Working with Finitely Presented Groups in Maple

Maplesoft

This Tips and Techniques article introduces Maple's facilities for working with finitely presented groups. A finitely presented group is a group defined by means of a finite number of generators, and a finite number of defining relations. It is one of the principal ways in which a group may be represented on the computer, and is virtually the only representation that effectively allows us to compute with many infinite groups.

This document uses a command-based approach, but note that many group theory operations are also available through the context-sensitive menus.

Overview

To begin working with groups in Maple, we first load the GroupTheory package, using the **with** command.

```
> with( GroupTheory );
```

(1.1)

[*<|>*, *AbelianInvariants*, *AllPerfectGroups*, *AllSmallGroups*, *AllTransitiveGroups*, *Alt*, *AlternatingGroup*, *AreConjugate*, *AreIsomorphic*, *BabyMonster*, *CayleyGraph*, *CayleyTable*, *CayleyTableGroup*, *Center*, *Centraliser*, *Centralizer*, *Centre*, *ComplexProduct*, *ConjugacyClass*, *ConjugacyClasses*, *Conjugator*, *ConwayGroup*, *Core*, *CustomGroup*, *CycleIndexPolynomial*, *CyclicGroup*, *Degree*, *DerivedLength*, *DerivedSeries*, *DerivedSubgroup*, *DicyclicGroup*, *DihedralGroup*, *DirectProduct*, *DrawCayleyTable*, *DrawSubgroupLattice*, *ElementOrder*, *ElementaryGroup*, *Elements*, *Embedding*, *ExceptionalGroup*, *Exponent*, *FPGroup*, *Factor*, *FischerGroup*, *FittingSubgroup*, *FrattiniSubgroup*, *FreeGroup*, *GL*, *GaloisGroup*, *GeneralLinearGroup*, *GeneralOrthogonalGroup*, *GeneralUnitaryGroup*, *Generators*, *Group*, *GroupOrder*, *HaradaNortonGroup*, *HeldGroup*, *HigmanSimsGroup*,

Hypercenter, *Hypercentre*, *IdentifySmallGroup*, *Index*, *Intersection*, *IsAbelian*, *IsAlternating*, *IsCommutative*, *IsCyclic*, *IsElementary*, *IsFinite*, *IsNilpotent*, *IsNormal*, *IsPerfect*, *IsPrimitive*, *IsRegular*, *IsSimple*, *IsSoluble*, *IsSolvable*, *IsSubgroup*, *IsSymmetric*, *IsTransitive*, *JankoGroup*, *Labels*, *LeftCoset*, *LeftCosets*, *LowerCentralSeries*, *LyonsGroup*, *MathieuGroup*, *McLaughlinGroup*, *MetacyclicGroup*, *Monster*, *NilpotencyClass*, *NilpotentResidual*, *NonRedundantGenerators*, *NormalClosure*, *Normaliser*, *NormaliserSubgroup*, *NormalizerSubgroup*, *NumGroups*, *NumPerfectGroups*, *NumTransitiveGroups*, *ONanGroup*, *Operations*, *Orbit*, *Orbits*, *OrthogonalGroup*, *PCore*, *PGL*, *PGU*, *PSL*, *PSU*, *PSp*, *PerfectGroup*, *PermApply*, *PermCommutator*, *PermConjugate*, *PermCycleType*, *PermDegree*, *PermFixed*, *PermInverse*, *PermLeftQuotient*, *PermOrder*, *PermParity*, *PermPower*, *PermProduct*, *PermRightQuotient*, *PermSupport*, *PermutationGroup*, *PresentationComplexity*, *ProjectiveGeneralLinearGroup*, *ProjectiveGeneralUnitaryGroup*, *ProjectiveSpecialLinearGroup*, *ProjectiveSpecialUnitaryGroup*, *ProjectiveSymplecticGroup*, *QuaternionGroup*, *RandomElement*, *Relators*, *RightCoset*, *RightCosets*, *RubiksCubeGroup*, *RudvalisGroup*, *SL*, *SearchSmallGroups*, *SearchTransitiveGroups*, *Simplify*, *SmallGroup*, *SolubleResidual*, *SolvableResidual*, *SpecialLinearGroup*, *SpecialOrthogonalGroup*, *SpecialUnitaryGroup*, *Stabiliser*, *Stabilizer*, *Subgroup*, *SubgroupLattice*, *SubgroupMembership*, *Supergroup*, *SuzukiGroup*, *SylowSubgroup*, *Symm*, *SymmetricGroup*, *SymplecticGroup*, *ThompsonGroup*, *TitsGroup*, *TransitiveGroup*, *TrivialGroup*, *UpperCentralSeries*]

This makes all the commands of the package available at the top level, and redefines certain syntaxes so that they are interpreted, in the correct context, as having group-theoretic significance. In particular, this action redefines the angle bracket operator so that it can be interpreted as a constructor for finite presentations, and not solely for creating vectors and matrices.

