The SmallCancellation Package

Metric and nonmetric small cancellation conditions

Version 1.0.3

Iván Sadofschi Costa

Iván Sadofschi Costa Email: isadofschi@dm.uba.ar
Homepage: http://mate.dm.uba.ar/~isadofschi

Copyright

Contents

Chapter 1

Installing and Loading the SmallCancellation Package

1.1 Compiling Binaries of the SmallCancellation Package

After unpacking the archive, go to the newly created `smallcancellation` directory and call `./configure` to use the default `../..` path to the GAP home directory or `./configure` *`path`* where *`path`* is the path to the GAP home directory, if the package is being installed in a non-default location. So for example if you install the package in the `~/.gap/pkg` directory and the GAP home directory is `~/gap4r5` then you have to call

Example
```
./configure ../../../gap4r5/
```

This will fetch the architecture type for which GAP has been compiled last and create a `Makefile`. Now simply call

Example
```
make
```

to compile the binary and to install it in the appropriate place.

If GAP cannot find a working binary, some computations may take more time (specially for large presentations).

1.2 Loading the SmallCancellation Package

To use the SmallCancellation Package you have to request it explicitly. This is done by calling `LoadPackage` (**Reference: LoadPackage**):

Example
```
gap> LoadPackage("smallcancellation");
-----------------------------------------------------------------
Loading  SmallCancellation 1.0.2
by Iván Sadofschi Costa (http://mate.dm.uba.ar/~isadofschi)
For help, type: ?SmallCancellation
-----------------------------------------------------------------
true
```

The SmallCancellation requires the Grape Package (version >=4.7).

If you want to load the SmallCancellation package by default, you can put the `LoadPackage` command into your `gaprc` file (see Section (**Reference: The gap.ini and gaprc files**)).

Chapter 2

Small Cancellation Theory

2.1 Introduction

A standard reference for Small Cancellation Theory is Lyndon-Schupp [LS01, Chapter V]. We review here some definitions and results.

A subset R of a free group F is called *symmetrized* if all elements of R are cyclically reduced and for each r in R all cyclically reduced conjugates of r and r^{-1} are also in R. If the set R is not symmetrized we may work instead with the *symmetrization* R^* of R (the smallest symmetrized set containing R).

A *piece* of R is a word that is a common prefix of two different words in the symmetrized set R^*.

We say that a presentation $P = \langle X \mid R \rangle$ *satisfies the small cancellation condition* $C(p)$ if no relator r in R^* is a product of fewer than p pieces.

We say that P *satisfies the small cancellation condition* $C'(\lambda)$ if for all r in R^*, if $r = bc$ and b is a piece then $|b| < \lambda |r|$.

We say that P *satisfies the small cancellation condition* $T(q)$ if for all h such that $3 \leq h < q$ and for all elements $r_1, \ldots, r_h$ in R^*, if no succesive elements r_i, r_{i+1} is an inverse pair, then at least one of the products $r_1 r_2, \ldots r_{h-1} r_h, r_h r_1$ is reduced without cancellation. Condition $T(q)$ may be rephrased in terms of the Whitehead graph of the presentation P.

If a group $G = \langle X \mid R \rangle$ satisfies $C'(1/6)$ then Dehn's algorithm (see [LS01, Chapter V, Section 4]) solves the word problem for G.

If a group $G = \langle X \mid R \rangle$ satisfies $C(6)$ or $C(4)\check{}T(4)$ or $C(3)\check{}T(6)$ then G has solvable word problem and solvable conjugacy problem (see [LS01, Chapter V, Sections 5 and 6]).

If a presentation $P = \langle X \mid R \rangle$ satisfies $C(6)$ and no relator of P is a proper power then the presentation complex of P is aspherical.

2.2 Pieces

2.2.1 PiecesOfGroup

▷ PiecesOfGroup(*G*) (function)

Returns the set of pieces of the FpGroup *G*.

Example
```
gap> F:=FreeGroup(["a","b"]);;
gap> AssignGeneratorVariables(F);;
```

```
#I  Assigned the global variables [ a, b ]
gap> r:=a*b*a*b^2*a*b^3;;
gap> G:=F/[r];;
gap> PiecesOfGroup(G);
[ a^-1, a, b^-1, b, a^-1*b^-1, a*b, b^-1*a^-1, b^-2, b*a, b^2, a^-1*b^-2,
  a*b^2, b^-1*a^-1*b^-1, b^-2*a^-1, b*a*b, b^2*a, b^-1*a^-1*b^-2,
  b^-2*a^-1*b^-1, b*a*b^2, b^2*a*b ]
```

2.2.2 PiecesOfPresentation

▷ `PiecesOfPresentation(P)` (function)

Returns the set of pieces of the presentation *P*.

