The Origami Package

Computiong Veechgroup of origamis

Version UNKNOWNEntity(VERSION)

UNKNOWNEntity(RELEASEDATE)

Pascal Kattler
Andrea Thevis

Pascal Kattler Email: kattler@math.uni-sb.de
Homepage: http://www.math.uni-sb.de/ag/weitze/
Address: AG Weitze-Schmithüsen
FR 6.1 Mathematik
Universität des Saarlandes
D-66041 Saarbrücken

Andrea Thevis Email: thevis@math.uni-sb.de
Homepage: http://www.math.uni-sb.de/ag/weitze/
Address: AG Weitze-Schmithüsen
FR 6.1 Mathematik
Universität des Saarlandes
D-66041 Saarbrücken

Copyright

Acknowledgements

Supported by Project I.8 of SFB-TRR 195 'Symbolic Tools in Mathematics and their Application' of the German Research Foundation (DFG).

Contents

Chapter 1

The Veechgroup of origamis

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 glue 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, which acts transitively on $\{1 \ldots d\}$ up to simultaneous conjugation. We introduce a new type of origamis, namely origami objects, which are created by this two permutations and its degree. The degree of an origami is the number of squares. Origamis are stored as such objects. 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 of $SL_2(\mathbb{Z})$

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

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 $\tilde{S}$ and $\tilde{T}$.

1.3 The Origmai Object

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

1.3.1 ActionOfSl

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

The group $SL_2(\mathbb{Z})$ acts on the set of origamis with a fixednumber of squares. This function lets act a word in the free group $Group(\tilde{S}, \tilde{T})$,representing an element of $Sl_2(\mathbb{Z})$ on an origami and returns $word.origami$. The word is given as string, as you can see in the following example.

Example
```
gap> ActionOfSl("ST",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 manni-folds. 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 function 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. This sqare is labeled 1 in the new numbering. The algorithm walks along the origami in the following way and numbers the squares in the order, they are visited. First it walks in horizontal direction until it reaches the square with number start again. 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. This function is used to determine a canonical origami independent of the start.

Example
```
gap> CanonicalOrigamiAndStart(Origami((1,10,7,6,8,9,2)(4,5),
> (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 all squares as start squares, independent of the given representation. Then it takes the minimum with respect to the order which GAP automatically uses to compare paires of permutations. Two origamis are equal if they are described by the same permu-tations in their canonical representation.

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((2,3,4,5,6,7,8)(9,10), (1,2,6)(3,5,7)(4,8,9,10), 10)
```

▷ `OrigamiFamily` (family)

The only sense of this familiy is, that origami does not fit in any other family.

▷ `HorizontalPerm(`*origami*`)` (attribute)

This function 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 function 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 function returns the degree of an origami.

Example

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

▷ `Stratum(`*Origami*`)` (attribute)

This function calculates the stratum of an Origami. That is a list of the degrees 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 function 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 function calculates the Veechgroup of an origami. This is a subgroup $SL_2(\mathbb{Z})$ of finite index. The group is stored as a 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 right coset G (Cosets(O) [i] VeechGroup = H) then we have for the coset H, associated to $\sigma_S(i)$, that $SG = H$. Analogously for σ_T. You get the coset Permutations from the ModularSubgroup as 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 function calculates the right cosets of the Veechgroup of an origami as a 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 ]
```

▷ `EquivalentOrigami(origami1, origami2)` (function)

This function tests wether origami1 is equal to origami2 up to renumbering 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)(2,6), 6));
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)(5,6), 6));
false
```