Congruence

Congruence subgroups of $SL_2(\mathbb{Z})$

Version 1.1.1

28 October 2014

Ann Dooms
Eric Jespers
Alexander Konovalov
Helena Verrill

Ann Dooms Email: andooms@vub.ac.be
Homepage: http://homepages.vub.ac.be/~andooms
Address: Department of Mathematics, Vrije Universiteit Brussel
Pleinlaan 2, Brussels, B-1050 Belgium

Eric Jespers Email: efjesper@vub.ac.be
Homepage: http://homepages.vub.ac.be/~efjesper
Address: Department of Mathematics, Vrije Universiteit Brussel
Pleinlaan 2, Brussels, B-1050 Belgium

Alexander Konovalov Email: alexk@mcs.st-andrews.ac.uk
Homepage: http://www.cs.st-andrews.ac.uk/~alexk/
Address: School of Computer Science
University of St Andrews
Jack Cole Building, North Haugh,
St Andrews, Fife, KY16 9SX, Scotland

Helena Verrill Email: verrill@math.lsu.edu
Homepage: http://www.math.lsu.edu/~verrill/
Address: Department of Mathematics
Louisiana State University
Baton Rouge, Louisiana, 70803-4918
USA

Abstract

The GAP package Congruence provides functionality to work with congruence subgroups of $SL_2(\mathbb{Z})$.

Copyright

© 2006-2014 by Ann Dooms, Eric Jespers, Alexander Konovalov and Helena Verrill.

Congruence is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. For details, see the FSF's own site http://www.gnu.org/licenses/gpl.html.

If you obtained Congruence, we would be grateful for a short notification sent to one of the authors.

If you publish a result which was partially obtained with the usage of Congruence, please cite it in the following form:

A. Dooms, E. Jespers, A. Konovalov and H. Verrill. *Congruence — Congruence subgroups of* $SL_2(\mathbb{Z})$*, Version 1.1.1;* 2014 (http://www.cs.st-andrews.ac.uk/~alexk/congruence/).

Acknowledgements

We are very grateful to Mong-Lung Lang, Chong-Hai Lim and Ser Peow Tan for their comments provided while implementing algorithms from [LLT95a] and [LLT95b], and to Francqui Stichting (Belgium) for the support of the third author.

Contents

1 Introduction **4**
- 1.1 General aims of Congruence package 4
- 1.2 Installation and system requirements 4

2 Construction of congruence subgroups **5**
- 2.1 Construction of congruence subgroups 5
- 2.2 Properties of congruence subgroups 8
- 2.3 Attributes of congruence subgroups 9
- 2.4 Operations for congruence subgroups 11

3 Farey symbols and their properties **13**
- 3.1 Construction of Farey symbols 13
- 3.2 Properties of Farey symbols 14

4 Farey symbols for congruence subgroups **16**
- 4.1 Computation of the Farey symbol for a finite index subgroup 16
- 4.2 Computation of generators of a finite index subgroup from its Farey symbol 17
- 4.3 Other properties derived from Farey symbols 19

5 Service functions of the Congruence package **20**
- 5.1 Additional information displayed by Congruence algorithms 20

References **21**

Index **22**

Chapter 1

Introduction

1.1 General aims of Congruence package

The GAP package Congruence provides functions to construct several types of canonical congruence subgroups in $SL_2(\mathbb{Z})$, and also intersections of a finite number of such subgroups.

Furthermore, it implements the algorithm for generating Farey symbols for congruence subgroups and using them to produce a system of independent generators for these subgroups.

Using the package, one can also determine indices of congruence subgroups and their intersections in $SL_2(\mathbb{Z})$ and in other congruence subgroups, generate their random elements and check element memberships. Success of other group theoretical constructions mostly depends on whether they could be expressed in terms of group generators or not.

For the theoretical backround, we refer to [LLT95b], [LLT95a], [CLLT93] and [Kul91].

1.2 Installation and system requirements

Congruence is distributed in standard formats (`tar.gz`, `tar.bz2`, `-win.zip`) and can be obtained from `http://www.cs.st-andrews.ac.uk/~alexk/congruence/`.

Congruence does not use external binaries and, therefore, works without restrictions on the operating system. It requires at least version GAP 4.5, and no compatibility with previous releases of GAP 4 is guaranteed.

