

The ModularGroup Package

Finite-index subgroups of $(P)SL_2(\mathbb{Z})$

Version 2.0.3

17 June 2026

**Sebastian Engelhardt
Luca L. Junk
Hannah Wagmann
Gabriela Weitze-Schmithüsen**

Sebastian Engelhardt Email: seen00001@stud.uni-saarland.de
Homepage: <https://www.uni-saarland.de/lehrstuhl/weitze-schmithuesen.html>

Luca L. Junk Email: junk@math.uni-sb.de
Homepage: <https://www.uni-saarland.de/lehrstuhl/weber-moritz/team/luca-junk.html>

Hannah Wagmann Email: wagmann@math.uni-sb.de
Homepage: [https://www.uni-saarland.de/lehrstuhl/weitze-schmithuesen/
team/hannah-wagmann.html](https://www.uni-saarland.de/lehrstuhl/weitze-schmithuesen/team/hannah-wagmann.html)

Gabriela Weitze-Schmithüsen Email: weitze@math.uni-sb.de
Homepage: [https://www.uni-saarland.de/lehrstuhl/weitze-schmithuesen/
team/gabriela-weitze-schmithuesen.html](https://www.uni-saarland.de/lehrstuhl/weitze-schmithuesen/team/gabriela-weitze-schmithuesen.html)

Copyright

© 2026 by Sebastian Engelhardt, Luca L. Junk, Hannah Wagmann and Gabriela Weitze-Schmithüsen

Acknowledgements

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

Contents

1	Introduction	4
1.1	General aims of the ModularGroup package	4
1.2	Technicalities	4
2	Subgroups of $SL_2(\mathbb{Z})$	5
2.1	Construction of modular subgroups	5
2.2	Getters for the coset action	7
2.3	Computing with modular subgroups	8
2.4	Miscellaneous	15
3	Subgroups of $PSL_2(\mathbb{Z})$	18
3.1	Construction of projective modular subgroups	18
3.2	Getters for the coset action	19
3.3	Computing with projective modular subgroups	19
	References	27
	Index	28

Chapter 1

Introduction

1.1 General aims of the `ModularGroup` package

This `GAP` package provides methods for computing with finite-index subgroups of the modular groups $\mathrm{SL}_2(\mathbb{Z})$ and $\mathrm{PSL}_2(\mathbb{Z})$. This includes, but is not limited to, computation of the generalized level, index or cusp widths. It also implements algorithms described in [Hsu96] and [HL14] for testing if a given group is a congruence subgroup. Hence it differs from the `Congruence` package [DJKV18], which can be used - among other things - to construct canonical congruence subgroups of $\mathrm{SL}_2(\mathbb{Z})$.

1.2 Technicalities

A convenient way to represent finite-index subgroups of $\mathrm{SL}_2(\mathbb{Z})$ is by specifying the action of generator matrices of $\mathrm{SL}_2(\mathbb{Z})$ on the right cosets by right multiplication. For example, one could choose the generators

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

and represent a subgroup as a tuple of transitive permutations (σ_S, σ_T) describing the action of S and T . This is exactly the way this package internally treats such subgroups. We use the convention that 1 corresponds to the coset of the identity matrix. Note that such a representation as a tuple of permutations is only unique up to relabelling of the cosets, i.e. up to simultaneous conjugation (fixing the 1 coset by our convention). To not restrict `ModularGroup` to right multiplication on right cosets, getters and setters are defined in two versions: one by right multiplication on right cosets ("`ViaRightAction`") and one by left multiplication on left cosets ("`ViaLeftAction`").

Chapter 2

Subgroups of $SL_2(\mathbb{Z})$

For representing finite-index subgroups of $SL_2(\mathbb{Z})$, this package introduces the new object `ModularSubgroup`. As stated in the introduction, a `ModularSubgroup` essentially consists of the two permutations σ_S and σ_T describing the coset graph with respect to the generators S and T (with the convention that 1 corresponds to the identity coset). So explicitly specifying these permutations is the canonical way to construct a `ModularSubgroup`.

Though you might not always have a coset graph of your subgroup at hand, but rather a list of generating matrices. Therefore we implement multiple constructors for `ModularSubgroup`: three that take as input two permutations describing the coset graph with respect to different pairs of generators of $SL_2(\mathbb{Z})$, and one that takes a list of $SL_2(\mathbb{Z})$ matrices as generators.

2.1 Construction of modular subgroups

2.1.1 `ModularSubgroup`

▷ `ModularSubgroupViaRightAction(s, t)` (operation)

Returns: A modular subgroup.

Constructs a `ModularSubgroup` object corresponding to the finite-index subgroup of $SL_2(\mathbb{Z})$ described by the permutations s and t .

This constructor tests if the given permutations actually describe the coset action of the matrices

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

by checking that they act transitively and satisfy the relations

$$s^4 = (s^3t)^3 = s^2ts^{-2}t^{-1} = 1$$

Upon creation, the cosets are renamed in a [standardized way](#) to make the internal interaction with existing GAP methods easier. (The fact that 1 corresponds to the identity coset is not changed by this)

Example

```
gap> G := ModularSubgroupViaRightAction(
> (1,2)(3,4)(5,6)(7,8)(9,10),
> (1,4)(2,5,9,10,8)(3,7,6));
<modular subgroup of index 10>
```

- ▷ `ModularSubgroupViaLeftAction(s, t)` (operation)
Returns: A modular subgroup.
 Analogous to `ModularSubgroupViaRightAction`, but with left coset actions.

2.1.2 ModularSubgroupST

- ▷ `ModularSubgroupSTViaRightAction(s, t)` (operation)
Returns: A modular subgroup.
 Synonymous for `ModularSubgroupViaRightAction` (2.1.1) (see above).
 ▷ `ModularSubgroupSTViaLeftAction(s, t)` (operation)
Returns: A modular subgroup.
 Synonymous for `ModularSubgroupViaLeftAction` (2.1.1) (see above).