To create a finitely presented group, we use the usual textbook notation involving angle brackets and a vertical bar (pipe) to separate the sequence of generators from the defining relations. Here is an example of a free abelian group of rank two on generators x and y with a defining relation expressing the fact that these two generators commute with one another. This single defining relation is sufficient to completely specify the group.

```
> A := < x, y | [x,y] = 1 >;
```

$$A := \langle x, y \mid x^{-1} y^{-1} x y \rangle \qquad \textbf{(1.2)}$$

We can ask Maple questions about the group, such as whether the group is abelian.

```
> IsAbelian( A );
```

$$true \tag{1.3}$$

The GroupOrder command is able to recognize that this group is infinite.

```
> GroupOrder( A );
```

$$\infty \tag{1.4}$$

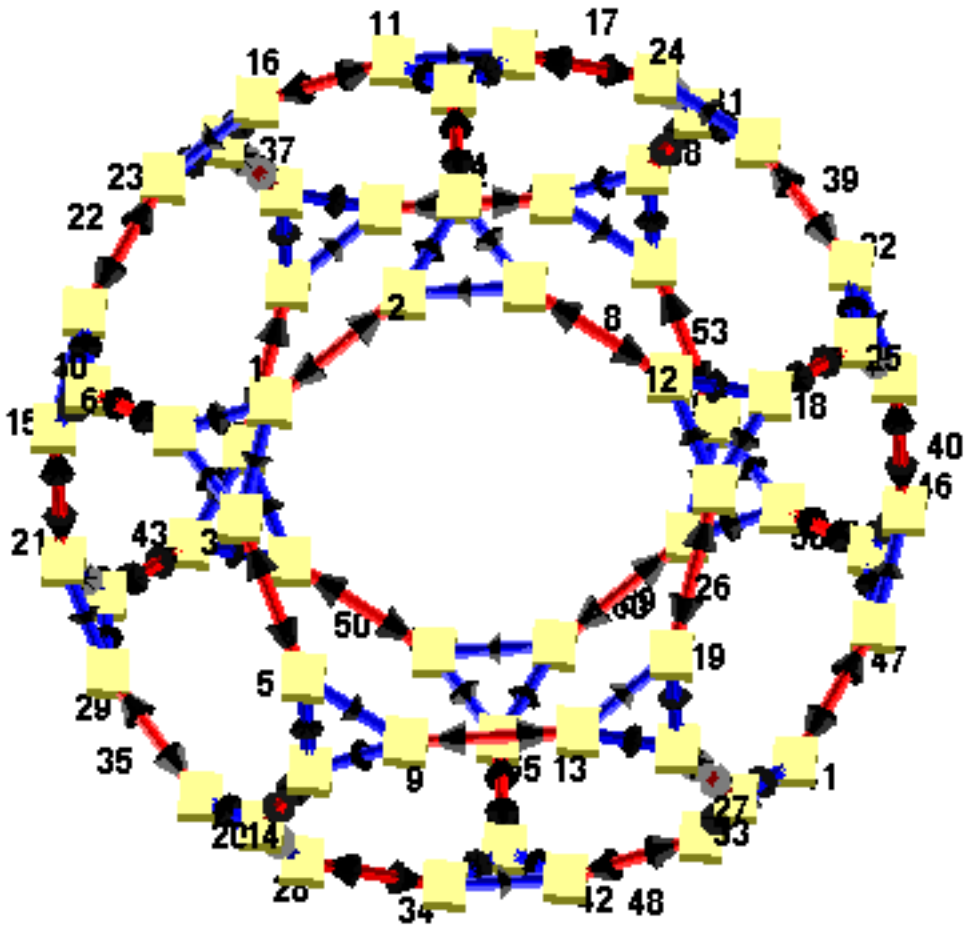
We can visualize small groups using their Cayley graph representation. Consider the alternating group A_5:

```
> A5 := < a, b | a^2, b^3, (a.b)^5 = 1 >;
```

$$A5 := \langle a, b \mid a^2, b^3, a\,b\,a\,b\,a\,b\,a\,b\,a\,b \rangle \tag{1.5}$$

