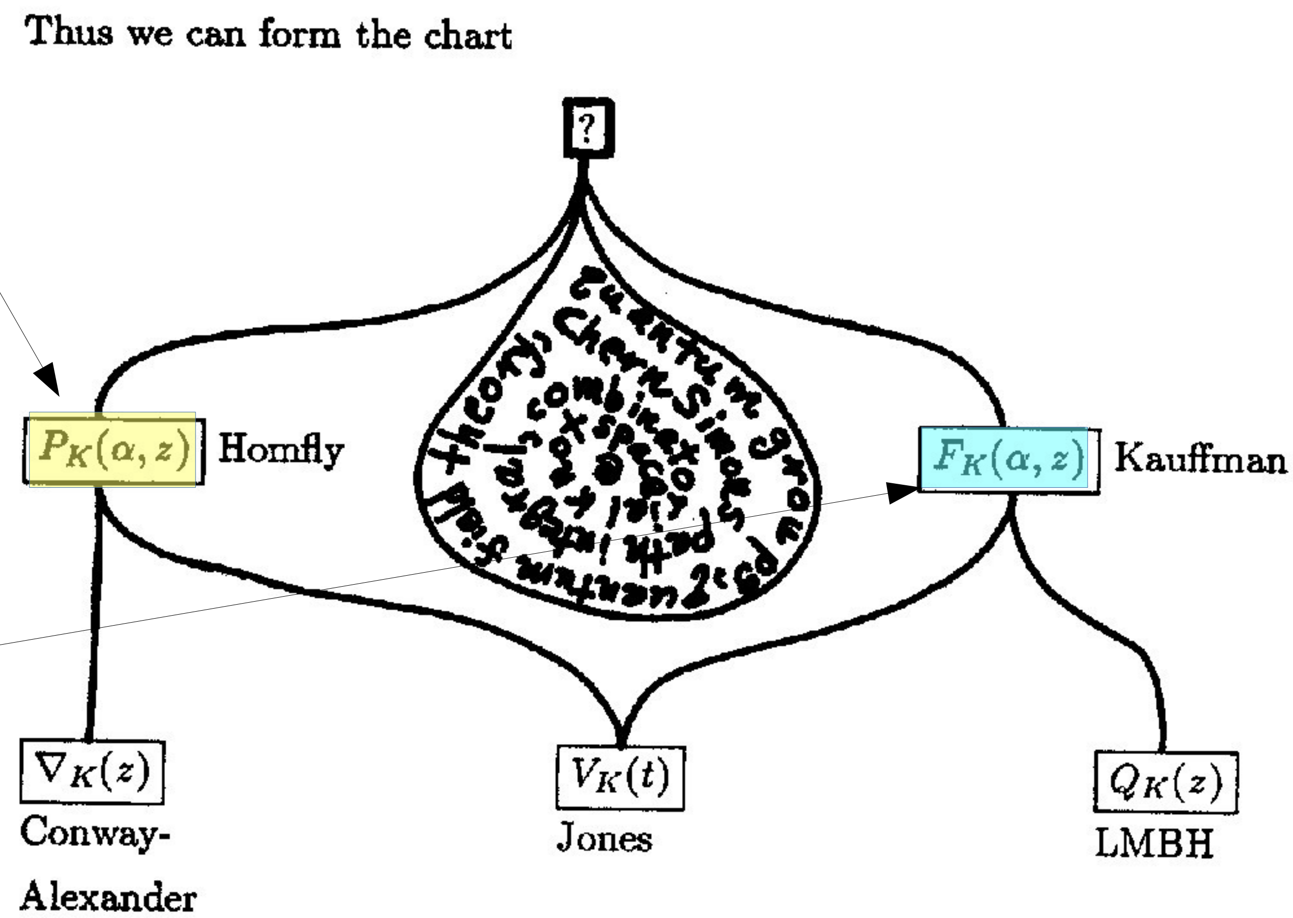
Why are cubic braid groups interesting?

Homfly-PT invariant	Markov trace on Iwahori Hecke algebra	deformation of group algebra of symmetric group
Kauffman invariant	Markov trace on cubic Hecke algebra	deformation of group algebra of cubic braid group

Why are cubic braid groups interesting?

Homfly-PT invariant

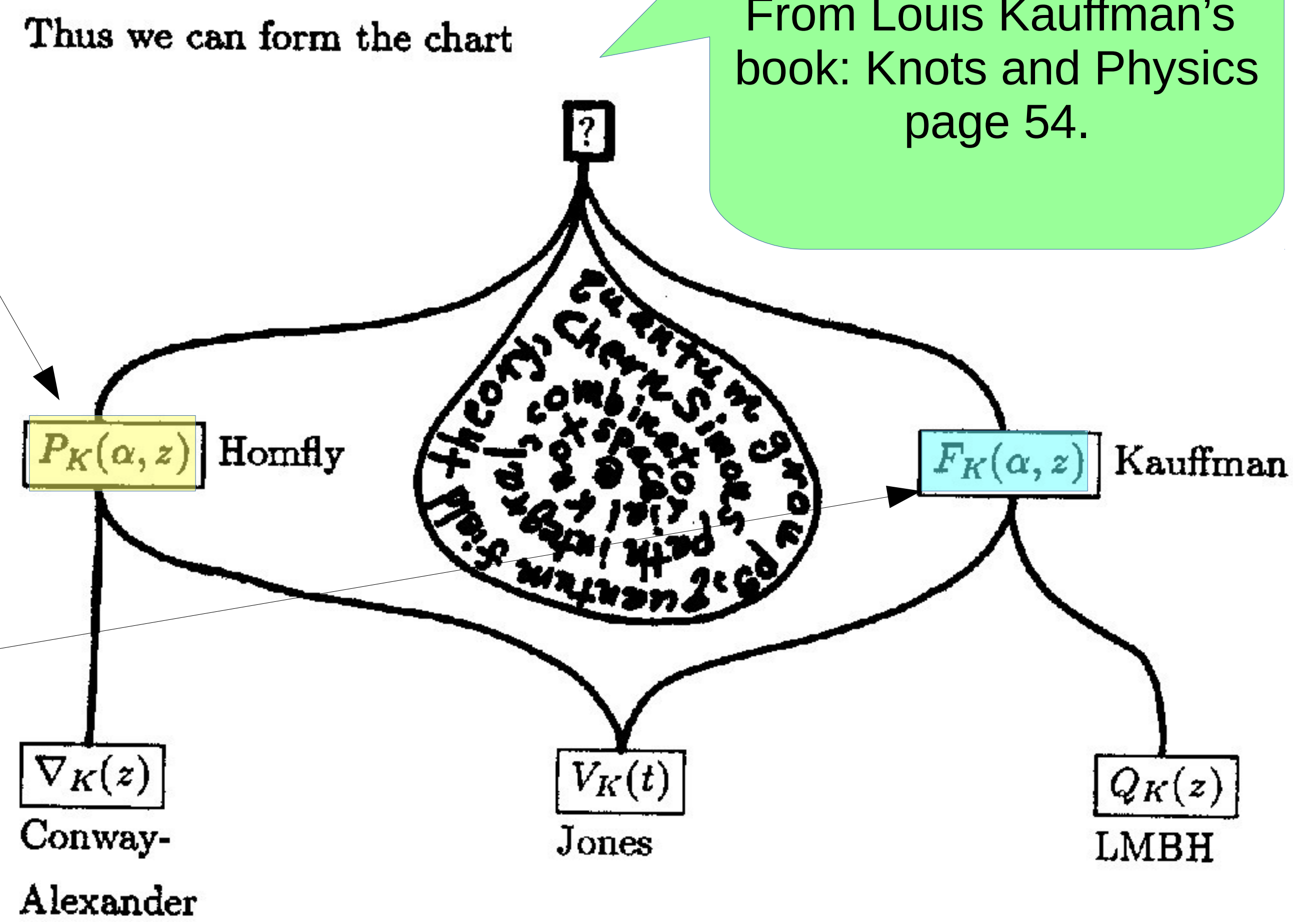
Kauffman invariant



Why are cubic braid groups interesting?