2.1.3 ModularSubgroupRT

- ▷ `ModularSubgroupRTViaRightAction(r, t)` (operation)
Returns: A modular subgroup.
 Constructs a `ModularSubgroup` object corresponding to the finite-index subgroup of $SL_2(\mathbb{Z})$ determined by the permutations r and t which describe the action of the matrices

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

on the right cosets.

A check is performed if the permutations actually describe such an action on the cosets of some subgroup.

Upon creation, the cosets are renamed in a [standardized way](#) to make the internal interaction with existing GAP methods easier. (The fact that 1 corresponds to the identity coset is not changed by this)

Example

```
gap> G := ModularSubgroupRTViaRightAction(
> (1,9,8,10,7)(2,6)(3,4,5),
> (1,3)(2,4,8,10,5)(6,9,7));
<modular subgroup of index 10>
```

- ▷ `ModularSubgroupRTViaLeftAction(r, t)` (operation)
Returns: A modular subgroup.
 Analogous to `ModularSubgroupRTViaRightAction`, but with left coset actions.

2.1.4 ModularSubgroupSJ

- ▷ `ModularSubgroupSJViaRightAction(s, j)` (operation)
Returns: A modular subgroup.
 Constructs a `ModularSubgroup` object corresponding to the finite-index subgroup of $SL_2(\mathbb{Z})$ determined by the permutations s and j which describe the action of the matrices

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

on the right cosets.

A check is performed if the permutations actually describe such an action on the cosets of some

subgroup.

Upon creation, the cosets are renamed in a [standardized way](#) to make the internal interaction with existing GAP methods easier. (The fact that 1 corresponds to the identity coset is not changed by this)

Example

```
gap> G := ModularSubgroupSJViaRightAction(
> (1,2)(3,6)(4,7)(5,9)(8,10),
> (1,5,6)(2,3,7)(4,9,10));
<modular subgroup of index 10>
```

▷ `ModularSubgroupSJViaLeftAction(s, j)` (operation)

Returns: A modular subgroup.

Analogous to `ModularSubgroupSJViaRightAction`, but with left coset actions.

2.1.5 ModularSubgroup (ModularSubgroupGens)

▷ `ModularSubgroup(gens)` (operation)

Returns: A modular subgroup.

Constructs a `ModularSubgroup` object corresponding to the finite-index subgroup of $SL_2(\mathbb{Z})$ generated by the matrices in `gens`.

No test is performed to check if the generated subgroup actually has finite index!

This constructor implicitly computes a coset table of the subgroup. Hence it might be slow for very large index subgroups.

Example

```
gap> G := ModularSubgroup([
> [[1,2], [0,1]],
> [[1,0], [2,1]],
> [[-1,0], [0,-1]]
> ]);
<modular subgroup of index 6>
```

2.2 Getters for the coset action

2.2.1 SAction

▷ `SRightAction(G)` (operation)

Returns: A permutation.

Returns the permutation σ_S describing the action of the matrix S on the right cosets of G .

▷ `SLeftAction(G)` (operation)

Returns: A permutation.

Analogous to `SRightAction`, but with left coset actions.

2.2.2 TAction

▷ `TRightAction(G)` (operation)

Returns: A permutation.

Returns the permutation σ_T describing the action of the matrix T on the right cosets of G .

▷ `TLeftAction(G)` (operation)

Returns: A permutation.

Analogous to `TRightAction`, but with left coset actions.

2.2.3 RAction

- ▷ `RRightAction(G)` (operation)
Returns: A permutation.
Returns the permutation σ_R describing the action of the matrix R on the right cosets of G .
- ▷ `RLeftAction(G)` (operation)
Returns: A permutation.
Analogous to `RRightAction`, but with left coset actions.

2.2.4 JAction

- ▷ `JRightAction(G)` (operation)
Returns: A permutation.
Returns the permutation σ_J describing the action of the matrix J on the right cosets of G .
- ▷ `JLeftAction(G)` (operation)
Returns: A permutation.
Analogous to `JRightAction`, but with left coset actions.

2.2.5 CosetActionOf

- ▷ `CosetRightActionOf(A, G)` (operation)
Returns: A permutation.
Returns the permutation σ_A describing the action of the matrix $A \in \mathrm{SL}_2(\mathbb{Z})$ on the right cosets of G .
- ▷ `CosetLeftActionOf(A, G)` (operation)
Returns: A permutation.
Analogous to `CosetRightActionOf`, but with left coset actions.

2.3 Computing with modular subgroups

2.3.1 Index (IndexSL2Z)

- ▷ `Index(G)` (attribute)
Returns: A natural number.
For a given modular subgroup G this method returns its index in $\mathrm{SL}_2(\mathbb{Z})$. As G is internally stored as permutations (s, t) this is just `LargestMovedPoint([s, t])` (or 1 if the permutations are trivial).

Example

```
gap> G := ModularSubgroup((1,2)(3,5)(4,6), (1,3)(2,4)(5,6));
<modular subgroup of index 6>
gap> Index(G);
6
```

2.3.2 GeneralizedLevel (GeneralizedLevelSL2Z)

- ▷ `GeneralizedLevel(G)` (attribute)
Returns: A natural number.
This method calculates the general Wohlfahrt level (i.e. the lowest common multiple of all cusp widths) of G as defined in [Woh64].

Example

```
gap> G := ModularSubgroup((1,2)(3,5)(4,6), (1,3)(2,4)(5,6));
<modular subgroup of index 6>
gap> GeneralizedLevel(G);
2
```

2.3.3 CosetRepresentatives

▷ `RightCosetRepresentatives(G)` (attribute)

Returns: A list of words.

This function returns a list of representatives of the (right) cosets of G as words in S and T .