We can using construct the DrawGraph command from the GraphTheory package to visualize its Cayley graph representation.

```
> GraphTheory:-DrawGraph( CayleyGraph( A5 ), dimension = 3 );
```



Working with Finitely Presented Groups

One of the difficulties you encounter when working with finitely presented groups is that almost any interesting question you might ask turns out to be algorithmically

undecidable, in general. This isn't simply a matter of nobody having been clever enough to work out how to answer such questions. It is a fact that, for most such questions, there can be no general algorithm to compute an answer. However, this does not preclude us from trying to answer interesting question in special cases, or for special classes of groups.

Let's try another example:

```
> G := < a, b, c, d | a^2, b^2, c^2, d^2, a.b = c, b.c = d, d.a =
  b >;
```

$$G := \langle a, b, c, d \mid a^2, b^2, c^2, d^2, a^{-1} d^{-1} b, b^{-1} a^{-1} c, c^{-1} b^{-1} d \rangle \quad \textbf{(2.1)}$$

We may ask whether this group is abelian.

```
> IsAbelian( G );
```

```
Error, (in IsAbelian) cannot determine whether a general
finitely presented group is Abelian.  If you know that your
group is finite, try converting it to a permutation group by
using the `PermutationGroup' command with your finitely
presented group as input.
```

Notice that Maple was unable (at least initially) to determine whether the group is abelian. However, by taking another route, we can see that the complicated presentation defining G actually results in a tiny group.

```
> GroupOrder( G );
```

$$2 \quad \textbf{(2.2)}$$

So, it turns out that the presentation used to define G is unnecessarily complicated. In some cases, Maple can produce a less complicated presentation for a given group. The Simplify command attempts to produce a less complicated presentation for a finitely presented group. It uses a heuristic algorithm to apply Tietze transformations to a finite presentation in an attempt to reduce the complexity of the presentation.

```
> Simplify( G );
```

$$\langle a \mid a^2 \rangle \quad \textbf{(2.3)}$$

Now that the presentation has been reduced to a particularly simple form, Maple is able to recognize that it defines an abelian group.

```
> IsAbelian( % );
```

$$true \quad \textbf{(2.4)}$$

Group Constructors and the Database of Small Groups

There are many ways to construct finitely presented groups in Maple other than by directly entering a presentation. Many group constructors have a form = "fpgroup" option that causes them to produce a finitely presented group. Maple includes a database of all the finite groups of order up to 511, and these can be accessed as finitely presented groups as follows.

```
> G := SmallGroup( 32, 5, 'form' = "fpgroup" );
```

$$G := \langle a1, a2, a3, a4, a5 \mid a2^2, a3^2, a5^2, a1^2\, a4^{-1}, a4^2\, a5^{-1}, a3^{-1}\, a1^{-1}\, a3\, a1, a3^{-1}\, a2^{-1}\, a3\, a2, a4^{-1}\, a1^{-1}\, a4\, a1, a4^{-1}\, a2^{-1}\, a4\, a2, a4^{-1}\, a3^{-1}\, a4\, a3, a5^{-1}\, a1^{-1}\, a5\, a1, a5^{-1}\, a2^{-1}\, a5\, a2, a5^{-1}\, a3^{-1}\, a5\, a3, a5^{-1}\, a4^{-1}\, a5\, a4, a2^{-1}\, a1^{-1}\, a2\, a1\, a3^{-1} \rangle \quad (3.1)$$