2.3 Metric small cancellation conditions

2.3.1 GroupSatisfiesCPrime

▷ `GroupSatisfiesCPrime(G, lambda[, explain])` (function)

Returns `true` if the FpGroup *G* satisfies the small cancellation condition *C'(lambda)*, `false` otherwise. If the optional argument *explain* is `true` instead of returning `false` returns an explanation of this result.

Example

```
gap> F:=FreeGroup(["a1","b1","a2","b2","a3","b3"]);;
gap> AssignGeneratorVariables(F);;
#I  Assigned the global variables [ a1, b1, a2, b2, a3, b3 ]
gap> r:=Comm(a1,b1)*Comm(a2,b2)*Comm(a3,b3);;
gap> G:=F/[r];;
gap> GroupSatisfiesCPrime(G,1/11);
true
gap> GroupSatisfiesCPrime(G,1/12);
false
gap> GroupSatisfiesCPrime(G,1/12,true);
[ false, a1^-1*b1^-1*a1*b1*a2^-1*b2^-1*a2*b2*a3^-1*b3^-1*a3*b3,
  " starts with a piece of length ", 1 ]
```

2.3.2 PresentationSatisfiesCPrime

▷ `PresentationSatisfiesCPrime(P, lambda[, explain])` (function)

Returns `true` if the presentation *P* satisfies the small cancellation condition *C'(lambda)*, `false` otherwise. If the optional argument *explain* is `true` instead of returning `false` returns an explanation of this result.

2.4 Non-metric small cancellation conditions

2.4.1 GroupSatisfiesC

▷ `GroupSatisfiesC(G, p[, explain])` (function)

Returns `true` if the FpGroup `G` satisfies the small cancellation condition `C(p)`, `false` otherwise. If the optional argument *explain* is `true` instead of returning `false` returns an explanation of this result.

Example

```
gap> GroupSatisfiesC(G,12);
true
gap> GroupSatisfiesC(G,13);
false
gap> GroupSatisfiesC(G,13,true);
[ false, a1^-1*b1^-1*a1*b1*a2^-1*b2^-1*a2*b2*a3^-1*b3^-1*a3*b3,
  " is the product of pieces of the following lengths: ",
  [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ] ]
```

2.4.2 PresentationSatisfiesC

▷ `PresentationSatisfiesC(P, p[, explain])` (function)

Returns `true` if the presentation `P` satisfies the small cancellation condition `C(p)`, `false` otherwise. If the optional argument *explain* is `true` instead of returning `false` returns an explanation of this result.

2.4.3 GroupSatisfiesT

▷ `GroupSatisfiesT(G, q)` (function)

Returns `true` if the FpGroup `G` satisfies the small cancellation condition `T(q)`, `false` otherwise.

Example

```
gap> GroupSatisfiesT(G,12);
true
gap> GroupSatisfiesT(G,13);
false
```

2.4.4 PresentationSatisfiesT

▷ `PresentationSatisfiesT(P, q)` (function)

Returns `true` if the presentation `P` satisfies the small cancellation condition `T(q)`, `false` otherwise.

2.5 Additional functions

2.5.1 GraphFromAdjacencyMatrix

▷ `GraphFromAdjacencyMatrix(A)` (function)

If A is the adjacency matrix of a simple and directed graph, returns a Grape graph representing the graph with adjacency matrix A.

2.5.2 WhiteheadGraphAdjacencyMatrix

▷ `WhiteheadGraphAdjacencyMatrix(G)` (function)

Returns the adjacency matrix of the Whitehead graph of the FpGroup G. Note that the Whitehead graph may not be simple.

2.5.3 ExponentMatrixOfGroup

▷ `ExponentMatrixOfGroup(G)` (function)

If G is a group with m relators and n generators, it returns the m by n matrix A such that $A_{i,j}$ is the total exponent of the generator g_j in the relator r_i. Returns the adjacency matrix of the Whitehead graph of the FpGroup G. Note that the Whitehead graph may not be simple.

2.5.4 ExponentMatrixOfPresentation

▷ `ExponentMatrixOfPresentation(P)` (function)

If P is a presentation with m relators and n generators, it returns the m by n matrix A such that $A_{i,j}$ is the total exponent of the generator g_j in the relator r_i. Returns the adjacency matrix of the Whitehead graph of the FpGroup G. Note that the Whitehead graph may not be simple.

2.6 Implementation details

Relators and their prefixes are stored in a data structure called trie [Wik19]. This provides an efficient way to write a word as a product of pieces using the minimum possible number of pieces.

References

[LS01] Roger C. Lyndon and Paul E. Schupp. *Combinatorial group theory*. Classics in Mathematics. Springer-Verlag, Berlin, 2001. Reprint of the 1977 edition. 6

[Wik19] Wikipedia. Trie. *Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Trie*, 2019. 9

Index