Example

```
gap> G := ModularSubgroup((1,2), (2,3));
<modular subgroup of index 3>
gap> RightCosetRepresentatives(G);
[ <identity ...>, S, S*T ]
```

▷ `LeftCosetRepresentatives(G)` (attribute)

Returns: A list of words.

This function returns a list of representatives of the (left) cosets of G as words in S and T .

2.3.4 WordGeneratorsOfGroup (GeneratorsOfGroupSL2Z)

▷ `WordGeneratorsOfGroup(G)` (attribute)

Returns: A list of words.

Calculates a list of generators (as words in S and T) of G . This list might include redundant generators (or even duplicates).

Example

```
gap> G := ModularSubgroup((1,2)(3,5)(4,6), (1,3)(2,4)(5,6));
<modular subgroup of index 6>
gap> WordGeneratorsOfGroup(G);
[ S^-2, T^-2, S*T^-2*S^-1 ]
```

2.3.5 GeneratorsOfGroup

▷ `GeneratorsOfGroup(G)` (attribute)

Returns: A list of matrices.

Calculates a list of generator matrices of G . This list might include redundant generators (or even duplicates).

Example

```
gap> G := ModularSubgroup((1,2)(3,5)(4,6), (1,3)(2,4)(5,6));
<modular subgroup of index 6>
gap> GeneratorsOfGroup(G);
[ [ [ -1, 0 ], [ 0, -1 ] ], [ [ 1, -2 ], [ 0, 1 ] ], [ [ 1, 0 ], [ 2, 1 ] ] ]
```

2.3.6 IsCongruence (IsCongruenceSL2Z)

▷ IsCongruence(G) (attribute)

Returns: True or false.

This method test whether a given modular subgroup G is a congruence subgroup. It is essentially an implementation of an algorithm described in [HL14].

Example

```
gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> IsCongruence(G);
true
```

2.3.7 Cusps (CuspsSL2Z)

▷ Cusps(G) (attribute)

Returns: A list of rational numbers and infinity.

This method computes a list of inequivalent cusp representatives with respect to G .

Example

```
gap> G := ModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
> );
<modular subgroup of index 24>
gap> Cusps(G);
[ infinity, 0, 1, 2, 3/2, 5/3 ]
```

2.3.8 CuspWidth (CuspWidthSL2Z)

▷ CuspWidth(c, G) (operation)

Returns: A natural number.

This method takes as input a cusp c (a rational number or infinity) and a modular group G and calculates the width of this cusp with respect to G .

Example

```
gap> G := ModularSubgroup(
> (1,2,6,3)(4,11,15,12)(5,13,16,14)(7,17,9,18)(8,19,10,20)(21,24,22,23),
> (1,4,5)(2,7,8)(3,9,10)(6,15,16)(11,20,21)(12,19,22)(13,23,17)(14,24,18)
> );
<modular subgroup of index 24>
gap> CuspWidth(-1, G);
3
gap> CuspWidth(infinity, G);
3
```

2.3.9 CuspsEquivalent (CuspsEquivalentSL2Z)

▷ `CuspsEquivalent(p , q , G)` (operation)

Returns: True or false.

Takes two cusps p and q and a modular subgroup G and checks if they are equivalent modulo G , i.e. if there exists a matrix $A \in G$ with $Ap = q$.

Example

```
gap> G := ModularSubgroup(
> (1,2,6,3)(4,11,15,12)(5,13,16,14)(7,17,9,18)(8,19,10,20)(21,24,22,23),
> (1,4,5)(2,7,8)(3,9,10)(6,15,16)(11,20,21)(12,19,22)(13,23,17)(14,24,18)
> );
<modular subgroup of index 24>
gap> CuspsEquivalent(infinity, 1, G);
false
gap> CuspsEquivalent(-1, 1/2, G);
true
```

2.3.10 CosetRepresentativeOfCusp (CosetRepresentativeOfCuspSL2Z)

▷ `CosetRepresentativeOfCusp(c , G)` (operation)

Returns: A word in S and T.

For a cusp c this function returns a right coset representative A of G such that $A\infty$ and c are equivalent with respect to G .

Example

```
gap> G := ModularSubgroup(
> (1,2,6,3)(4,11,15,12)(5,13,16,14)(7,17,9,18)(8,19,10,20)(21,24,22,23),
> (1,4,5)(2,7,8)(3,9,10)(6,15,16)(11,20,21)(12,19,22)(13,23,17)(14,24,18)
> );
<modular subgroup of index 24>
gap> CosetRepresentativeOfCusp(4, G);
T*S
```

2.3.11 IndexModN (IndexModNSL2Z)

▷ `IndexModN(G , N)` (operation)

Returns: A natural number.

For a modular subgroup G and a natural number N this method calculates the index of the projection \bar{G} of G in $SL_2(\mathbb{Z}/N\mathbb{Z})$.

Example

```
gap> G := ModularSubgroup((1,2)(3,5)(4,6), (1,3)(2,4)(5,6));
<modular subgroup of index 6>
gap> IndexModN(G, 2);
6
```

2.3.12 Deficiency (DeficiencySL2Z)

▷ `Deficiency(G , N)` (operation)

Returns: A natural number.

For a modular subgroup G and a natural number N this method calculates the so-called *deficiency* of G from being a congruence subgroup of level N .

The deficiency of a finite-index subgroup Γ of $\mathrm{SL}_2(\mathbb{Z})$ was introduced in [WS15]. It is defined as the index $[\Gamma(N) : \Gamma(N) \cap \Gamma]$ where $\Gamma(N)$ is the principal congruence subgroup of level N .

Example

```
gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> Deficiency(G, 2);
2
gap> Deficiency(G, 4);
1
```

2.3.13 Deficiency (DeficiencySL2ZAttr)

▷ Deficiency(G) (attribute)

Returns: A natural number.