Homfly-PT invariant

Kauffman invariant



From Louis Kauffman's book: Knots and Physics page 54.

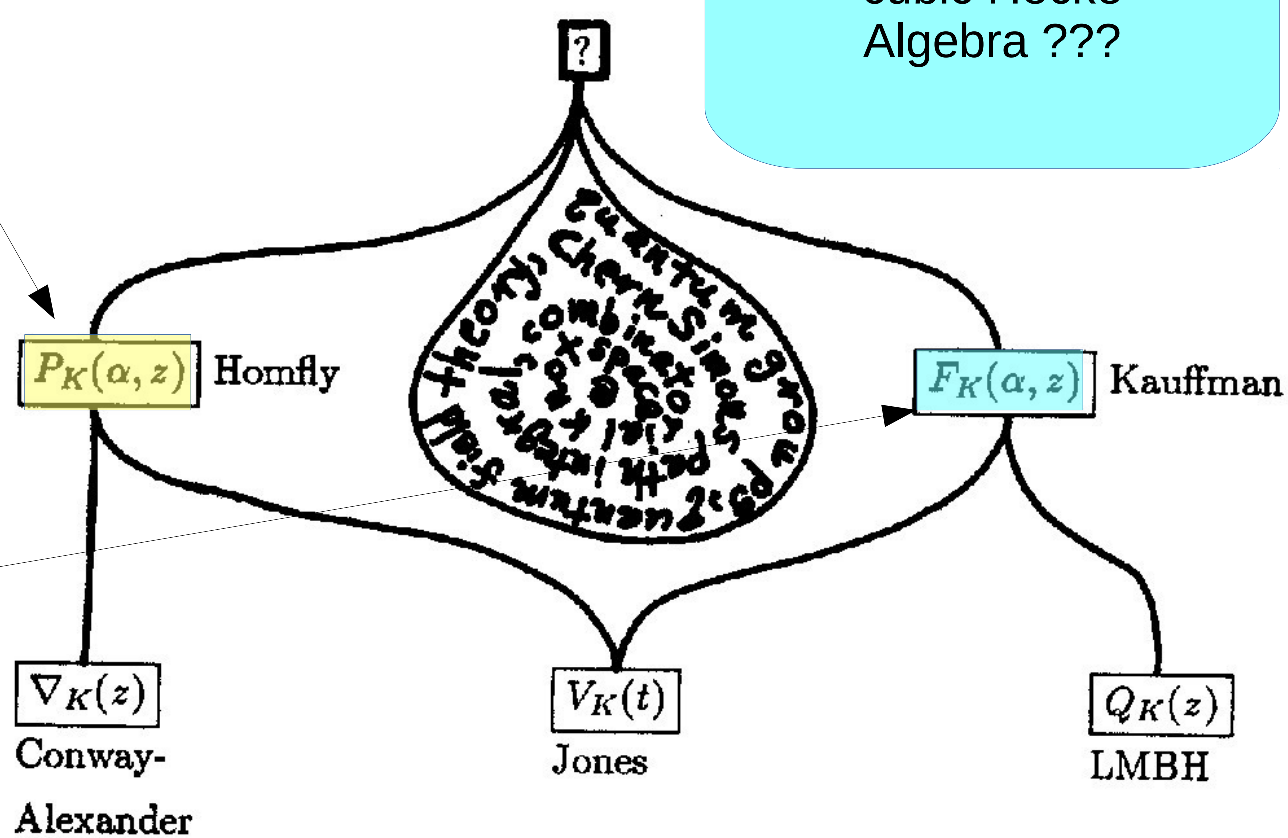
Why are cubic braid groups interesting?

Homfly-PT invariant

Kauffman invariant

??? Markov trace on cubic Hecke Algebra ???

Thus we can form the chart



What is known about them?

What is known about them?

Since the finite ones are irreducible complex reflection groups enumerated in the Shephard-Todd classification with ST-numbers 4, 25 and 32, a lot of results from this theory applies.

What is known about them?

Since the finite ones are irreducible complex reflection groups enumerated in the Shephard-Todd classification with ST-numbers 4, 25 and 32, a lot of results from this theory applies.

But this approach looks at each of the group as something individual.

What is known about them?

Since the finite ones are irreducible complex reflection groups enumerated in the Shephard-Todd classification with ST-numbers 4, 25 and 32, a lot of results from this theory applies.

But this approach looks at each of the group as something individual.

Here the focus will be on the whole tower of groups!

What is known about them?

Since the finite ones are irreducible complex reflection groups enumerated in the Shephard-Todd classification with ST-numbers 4, 25 and 32, a lot of results from this theory applies.

But this approach looks at each of the group as something individual.

Here the focus will be on **the whole tower** of groups!

What is known about them?

In "Einige endliche Faktorgruppen der Zopfgruppen" (Math. Z., 163 (1978), 291-302) J. Assion considered two series $S(m)$ and $U(m)$ of finite dimensional factors of these groups. The additional relations on the braid group generators $\{s_1, \cdots, s_{m-1}\}$ are

$$\begin{array}{rlll} s_3 s_1 t_2 s_1 t_2^{-1} t_3 t_2 s_1 t_2^{-1} t_3^{-1} = 1 & \text{for} & m >= 5 & \text{in case of} \quad S(m) \\ t_1 t_3 = 1 & \text{for} & m >= 5 & \text{in case of} \quad U(m) \end{array}$$

where $t_i = (s_i s_{i+1})^3$. He showed that each series of finite cubic braid group factors must be an epimorhic image of one of his two series, as long as the groups with less than 5 strands are the full cubic braid groups, whereas the group on 5 strands is not. He realized the groups $S(m)$ as symplectic groups over $GF(3)$ (resp. subgroups therein) and $U(m)$ as general unitary groups over $GF(4)$ (resp. subgroups therein).

What is known about them?

In “Einige endliche Faktorgruppen der Zopfgruppen” (Math. Z., 163 (1978), 291-302) J. Assion considered two series $S(m)$ and $U(m)$ of finite dimensional factors of these groups. The additional relations on the braid group generators $\{s_1, \cdots, s_{m-1}\}$ are

$$s_3 s_1 t_2 s_1 t_2^{-1} t_3 t_2 s_1 t_2^{-1} t_3^{-1} = 1 \quad \text{for} \quad m >= 5 \quad \text{in case of} \quad S(m)$$
$$t_1 t_3 = 1 \quad \text{for} \quad m >= 5 \quad \text{in case of} \quad U(m)$$

where $t_i = (s_i s_{i+1})^3$. He showed that each series of finite cubic braid group factors must be an epimorhic image of one of his two series, as long as the groups with less than 5 strands are the full cubic braid groups, whereas the group on 5 strands is not. He realized the groups $S(m)$ as symplectic groups over $GF(3)$ (resp. subgroups therein) and $U(m)$ as general unitary groups over $GF(4)$ (resp. subgroups therein).

