SGGI Package

Peter A. Brooksbank
Bucknell University
`pbrooksb@bucknell.edu`

Version 0.1
March 2, 2020

Contents

CHAPTER 1

Introduction

This documentation describes MAGMA functions that facilitate computation with *string groups generated by involutions* (SGGIs). Functions that compute with these structures already exist in distributed versions of MAGMA [**BCP**]. The purpose of the current package is to supplement—and in some cases improve upon—the existing machinery.

Citing SGGI. To cite the SGGI package, please use the following:

Peter A. Brooksbank, *SGGI*, version 0.1, GitHub, 2020. `https://github.com/galois60/SGGI`.

For AMSRefs:

```
\bib{SGGI}{misc}{
   author={Brooksbank, Peter A.},
   title={SGGI},
   publisher={GitHub},
   year={2020},
   edition={version 0.1},
   note={\texttt{https://github.com/galois60/SGGI}},
}
```

1.1. Overview

1.2. Version

CHAPTER 2

String groups generated by involutions

This package works with SGGIs as a data type. We begin by describing the essential ingredients. A **string group generated by involutions**, or **SGGI** consists of:

(1) a group H; and
(2) a list $h_1, \ldots, h_m$ of involutions of H.

The properties that make it an SGGI are first that $H = \langle h_1, \ldots, h_n \rangle$, and secondly that the involutions satisfy the **string condition**, namely

$$\forall i, j \in [m] \qquad |i - j| > 1 \implies h_i h_j = h_j h_i \tag{1}$$

The integer m is the **rank** of the SGGI. The package uses a data type `SGGI` for these objects.

2.1. Constructing SGGIs

The package works with the two representations of groups that are currently most relevant to computation with SGGIs, namely permutation groups and matrix groups.

A user can specify a SGGI in several related ways.

```
StringGroupGeneratedByInvolutions(S) : SeqEnum[GrpPermElt] -> SGGI
StringGroupGeneratedByInvolutions(S) : SeqEnum[GrpMatElt] -> SGGI
```

Provided S is a list of involutions satisfying (1) this returns the SGGI with group generated by S.

```
StringGroupGeneratedByInvolutions(G, S) : GrpPerm, SeqEnum[GrpPermElt] -> SGGI
StringGroupGeneratedByInvolutions(G, S) : GrpMat, SeqEnum[GrpMatElt] -> SGGI
```

Similar, but checks that the specified G is in fact generated by S. One can build the SGGI on the defining generators of input group G as follows:

```
StringGroupGeneratedByInvolutions(G) : GrpPerm -> SGGI
StringGroupGeneratedByInvolutions(G) : GrpMat  -> SGGI
```

Here are some basic access functions for SGGIs.

```
Group(H) : SGGI -> Grp
```

gives the underlying group of H, and

```
Generators(H) : SGGI -> SeqEnum
```

returns the sequence $h_1, \ldots, h_n$ of involutions generating that group.

```
Rank(H) : SGGI -> RngIntElt
```

is the rank of H, and

```
SchlafliType(H) : SGGI -> SeqEnum[RngIntElt]
```

return the Schlafli symbol of H, namely the sequence $\mathrm{ord}(h_i h_{i+1})$ for $i = 1, \ldots, n-1$.

```
Print(H) : SGGI
```

displays information about H.

2.1.1. Special constructions.

The symmetric group S_{n+1} with generators $(1\ 2)$, $(2\ 3)$, $\ldots$, $(n\ n{+}1)$ is the most obvious construction of a SGGI of rank n. The corresponding polytope is the simplex:

```
Simplex(n) : RngIntElt -> SGGI
```

Example 2.1. `Simplex`
We build the 5-simplex as an SGGI and access its basic features.

```
> H := Simplex(5);
> Group(H);
Permutation group acting on a set of cardinality 6
    (1, 2)
    (2, 3)
    (3, 4)
    (4, 5)
    (5, 6)
> Rank(H);
5
> SchlafliType(H);
[3, 3, 3, 3]
> Print(H);
SGGI with underlying group type GrpPerm.
```

SGGIs arise naturally as quotients of Coxeter groups.