Shorthand for Deficiency(G , GeneralizedLevel(G)).

Example

```
gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> Deficiency(G);
1
gap> Deficiency(G, GeneralizedLevel(G));
1
```

2.3.14 Projectivization

▷ Projectivization(G) (operation)

Returns: A projective modular subgroup.

For a given modular subgroup G this function calculates its image \bar{G} under the projection $\pi: \mathrm{SL}_2(\mathbb{Z}) \rightarrow \mathrm{PSL}_2(\mathbb{Z})$.

Example

```
gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> Projectivization(G);
<projective modular subgroup of index 6>
```

2.3.15 ConjugateGroup (ConjugateGroupSL2Z)

▷ `ConjugateGroup(G, A)` (operation)

Returns: A `ModularSubgroup`.

Conjugates the group G by the $SL_2(\mathbb{Z})$ matrix A and returns the group $A^{-1} * G * A$.

2.3.16 NormalCore (NormalCoreSL2Z)

▷ `NormalCore(G)` (attribute)

Returns: A modular subgroup.

Calculates the normal core of G in $SL_2(\mathbb{Z})$, i.e. the maximal subgroup of G that is normal in $SL_2(\mathbb{Z})$.

Example

```
gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> NormalCore(G);
<modular subgroup of index 48>
```

2.3.17 QuotientByNormalCore (QuotientByNormalCoreSL2Z)

▷ `QuotientByNormalCore(G)` (attribute)

Returns: A finite group.

Calculates the quotient of $SL_2(\mathbb{Z})$ by the normal core of G .

Example

```
gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> QuotientByNormalCore(G);
<permutation group with 2 generators>
```

2.3.18 AssociatedCharacterTable (AssociatedCharacterTableSL2Z)

▷ `AssociatedCharacterTable(G)` (attribute)

Returns: A character table.

Returns the character table of $SL_2(\mathbb{Z})/N$ where N is the normal core of G .

Example

```
gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> AssociatedCharacterTable(G);
CharacterTable( <permutation group of size 48 with 2 generators> )
```

2.3.19 IsElementOf (IsElementOfSL2Z)

▷ IsElementOf(A , G) (operation)

Returns: True or false.

This function checks if a given matrix A is an element of the modular subgroup G .

Example

```
gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> IsElementOf([[ -1,0],[0,-1]], G);
false
gap> IsElementOf([[ 1,4],[0,1]], G);
true
```

2.3.20 IsWordElementOf (IsWordElementOfSL2ZString)

▷ IsWordElementOf(s , G) (operation)

Returns: True or false.

This function checks if a given string s represents an element of the modular subgroup G , written in the generators S and T .

Example

```
gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> IsWordElementOf("S^4 * T^2", G);
true
```

2.3.21 IsWordElementOf (IsWordElementOfSL2Z)

▷ IsWordElementOf(w , G) (operation)

Returns: True or false.

This function checks if a given word w in the generators S and T represents an element of the modular subgroup G .

Example

```
gap> G := ModularSubgroup([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
<modular subgroup of index 12>
gap> F := FreeGroup("S", "T");
<free group on the generators [ S, T ]>
gap> S := F.1;
S
gap> T := F.2;
T
gap> SL2Z := F / ParseRelators([S, T], "S^4, (S^3*T)^3, S^2*T*S^-2*T^-1");
```

```

<fp group on the generators [ S, T ]>
gap> S := GeneratorsOfGroup(SL2Z)[1];
S
gap> T := GeneratorsOfGroup(SL2Z)[2];
T
gap> IsWordElementOf(S^4 * T^2, G);
true

```

2.3.22 Genus (GenusSL2Z)

▷ Genus(G) (attribute)

Returns: A non-negative integer.

Computes the genus of the quotient $G \backslash \mathbb{H}$ via an algorithm described in [Sch04].

Example

```

gap> G := ModularSubgroup((1,2),(2,3));
<modular subgroup of index 3>
gap> Genus(G);
0

```

2.4 Miscellaneous

The following functions are mostly helper functions used internally and are only documented for sake of completeness.

2.4.1 DefinesCosetActionST

▷ DefinesCosetActionST(s , t) (operation)

Returns: True or false.

Checks if two given permutations s and t describe the action of the generator matrices S and T on the cosets of some subgroup. This is the case if they satisfy the relations

$$s^4 = (s^3 t)^3 = s^2 t s^{-2} t^{-1} = 1$$

and act transitively.

Example

```

gap> s := (1,2)(3,4)(5,6)(7,8)(9,10);;
gap> t := (1,4)(2,5,9,10,8)(3,7,6);;
gap> DefinesCosetActionST(s,t);
true

```

2.4.2 DefinesCosetActionRT

▷ DefinesCosetActionRT(r , t) (operation)

Returns: True or false.

Checks if two given permutations r and t describe the action of the generator matrices R and T on the cosets of some subgroup. This is the case if they satisfy the relations

$$(rt^{-1}r)^4 = ((rt^{-1}r)^3t)^3 = (rt^{-1}r)^2t(rt^{-1}r)^{-2}t^{-1} = 1$$

and act transitively.

Example

```
gap> r := (1,9,8,10,7)(2,6)(3,4,5);;
gap> t := (1,3)(2,4,8,10,5)(6,9,7);;
gap> DefinesCosetActionRT(r,t);
true
```

2.4.3 DefinesCosetActionSJ

▷ `DefinesCosetActionSJ(s, j)` (operation)

Returns: True or false.

Checks if two given permutations s and j describe the action of the generator matrices S and J on the cosets of some subgroup. This is the case if they satisfy the relations

$$s^4 = (s^3 j^{-1} s^{-1})^3 = s^2 j^{-1} s^{-2} j = 1$$

and act transitively.