Installation of the package is standard and follows the guidelines from the GAP manual (see (**Reference: Installing a GAP Package**). After the package is installed, you can start GAP and load the Congruence package using the command:

Example

```
gap> LoadPackage("congruence");
```

Chapter 2

Construction of congruence subgroups

The package Congruence provides functions to construct several types of canonical congruence subgroups in $SL_2(\mathbb{Z})$, and also intersections of a finite number of such subgroups. They will return a matrix group in the category `IsCongruenceSubgroup`, which is defined as a subcategory of `IsMatrixGroup`, and which will have a distinguishing property determining whether it is a congruence subgroup of one of the canonical types, or an intersection of such congruence subgroups (if it can not be reduced to one of the canonical congruence subgroups). To start to work with the package, you need first to load it as follows:

Example

```
gap> LoadPackage("congruence");
-------------------------------------------------------------------------------
Loading  Congruence 1.1.0 (Congruence subgroups of SL(2,Integers))
by Ann Dooms (http://homepages.vub.ac.be/~andooms),
   Eric Jespers (http://homepages.vub.ac.be/~efjesper),
   Alexander Konovalov (http://www.cs.st-andrews.ac.uk/~alexk/), and
   Helena Verrill (http://www.math.lsu.edu/~verrill).
-------------------------------------------------------------------------------
true
```

2.1 Construction of congruence subgroups

2.1.1 PrincipalCongruenceSubgroup

▷ `PrincipalCongruenceSubgroup(N)` (operation)

Returns the principal congruence subgroup $\Gamma(N)$ of level `N` in $SL_2(\mathbb{Z})$.
This subgroup consists of all matrices of the form

$$\begin{pmatrix} 1+Na & Nb \\ Nc & 1+Nd \end{pmatrix}$$

where a,b,c,d are integers. The returned group will have the property `IsPrincipalCongruenceSubgroup` (2.2.1).

Example

```
gap> G_8:=PrincipalCongruenceSubgroup(8);
<principal congruence subgroup of level 8 in SL_2(Z)>
gap> IsGroup(G_8);
true
gap> IsMatrixGroup(G_8);
true
gap> DimensionOfMatrixGroup(G_8);
2
gap> MultiplicativeNeutralElement(G_8);
[ [ 1, 0 ], [ 0, 1 ] ]
gap> One(G);
[ [ 1, 0 ], [ 0, 1 ] ]
gap> [[1,2],[3,4]] in G_8;
false
gap> [[1,8],[8,65]] in G_8;
true
gap> SL_2:=SL(2,Integers);
SL(2,Integers)
gap> IsSubgroup(SL_2,G_8);
true
```

2.1.2 CongruenceSubgroupGamma0

▷ `CongruenceSubgroupGamma0(N)` (operation)

Returns the congruence subgroup $\Gamma_0(N)$ of level N in $SL_2(\mathbb{Z})$.
This subgroup consists of all matrices of the form

$$\begin{pmatrix} a & b \\ Nc & d \end{pmatrix}$$

where a,b,c,d are integers. The returned group will have the property `IsCongruenceSubgroupGamma0` (2.2.2).

Example

```
gap> G0_4:=CongruenceSubgroupGamma0(4);
<congruence subgroup CongruenceSubgroupGamma_0(4) in SL_2(Z)>
```

2.1.3 CongruenceSubgroupGammaUpper0

▷ `CongruenceSubgroupGammaUpper0(N)` (operation)

Returns the congruence subgroup $\Gamma^0(N)$ of level N in $SL_2(\mathbb{Z})$.
This subgroup consists of all matrices of the form

$$\begin{pmatrix} a & Nb \\ c & d \end{pmatrix}$$

where a,b,c,d are integers. The returned group will have the property IsCongruenceSubgroupGammaUpper0 (2.2.3).

Example

```
gap> GU0_2:=CongruenceSubgroupGammaUpper0(2);
<congruence subgroup CongruenceSubgroupGamma^0(2) in SL_2(Z)>
```

2.1.4 CongruenceSubgroupGamma1

▷ CongruenceSubgroupGamma1(*N*) (operation)

Returns the congruence subgroup $\Gamma_1(N)$ of level *N* in $SL_2(\mathbb{Z})$.
This subgroup consists of all matrices of the form

$$\begin{pmatrix} 1+Na & b \\ Nc & 1+Nd \end{pmatrix}$$

where a,b,c,d are integers. The returned group will have the property IsCongruenceSubgroupGamma1 (2.2.4).