What is known about them?

In "Einige endliche Faktorgruppen der Zopfgruppen" (Math. Z., 163 (1978), 291-302) J. Assion considered two series $S(m)$ and $U(m)$ of finite dimensional factors of these groups. The additional relations on the braid group generators $\{s_1, \cdots, s_{m-1}\}$ are

$$\begin{aligned} s_3 s_1 t_2 s_1 t_2^{-1} t_3 t_2 s_1 t_2^{-1} t_3^{-1} &= 1 \quad \text{for} \quad m >= 5 \quad \text{in case of} \quad S(m) \\ t_1 t_3 &= 1 \quad \text{for} \quad m >= 5 \quad \text{in case of} \quad U(m) \end{aligned}$$

where $t_i = (s_i s_{i+1})^3$. He showed that each series of finite cubic braid group factors must be an epimorhic image of one of his two series, as long as the groups with less than 5 strands are the full cubic braid groups, whereas the group on 5 strands is not. He realized the groups $S(m)$ as symplectic groups over $GF(3)$ (resp. subgroups therein) and $U(m)$ as general unitary groups over $GF(4)$ (resp. subgroups therein).

What is known about them?

In "Einige endliche Faktorgruppen der Zopfgruppen" (Math. Z., 163 (1978), 291-302) J. Assion considered two series $S(m)$ and $U(m)$ of finite dimensional factors of these groups. The additional relations on the braid group generators $\{s_1, \cdots, s_{m-1}\}$ are

$$\begin{aligned} s_3 s_1 t_2 s_1 t_2^{-1} t_3 t_2 s_1 t_2^{-1} t_3^{-1} = 1 \quad &\text{for} \quad m >= 5 \quad \text{in case of} \quad S(m) \\ t_1 t_3 = 1 \quad &\text{for} \quad m >= 5 \quad \text{in case of} \quad U(m) \end{aligned}$$

where $t_i = (s_i s_{i+1})^3$. He showed that each series of finite cubic braid group factors must be an epimorhic image of one of his two series, as long as the groups with less than 5 strands are the full cubic braid groups, whereas the group on 5 strands is not. He realized the groups $S(m)$ as symplectic groups over $GF(3)$ (resp. subgroups therein) and $U(m)$ as general unitary groups over $GF(4)$ (resp. subgroups therein).

Functionality of the class

This class implements all the groups conidered by Coxeter and Assion as finitely presented groups (via the gap interface) together with the classical realizations given by the authors. It also contains the coercion maps between the two ways of realization. In addition the user can construct other realizations and maps to matrix groups with help of the burau representation. In case gap3 and CHEVIE are installed under sage version 7.2 (or later) the reflection groups via the gap3 interface are availlable, too. The methods for all this functionality are:

Functionality of the class

This class implements all the groups conidered by Coxeter and Assion as finitely presented groups (via the gap interface) together with the classical realizations given by the authors. It also contains the coercion maps between the two ways of realization. In addition the user can construct other realizations and maps to matrix groups with help of the burau representation. In case gap3 and CHEVIE are installed under sage version 7.2 (or later) the reflection groups via the gap3 interface are availlable, too. The methods for all this functionality are:

Functionality of the class

This class implements all the groups conidered by Coxeter and Assion as finitely presented groups (via the gap interface) together with the classical realizations given by the authors. It also contains the coercion maps between the two ways of realization. In addition the user can construct other realizations and maps to matrix groups with help of the burau representation. In case gap3 and CHEVIE are installed under sage version 7.2 (or later) the reflection groups via the gap3 interface are availlable, too. The methods for all this functionality are:

Functionality of the class

This class implements all the groups conidered by Coxeter and Assion as finitely presented groups (via the gap interface) together with the classical realizations given by the authors. It also contains the coercion maps between the two ways of realization. In addition the user can construct other realizations and maps to matrix groups with help of the burau representation. In case gap3 and CHEVIE are installed under sage version 7.2 (or later) the reflection groups via the gap3 interface are availlable, too. The methods for all this functionality are:

Functionality of the class

This class implements all the groups conidered by Coxeter and Assion as finitely presented groups (via the gap interface) together with the classical realizations given by the authors. It also contains the coercion maps between the two ways of realization. In addition the user can construct other realizations and maps to matrix groups with help of the burau representation. In case gap3 and CHEVIE are installed under sage version 7.2 (or later) the reflection groups via the gap3 interface are availlable, too. The methods for all this functionality are:

Functionality of the class

This class implements all the groups conidered by Coxeter and Assion as finitely presented groups (via the gap interface) together with the classical realizations given by the authors. It also contains the coercion maps between the two ways of realization. In addition the user can construct other realizations and maps to matrix groups with help of the burau representation. In case gap3 and CHEVIE are installed under sage version 7.2 (or later) the reflection groups via the gap3 interface are availlable, too. The methods for all this functionality are:

Functionality of the class

This class implements all the groups conidered by Coxeter and Assion as finitely presented groups (via the gap interface) together with the classical realizations given by the authors. It also contains the coercion maps between the two ways of realization. In addition the user can construct other realizations and maps to matrix groups with help of the burau representation. In case gap3 and CHEVIE are installed under sage version 7.2 (or later) the reflection groups via the gap3 interface are availlable, too. The methods for all this functionality are:

- as_classical_group
- as_matrix_group
- as_reflection_group (needs sage version 7.2 up and gap3 + CHEVIE)
- as_permutation_group

Functionality of the class

This class implements all the groups conidered by Coxeter and Assion as finitely presented groups (via the gap interface) together with the classical realizations given by the authors. It also contains the coercion maps between the two ways of realization. In addition the user can construct other realizations and maps to matrix groups with help of the burau representation. In case gap3 and CHEVIE are installed under sage version 7.2 (or later) the reflection groups via the gap3 interface are availlable, too. The methods for all this functionality are:

- as_classical_group
- as_matrix_group
- as_reflection_group (needs sage version 7.2 up and gap3 + CHEVIE)
- as_permutation_group

Overview

Contents

Examples of general cubic braid groups

```
sage: from cubic_braid import *
sage: C3 = CubicBraidGroup (3) ; C3
Cubic Braid group on 3 strands
sage: C3Cl = C3.as_classical_group(); C3Cl
Subgroup of Unitary Group of degree 2 over Universal
Cyclotomic Field with respect to hermitian form
[-E(12)^7 + E(12)^11                  -1]
[                 -1 -E(12)^7 + E(12)^11] generated by:
([    E(3) E(12)^11]
[       0        1], [       1        0]
[E(12)^11     E(3)])
braid5_2inC3 = C3(braid5_2)
sage: braid5_2inC3Cl = C3Cl(braid5_2inC3); braid5_2inC3Cl
[-E(3)^2 E(12)^7]
[      0      -1]
sage: braid5_2inC3back = C3(braid5_2inC3Cl); braid5_2inC3back
c0*c1*c0^2*c1
sage: braid5_2inC3 == braid5_2inC3back
True
sage: braid5_2inC3.braid() == braid5_2inC3back.braid()
False
```