Example

```
gap> s := (1,2)(3,4)(5,6)(7,8)(9,10);;
gap> j := (1,5,6)(2,3,7)(4,9,10);;
gap> DefinesCosetActionSJ(s,j);
true
```

2.4.4 CosetActionFromGenerators

▷ `RightCosetActionFromGenerators(gens)` (operation)

Returns: A tuple of permutations.

Takes a list of generator matrices and calculates the right coset graph (as two permutations σ_S and σ_T) of the generated subgroup of $SL_2(\mathbb{Z})$.

Example

```
gap> RightCosetActionFromGenerators([
> [[1,2],[0,1]],
> [[1,0],[2,1]]
> ]);
[ (1,2,5,3)(4,8,10,9)(6,11,7,12), (1,4)(2,6)(3,7)(5,10)(8,12,9,11) ]
```

▷ `LeftCosetActionFromGenerators(gens)` (operation)

Returns: A tuple of permutations.

Analogous to `RightCosetActionFromGenerators`, but with left coset actions.

2.4.5 STDecomposition

▷ `STDecomposition(A)` (operation)

Returns: A word in S and T .

Takes a matrix $A \in SL_2(\mathbb{Z})$ and decomposes it into a word in the generator matrices S and T .

Example

```
gap> M := [ [ 4, 3 ], [ -3, -2 ] ];;
gap> STDecomposition(M);
S^2*T^-1*S^-1*T^2*S^-1*T^-1*S^-1
```

2.4.6 RTDecomposition

▷ `RTDecomposition(A)` (operation)

Returns: A word in R and T .

Takes a matrix $A \in \mathrm{SL}_2(\mathbb{Z})$ and decomposes it into a word in the generator matrices R and T .

Example

```
gap> M := [ [ 4, 3 ], [ -3, -2 ] ];;
gap> RTDecomposition(M);
(R*T^-1*R)^2*T^-1*R^-1*(T*R^-1*T)^2*R^-1*T^-1*R^-1*T*R^-1
```

2.4.7 SJDecomposition

▷ `SJDecomposition(A)` (operation)

Returns: A word in S and J .

Takes a matrix $A \in \mathrm{SL}_2(\mathbb{Z})$ and decomposes it into a word in the generator matrices S and J .

Example

```
gap> M := [ [ 4, 3 ], [ -3, -2 ] ];;
gap> SJDecomposition(M);
S^3*J*(S^-1*J^-1)^2*S^-1*J*S^-1
```

2.4.8 STDecompositionAsList

▷ `STDecompositionAsList(A)` (operation)

Returns: A list representing a word in S and T .

Takes a matrix $A \in \mathrm{SL}_2(\mathbb{Z})$ and decomposes it into a word in the generator matrices S and T . The word is represented as a list in the format `[[generator, exponent], ...]`

Example

```
gap> M := [ [ 4, 3 ], [ -3, -2 ] ];;
gap> STDecompositionAsList(M);
[ [ "S", 2 ], [ "T", -1 ], [ "S", -1 ], [ "T", 2 ], [ "S", -1 ], [ "T", -1 ],
  [ "S", -1 ], [ "T", 0 ] ]
```

Chapter 3

Subgroups of $\mathrm{PSL}_2(\mathbb{Z})$

Analogous to finite-index subgroups of $\mathrm{SL}_2(\mathbb{Z})$, we define a new type `ProjectiveModularSubgroup` for representing subgroups of $\mathrm{PSL}_2(\mathbb{Z})$. It consists essentially of two permutations $\sigma_{\bar{S}}$ and $\sigma_{\bar{T}}$ describing the action of \bar{S} and \bar{T} on the cosets of the given subgroup, where \bar{S} and \bar{T} are the images of S and T in $\mathrm{PSL}_2(\mathbb{Z})$.

The methods implemented for $\mathrm{PSL}_2(\mathbb{Z})$ subgroups are mostly the same as for $\mathrm{SL}_2(\mathbb{Z})$ subgroups and behave more or less identically. Nevertheless we list them here.

3.1 Construction of projective modular subgroups

3.1.1 `ProjectiveModularSubgroup`

▷ `ProjectiveModularSubgroupViaRightAction(s, t)` (operation)

Returns: A projective modular subgroup.

Constructs a `ProjectiveModularSubgroup` object corresponding to the finite-index subgroup of $\mathrm{PSL}_2(\mathbb{Z})$ described by the permutations s and t .

This constructor tests if the given permutations actually describe the coset action of \bar{S} and \bar{T} on some subgroup by checking that they act transitively and satisfy the relations

$$s^2 = (st)^3 = 1$$

Upon creation, the cosets are renamed in a [standardized way](#) to make the internal interaction with existing `GAP` methods easier.

Example

```
gap> G := ProjectiveModularSubgroupViaRightAction(  
> (1,2)(3,4)(5,6)(7,8)(9,10),  
> (1,4)(2,5,9,10,8)(3,7,6));  
<projective modular subgroup of index 10>
```

If you want to construct a `ProjectiveModularSubgroup` from a list of generators, you can lift each generator to a matrix in $\mathrm{SL}_2(\mathbb{Z})$, construct from these a `ModularSubgroup`, and then project it to $\mathrm{PSL}_2(\mathbb{Z})$ via `Projectivization`.

▷ `ProjectiveModularSubgroupViaLeftAction(s, t)` (operation)

Returns: A projective modular subgroup.

Analogous to `ProjectiveModularSubgroupViaRightAction`, but with left coset actions.

3.2 Getters for the coset action

3.2.1 SAction

- ▷ `SRightAction(G)` (operation)
Returns: A permutation.
Returns the permutation $\sigma_{\bar{S}}$ describing the action of \bar{S} on the right cosets of G .
- ▷ `SLeftAction(G)` (operation)
Returns: A permutation.
Analogous to `SRightAction`, but with left coset actions.

