The ThreeKPlusOne Package

Version 1

Pascal Kattler

Pascal Kattler Email: kattler@math.uni-sb.de

Contents

Chapter 1

The $3k+1$ Problem

1.1 Introduction

This package provides calculations with Origamis. An Origami can be obtained in the following way from two permutations $\sigma_a, \sigma_b \in S_d$. We take d Squares $Q_1, \ldots, Q_d$ and clue the lower side of Q_i to the upper side of $Q_{\sigma_y(i)}$ and the right side of Q_i to the left side of $Q_{\sigma_x(i)}$. So in this Package we identify an Origami with a pair of permutations, witch acts transitive on $\{1 \ldots d\}$ up to simultan conjugation. We store an Origami as Origami object. We are mainly interested in the Veechgroup of an Origami. It can be shown that the Veechgroup of an Origami is a subgroup of $Sl_2(\mathbb{Z})$ of finite index. So we fix two generators

$$S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

and

$$T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

1.2 The Free Group

In this package we fix the Free Group F generated by S and T.

1.3 The Origmai Object

In this section we describe the main function of this package.

1.3.1 ActionOfSl

▷ `ActionOfSl(word, Origami)` (function)

This function lets act a word in the free group $Group(S,T)$,representing an element of $Sl_2(\mathbb{Z})$ on an Origami and returns $word.Origami$.

Example

```
gap> ActionOfSl(S*T,Origami((1,3,5), (1,3)(2,4,5), 5));
Origami((1,3)(2,5,4), (2,4,5,3), 5)
```

▷ `ActionOfF2ViaCanonical(`*`Origami, word`*`)` (function)

This lets act a word in the free group $Group(S,T)$, representing an element of $Sl_2(\mathbb{Z})$, on an Origami and returns $word.Origami$. But in contrast to `ActionOfSl` the result is stored in the canonical representation. ATTENTION: the order of arguments is here reserved.

Example

```
  gap> ActionOfF2ViaCanonical(Origami((1,2), (1,3), 3), S);
Origami((1,2), (2,3), 3)
```

▷ `RightActionOfF2ViaCanonical(`*`Origami, word`*`)` (function)

This lets act a word in the free group $Group(S,T)$ on an Origami from right and returns $Origami.word = word^{-}1.Origami$, where the left action is the common action of $Sl_2(\mathbb{Z})$ on 2 mannifolds. This action has the same Veechgroup and orbits as the left action. In contrast to `ActionOfSl` the result is stored in the canonical representation. ATTENTION: the order of arguments is here reserved.

Example

```
gap> RightActionOfF2ViaCanonical(Origami((2,3), (1,3,2), 3),T);
Origami((1,2), (2,3), 3)
```

▷ `CanonicalOrigamiViaDelecroixAndStart(`*`Origami, start`*`)` (function)

This calculates a canonical representation of an origami depending on a given number start (Between 1 and the degree of of the Origami). To determine a canonical numbering the algorithm starts at the square with number start and walks over the origami in a certain way and numbers the squares in the order, they are visited . First it walks in horizontal direction one loop. Then it walks one step up (in vertical direection) and then again a loop in horizontal direction. This wil be repeated until the vertical loop is complete or all squares have been visited. If there are unvisited squares, we continue with the smallest number (in the new numbering), that has not been in a vertical loop. An Origami is connected, so that number exists. Two origamis are equal if they are described by the same permutations in their canonical representation.

Example

```
gap> CanonicalOrigamiViaDelecroixAndStart(Origami((1,10,7,6,8,9,2)(4,5),
gap> (1,8,5,4)(2,10,6)(3,9,7), 10), 1);
Origami((1,2,3,4,5,6,7)(8,9), (1,5,8,9)(2,4,7)(3,10,6), 10)
```

▷ `CanonicalOrigamiViaDelecroix(`*`Origami`*`)` (function)

This calculates a canonical representation of an origami. It calculates the representation from CanonicalOrigamiViaDelecroixAndStart with several start squares, independent of the given representation. Then it takes the minimum with respect to some order. Two origamis are equal if they are described by the same permutations in their canonical representation.

