YangBaxter

Combinatorial Solutions for the Yang-Baxter equation

0.10.2

12 December 2022

Leandro Vendramin

Olexandr Konovalov

Leandro Vendramin
Email: Leandro.Vendramin@vub.be
Homepage: https://vendramin.github.io/
Address: Vrije Universiteit Brussel
Faculty of Sciences
Department of Mathematics and Data Science
Pleinlaan 2, B-1050
Brussel, Belgium

Olexandr Konovalov
Email: obk1@st-andrews.ac.uk
Homepage: https://alex-konovalov.github.io/
Address: School of Computer Science
University of St Andrews
Jack Cole Building, North Haugh,
St Andrews, Fife, KY16 9SX, Scotland

Contents

Chapter 1

Preliminaries

In this section we define skew braces and list some of their main properties [GV17].

1.1 Definition and examples

A skew brace is a triple $(A,+,\circ)$, where $(A,+)$ and $(A,\circ)$ are two (not necessarily abelian) groups such that the compatibility $a\circ(b+c) = a\circ b - a + a\circ c$ holds for all $a,b,c\in A$. Ones proves that the map $\lambda\colon(A,\circ)\to\mathrm{Aut}(A,+)$, $a\mapsto\lambda_a(b)$, $\lambda_a(b) = -a+a\circ b$, is a group homomorphism. Notation: For $a,b\in A$, we write $a*b=\lambda_a(b)-b$.

1.1.1 IsSkewbrace (for IsAttributeStoringRep)

▷ `IsSkewbrace(arg)` (filter)

Returns: `true` or `false`

1.1.2 Skewbrace (for IsList)

▷ `Skewbrace(list)` (operation)

Returns: a skew brace

The argument `list` is a list of pairs of elements in a group. By Proposition 5.11 of [GV17], skew braces over an abelian group A are equivalent to pairs (G,π), where G is a group and $\pi\colon G\to A$ is a bijective 1-cocycle, a finite skew brace can be constructed from the set $\{(a_j,g_j) : 1\le j\le n\}$, where $G=\{g_1,\dots,g_n\}$ and $A=\{a_1,\dots,a_n\}$ are permutation groups. This function is used to construct skew braces.

Example

```
gap> Skewbrace([[(),()]]);
<brace of size 1>
gap> Skewbrace([[(),()],[(1,2),(1,2)]]);
<brace of size 2>
```

1.1.3 SmallSkewbrace (for IsInt, IsInt)

▷ `SmallSkewbrace(n, k)` (operation)

Returns: a skew brace

The function returns the `k`-th skew brace from the database of skew braces of order `n`.

Example
```
gap> SmallSkewbrace(8,3);
<brace of size 8>
```

1.1.4 TrivialBrace (for IsGroup)

▷ `TrivialBrace(abelian_group)` (operation)

Returns: a brace

This function returns the trivial brace over the abelian group *abelian_group*. Here *abelian_group* should be an abelian group!

Example
```
gap> TrivialBrace(CyclicGroup(IsPermGroup, 5));
<brace of size 5>
```

1.1.5 TrivialSkewbrace (for IsGroup)

▷ `TrivialSkewbrace(group)` (operation)

Returns: a skew brace

This function returns the trivial skew brace over *group*.

Example
```
gap> TrivialSkewbrace(DihedralGroup(10));
<skew brace of size 10>
```

1.1.6 SmallBrace (for IsInt, IsInt)

▷ `SmallBrace(n, k)` (operation)

Returns: a brace of abelian type

The function returns the *k*-th brace (of abelian type) from the database of braces of order *n*.

Example
```
gap> SmallBrace(8,3);
<brace of size 8>
```

1.1.7 IdSkewbrace (for IsSkewbrace)

▷ `IdSkewbrace(obj)` (attribute)

Returns: a list

The function returns *[n, k]* if the skew brace *obj* is isomorphic to *SmallSkewbrace(n,k)*.

Example
```
gap> IdSkewbrace(SmallSkewbrace(8,5));
[ 8, 5 ]
```

1.1.8 AutomorphismGroup (for IsSkewbrace)

▷ `AutomorphismGroup(obj)` (attribute)

Returns: a list

The function computes the automorphism group of a skew brace.