Examples of general cubic braid groups

```
sage: from cubic_braid import *
sage: C3 = CubicBraidGroup(3); C3
Cubic Braid group on 3 strands
sage: C3Cl = C3.as_classical_group(); C3Cl
Subgroup of Unitary Group of degree 2 over Universal
Cyclotomic Field with respect to hermitian form
[-E(12)^7 + E(12)^11                  -1]
[                 -1 -E(12)^7 + E(12)^11] generated by:
([    E(3) E(12)^11]
[       0         1], [        1        0]
[E(12)^11     E(3)])
braid5_2inC3 = C3(braid5_2)
sage: braid5_2inC3Cl = C3Cl(braid5_2inC3); braid5_2inC3Cl
[-E(3)^2 E(12)^7]
[      0      -1]
sage: braid5_2inC3back = C3(braid5_2inC3Cl); braid5_2inC3back
c0*c1*c0^2*c1
sage: braid5_2inC3 == braid5_2inC3back
True
sage: braid5_2inC3.braid() == braid5_2inC3back.braid()
False
```

Examples of general cubic braid groups

```
sage: from cubic_braid import *
sage: C3 = CubicBraidGroup (3) ; C3
Cubic Braid group on 3 strands
sage: C3Cl = C3.as_classical_group(); C3Cl
Subgroup of Unitary Group of degree 2 over Universal
Cyclotomic Field with respect to hermitian form
[-E(12)^7 + E(12)^11                  -1]
[                 -1 -E(12)^7 + E(12)^11] generated by:
([    E(3) E(12)^11]
[       0        1], [       1        0]
[E(12)^11     E(3)])
braid5_2inC3 = C3(braid5_2)
sage: braid5_2inC3Cl = C3Cl(braid5_2inC3); braid5_2inC3Cl
[-E(3)^2 E(12)^7]
[      0      -1]
sage: braid5_2inC3back = C3(braid5_2inC3Cl); braid5_2inC3back
c0*c1*c0^2*c1
sage: braid5_2inC3 == braid5_2inC3back
True
sage: braid5_2inC3.braid() == braid5_2inC3back.braid()
False
```

Examples of general cubic braid groups

```
sage: from cubic_braid import *
sage: C3 = CubicBraidGroup (3) ; C3
Cubic Braid group on 3 strands
sage: C3Cl = C3.as_classical_group(); C3Cl
Subgroup of Unitary Group of degree 2 over Universal
Cyclotomic Field with respect to hermitian form
[-E(12)^7 + E(12)^11                    -1]
[                  -1 -E(12)^7 + E(12)^11] generated by:
([    E(3) E(12)^11]
[       0         1], [       1         0]
[E(12)^11      E(3)])
braid5_2inC3 = C3(braid5_2)
sage: braid5_2inC3Cl = C3Cl(braid5_2inC3); braid5_2inC3Cl
[-E(3)^2 E(12)^7]
[      0      -1]
sage: braid5_2inC3back = C3(braid5_2inC3Cl); braid5_2inC3back
c0*c1*c0^2*c1
sage: braid5_2inC3 == braid5_2inC3back
True
sage: braid5_2inC3.braid() == braid5_2inC3back.braid()
False
```

Examples of general cubic braid groups

```
sage: from cubic_braid import *
sage: C3 = CubicBraidGroup (3) ; C3
Cubic Braid group on 3 strands
sage: C3Cl = C3.as_classical_group(); C3Cl
Subgroup of Unitary Group of degree 2 over Universal
Cyclotomic Field with respect to hermitian form
[-E(12)^7 + E(12)^11                   -1]
[                  -1 -E(12)^7 + E(12)^11] generated by:
([    E(3) E(12)^11]
[       0         1], [        1         0]
[E(12)^11      E(3)])
braid5_2inC3 = C3(braid5_2)
sage: braid5_2inC3Cl = C3Cl(braid5_2inC3); braid5_2inC3Cl
[-E(3)^2 E(12)^7]
[      0      -1]
sage: braid5_2inC3back = C3(braid5_2inC3Cl); braid5_2inC3back
c0*c1*c0^2*c1
sage: braid5_2inC3 == braid5_2inC3back
True
sage: braid5_2inC3.braid() == braid5_2inC3back.braid()
False
```

Examples of general cubic braid groups

```
sage: from cubic_braid import *
sage: C3 = CubicBraidGroup (3) ; C3
Cubic Braid group on 3 strands
sage: C3Cl = C3.as_classical_group(); C3Cl
Subgroup of Unitary Group of degree 2 over Universal
Cyclotomic Field with respect to hermitian form
[-E(12)^7 + E(12)^11                  -1]
[                 -1 -E(12)^7 + E(12)^11] generated by:
([    E(3) E(12)^11]
[       0         1], [        1         0]
[E(12)^11      E(3)])
braid5_2inC3 = C3(braid5_2)
sage: braid5_2inC3Cl = C3Cl(braid5_2inC3); braid5_2inC3Cl
[-E(3)^2 E(12)^7]
[      0      -1]
sage: braid5_2inC3back = C3(braid5_2inC3Cl); braid5_2inC3back
c0*c1*c0^2*c1
sage: braid5_2inC3 == braid5_2inC3back
True
sage: braid5_2inC3.braid() == braid5_2inC3back.braid()
False
```

Examples of general cubic braid groups

```
sage: from cubic_braid import *
sage: C3 = CubicBraidGroup (3) ; C3
Cubic Braid group on 3 strands
sage: C3Cl = C3.as_classical_group(); C3Cl
Subgroup of Unitary Group of degree 2 over Universal
Cyclotomic Field with respect to hermitian form
[-E(12)^7 + E(12)^11                    -1]
[                  -1 -E(12)^7 + E(12)^11]  generated by:
([    E(3) E(12)^11]
[       0         1], [        1         0]
[E(12)^11      E(3)])
braid5_2inC3 = C3(braid5_2)
sage: braid5_2inC3Cl = C3Cl(braid5_2inC3); braid5_2inC3Cl
[-E(3)^2 E(12)^7]
[      0      -1]
sage: braid5_2inC3back = C3(braid5_2inC3Cl); braid5_2inC3back
c0*c1*c0^2*c1
sage: braid5_2inC3 == braid5_2inC3back
True
sage: braid5_2inC3.braid() == braid5_2inC3back.braid()
False
```

converting braid5_2 to the classical realization

Examples of general cubic braid groups

```
sage: from cubic_braid import *
sage: C3 = CubicBraidGroup (3) ; C3
Cubic Braid group on 3 strands
sage: C3Cl = C3.as_classical_group(); C3Cl
Subgroup of Unitary Group of degree 2 over Universal
Cyclotomic Field with respect to hermitian form
[-E(12)^7 + E(12)^11                  -1]
[                 -1 -E(12)^7 + E(12)^11] generated by:
([     E(3) E(12)^11]
[        0         1], [        1         0]
[E(12)^11      E(3)])
braid5_2inC3 = C3(braid5_2)
sage: braid5_2inC3Cl = C3Cl(braid5_2inC3); braid5_2inC3Cl
[-E(3)^2 E(12)^7]
[      0      -1]
sage: braid5_2inC3back = C3(braid5_2inC3Cl); braid5_2inC3back
c0*c1*c0^2*c1
sage: braid5_2inC3 == braid5_2inC3back
True
sage: braid5_2inC3.braid() == braid5_2inC3back.braid()
False
```