3.2.2 TAction

- ▷ `TRightAction(G)` (operation)
Returns: A permutation.
Returns the permutation $\sigma_{\bar{T}}$ describing the action of \bar{T} on the right cosets of G .
- ▷ `TLeftAction(G)` (operation)
Returns: A permutation.
Analogous to `TRightAction`, but with left coset actions.

3.3 Computing with projective modular subgroups

3.3.1 Index (IndexPSL2Z)

- ▷ `Index(G)` (attribute)
Returns: A natural number.
For a given projective modular subgroup G this method returns its index in $\mathrm{PSL}_2(\mathbb{Z})$. As G is internally stored as permutations (s, t) this is just

$$\mathrm{LargestMovedPoint}(s, t)$$

(or 1 if the permutations are trivial).

Example

```
gap> G := ProjectiveModularSubgroup((1,2),(2,3));
<projective modular subgroup of index 3>
gap> Index(G);
3
```

3.3.2 GeneralizedLevel (GeneralizedLevelPSL2Z)

- ▷ `GeneralizedLevel(G)` (attribute)
Returns: A natural number.
This method calculates the general Wohlfahrt level (i.e. the lowest common multiple of all cusp widths) of G as defined in [Woh64].

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
```

```

> );
<projective modular subgroup of index 24>
gap> GeneralizedLevel(G);
12

```

3.3.3 CosetRepresentatives

▷ `RightCosetRepresentatives(G)` (attribute)

Returns: A list of words.

This function returns a list of representatives of the (right) cosets of G as words in \bar{S} and \bar{T} .

Example

```

gap> G := ProjectiveModularSubgroup((1,2),(2,3));
<projective modular subgroup of index 3>
gap> RightCosetRepresentatives(G);
[ <identity ...>, S, S*T ]

```

▷ `LeftCosetRepresentatives(G)` (attribute)

Returns: A list of words.

Analogous to `LeftCosetRepresentatives`, but with left coset actions.

3.3.4 GeneratorsOfGroup (GeneratorsOfGroupPSL2Z)

▷ `GeneratorsOfGroup(G)` (attribute)

Returns: A list of words.

Calculates a list of generators (as words in \bar{S} and \bar{T}) of G . This list might include redundant generators.

Example

```

gap> G := ProjectiveModularSubgroup((1,2)(3,5)(4,6), (1,3)(2,4)(5,6));
<projective modular subgroup of index 6>
gap> GeneratorsOfGroup(G);
[ T^-2, S*T^-2*S^-1 ]

```

3.3.5 IsCongruence (IsCongruencePSL2Z)

▷ `IsCongruence(G)` (attribute)

Returns: True or false.

This method test whether a given modular subgroup G is a congruence subgroup. It is essentially an implementation of an algorithm described in [Hsu96].

Example

```

gap> G := ProjectiveModularSubgroup(
> (1,2)(3,5)(4,6),
> (1,3)(2,4)(5,6)
> );
<projective modular subgroup of index 6>
gap> IsCongruence(G);
true

```

3.3.6 Cusps (CuspsPSL2Z)

▷ Cusps(G) (attribute)

Returns: A list of rational numbers and infinity.

This method computes a list of inequivalent cusp representatives with respect to G .

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
> );
<projective modular subgroup of index 24>
gap> Cusps(G);
[ infinity, 0, 1, 2, 3/2, 5/3 ]
```

3.3.7 CuspWidth (CuspWidthPSL2Z)

▷ CuspWidth(c, G) (operation)

Returns: A natural number.

This method takes as input a cusp c (a rational number or infinity) and a modular group G and calculates the width of this cusp with respect to G .

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,7)(4,8)(5,9)(6,10)(11,12),
> (1,3,4)(2,5,6)(7,10,11)(8,12,9)
> );
<projective modular subgroup of index 12>
gap> CuspWidth(-1, G);
3
gap> CuspWidth(infinity, G);
3
```

3.3.8 CuspsEquivalent (CuspsEquivalentPSL2Z)

▷ CuspsEquivalent(p, q, G) (operation)

Returns: True or false.

Takes two cusps p and q and a projective modular subgroup G and checks if they are equivalent modulo G , i.e. if there exists $A \in G$ with $Ap = q$.

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,7)(4,8)(5,9)(6,10)(11,12),
> (1,3,4)(2,5,6)(7,10,11)(8,12,9)
> );
<projective modular subgroup of index 12>
gap> CuspsEquivalent(infinity, 1, G);
false
gap> CuspsEquivalent(-1, 1/2, G);
true
```

3.3.9 CosetRepresentativeOfCusp (CosetRepresentativeOfCuspPSL2Z)

▷ `CosetRepresentativeOfCusp(c, G)` (operation)

Returns: A word in S and T .

For a cusp c this function returns a right coset representative A of G such that $A\infty$ and c are equivalent with respect to G .

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,7)(4,8)(5,9)(6,10)(11,12),
> (1,3,4)(2,5,6)(7,10,11)(8,12,9)
> );
<projective modular subgroup of index 12>
gap> CosetRepresentativeOfCusp(4, G);
T*S
```

3.3.10 LiftToSL2ZEven

▷ `LiftToSL2ZEven(G)` (operation)

Returns: A modular subgroup.

Lifts a given subgroup G of $\mathrm{PSL}_2(\mathbb{Z})$ to an even subgroup of $\mathrm{SL}_2(\mathbb{Z})$, i.e. a group that contains -1 and whose projection to $\mathrm{PSL}_2(\mathbb{Z})$ is G .

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,7)(4,8)(5,9)(6,10)(11,12),
> (1,3,4)(2,5,6)(7,10,11)(8,12,9)
> );
<projective modular subgroup of index 12>
gap> LiftToSL2ZEven(G);
<modular subgroup of index 12>
```

3.3.11 LiftToSL2ZOdd

▷ `LiftToSL2ZOdd(G)` (operation)

Returns: A modular subgroup.