Example
```
gap> br := SmallSkewbrace(8,20);;
gap> AutomorphismGroup(br);
<group with 8 generators>
gap> StructureDescription(last);
"D8"
```

Example
```
gap> br := SmallSkewbrace(8,25);;
gap> aut := AutomorphismGroup(br);;
gap> f := Random(aut);;
gap> x := Random(br);;
gap> ImageElm(f, x) in br;
true
```

1.1.9 IdBrace (for IsSkewbrace)

▷ `IdBrace(obj)` (attribute)

Returns: a list

The function returns `[ n, k ]` if the brace of abelian type `obj` is isomorphic to `SmallBrace(n,k)`.

Example
```
gap> IdBrace(SmallBrace(8,5));
[ 8, 5 ]
```

1.1.10 IsomorphismSkewbraces

▷ `IsomorphismSkewbraces(obj1, obj2)` (function)

Returns: an isomorphism of skew braces if `obj1` and `obj2` are isomorphic and `fail` otherwise.

If A and B are skew braces, a skew brace homomorphism is a map $f\colon A \to B$ such that

$$f(a+b) = f(a) + f(b) \quad f(a \circ b) = f(a) \circ f(b)$$

hold for all $a, b \in A$. A skew brace isomorphism is a bijective skew brace homomorphism. `IsomorphismSkewbraces` first computes all injective homomorphisms from $(A,+)$ to $(B,+)$ and then tries to find one f such that $f(a \circ b) = f(a) \circ f(b)$ for all $a, b \in A$.

1.1.11 DirectProductSkewbraces (for IsSkewbrace, IsSkewbrace)

▷ `DirectProductSkewbraces(obj1, obj2)` (operation)

Returns: the direct product of `obj1` and `obj2`

Example
```
gap> br1 := SmallBrace(8,18);;
gap> br2 := SmallBrace(12,2);;
gap> br := DirectProductSkewbraces(br1,br2);;
gap> IsLeftNilpotent(br);
false
gap> IsRightNilpotent(br);
false
gap> IsSolvable(br);
true
```

1.1.12 DirectProductOp (for IsList, IsSkewbrace)

▷ DirectProductOp(*arg1, arg2*) (operation)

1.1.13 IsTwoSided (for IsSkewbrace)

▷ IsTwoSided(*obj*) (property)

Returns: *true* if the skew brace is two sided, *false* otherwise

A skew brace A is said to be *two-sided* if $(a+b)\circ c = a\circ c - c + b\circ c$ holds for all $a,b,c\in A$.

Example
```
gap> IsTwoSided(SmallSkewbrace(8,2));
false
gap> IsTwoSided(SmallSkewbrace(8,4));
true
```

1.1.14 IsAutomorphismGroupOfSkewbrace (for IsAutomorphismGroup)

▷ IsAutomorphismGroupOfSkewbrace(*obj*) (property)

Returns: *true* if the group is the automorphism group of a skew braces, *false* otherwise

Example
```
gap> br := SmallSkewbrace(8,25);;
gap> aut := AutomorphismGroup(br);;
gap> Order(aut);
4
gap> IsAutomorphismGroupOfSkewbrace(aut);
true
```

1.1.15 IsClassical (for IsSkewbrace)

▷ IsClassical(*obj*) (property)

Returns: *true* if the skew brace is of abelian type, *false* otherwise

Let $\mathscr{X}$ be a property of groups. A skew brace A is said to be of $\mathscr{X}$-type if its additive group belongs to $\mathscr{X}$. In particular, skew braces of abelian type are those skew braces with abelian additive group. Such skew braces were introduced by Rump in [Rum07].