Converting back

Examples of general cubic braid groups

```
braid5_2inC3 = C3(braid5_2)
sage: braid5_2inC3Cl = C3Cl(braid5_2inC3); braid5_2inC3Cl
[-E(3)^2 E(12)^7]
[      0      -1]
sage: braid5_2inC3back = C3(braid5_2inC3Cl); braid5_2inC3back
c0*c1*c0^2*c1
sage: braid5_2inC3 == braid5_2inC3back
True
sage: braid5_2inC3.braid() == braid5_2inC3back.braid()
False
sage: braid5_2inC3back.braid().plot()
Launched png viewer for Graphics object consisting of 20
graphics primitives
sage: Link(braid5_2inC3back.braid()).plot()
Launched png viewer for Graphics object consisting of 24
graphics primitives
```

Converting back

Examples of general cubic braid groups

```
braid5_2inC3 = C3(braid5_2)
sage: braid5_2inC3Cl = C3Cl(braid5_2inC3); braid5_2inC3Cl
[-E(3)^2 E(12)^7]
[      0       -1]
sage: braid5_2inC3back = C3(braid5_2inC3Cl); braid5_2inC3back
c0*c1*c0^2*c1
sage: braid5_2inC3 == braid5_2inC3back
True
sage: braid5_2inC3.braid() == braid5_2inC3back.braid()
False
sage: braid5_2inC3back.braid().plot()
Launched png viewer for Graphics object consisting of 20
graphics primitives
sage: Link(braid5_2inC3back.braid()).plot()
Launched png viewer for Graphics object consisting of 24
graphics primitives
```

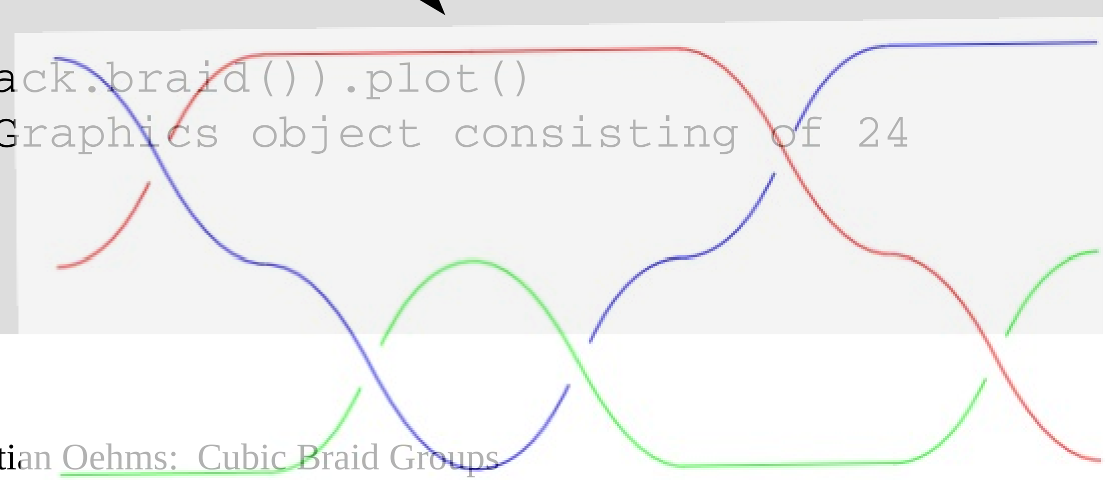
Examples of general cubic braid groups

```
braid5_2inC3 = C3(braid5_2)
sage: braid5_2inC3Cl = C3Cl(braid5_2inC3); braid5_2inC3Cl
[-E(3)^2 E(12)^7]
[      0      -1]
sage: braid5_2inC3back = C3(braid5_2inC3Cl); braid5_2inC3back
c0*c1*c0^2*c1
sage: braid5_2inC3 == braid5_2inC3back
True
sage: braid5_2inC3.braid() == braid5_2inC3back.braid()
False
sage: braid5_2inC3back.braid().plot()
Launched png viewer for Graphics object consisting of 20
graphics primitives
sage: Link(braid5_2inC3back.braid()).plot()
Launched png viewer for Graphics object consisting of 24
graphics primitives
```

pre-image in the braid group

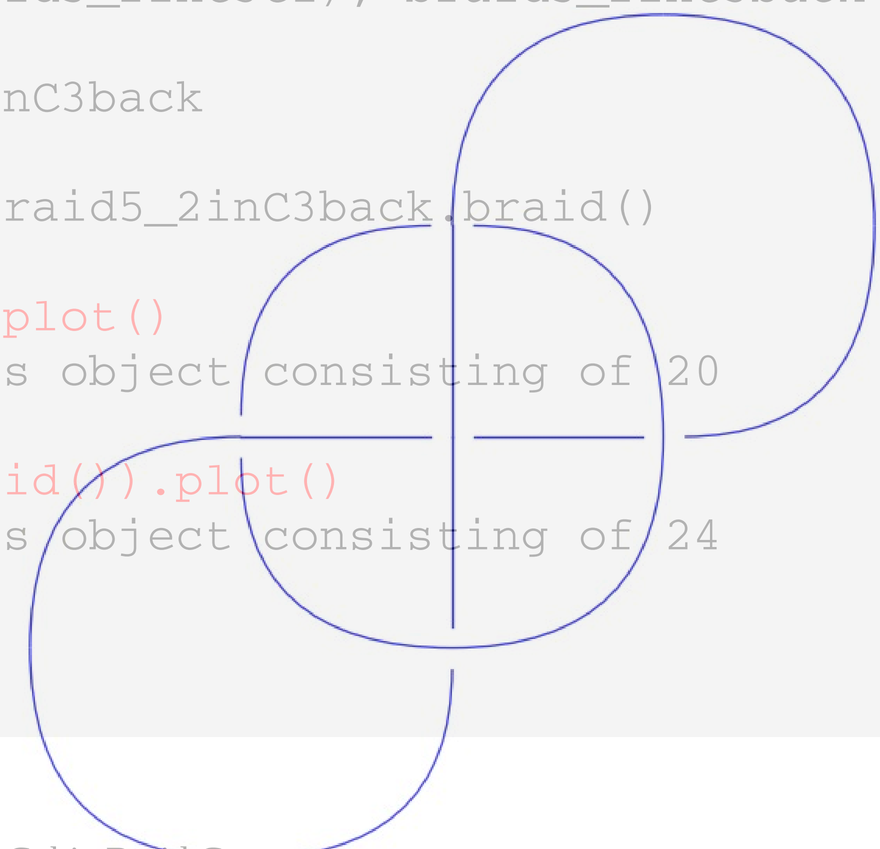
Examples of general cubic braid groups

```
braid5_2inC3 = C3(braid5_2)
sage: braid5_2inC3Cl = C3Cl(braid5_2inC3); braid5_2inC3Cl
[-E(3)^2 E(12)^7]
[      0       -1]
sage: braid5_2inC3back = C3(braid5_2inC3Cl); braid5_2inC3back
c0*c1*c0^2*c1
sage: braid5_2inC3 == braid5_2inC3back
True
sage: braid5_2inC3.braid() == braid5_2inC3back.braid()
False
sage: braid5_2inC3back.braid().plot()
Launched png viewer for Graphics object consisting of 20
graphics primitives
sage: Link(braid5_2inC3back.braid()).plot()
Launched png viewer for Graphics object consisting of 24
graphics primitives
```