Lifts a given subgroup G of $\mathrm{PSL}_2(\mathbb{Z})$ to an odd subgroup of $\mathrm{SL}_2(\mathbb{Z})$, i.e. a group that does not contain -1 and whose projection to $\mathrm{PSL}_2(\mathbb{Z})$ is G .

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,7)(4,8)(5,9)(6,10)(11,12),
> (1,3,4)(2,5,6)(7,10,11)(8,12,9)
> );
<projective modular subgroup of index 12>
gap> LiftToSL2ZOdd(G);
<modular subgroup of index 24>
```

3.3.12 IndexModN (IndexModNPSL2Z)

▷ `IndexModN(G, N)` (operation)

Returns: A natural number.

For a projective modular subgroup G and a natural number N this method calculates the index of the projection \bar{G} of G in $\mathrm{PSL}_2(\mathbb{Z}/N\mathbb{Z})$.

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
> );
<projective modular subgroup of index 24>
gap> IndexModN(G, 2);
6
```

3.3.13 Deficiency (DeficiencyPSL2Z)

▷ Deficiency(G, N) (operation)

Returns: A natural number.

For a projective modular subgroup G and a natural number N this method calculates the so-called *deficiency* of G from being a congruence subgroup of level N .

The deficiency of a finite-index subgroup Γ of $\mathrm{PSL}_2(\mathbb{Z})$ was introduced in [WS15]. It is defined as the index $[\Gamma(N) : \Gamma(N) \cap \Gamma]$ where $\Gamma(N)$ is the principal congruence subgroup of level N .

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
> );
<projective modular subgroup of index 24>
gap> Deficiency(G, 4);
4
```

3.3.14 Deficiency (DeficiencyPSL2ZAttr)

▷ Deficiency(G) (attribute)

Returns: A natural number.

Shorthand for Deficiency($G, \text{GeneralizedLevel}(G)$).

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
> );
<projective modular subgroup of index 24>
gap> Deficiency(G);
4
gap> Deficiency(G, GeneralizedLevel(G));
4
```

3.3.15 ConjugateGroup (ConjugateGroupPSL2Z)

▷ ConjugateGroup(G, A) (operation)

Returns: A ProjectiveModularSubgroup.

For an $\mathrm{SL}_2(\mathbb{Z})$ matrix A , conjugates the group G by (the residue class in $\mathrm{PSL}_2(\mathbb{Z})$ of) A and returns the group $A^{-1} * G * A$.

3.3.16 NormalCore (NormalCorePSL2Z)

▷ NormalCore(G) (attribute)

Returns: A projective modular subgroup.

Calculates the normal core of G in $\mathrm{PSL}_2(\mathbb{Z})$, i.e. the maximal subgroup of G that is normal in $\mathrm{PSL}_2(\mathbb{Z})$.

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
> );
<projective modular subgroup of index 24>
gap> NormalCore(G);
<projective modular subgroup of index 3456>
```

3.3.17 QuotientByNormalCore (QuotientByNormalCorePSL2Z)

▷ QuotientByNormalCore(G) (attribute)

Returns: A finite group.

Calculates the quotient of $\mathrm{PSL}_2(\mathbb{Z})$ by the normal core of G .

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
> );
<projective modular subgroup of index 24>
gap> QuotientByNormalCore(G);
<permutation group with 2 generators>
```

3.3.18 AssociatedCharacterTable (AssociatedCharacterTablePSL2Z)

▷ AssociatedCharacterTable(G) (attribute)

Returns: A character table.

Returns the character table of $\mathrm{PSL}_2(\mathbb{Z})/N$ where N is the normal core of G .

Example

```
gap> G := ProjectiveModularSubgroup(
> (1,2)(3,6)(4,8)(5,9)(7,11)(10,13)(12,15)(14,17)(16,19)(18,21)(20,23)(22,24),
> (1,3,7,4)(2,5)(6,9,8,12,14,10)(11,13,16,20,18,15)(17,21,22,19)(23,24)
> );
<projective modular subgroup of index 24>
gap> AssociatedCharacterTable(G);
CharacterTable( <permutation group of size 3456 with 2 generators> )
```

3.3.19 IsMatrixElementOf (IsElementOfPSL2Z)

▷ IsMatrixElementOf(A , G) (operation)

Returns: True or false.

This function checks if the image of a given matrix A in $\mathrm{PSL}_2(\mathbb{Z})$ is an element of the group G .

Example

```
gap> G := ProjectiveModularSubgroup((1,2),(2,3));
<projective modular subgroup of index 3>
gap> IsMatrixElementOf([[1,1],[0,1]], G);
true
gap> IsMatrixElementOf([[0,-1],[1,0]], G);
false
```

3.3.20 IsWordElementOf (IsWordElementOfPSL2ZString)

▷ IsWordElementOf(s , G) (operation)

Returns: True or false.

This function checks if a given string s represents an element of the group G , written in the generators S and T .

Example

```
gap> G := ProjectiveModularSubgroup((1,2),(2,3));
<projective modular subgroup of index 3>
gap> IsWordElementOf("S^2 * T", G);
true
```

3.3.21 IsElementOf (IsWordElementOfPSL2Z)

▷ IsElementOf(w , G) (operation)

Returns: True or false.

This function checks if a given word w in the generators S and T represents an element of the group G .

Example

```
gap> G := ProjectiveModularSubgroup((1,2),(2,3));
<projective modular subgroup of index 3>
gap> F := FreeGroup("S", "T");
<free group on the generators [ S, T ]>
gap> S := F.1;
S
gap> T := F.2;
T
gap> PSL2Z := F / ParseRelators([S, T], "S^2, (S*T)^3");
<fp group on the generators [ S, T ]>
gap> S := GeneratorsOfGroup(PSL2Z)[1];
S
gap> T := GeneratorsOfGroup(PSL2Z)[2];
T
gap> IsElementOf(S^2 * T, G);
true
```

3.3.22 Genus (GenusPSL2Z)

▷ Genus(G) (attribute)

Returns: A non-negative integer.