1.1.16 IsOfAbelianType (for IsSkewbrace)

▷ IsOfAbelianType(*arg*) (property)

Returns: true or false

1.1.17 IsBiSkewbrace (for IsSkewbrace)

▷ IsBiSkewbrace(*obj*) (property)

Returns: *true* if the skew brace is a bi-skew brace, *false* otherwise

A skew brace $(A,+,\circ)$ is said to be a bi-skew brace if $(A,\circ,+)$ is a skew brace

Example
```
gap> Number([1..NrSmallSkewbraces(8)], k->IsBiSkewbrace(SmallSkewbrace(8,k)));
39
```

1.1.18 IsOfNilpotentType (for IsSkewbrace)

▷ `IsOfNilpotentType(obj)` (property)

Returns: `true` if the skew brace is of nilpotent type, `false` otherwise

Let $\mathscr{X}$ be a property of groups. A skew brace A is said to be of $\mathscr{X}$-type if its additive group belongs to $\mathscr{X}$. In particular, skew braces of nilpotent type are those skew braces with nilpotent additive group.

1.1.19 IsTrivialSkewbrace (for IsSkewbrace)

▷ `IsTrivialSkewbrace(obj)` (property)

Returns: `true` if the skew brace is trivial, `false` otherwise

The function returns `true` if the skew brace A is trivial, i.e., $a \circ b = a + b$ for all $a, b \in A$. WARNING: The property IsTrivial applied to a skew brace will return true if and only if the skew brace has only one element.

Example
```
gap> br := SmallSkewbrace(9,1);;
gap> IsTrivialSkewbrace(br);
true
gap> IsTrivial(br);
false
```

1.1.20 Skewbrace2YB (for IsSkewbrace)

▷ `Skewbrace2YB(obj)` (attribute)

Returns: the set-theoretic solution associated with the skew brace `obj`

If A is a skew brace, the map $r_A\colon A \times A \to A \times A$

$$r_A(a,b) = (\lambda_a(b), \lambda_a(b)' \circ a \circ b)$$

is a non-degenerate set-theoretic solution of the Yang–Baxter equation. Furthermore, r_A is involutive if and only if A is of abelian type (i.e., the additive group of A is abelian).

Example
```
gap> Skewbrace2YB(TrivialBrace(CyclicGroup(6)));
<A set-theoretical solution of size 6>
```

1.1.21 Brace2YB (for IsSkewbrace)

▷ `Brace2YB(arg)` (attribute)

1.1.22 SkewbraceSubset2YB (for IsSkewbrace, IsCollection)

▷ `SkewbraceSubset2YB(obj)` (operation)

Returns: the set-theoretic solution associated with a given subset of a skew brace

Example
```
gap> br := TrivialSkewbrace(SymmetricGroup(3));;
gap> AsList(br);
[ <()>, <(2,3)>, <(1,2)>, <(1,2,3)>, <(1,3,2)>, <(1,3)> ]
gap> SkewbraceSubset2YB(br, last{[4,5]});
<A set-theoretical solution of size 2>
```

1.1.23 SemidirectProduct (for IsSkewbrace, IsSkewbrace, IsGeneralMapping)

▷ `SemidirectProduct(A, B, s)` (operation)

Returns: the semidirect product of skew braces

Let A and B be two skew braces and σ be a skew brace action of B on A, this is a group homomorphism $\sigma\colon (B,\circ) \to Aut_{\mathrm{Br}}(A)$ from the multiplicative group of B to the skew brace automorphism of A. The semidirect product of A and B with with respect to σ is the skew brace $A \rtimes_\sigma B$ with operations

$$(a_1,b_1)+(a_2,b_2)=(a_1+a_2,b_1+b_2), \quad (a_1,b_1)\circ(b_2,b_2)=(a_1\circ\sigma(b_1)(a_2),b_1\circ b_2)$$

Example

```
gap> A := SmallSkewbrace(4,2);;
gap> B := SmallSkewbrace(3,1);;
gap> s := SkewbraceActions(B,A);;
gap> Size(s);
1
gap> IdSkewbrace(SemidirectProduct(A,B,s[1]));
[ 12, 11 ]
gap> IdSkewbrace(DirectProduct(A,B));
[ 12, 11 ]
```

1.1.24 UnderlyingAdditiveGroup (for IsSkewbrace)

▷ `UnderlyingAdditiveGroup(A)` (attribute)

Returns: the underlying multiplicative group of the skew brace

Example

```
gap> br := SmallBrace(4,2);;
gap> G:=UnderlyingMultiplicativeGroup(br);;
gap> StructureDescription(G);
"C2 x C2"
```