Examples of general cubic braid groups

```
braid5_2inC3 = C3(braid5_2)
sage: braid5_2inC3Cl = C3Cl(braid5_2inC3); braid5_2inC3Cl
[-E(3)^2 E(12)^7]
[      0       -1]
sage: braid5_2inC3back = C3(braid5_2inC3Cl); braid5_2inC3back
c0*c1*c0^2*c1
sage: braid5_2inC3 == braid5_2inC3back
True
sage: braid5_2inC3.braid() == braid5_2inC3back.braid()
False
sage: braid5_2inC3back.braid().plot()
Launched png viewer for Graphics object consisting of 20
graphics primitives
sage: Link(braid5_2inC3back.braid()).plot()
Launched png viewer for Graphics object consisting of 24
graphics primitives
```



Examples of Assion Groups

```
sage: S3 = AssionGroupS(3); S3
Assion group on 3 strands of type S
sage: U3 = AssionGroupU(3); U3
Assion group on 3 strands of type U
sage: C3.is_isomorphic(S3)
True
sage: C3.is_isomorphic(U3)
True
sage: C3 == S3
False
sage: C3 == U3
False
sage: S3 == U3
False
```

Examples of Assion Groups

```
sage: S3 = AssionGroupS(3); S3
Assion group on 3 strands of type S
sage: U3 = AssionGroupU(3); U3
Assion group on 3 strands of type U
sage: C3.is_isomorphic(S3)
True
sage: C3.is_isomorphic(U3)
True
sage: C3 == S3
False
sage: C3 == U3
False
sage: S3 == U3
False
```

Examples of Assion Groups

```
sage: S3 = AssionGroupS(3); S3
Assion group on 3 strands of type S
sage: U3 = AssionGroupU(3); U3
Assion group on 3 strands of type U
sage: C3.is_isomorphic(S3)
True
sage: C3.is_isomorphic(U3)
True
sage: C3 == S3
False
sage: C3 == U3
False
sage: S3 == U3
False
```

Assion Groups of higher order

```
sage: S5.order()
51840
sage: U5.order()
77760
sage: C5.order()
155520
sage: S5Cl = S5.as_classical_group(); S5Cl
Symplectic Group of degree 4 over Finite Field
of size 3
sage:
sage: U5Cl = U5.as_classical_group(); U5Cl
General Unitary Group of degree 4 over Finite
Field in a of size 2^2
```

Assion Groups of higher order

```
sage: S5.order()
51840
sage: U5.order()
77760
sage: C5.order()
155520
sage: S5Cl = S5.as_classical_group(); S5Cl
Symplectic Group of degree 4 over Finite Field
of size 3
sage:
sage: U5Cl = U5.as_classical_group(); U5Cl
General Unitary Group of degree 4 over Finite
Field in a of size 2^2
```

Assion Groups of higher order

```
sage: S5.order()
51840
sage: U5.order()
77760
sage: C5.order()
155520
sage: S5Cl = S5.as_classical_group(); S5Cl
Symplectic Group of degree 4 over Finite Field
of size 3
sage:
sage: U5Cl = U5.as_classical_group(); U5Cl
General Unitary Group of degree 4 over Finite
Field in a of size 2^2
```

Assion Groups of higher order

```
sage: S5.order()
51840
sage: U5.order()
77760
sage: C5.order()
155520
sage: S5Cl = S5.as_classical_group(); S5Cl
Symplectic Group of degree 4 over Finite Field
of size 3
sage:
sage: U5Cl = U5.as_classical_group(); U5Cl
General Unitary Group of degree 4 over Finite
Field in a of size 2^2
```

Overview

Contents

Exceptions in Assion Group Series

```
sage: U3Cl = U3.as_classical_group(); U3Cl
Subgroup of (The projective general unitary
group of degree 3 over Finite Field of size 2)
generated by [(1,7,6)(3,19,14)(4,15,10)(5,11,18)
(12,16,20), (1,12,13)(2,15,19)(4,9,14)(5,18,8)
(6,21,16)]
sage: U3Clemb =
U3.as_classical_group(embedded=True); U3Clemb
Matrix group over Finite Field in a of size 2^2
with 2 generators (
[0 0 a]  [a + 1     a     a]
[0 1 0]  [    a a + 1     a]
[a 0 a], [    a     a a + 1]
)
```

Exceptions in Assion Group Series

```
sage: U3Cl = U3.as_classical_group(); U3Cl
Subgroup of (The projective general unitary
group of degree 3 over Finite Field of size 2)
generated by [(1,7,6)(3,19,14)(4,15,10)(5,11,18)
(12,16,20), (1,12,13)(2,15,19)(4,9,14)(5,18,8)
(6,21,16)]
sage: U3Clemb =
U3.as_classical_group(embedded=True); U3Clemb
Matrix group over Finite Field in a of size 2^2
with 2 generators (
[0 0 a]  [a + 1     a     a]
[0 1 0]  [    a a + 1     a]
[a 0 a], [    a     a a + 1]
)
```

Exceptions in Assion Group Series

```
sage: U3Cl = U3.as_classical_group(); U3Cl
Subgroup of (The projective general unitary
group of degree 3 over Finite Field of size 2)
generated by [(1,7,6)(3,19,14)(4,15,10)(5,11,18)
(12,16,20), (1,12,13)(2,15,19)(4,9,14)(5,18,8)
(6,21,16)]
sage: U3Clemb =
U3.as_classical_group(embedded=True); U3Clemb
Matrix group over Finite Field in a of size 2^2
with 2 generators (
[0 0 a]  [a + 1     a     a]
[0 1 0]  [    a a + 1     a]
[a 0 a], [    a     a a + 1]
)
```

Exceptions in Assion Group Series

```
sage: U3Cl = U3.as_classical_group(); U3Cl
Subgroup of (The projective general unitary
group of degree 3 over Finite Field of size 2)
generated by [(1,7,6)(3,19,14)(4,15,10)(5,11,18)
(12,16,20), (1,12,13)(2,15,19)(4,9,14)(5,18,8)
(6,21,16)]
sage: U3Clemb =
U3.as_classical_group(embedded=True); U3Clemb
Matrix group over Finite Field in a of size 2^2
with 2 generators (
[0 0 a]   [a + 1      a      a]
[0 1 0]   [    a  a + 1      a]
[a 0 a],  [    a      a  a + 1]
)
```

Overview

Contents

Other Realizations

```
sage: C3RG = C3.as_reflection_group(); C3RG
Irreducible complex reflection group of rank 2 and
type ST4
sage: coxelem = C3RG.coxeter_element(); coxelem
(1,7,6,12,23,20)(2,8,17,24,9,5)(3,16,10,19,15,21)
(4,14,11,22,18,13)
sage: C3(coxelem)
c0*c1
sage: C3M5 = sage: C3M5 =
C3.as_matrix_group(characteristic=5); C3M5
Matrix group over Finite Field in rI of size 5^2
with 2 generators (
[2*rI + 2 3*rI + 4        0]  [       1        0        0]
[       1        0        0]  [       0 2*rI + 2 3*rI + 4]
[       0        0        1], [       0        1        0]
)
```

Other Realizations

```
sage: C3RG = C3.as_reflection_group(); C3RG
Irreducible complex reflection group of rank 2 and
type ST4
sage: coxelem = C3RG.coxeter_element(); coxelem
(1,7,6,12,23,20)(2,8,17,24,9,5)(3,16,10,19,15,21)
(4,14,11,22,18,13)
sage: C3(coxelem)
c0*c1
sage: C3M5 = sage: C3M5 =
C3.as_matrix_group(characteristic=5); C3M5
Matrix group over Finite Field in rI of size 5^2
with 2 generators (
[2*rI + 2 3*rI + 4        0]  [       1        0        0]
[       1        0        0]  [       0 2*rI + 2 3*rI + 4]
[       0        0        1], [       0        1        0]
)
```