Example

```
gap> G1_6:=CongruenceSubgroupGamma1(6);
<congruence subgroup CongruenceSubgroupGamma_1(6) in SL_2(Z)>
```

2.1.5 CongruenceSubgroupGammaUpper1

▷ CongruenceSubgroupGammaUpper1(*N*) (operation)

Returns the congruence subgroup $\Gamma^1(N)$ of level *N* in $SL_2(\mathbb{Z})$.
This subgroup consists of all matrices of the form

$$\begin{pmatrix} 1+Na & Nb \\ c & 1+Nd \end{pmatrix}$$

where a,b,c,d are integers. The returned group will have the property IsCongruenceSubgroupGammaUpper1 (2.2.5).

Example

```
gap> GU1_4:=CongruenceSubgroupGammaUpper1(4);
<congruence subgroup CongruenceSubgroupGamma^1(4) in SL_2(Z)>
```

2.1.6 IntersectionOfCongruenceSubgroups

▷ IntersectionOfCongruenceSubgroups(*G1, G2, ..., GN*) (function)
▷ Intersection(*G1, G2, ..., GN*) (function)

Returns the intersection of its arguments, which can be congruence subgroups or their intersections, constructed with the same function. It is not necessary for the user to use `IntersectionOfCongruenceSubgroups`, since it will be called automatically from `Intersection`.

The returned group will have the property `IsIntersectionOfCongruenceSubgroups` (2.2.6).

The list of congruence subgroups that form the intersection can be obtained using `DefiningCongruenceSubgroups` (2.3.3). Note, that when the intersection appears to be one of the canonical congruence subgroups, the package will recognize this and will return a canonical subgroup of the appropriate type.

Example

```
gap> I:=IntersectionOfCongruenceSubgroups(G0_4,GU1_4);
<principal congruence subgroup of level 4 in SL_2(Z)>
gap> J:=IntersectionOfCongruenceSubgroups(G0_4,G1_6);
<intersection of congruence subgroups of resulting level 12 in SL_2(Z)>
```

2.2 Properties of congruence subgroups

A congruence subgroup constructed by one of the five above listed functions will have certain properties determining its type. These properties will be used for method selection by Congruence algorithms. Note that they do not provide an actual test whether a certain matrix group is a congruence subgroup or not.

2.2.1 IsPrincipalCongruenceSubgroup

▷ `IsPrincipalCongruenceSubgroup(`*G*`)` (property)

For a congruence subgroup *G* in the category `IsCongruenceSubgroup`, returns `true` if *G* was constructed by `PrincipalCongruenceSubgroup` (2.1.1) (or reduced to one as a result of an intersection) and returns `false` otherwise.

Example

```
gap> IsPrincipalCongruenceSubgroup(G_8);
true
gap> IsPrincipalCongruenceSubgroup(G0_4);
false
gap> IsPrincipalCongruenceSubgroup(I);
true
```

2.2.2 IsCongruenceSubgroupGamma0

▷ `IsCongruenceSubgroupGamma0(`*G*`)` (property)

For a congruence subgroup *G* in the category `IsCongruenceSubgroup`, returns `true` if *G* was constructed by `CongruenceSubgroupGamma0` (2.1.2) (or reduced to one as a result of an intersection) and returns `false` otherwise.

2.2.3 IsCongruenceSubgroupGammaUpper0

▷ `IsCongruenceSubgroupGammaUpper0(G)` (property)

For a congruence subgroup `G` in the category `IsCongruenceSubgroup`, returns `true` if `G` was constructed by `CongruenceSubgroupGammaUpper0` (2.1.3) (or reduced to one as a result of an intersection) and returns `false` otherwise.

2.2.4 IsCongruenceSubgroupGamma1

▷ `IsCongruenceSubgroupGamma1(G)` (property)

For a congruence subgroup `G` in the category `IsCongruenceSubgroup`, returns `true` if `G` was constructed by `CongruenceSubgroupGamma1` (2.1.4) (or reduced to one as a result of an intersection) and returns `false` otherwise.

2.2.5 IsCongruenceSubgroupGammaUpper1

▷ `IsCongruenceSubgroupGammaUpper1(G)` (property)

For a congruence subgroup `G` in the category `IsCongruenceSubgroup`, returns `true` if `G` was constructed by `CongruenceSubgroupGammaUpper1` (2.1.5) (or reduced to one as a result of an intersection) and returns `false` otherwise.

2.2.6 IsIntersectionOfCongruenceSubgroups

▷ `IsIntersectionOfCongruenceSubgroups(G)` (property)

For a congruence subgroup `G` in the category `IsCongruenceSubgroup`, returns `true` if `G` was constructed by `IntersectionOfCongruenceSubgroups` (2.1.6) and without being one of the canonical congruence subgroups, otherwise it returns `false`.