1.1.25 UnderlyingMultiplicativeGroup (for IsSkewbrace)

▷ `UnderlyingMultiplicativeGroup(A)` (attribute)

Returns: the underlying additive group of the skew brace

Example

```
gap> br := SmallSkewbrace(6,2);;
gap> G:=UnderlyingAdditiveGroup(br);;
gap> IsAbelian(G);
false
```

Chapter 2

Algebraic Properties of Braces

2.1 Braces and Radical Rings

2.1.1 AdditiveGroupOfRing (for IsRing)

▷ `AdditiveGroupOfRing(ring)` (attribute)

Returns: a group

This function returns a permutation representation of the additive group of the given ring.

Example

```
gap> rg := SmallRing(8,10);;
gap> StructureDescription(AdditiveGroupOfRing(rg));
"C4 x C2"
```

2.1.2 IsJacobsonRadical (for IsRing)

▷ `IsJacobsonRadical(ring)` (attribute)

Returns: true if the ring is radical and false otherwise.

This function checks whether a ring is Jacobson radical.

Example

```
gap> rg := SmallRing(8,11);;
gap> IsJacobsonRadical(rg);
true
gap> rg := SmallRing(8,20);;
gap> IsJacobsonRadical(rg);
false
```

2.2 Braces and Yang-Baxter Equation

2.2.1 Table2YB (for IsList)

▷ `Table2YB(table)` (operation)

Returns: the solution given by the table

Given the table with $r(x,y)$ in the position (x,y) find the corresponding r

Example

```
gap> l := Table(SmallIYB(4,13));;
gap> t := Table2YB(l);;
```

```
gap> IdCycleSet(YB2CycleSet(t));
[ 4, 13 ]
```

2.2.2 Evaluate (for IsYB, IsList)

▷ Evaluate(*obj*, *pair*) (operation)

Returns: a pair of two integers

Given the pair (x,y) this function returns $r(x,y)$.

Example
```
gap> cs := SmallCycleSet(4,13);;
gap> yb := CycleSet2YB(cs);;
gap> Permutations(yb);
[ [ (3,4), (1,3,2,4), (1,4,2,3), (1,2) ],
  [ (2,4), (1,4,3,2), (1,2,3,4), (1,3) ] ]
gap> Evaluate(yb, [1,2]);
[ 2, 4 ]
gap> Evaluate(yb, [1,3]);
[ 4, 2 ]
```

2.2.3 LyubashenkoYB (for IsInt, IsPerm, IsPerm)

▷ LyubashenkoYB(*size*, *f*, *g*) (operation)

Returns: a permutation solution to the YBE

Finite Lyubashenko (or permutation) solutions are defined as follows: Let $X = \{1,\dots,n\}$ and $f,g\colon X \to X$ be bijective functions such that $fg = gf$. Then (X,r), where $r(x,y) = (f(y),g(x))$, is a set-theoretic solution to the YBE.

Example
```
gap> yb := LyubashenkoYB(4, (1,2),(3,4));
<A set-theoretical solution of size 4>
gap> Permutations(last);
[ [ (1,2), (1,2), (1,2), (1,2) ], [ (3,4), (3,4), (3,4), (3,4) ] ]
```

2.2.4 IsIndecomposable (for IsYB)

▷ IsIndecomposable(*X*) (property)

Returns: true if the involutive solutions is indecomposable

2.2.5 Table (for IsYB)

▷ Table(*obj*) (attribute)

Returns: a table with the image of the solution

The table shows the value of $r(x,y)$ for each (x,y)

Example
```
gap> yb := SmallIYB(3,2);;
gap> Table(yb);
[ [ [ 1, 1 ], [ 2, 1 ], [ 3, 2 ] ], [ [ 1, 2 ], [ 2, 2 ], [ 3, 1 ] ], [ [ 2, 3 ], [ 1, 3 ], [ 3,
```

2.2.6 DehornoyClass (for IsYB)

▷ DehornoyClass(*obj*) (attribute)