`ModularReflectionGroup(L, p) : SeqEnum[RngIntElt], RngIntElt -> SGGI`

Given a list $L = [\ell_1, \ldots, \ell_{n-1}]$ of natural numbers such that the reflection group G_0 with Schlafli symbol L is crystallographic, return the reduction of G_0 modulo the specified prime p. (Note, a "0" entry in L is understood as "∞".)

2.1.2. Duals. If H is an SGGI of rank n with distinguished involution sequence $[h_1, \ldots, h_n]$, evidently there is a **dual** SGGI (on the same group) of rank n with involution sequence $[h_n, \ldots, h_1]$.

`Dual(H) : SGGI -> SGGI`

One can also form the **Petrie dual** of an SGGI:

`PetrieDual(H) : SGGI -> SGGI`

returns the SGGI (on the same group) of rank n whose involution sequence has h_3 replaced by h_1h_3. The Petrie construction can also be used to reduce the rank of an SGGI:

`PetrieReduction(H) : SGGI -> SGGI`

returns the SGGI of rank $n-1$ on involution sequence $[h_2, h_1h_3, h_4, \ldots, h_n]$. (Note, this may or may not have the same underlying group as H.)

Example 2.2. `Reflection Group`

We build the SGGI that is the reduction modulo 7 of the crystallographic Coxeter group of type $[3, 3, \infty, 3]$, together with various duals and rank-reduced versions.

```
> H := ModularReflectionGroup([3,3,0,3], 7);
> SchlafliType(H);
[3, 3, 7, 3]
> Generic (Group(H));
GL(5, GF(7))
> D := Dual(H); SchlafliType(D);
[3, 7, 3, 3]
> PD := PetrieDual(H); SchlafliType(PD);
[3, 4, 14, 3]
> PR := PetrieReduction(H); SchlafliType(PR);
[4, 14, 3]
> Group(PR) eq Group(H);
true
```

2.2. Isomorphism Testing

Two SGGIs H and J of rank n, with distinguished involution sequences $[h_1, \ldots, h_n]$ and $[j_1, \ldots, j_n]$ are **isomorphic** if there is an isomorphism of the underlying groups sending one generating sequence to the other. The SGGIs are **equivalent** is either they are isomorphic, or H is isomorphic to the dual of J. These conditions are tested as follows:

```
IsIsomorphic(H,J) : SGGI , SGGI -> BoolElt
IsEquivalent(H,J) : SGGI , SGGI -> BoolElt
```

CHAPTER 3

String C-groups

An SGGI H of rank n with distinguished involution sequence $[h_1, \ldots, h_n]$ satisfies the **intersection property** if

$$\forall I, J \subseteq [n] \qquad \langle h_i \mid i \in I\rangle \cap \langle h_j \mid j \in J\rangle = \langle h_k \mid k \in I \cap J\rangle. \tag{2}$$

An SGGI that satisfies this condition is called a **string C-group**.

```
HasIntersectionProperty(H) : SGGI -> BoolElt
IsStringCGroup(H) : SGGI  -> BoolElt
```

3.1. Search Functions

Given a group G one can try to build all sequences of involutions that generate G as a string C-group of a given rank n. This can be done as follows:

```
AllStringCReps(G,n) : GrpPerm , RngIntElt -> SeqEnum[SGGI]
AllStringCReps(G,n) : GrpMat , RngIntElt -> SeqEnum[SGGI]
```

The user is warned that this can be a time (and memory) consuming computation when G is a large group.

Bibliography

[BCP] Wieb Bosma, John Cannon, and Catherine Playoust, *The Magma algebra system. I. The user language*, J. Symbolic Comput. **3-4** (1997), 235-265.

[BL] Peter A. Brooksbank and Dimitri Leemans, *Rank reduction of string C-group representations*, Proc. Amer. Math. Soc.