The Origami Package

Computing Veechgroups 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 (also known as square-tiled surface) is a finite covering of a torus which is ramified at most over one point. It can be described in the following way from two permutations $\sigma_x, \sigma_y \in S_d$. We take d squares $Q_1, \dots, Q_d$ 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 $Q_{\sigma_x(i)}$. We require origamis to be connected and thus the group generated by σ_x and σ_y acts transitively on $\{1, \dots, d\}$. In this package we identify an origami with a pair of permutations, which acts transitively on $\{1, \dots, d\}$ 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 $\tilde{T}$. We consider the canonical epimorphism $\pi : F \to SL_2(\mathbb{Z})$ with $\pi(\tilde{S}) = S$ and $\pi(\tilde{T}) = T$.

1.3 The Origami Object

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

1.3.1 ActionOfSpecialLinearGroup

▷ `ActionOfSpecialLinearGroup(`*word, origami*`)` (function)

The group $SL_2(\mathbb{Z})$ acts on the set of origamis of fixed degree. Given a word in the free group $Group(\tilde{S},\tilde{T})$ this function computes $\pi(word) \in SL_2(\mathbb{Z})$ and returns $\pi(word).origami$. The word is given as a string, as shown in the following example.

Example
```
gap> ActionOfSpecialLinearGroup("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)

Given a word in the free group $Group(\tilde{S},\tilde{T})$ this function computes $\pi(word) \in SL_2(\mathbb{Z})$ and returns $\pi(word).origami$. But in contrast to `ActionOfSpecialLinearGroup` the result is stored in the canonical representation. ATTENTION: the order of arguments is switched compared to the order of the arguments in the function ActionOfSpecialLinearGroup.

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

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

This function computes the right action of the projection of a word in the free group $Group(\tilde{S},\tilde{T})$ on an Origami. It returns $Origami.\pi(word) = \pi(word)^{-}1.Origami$, where the left action is the common action of $_2(\mathbb{Z})$ on 2-mannifolds. This action has the same orbits as the left action. For the Veechgroup computation both actions can be used and give the same result. In contrast to `ActionOfSpecialLinearGroup` the result is stored in the canonical representation. ATTENTION: the order of arguments is switched compared to the order of the arguments in the function ActionOf-SpecialLinearGroup.

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 as 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 direction) and then again a loop in horizontal direction. This will 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 starting point. The idea for the algorithm derives from the algorithm to_standard_form implemented in the SageMath package surface_dynamics. (REFERENZ HINZUFÜGEN!)

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 function computes 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 (up to renumbering of the squares) if they are described by the same permutations 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)

Since origamis do not fit in any existing family in GAP, we introduce a new family for origami objects called OrigamiFamily. ▷ `HorizontalPerm(origami)` (attribute)

This function returns the horizontal permutation σ_y of an origami.

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

▷ `VerticalPerm(origami)` (attribute)

This function returns the vertical permutation σ_x of an 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. The stratum of an origami is a list of the nonzero degrees of the singularities. For a singularity with cone angle $2\pi k$ the degree of the singularity is $k-1$.

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. The Veechgroup G of an origami is a subgroup $SL_2(\mathbb{Z})$ of finite index. The group is stored as a ModularSubgroup from the ModularSubgroup package. It is represented by two permutations σ_S and σ_T describing how the generators S and T of $SL_2(\mathbb{Z})$ act on the cosets of G in $SL_2(\mathbb{Z})$. E.g, if $SH_i = H_j$ and H_i, H_j are the cosets associated to the integers i, j, respectively, then $\sigma_S(i) = j$. The algorithm was introduced by Gabriela Weitze-Schmithüsen in (REFERENZ HINZUFÜGEN!). You get the coset permutations using the ModularSubgroup package 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 whether 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
```