Other Realizations

```
sage: C3RG = C3.as_reflection_group(); C3RG
Irreducible complex reflection group of rank 2 and
type ST4
sage: coxelem = C3RG.coxeter_element(); coxelem
(1,7,6,12,23,20)(2,8,17,24,9,5)(3,16,10,19,15,21)
(4,14,11,22,18,13)
sage: C3(coxelem)
c0*c1
sage: C3M5 = sage: C3M5 =
C3.as_matrix_group(characteristic=5); C3M5
Matrix group over Finite Field in rI of size 5^2
with 2 generators (
[2*rI + 2 3*rI + 4        0]   [      1          0          0]
[       1        0        0]   [      0 2*rI + 2 3*rI + 4]
[       0        0        1],  [      0          1          0]
)
```

Other Realizations

```
sage: C3RG = C3.as_reflection_group(); C3RG
Irreducible complex reflection group of rank 2 and
type ST4
sage: coxelem = C3RG.coxeter_element(); coxelem
(1,7,6,12,23,20)(2,8,17,24,9,5)(3,16,10,19,15,21)
(4,14,11,22,18,13)
sage: C3(coxelem)
c0*c1
sage: C3M5 = sage: C3M5 =
C3.as_matrix_group(characteristic=5); C3M5
Matrix group over Finite Field in rI of size 5^2
with 2 generators (
[2*rI + 2 3*rI + 4        0]   [      1        0        0]
[       1        0        0]   [      0 2*rI + 2 3*rI + 4]
[       0        0        1],  [      0        1        0]
)
```

Implementation

```
sage/CubicBraidGroup $ ls  -ltr lib/*.py
      0 Dez 30  2016 lib/__init__.py
  27620 Apr  3  2017 lib/utils_sys.py
  41473 Apr  3  2017 lib/utils_gap_interface.py
  17263 Apr  3  2017 lib/local_braid.py
  21185 Apr  3  2017 lib/local_matrix_group.py
  23680 Apr  6  2017 lib/local_permgroup.py
 133736 Jun 10 22:17 lib/cubic_braid.py
sage/CubicBraidGroup $
```

Implementation

```
sage/CubicBraidGroup $ ls  -ltr lib/*.py
      0 Dez 30  2016 lib/__init__.py
  27620 Apr  3  2017 lib/utils_sys.py
  41473 Apr  3  2017 lib/utils_gap_interface.py
  17263 Apr  3  2017 lib/local_braid.py
  21185 Apr  3  2017 lib/local_matrix_group.py
  23680 Apr  6  2017 lib/local_permgroup.py
 133736 Jun 10 22:17 lib/cubic_braid.py
sage/CubicBraidGroup $
```

Implementation

```
sage/CubicBraidGroup $ ls  -ltr lib/*.py
      0 Dez 30  2016 lib/__init__.py
  27620 Apr  3  2017 lib/utils_sys.py
  41473 Apr  3  2017 lib/utils_gap_interface.py
  17263 Apr  3  2017 lib/local_braid.py
  21185 Apr  3  2017 lib/local_matrix_group.py
  23680 Apr  6  2017 lib/local_permgroup.py
 133736 Jun 10 22:17 lib/cubic_braid.py
sage/CubicBraidGroup $
```

Implementation

```
sage/CubicBraidGroup $ ls  -ltr lib/*.py
      0 Dez 30  2016 lib/__init__.py
  27620 Apr  3  2017 lib/utils_sys.py
  41473 Apr  3  2017 lib/utils_gap_interface.py
  17263 Apr  3  2017 lib/local_braid.py
  21185 Apr  3  2017 lib/local_matrix_group.py
  23680 Apr  6  2017 lib/local_permgroup.py
 133736 Jun 10 22:17 lib/cubic_braid.py
sage/CubicBraidGroup $
```

These modules contain suggestions for improvements of sage

Implementation

```
sage/CubicBraidGroup $ ls  -ltr lib/*.py
      0 Dez 30  2016 lib/__init__.py
  27620 Apr  3  2017 lib/utils_sys.py
  41473 Apr  3  2017 lib/utils_gap_interface.py
  17263 Apr  3  2017 lib/local_braid.py
  21185 Apr  3  2017 lib/local_matrix_group.py
  23680 Apr  6  2017 lib/local_permgroup.py
 133736 Jun 10 22:17 lib/cubic_braid.py
sage/CubicBraidGroup $
```

For example Tickets #25686, #25706

Sage-problems I had to solve

the following I noticed when I tried to obtain the natural projection from finite unitary and symplectic groups to the corresponding projective groups

Sage-problems I had to solve

```
sage: G = GU(3,2); G
General Unitary Group of degree 3 over Finite
Field in a of size 2^2
sage: MG = G.as_matrix_group(); MG
Matrix group over Finite Field in a of size 2^2
with 2 generators (
[a 0 0]   [a 1 1]
[0 1 0]   [1 1 0]
[0 0 a],  [1 0 0]
)
sage: mg = MG.an_element(); mg
[a + 1     a     a]
[    1     1     0]
[    a     0     0]
```

Sage-problems I had to solve

```
sage: G = GU(3,2); G
General Unitary Group of degree 3 over Finite
Field in a of size 2^2
sage: MG = G.as_matrix_group(); MG
Matrix group over Finite Field in a of size 2^2
with 2 generators (
[a 0 0]   [a 1 1]
[0 1 0]   [1 1 0]
[0 0 a],  [1 0 0]
)
sage: mg = MG.an_element(); mg
[a + 1     a     a]
[    1     1     0]
[    a     0     0]
```

Sage-problems I had to solve

```
sage: G = GU(3,2); G
General Unitary Group of degree 3 over Finite
Field in a of size 2^2
sage: MG = G.as_matrix_group(); MG
Matrix group over Finite Field in a of size 2^2
with 2 generators (
[a 0 0]   [a 1 1]
[0 1 0]   [1 1 0]
[0 0 a], [1 0 0]
)
sage: mg = MG.an_element(); mg
[a + 1     a     a]
[    1     1     0]
[    a     0     0]
```

Sage-problems I had to solve

```
sage: G = GU(3,2); G
General Unitary Group of degree 3 over Finite
Field in a of size 2^2
sage: MG = G.as_matrix_group(); MG
Matrix group over Finite Field in a of size 2^2
with 2 generators (
[a 0 0]   [a 1 1]
[0 1 0]   [1 1 0]
[0 0 a], [1 0 0]
)
sage: mg = MG.an_element(); mg
[a + 1     a     a]
[    1     1     0]
[    a     0     0]
```

Sage-problems I had to solve

```
sage: mg = MG.an_element(); mg
[a + 1     a     a]
[    1     1     0]
[    a     0     0]
PG = MG.as_permutation_group()
sage: PG(mg)
```

Sage-problems I had to solve

```
sage: mg = MG.an_element(); mg
[a + 1       a       a]
[    1       1       0]
[    a       0       0]
PG = MG.as_permutation_group()
sage: PG(mg)
```

Now, try to obtain
the image of mg
in the permutation group

Sage-problems I had to solve

```
sage: mg = MG.an_element(); mg
[a + 1     a     a]
[    1     1     0]
[    a     0     0]
PG = MG.as_permutation_group()
sage: PG(mg)
----------------------------------------------
TypeError:
..................
'sage.groups.matrix_gps.group_element.MatrixGrou
pElement_gap' object is not iterable
```