Example

```
gap> IsIntersectionOfCongruenceSubgroups(I);
false
gap> IsIntersectionOfCongruenceSubgroups(J);
true
```

2.3 Attributes of congruence subgroups

The next three attributes store key properties of congruence subgroups.

2.3.1 LevelOfCongruenceSubgroup

▷ `LevelOfCongruenceSubgroup(G)` (attribute)

Stores the level of the congruence subgroup `G`. The (arithmetic) level of a congruence subgroup G is the smallest positive number N such that G contains the principal congruence subgroup of level N.

Example

```
gap> LevelOfCongruenceSubgroup(G_8);
8
gap> LevelOfCongruenceSubgroup(G1_6);
6
gap> LevelOfCongruenceSubgroup(I);
4
gap> LevelOfCongruenceSubgroup(J);
12
```

2.3.2 IndexInSL2Z

▷ `IndexInSL2Z(G)` (attribute)

Stores the index of the congruence subgroup `G` in $SL_2(\mathbb{Z})$.

Example

```
gap> IndexInSL2Z(G_8);
384
gap> G_2:=PrincipalCongruenceSubgroup(2);
<principal congruence subgroup of level 2 in SL_2(Z)>
gap> IndexInSL2Z(G_2);
12
gap> IndexInSL2Z(GU1_4);
12
```

2.3.3 DefiningCongruenceSubgroups

▷ `DefiningCongruenceSubgroups(G)` (attribute)

Returns: list of congruence subgroups

For an intersection of congruence subgroups, returns the list of congruence subgroups forming this intersection. For a canonical congruence subgroup returns a list of length one containing that subgroup.

Example

```
gap> DefiningCongruenceSubgroups(J);
[ <congruence subgroup CongruenceSubgroupGamma_0(4) in SL_2(Z)>,
  <congruence subgroup CongruenceSubgroupGamma_1(6) in SL_2(Z)> ]
gap> P:=PrincipalCongruenceSubgroup(6);
<principal congruence subgroup of level 6 in SL_2(Z)>
gap> Q:=PrincipalCongruenceSubgroup(10);
<principal congruence subgroup of level 10 in SL_2(Z)>
gap> G:=IntersectionOfCongruenceSubgroups(Q,P);
<principal congruence subgroup of level 30 in SL_2(Z)>
```

```
gap> DefiningCongruenceSubgroups(G);
[ <principal congruence subgroup of level 30 in SL_2(Z)> ]
```

2.4 Operations for congruence subgroups

Congruence installs several special methods for operations already available in GAP.

2.4.1 Random (one and two argument versions)

▷ `Random(G)` (operation)
▷ `Random(G, m)` (operation)

For a congruence subgroup `G` in the category `IsCongruenceSubgroup`, returns random element. In the two-argument form, the second parameter will control the absolute value of randomly selected entries of the matrix.

Example

```
gap> Random(G_2) in G_2;
true
gap> Random(G_8,2) in G_8;
true
```

2.4.2 \in

▷ `\in(m, G)` (operation)

It is easy to implement the membership test for congruence subgroups and their intersections.

Example

```
gap> \in([ [ 21, 10 ], [ 2, 1 ] ],G_2);
true
gap> \in([ [ 21, 10 ], [ 2, 1 ] ],G_8);
false
```

2.4.3 CanEasilyCompareCongruenceSubgroups

▷ `CanEasilyCompareCongruenceSubgroups(G, H)` (operation)

For congruence subgroups `G,H` in the category `IsCongruenceSubgroup`, returns `true` if `G` and `H` are of the same type listed in `PrincipalCongruenceSubgroup` (2.1.1) –> `CongruenceSubgroupGammaUpper1` (2.1.5) and have the same `LevelOfCongruenceSubgroup` (2.3.1) or if `G` and `H` are of the type `IntersectionOfCongruenceSubgroups` (2.1.6) and the groups from `DefiningCongruenceSubgroups` (2.3.3) are in one to one correspondence, otherwise it returns `false`.

Example

```
gap> CanEasilyCompareCongruenceSubgroups(G_8,I);
false
```

2.4.4 IsSubset

▷ IsSubset(*G*, *H*) (operation)

Congruence provides methods for IsSubset for congruence subgroups. IsSubset returns true if *H* is a subset of *G*. These methods make it possible to use IsSubgroup operation for congruence subgroups.

