ModularGroup

Computing with finite-index subgroups of SL(2,Z)

0.0.1

10 January 2018

Max Mustermann

John Doe

Max Mustermann
Email: max.mustermann@example.com
Homepage: http://www.math.uni-sb.de/ag/weitze/
Address: AG Weitze-Schmithüsen
FR 6.1 Mathematik
Universität des Saarlandes
D-66041 Saarbrücken

John Doe
Email: john.doe@example.com
Homepage: http://www.math.uni-sb.de/ag/weitze/
Address: AG Weitze-Schmithüsen
FR 6.1 Mathematik
Universität des Saarlandes
D-66041 Saarbrücken

Contents

Chapter 1

The Modular Group and its subgroups

This package contains methods for computing with finite-index subgroups of the modular group $\mathrm{SL}(2,\mathbb{Z})$ which are given by a coset permutation representation with respect to the generators

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

We will call these subgroups 'modular subgroups'.

1.1 Construction of modular subgroups

In this section we describe how to construct modular subgroups from a given coset permutation representation or from a list of generator matrices and some related methods.

1.1.1 ModularSubgroup (for two permutations)

▷ `ModularSubgroup(`*s, t*`)` (function)

Returns: an object representing a modular subgroup

This method constructs a modular subgroup from two given permutations (provided they describe a coset action).

1.1.2 ModularSubgroup (for a list of matrices)

▷ `ModularSubgroup(`*gens*`)` (function)

Returns: an object representing a modular subgroup

This is another constructor for a modular subgroup, with the difference that it takes a list of generator matrices as input and calculates the coset graph of the generated group. One has to be careful though when using this method, because no check is performed if the generated group has finite index! Internally, when trying to calculate the coset graph, we just enumerate all cosets until we are done or some limit fixed is reached. This also exposes another weakness of this method: The coset enumeration might be very time-consuming, so constructing modular subgroups from a list of generators is not always feasible. On the other hand, the clear advantage of constructing a modular subgroup in this way is that it will alreasy know its generators. So future computations with this group involving generators will most likely be faster.

1.1.3 DefinesCosetAction (for two permutations)

▷ `DefinesCosetAction(s, t)` (function)

Returns: true or false

This is an auxiliary method that takes two permutations as input and checks if they describe the action of the generators S and T on the cosets of some group. This check is for example performed in the constructor for a modular subgroup.

1.1.4 CosetActionFromGenerators (for a list of generator matrices)

▷ `CosetActionFromGenerators(gens)` (function)

Returns: two permutations

Takes a list of generator matrices and calculates the coset permutation representation of the generated subgroup. The same warning as above applies: No check is performed if the generated subgroup actually has finite index.

1.1.5 STDecomposition (for a matrix in SL(2,Z))

▷ `STDecomposition(M)` (function)

Returns: a word in S and T

Takes a matrix in $\mathrm{SL}(2,\mathbb{Z})$ and decomposes it in a word in the generators S and T.

1.1.6 SAction (for a modular subgroup)

▷ `SAction(G)` (function)

Returns: a permutation

Takes a modular subgroup and returns a permutation corresponding to the action of the generator matrix S.

1.1.7 TAction (for a modular subgroup)

▷ `TAction(G)` (function)

Returns: a permutation

Takes a modular subgroup and returns a permutation corresponding to the action of the generator matrix T.

1.2 Computations with modular subgroups

In this section we describe the implemented method for computing with modular subgroups.

1.2.1 Index (for a modular subgroup)

▷ `Index(G)` (function)

Returns: a natural number

Takes a modular subgroup and returns its index in $\mathrm{SL}(2,\mathbb{Z})$.

1.2.2 IsCongruenceSubgroup (for a modular subgroup)

▷ `IsCongruenceSubgroup(G)` (function)

Returns: true or false

Tests whether a given modular subgroup is a congruence subgroup.

1.2.3 RightCosetRepresentatives (for a modular subgroup)

▷ `RightCosetRepresentatives(G)` (function)

Returns: a list of matrices

Calculates a list of representatives of the right cosets of a given modular subgroup.

1.2.4 GeneralizedLevel (for a modular subgroup)

▷ `GeneralizedLevel(G)` (function)

Returns: a natural number

Computes the generalized level (i.e. the lowest common multiple of all cusp widths) of a given modular subgroup.

1.2.5 GeneratorsOfGroup (for a modular subgroup)

▷ `GeneratorsOfGroup(G)` (function)

Returns:

Calculates a list of generators for a given modular subgroup. Note: The returned list might contain redundant generators (or even duplicates). This calculation involves enumerating the cosets of the given group and might become very slow for large index.

1.2.6 IsElementOf (for a matrix in SL(2,Z) and a modular subgroup)

▷ `IsElementOf(A, G)` (function)

Returns: true or false

This is a membership test for modular subgroups given by a coset permutation representation.

1.2.7 CuspWidth (for a rational number or infinity and a modular subgroup)

▷ `CuspWidth(c, G)` (function)

Returns: a natural number

Calculates the width of c with respect to a given modular subgroup, i.e. the smallest k such that $\pm gT^kg^{-1} \in G$ where $g \in \mathrm{SL}(2,\mathbb{Z})$ such that $g\infty = c$.

1.2.8 CuspsEquivalent (for two cusps (i.e. rational numbers or infinity) and a modular subgroup)

▷ `CuspsEquivalent(c1, c2, G)` (function)

Returns: true or false

Checks if two cusps are equivalent with respect to a given modular subgroup.

1.2.9 Cusps (for a modular subgroup)

▷ `Cusps(G)` (function)

Returns: a list of cusps

Calculates a list of inequivalent cusp representative for a given modular subgroup.

1.2.10 CuspsRedundant (for a modular subgroup)

▷ `CuspsRedundant(G)` (function)

Returns: a list of cusps

Calculates a list of cusp representatives for a given modular subgroup. Will most likely produce redundant cusps, i.e. there will be cusps which are equivalent. This method has the advantage that it is rather fast, compared to the one above though. For example it is used in the calculation of the generalized level, since when computing the lcm of the cusp widths, we do not care about duplicates.

Index