Sage-problems I had to solve

```
sage: conv_ori = PG.convert_map_from(MG);
conv_ori
Call morphism:
  From: Matrix group over Finite Field in a of
size 2^2 with 2 generators (….)
  To:   Permutation Group with generators
[(2,3,5)(4,7,12)…. ]
sage: img_mg = conv_ori(mg)
----------------------------------------------
TypeError:
……………….. . .
'sage.groups.matrix_gps.group_element.MatrixGrou
pElement_gap' object is not iterable
```

Sage-problems I had to solve

```
sage: conv_ori = PG.convert_map_from(MG); 
conv_ori
Call morphism:
  From: Matrix group over Finite Field in a of 
size 2^2 with 2 generators (….)
  To:   Permutation Group with generators 
[(2,3,5)(4,7,12)…. ]
sage: img_mg = conv_ori(mg)
----------------------------------------------
TypeError:
…………………...
'sage.groups.matrix_gps.group_element.MatrixGrou
pElement_gap' object is not iterable
```

Second try

Sage-problems I had to solve

```
sage: conv_ori = PG.convert_map_from(MG); conv_ori
Call morphism:
  From: Matrix group over Finite Field in a of size 2^2 with 2 generators (….)
  To:   Permutation Group with generators [(2,3,5)(4,7,12)…. ]
sage: img_mg = conv_ori(mg)
---------------------------------------------
TypeError:
……………….. . .
'sage.groups.matrix_gps.group_element.MatrixGroupElement_gap' object is not iterable
```

Sage-problems I had to solve

```
sage: conv_ori = PG.convert_map_from(MG); conv_ori
Call morphism:
  From: Matrix group over Finite Field in a of size 2^2 with 2 generators (….)
  To:   Permutation Group with generators [(2,3,5)(4,7,12)…. ]
sage: img_mg = conv_ori(mg)
---------------------------------------------
TypeError:
……………….. . .
'sage.groups.matrix_gps.group_element.MatrixGroupElement_gap' object is not iterable
```

Sage-problems I had to solve

The problem consists of two issues:

1) Repair the registered conversion map
2) Make the conversion work with __call__

Sage-problems I had to solve

The problem consists of two issues:

1) Repair the registered conversion map
2) Make the conversion work with __call__

Sage-problems I had to solve

```
sage: gap_hom = MG.gap().GroupHomomorphismByImages
sage: MGgens = MG.gap().GeneratorsOfGroup()
sage: PGgens = gap(PG).GeneratorsOfGroup()
sage: conv_map_gap = gap_hom(MGgens, PGgens)
sage: def conv_map_func(elem):
....:     res = conv_map_gap.ImageElm(elem.gap())
....:     return res.sage()
sage: conv_map = Hom(MG, PG)(conv_map_func)
```

Sage-problems I had to solve

```
sage: gap_hom = MG.gap().GroupHomomorphismByImages
sage: MGgens = MG.gap().GeneratorsOfGroup()
sage: PGgens = gap(PG).GeneratorsOfGroup()
sage: conv_map_gap = gap_hom(MGgens, PGgens)
sage: def conv_map_func(elem):
....:     res = conv_map_gap.ImageElm(elem.gap())
....:     return res.sage()
sage: conv_map = Hom(MG, PG)(conv_map_func)
```

Try to construct the homomorphism via GAP

Sage-problems I had to solve

```
sage: gap_hom = MG.gap().GroupHomomorphismByImages
sage: MGgens = MG.gap().GeneratorsOfGroup()
sage: PGgens = gap(PG).GeneratorsOfGroup()
sage: conv_map_gap = gap_hom(MGgens, PGgens)
sage: def conv_map_func(elem):
....:     res = conv_map_gap.ImageElm(elem.gap())
....:     return res.sage()
sage: conv_map = Hom(MG, PG)(conv_map_func)
```

Sage-problems I had to solve

```
sage: gap_hom = MG.gap().GroupHomomorphismByImages
sage: MGgens = MG.gap().GeneratorsOfGroup()
sage: PGgens = gap(PG).GeneratorsOfGroup()
sage: conv_map_gap = gap_hom(MGgens, PGgens)
sage: def conv_map_func(elem):
....:     res = conv_map_gap.ImageElm(elem.gap())
....:     return res.sage()
sage: conv_map = Hom(MG, PG)(conv_map_func)
```

Sage-problems I had to solve

```
sage: gap_hom = MG.gap().GroupHomomorphismByImages
sage: MGgens = MG.gap().GeneratorsOfGroup()
sage: PGgens = gap(PG).GeneratorsOfGroup()
sage: conv_map_gap = gap_hom(MGgens, PGgens)
sage: def conv_map_func(elem):
....:     res = conv_map_gap.ImageElm(elem.gap())
....:     return res.sage()
sage: conv_map = Hom(MG, PG)(conv_map_func)
```

Sage-problems I had to solve

```
sage: gap_hom = MG.gap().GroupHomomorphismByImages
sage: MGgens = MG.gap().GeneratorsOfGroup()
sage: PGgens = gap(PG).GeneratorsOfGroup()
sage: conv_map_gap = gap_hom(MGgens, PGgens)
sage: def conv_map_func(elem):
....:     res = conv_map_gap.ImageElm(elem.gap())
....:     return res.sage()
sage: conv_map = Hom(MG, PG)(conv_map_func)
```

Sage-problems I had to solve

```
sage: gap_hom = MG.gap().GroupHomomorphismByImages
sage: MGgens = MG.gap().GeneratorsOfGroup()
sage: PGgens = gap(PG).GeneratorsOfGroup()
sage: conv_map_gap = gap_hom(MGgens, PGgens)
sage: def conv_map_func(elem):
....:     res = conv_map_gap.ImageElm(elem.gap())
....:     return res.sage()
sage: conv_map = Hom(MG, PG)(conv_map_func)
sage: img_mg = conv_map(mg); img_mg
(1,2,6,19,35,33)(3,9,26,14,31,23)(4,13,5)(7,22,17)
(8,24,12)(10,16,32,27,20,28)(11,30,18)
(15,25,36,34,29,21)
```

Sage-problems I had to solve

```
sage: gap_hom = MG.gap().GroupHomomorphismByImages
sage: MGgens = MG.gap().GeneratorsOfGroup()
sage: PGgens = gap(PG).GeneratorsOfGroup()
sage: conv_map_gap = gap_hom(MGgens, PGgens)
sage: def conv_map_func(elem):
....:       res = conv_map_gap.ImageElm(elem.gap())
....:       return res.sage()
sage: conv_map = Hom(MG, PG)(conv_map_func)
sage: img_mg = conv_map(mg); img_mg
(1,2,6,19,35,33)(3,9,26,14,31,23)(4,13,5)(7,22,17)
(8,24,12)(10,16,32,27,20,28)(11,30,18)
(15,25,36,34,29,21)
```

Idea: Integrate this in the as_permutation_group method of FinitelyGeneratedMatrixGroup_gap

Sage-problems I had to solve

The problem consists of two issues:

1) Repair the registered conversion map
2) Make the conversion work with __call__

Sage-problems I had to solve

1) Repair the registered conversion map
2) Make the conversion work with __call__

I solved this overloading the __call__ method of PermutationGroup_generic locally

Sage-problems I had to solve

1) Repair the registered conversion map
2) Make the conversion work with __call__

I solved this overloading the __call__ method of PermutationGroup_generic locally

What would be the right way to solve this problem in Sage?