Example

```
gap> IsSubset(G_2,G_8);
true
gap> IsSubset(G_8,G_2);
false
gap> f:=[PrincipalCongruenceSubgroup,CongruenceSubgroupGamma1,CongruenceSubgroupGammaUpper1,Cong
gap> g1:=List(f, t -> t(2));;
gap> g2:=List(f, t -> t(4));;
gap> for g in g2 do
> Print( List( g1, x -> IsSubgroup(x,g) ), "\n");
> od;
[ true, true, true, true, true ]
[ false, true, false, true, false ]
[ false, false, true, false, true ]
[ false, false, false, true, false ]
[ false, false, false, false, true ]
```

2.4.5 Index

▷ Index(*G*, *H*) (operation)

If a congruence subgroup *H* is a subgroup of a congruence subgroup *G*, we can easily compute the index of *H* in *G*, since we know the index of both subgroups in $SL_2(\mathbb{Z})$.

Example

```
gap> Index(G_2,G_8);
32
```

Chapter 3

Farey symbols and their properties

A Farey symbol is a compact and useful way to represent a subgroup of finite index in $SL_2(\mathbb{Z})$ from which one can deduce independent generators for this subgroup. It consists of two components, namely a so-called generalised Farey sequence (*gfs*) and an ordered list of labels, giving additional structure to the *gfs*.

A generalised Farey sequence (g.F.S.) is an ordered list of the form $-infinity, x_0, x_1, ..., x_n, infinity$, where

1. the $x_i = a_i/b_i$ are rational numbers in reduced form arranged in increasing order for $i = 0, ..., n$;
2. $x_0, ..., x_n \in Z$, and some $x_i = 0$;
3. we define $x_{-1} = -infinity = -1/0$ and $x_{n+1} = infinity = 1/0$;
4. $a_{i+1}b_i - a_ib_{i+1} = 1$ for $i = -1, ..., n$.

The ordered list of labels of a Farey symbol gives an additional structure to the *gfs*. The labels correspond to each consecutive pair of x_i's and are of the following types:

1. even,
2. odd,
3. a natural number, which occurs in the list of labels exactly twice or not at all.

Note that the actual values of numerical labels are not important; it is the pairing of two intervals that matters.

The package Congruence provides functions to construct Farey symbols by the given generalised Farey sequence and corresponding list of labels. The returned Farey symbol will belong to the category `IsFareySymbol` and will have the representation `IsFareySymbolDefaultRep`.

3.1 Construction of Farey symbols

3.1.1 FareySymbolByData

▷ `FareySymbolByData(`*gfs, labels*`)` (function)

This constructor creates the Farey symbol with the given generalized Farey sequence and list of labels. It also checks conditions from the definition of Farey symbol and returns an error if they are not satisfied. The data used to create the Farey symbol are stored as its attributes `GeneralizedFareySequence` (3.2.1) and `LabelsOfFareySymbol` (3.2.4).

Example

```
gap> fs:=FareySymbolByData([infinity,0,1,2,infinity],[1,2,2,1]);
```

```
[ infinity, 0, 1, 2, infinity ]
[ 1, 2, 2, 1 ]
```

3.1.2 IsValidFareySymbol

▷ IsValidFareySymbol(*fs*) (function)

This function is used in FareySymbolByData (3.1.1) to validate its output.

Example

```
gap> IsValidFareySymbol(fs);
true
```

3.2 Properties of Farey symbols

3.2.1 GeneralizedFareySequence

▷ GeneralizedFareySequence(*fs*) (attribute)

Returns the generalized Farey sequence *gfs* of the Farey symbol.

Example

```
gap> GeneralizedFareySequence(fs);
[ infinity, 0, 1, 2, infinity ]
```

3.2.2 NumeratorOfGFSElement

▷ NumeratorOfGFSElement(*gfs, i*) (function)

Returns: integer

Returns the numerator of the i-th term of the generalised Farey sequence *gfs*: for the 1st infinite entry returns -1, for the last one returns 1, for all other entries returns the usual numerator.

Example

```
gap> List([1..5], i -> NumeratorOfGFSElement(GeneralizedFareySequence(fs),i));
[ -1, 0, 1, 2, 1 ]
```

3.2.3 DenominatorOfGFSElement

▷ DenominatorOfGFSElement(*gfs, i*) (function)

Returns: integer

Returns the denominator of the i-th term of the generalised Farey sequence *gfs*: for both infinite entries returns 0, for the other ones returns the usual denominator.