Many of the standard groups constructors also accept the form = ``fpgroup`` option.

```
> DihedralGroup( 5, 'form' = "fpgroup" );
```

$$D_5 \quad (3.2)$$

```
> QuaternionGroup( 'form' = "fpgroup" );
```

$$Q \quad (3.3)$$

```
> AlternatingGroup( 6, 'form' = "fpgroup" );
```

$$\langle s, t \mid t^3, s^4, t^{-1}\, s^{-1}\, t\, s\, t^{-1}\, s^{-1}\, t\, s, s\, t\, s\, t\, s\, t\, s\, t\, s\, t \rangle \quad (3.4)$$

```
> CyclicGroup( 12, 'form' = "fpgroup" );
```

$$\langle g \mid g^{12} \rangle \quad (3.5)$$

Subgroups of Finitely Presented Groups

Another important way in which finitely presented groups arise is as subgroups of other finitely presented groups. To see how this works, let's start with a free group of rank 2.

```
> F := FreeGroup( [ a, b ] );
```

$$F := \langle a, b \mid \rangle \quad (4.1)$$

Now define a subgroup by listing the subgroup generators as words on the generators of the parent group. Maple responds by computing a presentation for the subgroup. Of course, by Nielsen-Schreier, this is itself a free group, in this case, of rank 5.

```
> H := Subgroup( { a^2, b^2, (a.b)^2, (b.a)^2, (a.b^(-1))^2 }, F )
  ;
```

$$H := \langle \_G, \_G0, \_G1, \_G2, \_G3 \mid \rangle \quad (4.2)$$

We can also give the subgroup generators new names.

```
> H := Subgroup( { x = a^2, y = b^2, z = (a.b)^2, u = (b.a)^2, v =
```

```
  (a.b^(-1))^2 }, F  );
```

$$H := \langle u, v, x, y, z \mid \rangle \quad \textbf{(4.3)}$$

We can compute the (finite) index of H in F by using the Index command.

```
> Index( H, F );
```

$$4 \quad \textbf{(4.4)}$$

Observe that the Schreier Index Formula is satisfied:

```
> nops( Generators( H ) ) = Index( H, F ) * ( nops( Generators( F
  ) ) - 1 ) + 1;
```

$$5 = 5 \quad \textbf{(4.5)}$$

We can also compute the left or right cosets of H in F.

```
> RightCosets( H, F );
```

$$\{\langle u, v, x, y, z \mid \rangle \,.\, b, \langle u, v, x, y, z \mid \rangle \,.\, b\,a, \langle u, v, x, y, z \mid \rangle \,.\, b\,a\,b, \langle u, v, x, y, z \mid \rangle\} \quad \textbf{(4.6)}$$

The Representative method produces a representative for each coset.

```
> map( Representative, % );
```

$$\{1, b, b\,a, b\,a\,b\} \quad \textbf{(4.7)}$$

Permutation Groups

As we noted earlier, many questions cannot be answered for general finitely presented groups. However, for a finitely presented group that happens to be finite, we can often answer questions by first converting to a permutation group.

```
> G := SmallGroup( 12, 1, 'form' = "fpgroup" );
```

$$G := \langle a1, a2, a3 \mid a2^2, a1^2\,a2^{-1}, a3^3, a2^{-1}\,a1^{-1}\,a2\,a1, a3^{-1}\,a2^{-1}\,a3\,a2, a3^{-1}\,a1^{-1}\,a3\,a1\,a3^{-1} \rangle \quad \textbf{(5.1)}$$

```
> Simplify( G );
```

$$\langle a1, a3 \mid a3^3, a1^4, a3^{-1}\,a1^{-1}\,a3^{-1}\,a1 \rangle \quad \textbf{(5.2)}$$

```
> IsSoluble( G );
```

```
Error, (in DerivedSubgroup) cannot compute the derived
subgroup of a general finitely presented group.  If you know
that your group is finite, try converting it to a permutation
group by using the `PermutationGroup' command with your
finitely presented group as input.
```

Since that didn't work, we convert our group to a permutation group, and then ask questions about the group using the isomorphic permutation group. This approach

gives us much more information about our group.

```
> P := PermutationGroup( G );
```

$$P := \langle (1,2,4,3)(5,8,11,10)(6,7,12,9), (1,4)(2,3)(5,11)(6,12)(7,9)(8,10), (1,5,6)(2,7,8)(3,9,10)(4,11,12)\rangle \quad (5.3)$$

```
> GroupTheory:-IsSoluble( P );
```

$$true \quad (5.4)$$

```
> GroupTheory:-Hypercentre((5.3));
```

$$\langle (1,4)(2,3)(5,11)(6,12)(7,9)(8,10)\rangle \quad (5.5)$$

```
> GroupTheory:-IsNilpotent((5.3));
```

$$false \quad (5.6)$$

For more information on computing with and visualizing finite groups in Maple, including examples using permutation groups, Cayley table groups, and more, see the GroupTheory help page.