Returns: The class of an involutive solution

Example
```
gap> cs := SmallCycleSet(4,13);;
gap> yb := CycleSet2YB(cs);;
gap> DehornoyClass(yb);
2
gap> cs := SmallCycleSet(4,19);;
gap> yb := CycleSet2YB(cs);;
gap> DehornoyClass(yb);
4
```

2.2.7 DehornoyRepresentationOfStructureGroup (for IsYB, IsObject)

▷ DehornoyRepresentationOfStructureGroup(*obj*, *variable*) (operation)

Returns: A faithful linear representation of the structure group of `obj`

Example
```
gap> cs := SmallCycleSet(4,13);;
gap> yb := CycleSet2YB(cs);;
gap> Permutations(yb);
[ [ (3,4), (1,3,2,4), (1,4,2,3), (1,2) ],
  [ (2,4), (1,4,3,2), (1,2,3,4), (1,3) ] ]
gap> field := FunctionField(Rationals, 1);;
gap> q := IndeterminatesOfFunctionField(field)[1];;
gap> G := DehornoyRepresentationOfStructureGroup(yb, q);;
gap> x1 := G.1;;
gap> x2 := G.2;;
gap> x3 := G.3;;
gap> x4 := G.4;;
gap> x1*x2=x2*x4;
true
gap> x1*x3=x4*x2;
true
gap> x1*x4=x3*x3;
true
gap> x2*x1=x3*x4;
true
gap> x2*x2=x4*x1;
true
gap> x3*x1=x4*x3;
true
```

2.2.8 IdYB (for IsYB)

▷ IdYB(*obj*) (attribute)

Returns: the identification number of `obj`

Example
```
gap> cs := SmallCycleSet(5,10);;
gap> IdCycleSet(cs);
```

```
[ 5, 10 ]
gap> cs := SmallCycleSet(4,3);;
gap> yb := CycleSet2YB(cs);;
gap> IdYB(yb);
[ 4, 3 ]
```

2.2.9 LinearRepresentationOfStructureGroup (for IsYB)

▷ LinearRepresentationOfStructureGroup(*obj*) (attribute)

Returns: the permutation brace of the involutive solution of *obj* a linear representation of the structure group of a finite involutive solution

Example

```
gap> yb := SmallIYB(5,86);;
gap> IdBrace(IYBBrace(yb));
[ 6, 2 ]
```

Example

```
gap> yb := SmallIYB(5,86);;
gap> gr := LinearRepresentationOfStructureGroup(yb);;
gap> gens := GeneratorsOfGroup(gr);;
gap> Display(gens[1]);
[ [  0,  1,  0,  0,  0,  1 ],
  [  1,  0,  0,  0,  0,  0 ],
  [  0,  0,  0,  0,  1,  0 ],
  [  0,  0,  1,  0,  0,  0 ],
  [  0,  0,  0,  1,  0,  0 ],
  [  0,  0,  0,  0,  0,  1 ] ]
```

Chapter 3

YangBaxter automatic generated documentation

3.1 YangBaxter automatic generated documentation of properties

3.1.1 IsIndecomposable (for IsCycleSet)

▷ `IsIndecomposable(arg)` (property)

Returns: true if the cycle set is indecomposable

Let X be a cycle set. We say that X is indecomposable if the group $\mathscr{G}(X) = \langle \varphi_x : x \in X \rangle$ acts transitively on X.

Chapter 4

Ideals and left ideals

In this section we describe several functions related to ideals and left ideals of skew braces. References: [GV17] and [SV18].

4.1 Left ideals

An left ideal I of a skew brace A is a subgroup I of the additive group of A such that $\lambda_a(I) \subseteq I$ for all $a \in A$.