Example

```
gap> List([1..5], i -> DenominatorOfGFSElement(GeneralizedFareySequence(fs),i));
[ 0, 1, 1, 1, 0 ]
```

3.2.4 LabelsOfFareySymbol

▷ LabelsOfFareySymbol(*fs*) (attribute)

Returns the list of labels of the Farey symbol. This list has "odd", "even" and paired integers as entries.

Example

```
gap> LabelsOfFareySymbol(fs);
[ 1, 2, 2, 1 ]
```

Chapter 4

Farey symbols for congruence subgroups

The package Congruence provides functions to construct Farey symbols for finite index subgroups. The algorithm used in the package allows to construct a Farey symbol for any finite index subgroup of $SL_2(\mathbb{Z})$ for which it is possible to check whether a given matrix belongs to this subgroup or not.

The development of an algorithm to determine the Farey symbol for a subgroup G of a finite index in $SL_2(\mathbb{Z})$ was started by Ravi Kulkarni in [Kul91] and later it was improved by Mong-Lung Lang, Chong-Hai Lim and Ser-Peow Tan in [LLT95b], [LLT95a].

4.1 Computation of the Farey symbol for a finite index subgroup

4.1.1 FareySymbol

▷ `FareySymbol(G)` (attribute)

For a subgroup of a finite index G, this attribute stores one of the Farey symbols corresponding to the congruence subgroup `G`. The algorithm for its computation will work for any matrix group for which a membership test is available.

Example

```
gap> FareySymbol(PrincipalCongruenceSubgroup(8));
[ infinity, 0, 1/4, 1/3, 3/8, 2/5, 1/2, 3/5, 5/8, 2/3, 3/4, 1, 5/4, 4/3,
  11/8, 7/5, 3/2, 8/5, 13/8, 5/3, 7/4, 2, 9/4, 7/3, 19/8, 12/5, 5/2, 13/5,
  21/8, 8/3, 11/4, 3, 13/4, 10/3, 27/8, 17/5, 7/2, 18/5, 29/8, 11/3, 15/4, 4,
  17/4, 13/3, 9/2, 14/3, 19/4, 5, 21/4, 16/3, 11/2, 17/3, 23/4, 6, 25/4,
  19/3, 13/2, 20/3, 27/4, 7, 29/4, 22/3, 15/2, 23/3, 31/4, 8, infinity ]
[ 1, 17, 10, 26, 32, 18, 19, 27, 30, 5, 2, 2, 13, 28, 26, 20, 21, 29, 27, 7,
  3, 3, 16, 31, 28, 22, 23, 33, 29, 9, 4, 4, 5, 30, 31, 24, 25, 32, 33, 12,
  6, 6, 7, 19, 18, 15, 8, 8, 9, 21, 20, 10, 11, 11, 12, 23, 22, 13, 14, 14,
  15, 25, 24, 16, 17, 1 ]
gap> FareySymbol(CongruenceSubgroupGamma0(20));
[ infinity, 0, 1/5, 1/4, 2/7, 3/10, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 1,
  infinity ]
[ 1, 3, 4, 6, 7, 7, 5, 2, 2, 3, 6, 4, 5, 1 ]
```

4.2 Computation of generators of a finite index subgroup from its Farey symbol

If `fs` is the Farey symbol for a group G with r_1 even labels, r_2 odd labels and r_3 pairs of intervals, then G is generated by $r_1+r_2+r_3$ matrices, which form a set of independent generators for G. These matrices are constructed as follows:

for each even interval $[x_i, x_{i+1}]$, take the matrix

$$A = \begin{pmatrix} a_{i+1}b_{i+1}+a_ib_i & -a_i^2-a_{i+1}^2 \\ b_i^2+b_{i+1}^2 & -a_{i+1}b_{i+1}-a_ib_i \end{pmatrix}$$

for each odd interval $[x_j, x_{j+1}]$, take the matrix

$$B = \begin{pmatrix} a_{j+1}b_{j+1}+a_jb_{j+1}+a_jb_j & -a_j^2-a_ja_{j+1}-a_{j+1}^2 \\ b_j^2+b_jb_{j+1}+b_{j+1}^2 & -a_{j+1}b_{j+1}-a_{j+1}b_j-a_jb_j \end{pmatrix}$$

for each pair of free intervals $[x_k, x_{k+1}]$ and $[x_s, x_{s+1}]$, take the matrix

$$\begin{pmatrix} a_{s+1}b_{k+1}+a_sb_k & -a_sa_k-a_{s+1}a_{k+1} \\ b_sb_k-b_{s+1}b_{k+1} & -a_{k+1}b_{s+1}-a_kb_s \end{pmatrix}$$

4.2.1 MatrixByEvenInterval

▷ `MatrixByEvenInterval(gfs, i)` (function)

Returns the matrix corresponding to the even interval i in the generalized Farey sequence `gfs`.