Computes the genus of the quotient $G \backslash \mathbb{H}$ via an algorithm described in [Sch04].

Example

```
gap> G := ProjectiveModularSubgroup((1,2),(2,3));  
<projective modular subgroup of index 3>  
gap> Genus(G);  
0
```

References

- [DJKV18] Ann Dooms, Eric Jespers, Alexander Konovalov, and Helena Verrill. Congruence, congruence subgroups of $SL(2, \mathbb{Z})$, Version 1.2.2, Feb 2018. GAP package. [4](#)
- [HL14] Thomas Hamilton and David Loeffler. Congruence testing for odd subgroups of the modular group. *LMS Journal of Computation and Mathematics*, 17(1):206–208, 2014. [4](#), [10](#)
- [Hsu96] Tim Hsu. Identifying congruence subgroups of the modular group. *Proceedings of the American Mathematical Society*, 124(5):1351–1359, April 1996. [4](#), [20](#)
- [Sch04] Gabriela Schmithuesen. An algorithm for finding the Veech group of an origami. *Experimental Mathematics*, 13(4):459–472, 2004. [15](#), [25](#)
- [Woh64] Klaus Wohlfahrt. An extension of F. Klein’s level concept. *Illinois J. Math.*, 8(3):529–535, 09 1964. [8](#), [19](#)
- [WS15] Gabriela Weitze-Schmithuesen. The deficiency of being a congruence group for Veech groups of origamis. *International Mathematics Research Notices*, 2015(6):1613–1637, 2015. [12](#), [23](#)

Index

- AssociatedCharacterTable
 - AssociatedCharacterTablePSL2Z, 24
 - AssociatedCharacterTableSL2Z, 13
- ConjugateGroup
 - ConjugateGroupPSL2Z, 23
 - ConjugateGroupSL2Z, 13
- CosetLeftActionOf, 8
- CosetRepresentativeOfCusp
 - CosetRepresentativeOfCuspPSL2Z, 22
 - CosetRepresentativeOfCuspSL2Z, 11
- CosetRightActionOf, 8
- Cusps
 - CuspsPSL2Z, 21
 - CuspsSL2Z, 10
- CuspsEquivalent
 - CuspsEquivalentPSL2Z, 21
 - CuspsEquivalentSL2Z, 11
- CuspWidth
 - CuspWidthPSL2Z, 21
 - CuspWidthSL2Z, 10
- Deficiency
 - DeficiencyPSL2Z, 23
 - DeficiencyPSL2ZAttr, 23
 - DeficiencySL2Z, 11
 - DeficiencySL2ZAttr, 12
- DefinesCosetActionRT, 15
- DefinesCosetActionSJ, 16
- DefinesCosetActionST, 15
- GeneralizedLevel
 - GeneralizedLevelPSL2Z, 19
 - GeneralizedLevelSL2Z, 8
- GeneratorsOfGroup, 9
 - GeneratorsOfGroupPSL2Z, 20
- Genus
 - GenusPSL2Z, 25
 - GenusSL2Z, 15
- Index
 - IndexPSL2Z, 19
 - IndexSL2Z, 8
- IndexModN
 - IndexModNPSL2Z, 22
 - IndexModNSL2Z, 11
- IsCongruence
 - IsCongruencePSL2Z, 20
 - IsCongruenceSL2Z, 10
- IsElementOf
 - IsElementOfSL2Z, 14
 - IsWordElementOfPSL2Z, 25
- IsMatrixElementOf
 - IsElementOfPSL2Z, 24
- IsWordElementOf
 - IsWordElementOfPSL2ZString, 25
 - IsWordElementOfSL2Z, 14
 - IsWordElementOfSL2ZString, 14
- JLeftAction, 8
- JRightAction, 8
- LeftCosetActionFromGenerators, 16
- LeftCosetRepresentatives
 - LeftCosetRepresentativesPSL2Z, 20
 - LeftCosetRepresentativesSL2Z, 9
- LiftToSL2ZEven, 22
- LiftToSL2ZOdd, 22
- ModularSubgroup
 - ModularSubgroupGens, 7
- ModularSubgroupRTViaLeftAction, 6
- ModularSubgroupRTViaRightAction, 6
- ModularSubgroupSJViaRightAction, 6
- ModularSubgroupSTViaLeftAction, 6
- ModularSubgroupSTViaRightAction, 6
- ModularSubgroupViaLeftAction, 6
- ModularSubgroupViaRightAction, 5
- ModularSubgroupSJViaLeftAction, 7

- NormalCore
 - NormalCorePSL2Z, 24
 - NormalCoreSL2Z, 13
- ProjectiveModularSubgroupViaLeftAction, 18
- ProjectiveModularSubgroupViaRightAction, 18
- Projectivization, 12
- QuotientByNormalCore
 - QuotientByNormalCorePSL2Z, 24
 - QuotientByNormalCoreSL2Z, 13
- RightCosetActionFromGenerators, 16
- RightCosetRepresentatives
 - RightCosetRepresentativesPSL2Z, 20
 - RightCosetRepresentativesSL2Z, 9
- RLeftAction, 8
- RRightAction, 8
- RTDecomposition, 17
- SJDecomposition, 17
- SLeftAction, 7
 - SLeftActionPSL2Z, 19
- SRightAction, 7
 - SRightActionPSL2Z, 19
- STDecomposition, 16
- STDecompositionAsList, 17
- TLeftAction, 7
 - TLeftActionPSL2Z, 19
- TRightAction, 7
 - TRightActionPSL2Z, 19
- WordGeneratorsOfGroup
 - GeneratorsOfGroupSL2Z, 9