Example

```
gap> CanonicalOrigamiViaDelecroix(Origami((1,10,7,6,8,9,2)(4,5), (1,8,5,4)(2,10,6)(3,9,7), 10));
Origami((2,3,4,5,6,7,8)(9,10), (1,2,6)(3,5,7)(4,8,9,10), 10)
```

▷ `CanonicalOrigami(`*`Origami`*`)` (function)

This calculates a canonical representation of an origami. Two origamis are equal if they are described by the same permutations in their canonical representation. This is an older slower Version.

Example

```
  gap> CanonicalOrigami(Origami((1,10,7,6,8,9,2)(4,5), (1,8,5,4)(2,10,6)(3,9,7), 10));
Origami((1,2)(3,4,5,6,7,8,9), (1,2,3,7)(4,6,9)(5,10,8), 10)
```

▷ `OrigamiFamily` (family)

The only sense of this Familiy is, that Origami does not fit in any other

▷ `HorizontalPerm(`*`Origami`*`)` (attribute)

This returns the vertical permutation σ_y of the Origami.

Example

```
          gap> HorizontalPerm(Origami((1,3,5), (1,3)(2,4,5), 5));
(1,3,5)
```

▷ `VerticalPerm(`*`Origami`*`)` (attribute)

This returns the horizontal permutation σ_x of the Origami.

Example

```
          gap> VerticalPerm( Origami((1,3,5), (1,3)(2,4,5), 5));
            (1,3)(2,4,5)
```

▷ `DegreeOrigami(`*`Origami`*`)` (attribute)

This returns the degree of an Origami.

Example

```
          gap> DegreeOrigami(Origami((1,3,5), (1,3)(2,4,5), 5));
5
```

▷ `Stratum(`*`Origami`*`)` (attribute)

This calculates the stratum of an Origami. That is a list of the orders of the singularities

Example

```
          gap> Stratum(Origami((1,6,4,7,5,3)(2,8), (1,4,5,3,8,2,6), 8));
[ 1, 5 ]
```

▷ `Genus(`*`Origami`*`)` (attribute)

This calculates the genus of the Origami surface.

Example

```
          gap> Genus( Origami((1,2,3,4),(1,2)(3,4), 4) );
2
```

▷ VeechGroup(*Origami*) (attribute)

This calculates the Veechgroup of an Origami. This is a subgroup of $Sl_2(\mathbb{Z})$ of finite degree. The group is stored as ModularSubgroup from the ModularSubgroup package. The Veechgroup is represented as the coset permutations σ_S and σ_T with respect to the generators S and T. This means if i is the integer associated to the rigth coset G (Cosets(O) [i] VeechGroup = H) then we have for the coset H, associated to $\sigma_S(i)$, that $SG = H$. Dito for σ_T. You get the coset Permutations from the ModularSubgroup like in the following Example.

Example

```
gap> SAction(VeechGroup(Origami((1,2,5)(3,4,6), (1,2)(5,6), 6)));
(1,3)(2,5)(4,7)(6,8)(9,10)
gap> TAction(VeechGroup(Origami((1,2,5)(3,4,6), (1,2)(5,6), 6)));
(1,2,4)(3,6)(5,8,7,9,10)
```

▷ Cosets(*Origami*) (attribute)

This Calculates the right cosets of the Veechgroup of an Origami as list of words in S and T.

Example

```
          gap> Cosets(Origami((1,2,5)(3,4,6), (1,2)(5,6), 6));
[ < identity ...>, T, S, T^2, T*S, S*T, T^2*S, T*S*T, T^2*S*T, T^2*S*T^2 ]
```

▷ (*Origami1, Origami2*) (function)

This tests wether Origami1 is equal up to Origami2 up to numbering of the squares.

Example

```
gap> EquivalentOrigami(Origami((1,4)(2,6,3), (1,5)(2,3,6,4), 6), Origami((1,4,3)(2,5), (1,5,3,4)
true

gap> EquivalentOrigami(Origami((1,4)(2,6,3), (1,5)(2,3,6,4), 6), Origami((1,2,5)(3,4,6), (1,2)(5
false
```