Example

```
gap> H:=CongruenceSubgroupGamma0(5);
<congruence subgroup CongruenceSubgroupGamma_0(5) in SL_2(Z)>
gap> fs:=FareySymbol(H);
[ infinity, 0, 1/2, 1, infinity ]
[ 1, "even", "even", 1 ]
gap> gfs:=GeneralizedFareySequence(fs);
[ infinity, 0, 1/2, 1, infinity ]
gap> MatrixByEvenInterval(gfs,2);
[ [ 2, -1 ], [ 5, -2 ] ]
```

4.2.2 MatrixByOddInterval

▷ `MatrixByOddInterval(gfs, i)` (function)

Returns the matrix corresponding to the odd interval i in the generalized Farey sequence `gfs`.

Example

```
gap> fs_oo:=FareySymbolByData([infinity,0,infinity],["odd","odd"]);;
gap> gfs_oo:=GeneralizedFareySequence(fs_oo);
[ infinity, 0, infinity ]
gap> MatrixByOddInterval(gfs_oo,1);
[ [ -1, -1 ], [ 1, 0 ] ]
```

4.2.3 MatrixByFreePairOfIntervals

▷ `MatrixByFreePairOfIntervals(gfs, k, kp)` (function)

Returns the matrix corresponding to the pair of free intervals k and kp in the generalized Farey sequence *gfs*.

Example

```
gap> fs_free:=FareySymbolByData([infinity,0,1,2,infinity],[1,2,2,1]);;
gap> gfs_free:=GeneralizedFareySequence(fs_free);;
gap> MatrixByFreePairOfIntervals(gfs_free,2,3);
[ [ 3, -2 ], [ 2, -1 ] ]
```

4.2.4 GeneratorsByFareySymbol

▷ `GeneratorsByFareySymbol(fs)` (function)

Returns a set of matrices constructed as above.

Example

```
gap> fs_eo:=FareySymbolByData([infinity,0,infinity],["even","odd"]);;
gap> GeneratorsByFareySymbol(last);
[ [ [ 0, -1 ], [ 1, 0 ] ], [ [ 0, -1 ], [ 1, -1 ] ] ]
gap> GeneratorsByFareySymbol(fs);
[ [ [ 1, 1 ], [ 0, 1 ] ], [ [ 2, -1 ], [ 5, -2 ] ], [ [ 3, -2 ], [ 5, -3 ] ] ]
gap> GeneratorsByFareySymbol(fs_oo);
[ [ [ -1, -1 ], [ 1, 0 ] ], [ [ 0, -1 ], [ 1, -1 ] ] ]
gap> GeneratorsByFareySymbol(fs_free);
[ [ [ 1, 2 ], [ 0, 1 ] ], [ [ 3, -2 ], [ 2, -1 ] ] ]
```

4.2.5 GeneratorsOfGroup

▷ `GeneratorsOfGroup(G)` (function)

Returns a set of generators for the finite index group G in $SL_2(Z)$.

Example

```
gap> G:=PrincipalCongruenceSubgroup(2);
<principal congruence subgroup of level 2 in SL_2(Z)>
gap> FareySymbol(G);
[ infinity, 0, 1, 2, infinity ]
[ 2, 1, 1, 2 ]
gap> GeneratorsOfGroup(G);
#I  Using the Congruence package for GeneratorsOfGroup ...
[ [ [ 1, 2 ], [ 0, 1 ] ], [ [ 3, -2 ], [ 2, -1 ] ] ]
gap> H:=CongruenceSubgroupGamma0(5);
<congruence subgroup CongruenceSubgroupGamma_0(5) in SL_2(Z)>
gap> GeneratorsOfGroup(H);
```

```
#I  Using the Congruence package for GeneratorsOfGroup ...
[ [ [ 1, 1 ], [ 0, 1 ] ], [ [ 2, -1 ], [ 5, -2 ] ], [ [ 3, -2 ], [ 5, -3 ] ] ]
gap> I:=IntersectionOfCongruenceSubgroups(PrincipalCongruenceSubgroup(2),CongruenceSubgroupGamma
<intersection of congruence subgroups of resulting level 6 in SL_2(Z)>
gap> FareySymbol(I);
[ infinity, 0, 1/3, 1/2, 2/3, 1, 4/3, 3/2, 5/3, 2, infinity ]
[ 1, 5, 4, 3, 2, 2, 3, 4, 5, 1 ]
gap> GeneratorsOfGroup(I);
#I  Using the Congruence package for GeneratorsOfGroup ...
[ [ [ 1, 2 ], [ 0, 1 ] ], [ [ 11, -2 ], [ 6, -1 ] ],
  [ [ 19, -8 ], [ 12, -5 ] ], [ [ 17, -10 ], [ 12, -7 ] ],
  [ [ 7, -6 ], [ 6, -5 ] ] ]
```