4.1.1 LeftIdeals (for IsSkewbrace)

▷ `LeftIdeals(obj)` (attribute)

Returns: a list with the left ideals of the skew brace `obj`

4.1.2 StrongLeftIdeals (for IsSkewbrace)

▷ `StrongLeftIdeals(obj)` (attribute)

Returns: a list with the left ideals of the skew brace `obj` that are normal in the additive group of A

4.1.3 IsLeftIdeal (for IsSkewbrace, IsCollection)

▷ `IsLeftIdeal(obj)` (operation)

Returns: `true` if the subset is a left ideal of `obj`

Example

```
gap> br := SmallBrace(8,4);
<brace of size 8>
gap> leftideals := LeftIdeals(br);
[ <brace of size 1>, <brace of size 2>, <brace of size 4>, <brace of size 8> ]
gap> List(leftideals, x->IsLeftIdeal(br, x));
[ true, true, true, true ]
gap> List(leftideals, IdBrace);
[ [ 1, 1 ], [ 2, 1 ], [ 4, 1 ], [ 8, 4 ] ]
```

4.2 Ideals

An ideal I of a skew brace A is a normal subgroup I of the additive group of A such that $\lambda_a(I) \subseteq I$ and $a \circ I = I \circ a$ for all $a \in A$.

4.2.1 IsIdeal (for IsSkewbrace, IsCollection)

▷ `IsIdeal(obj, subset)` (operation)

Returns: *true* if the *subset* is a left ideal of *obj*

Example
```
gap> br := SmallBrace(8,4);
<brace of size 8>
gap> leftideals := LeftIdeals(br);
[ <brace of size 1>, <brace of size 2>, <brace of size 4>, <brace of size 8> ]
gap> List(leftideals, x->IsLeftIdeal(br, x));
[ true, true, true, true ]
gap> List(leftideals, IdBrace);
[ [ 1, 1 ], [ 2, 1 ], [ 4, 1 ], [ 8, 4 ] ]
```

4.2.2 Ideals (for IsSkewbrace)

▷ `Ideals(obj)` (attribute)

Returns: a list with the ideals of the skew brace *obj*

4.2.3 AsIdeal (for IsSkewbrace, IsCollection)

▷ `AsIdeal(arg1, arg2)` (operation)

4.2.4 IdealGeneratedBy (for IsSkewbrace, IsCollection)

▷ `IdealGeneratedBy(obj, subset)` (operation)

Returns: the ideal of *obj* generated by the given *subset*

The ideal of a skew brace A generated by a subset X is the intersection of all the ideals of A containing X.

Example
```
gap> br := SmallSkewbrace(6,6);;
gap> AsList(br);
[ <()>, <(1,2,3)(4,5,6)>, <(1,3,2)(4,6,5)>, <(1,4)(2,5)(3,6)>,
  <(1,5,3,4,2,6)>, <(1,6,2,4,3,5)> ]
gap> IdealGeneratedBy(br, [last[2]]);
<brace of size 3>
```

4.2.5 IntersectionOfTwoIdeals (for IsSkewbrace and IsIdealInParent, IsSkewbrace and IsIdealInParent)

▷ `IntersectionOfTwoIdeals(ideal1, ideal2)` (operation)

Returns: the intersection of *ideal1* and *ideal2*

Example
```
gap> br := SmallSkewbrace(6,6);;
gap> Ideals(br);;
gap> IntersectionOfTwoIdeals(last[2],last[3]);
<brace of size 1>
```

4.2.6 SumOfTwoIdeals (for IsSkewbrace and IsIdealInParent, IsSkewbrace and IsIdealInParent)

▷ `SumOfTwoIdeals(`*`ideal1, ideal2`*`)` (operation)

Returns: the sum of *`ideal1`* and *`ideal2`*

Example
```
gap> br := SmallSkewbrace(6,6);;
gap> Ideals(br);;
gap> SumOfTwoIdeals(last[2],last[3]);
<brace of size 6>
```

4.3 Sequences (left) ideals

4.3.1 LeftSeries (for IsSkewbrace)

▷ `LeftSeries(`*`obj`*`)` (attribute)

Returns: the left ideals of the left series of *`obj`*

The left series of a skew brace A is defined recursively as $A^1 = A$ and $A^{n+1} = A * A^n$ for $n \geq 1$, where $a * b = \lambda_a(b) - b$. Each A^n is a left ideal.