4.3 Other properties derived from Farey symbols

4.3.1 IndexInPSL2ZByFareySymbol

▷ `IndexInPSL2ZByFareySymbol(`*fs*`)` (function)

By Proposition 7.2 in [Kulkarni], for the Farey symbol with underlying generalized Farey sequence [infinity, x0, x1, ..., xn, infinity], the index in $PSL_2(Z)$ is given by the formula d = 3*n + e3, where e3 is the number of odd intervals.

Example

```
gap> IndexInPSL2ZByFareySymbol(fs);
6
gap> IndexInPSL2ZByFareySymbol(fs_oo);
2
gap> IndexInPSL2ZByFareySymbol(fs_free);
6
```

Chapter 5

Service functions of the Congruence package

5.1 Additional information displayed by Congruence algorithms

5.1.1 InfoCongruence

▷ `InfoCongruence` (info class)

`InfoCongruence` is a special Info class for Congruence algorithms. It has 3 levels: 0, 1 (default) and 2. To change the info level to `k`, use the command `SetInfoLevel(InfoCongruence, k)`.

In the example below we use this mechanism to see more details during the Farey symbol construction for a congruence subgroup.

References

[CLLT93] Shih-Ping Chan, Mong-Lung Lang, Chong-Hai Lim, and Ser Peow Tan. Special polygons for subgroups of the modular group and applications. *Internat. J. Math.*, 4(1):11–34, 1993. 4

[Kul91] Ravi S. Kulkarni. An arithmetic-geometric method in the study of the subgroups of the modular group. *Amer. J. Math.*, 113(6):1053–1133, 1991. 4, 16

[LLT95a] Mong-Lung Lang, Chong-Hai Lim, and Ser Peow Tan. An algorithm for determining if a subgroup of the modular group is congruence. *J. London Math. Soc. (2)*, 51(3):491–502, 1995. 2, 4, 16

[LLT95b] Mong-Lung Lang, Chong-Hai Lim, and Ser Peow Tan. Independent generators for congruence subgroups of Hecke groups. *Math. Z.*, 220(4):569–594, 1995. 2, 4, 16

Index

CanEasilyCompareCongruenceSubgroups, 11
Congruence package, 2
CongruenceSubgroupGamma0, 6
CongruenceSubgroupGamma1, 7
CongruenceSubgroupGammaUpper0, 6
CongruenceSubgroupGammaUpper1, 7

DefiningCongruenceSubgroups, 10
DenominatorOfGFSElement, 14

FareySymbol, 16
FareySymbolByData, 13

GeneralizedFareySequence, 14
GeneratorsByFareySymbol, 18
GeneratorsOfGroup, 18

\in, 11
Index, 12
IndexInPSL2ZByFareySymbol, 19
IndexInSL2Z, 10
InfoCongruence, 20
Intersection, 7
IntersectionOfCongruenceSubgroups, 7
IsCongruenceSubgroup, 4, 5
IsCongruenceSubgroupGamma0, 8
IsCongruenceSubgroupGamma1, 9
IsCongruenceSubgroupGammaUpper0, 9
IsCongruenceSubgroupGammaUpper1, 9
IsFareySymbol, 13
IsFareySymbolDefaultRep, 13
IsIntersectionOfCongruenceSubgroups, 9
IsPrincipalCongruenceSubgroup, 8
IsSubset, 12
IsValidFareySymbol, 14

LabelsOfFareySymbol, 15
LevelOfCongruenceSubgroup, 9

MatrixByEvenInterval, 17
MatrixByFreePairOfIntervals, 18
MatrixByOddInterval, 17

NumeratorOfGFSElement, 14

PrincipalCongruenceSubgroup, 5

Random, 11
    one and two argument versions, 11