Example
```
gap> br := SmallSkewbrace(8,20);
<skew brace of size 8>
gap> LeftSeries(br);
[ <skew brace of size 8>, <brace of size 2>, <brace of size 1> ]
```

4.3.2 RightSeries (for IsSkewbrace)

▷ `RightSeries(`*`obj`*`)` (attribute)

Returns: the ideals of the right series of *`obj`*

The right series of a skew brace $0A$ is defined recursively as $A^{(1)} = A$ and $A^{(n+1)} = A * A^{(n)}$ for $n \geq 1$, where $a * b = \lambda_a(b) - b$

Example
```
gap> br := SmallSkewbrace(8,20);
<skew brace of size 8>
gap> RightSeries(br);
[ <skew brace of size 8>, <brace of size 2>, <brace of size 1> ]
```

4.3.3 IsLeftNilpotent (for IsSkewbrace)

▷ `IsLeftNilpotent(`*`obj`*`)` (property)

Returns: *`true`* if the skew brace *`obj`* is left nilpotent.

A skew brace A is said to be left nilpotent if there exists $n \geq 1$ such that $A^n = 0$.

Example
```
gap> IsLeftNilpotent(SmallBrace(8,18));
true
gap> IsLeftNilpotent(SmallBrace(12,2));
false
```

4.3.4 IsSimpleSkewbrace (for IsSkewbrace)

▷ `IsSimpleSkewbrace(obj)` (property)

Returns: `true` if the skew brace `obj` is simple.

A skew brace A is said to be simple if $\{0\}$ and A are its only ideals.

Example
```
gap> IsSimple(SmallSkewbrace(12,22));
true
gap> IsSimple(SmallSkewbrace(12,21));
false
```

4.3.5 IsRightNilpotent (for IsSkewbrace)

▷ `IsRightNilpotent(obj)` (property)

Returns: `true` if the skew brace `obj` is right nilpotent.

A skew brace A is said to be right nilpotent if there exists $n \geq 1$ such that $A^{(n)} = 0$.

Example
```
gap> IsRightNilpotent(SmallBrace(8,18));
false
gap> IsRightNilpotent(SmallBrace(12,2));
true
```

4.3.6 LeftNilpotentIdeals (for IsSkewbrace)

▷ `LeftNilpotentIdeals(obj)` (attribute)

Returns: the list of right or left nilpotent ideals of `obj`

An ideal I of a skew brace A is said to be left if it is left nilpotent as a skew brace.

4.3.7 RightNilpotentIdeals (for IsSkewbrace)

▷ `RightNilpotentIdeals(obj)` (attribute)

Returns: the list of right or left nilpotent ideals of `obj`

An ideal I of a skew brace A is said to be right nilpotent if An ideal I of a skew brace A is said to be left if it is right nilpotent as a skew brace.

Example
```
gap> br := SmallBrace(8,18);;
gap> IsLeftNilpotent(br);
true
gap> IsRightNilpotent(br);
false
gap> Length(LeftNilpotentIdeals(br));
3
gap> Length(RightNilpotentIdeals(br));
2
```

4.3.8 SmoktunowiczSeries (for IsSkewbrace, IsInt)

▷ SmoktunowiczSeries(*obj*, *bound*) (operation)

Returns: a list of *bound* left ideals of the Smoktunowicz's series of *obj*

The Smoktunowicz's series of a skew brace A is defined recursively as $A^{[1]} = A$ and $A^{[n+1]}$ is the additive subgroup of A generated by $A^{[i]} * A^{[n+1-i]}$ for $1 \leq i+j \leq n+1$, where $a*b = \lambda_a(b) - b$.

Example
```
gap> br := SmallBrace(16,145);;
gap> SmoktunowiczSeries(br,4);
[ <brace of size 16>, <brace of size 8>, <brace of size 4>, <brace of size 2>,
  <brace of size 2> ]
gap> SmoktunowiczSeries(br,5);
[ <brace of size 16>, <brace of size 8>, <brace of size 4>, <brace of size 2>,
  <brace of size 2>, <brace of size 1> ]
```

4.3.9 Socle (for IsSkewbrace)

▷ Socle(*obj*) (attribute)

Returns: the socle of *obj*

The socle of a skew brace A is the ideal $\ker\lambda \cap Z(A,+)$.

Example
```
gap> Socle(SmallSkewbrace(6,2));
<brace of size 1>
gap> Socle(SmallBrace(8,20));
<brace of size 8>
gap> Socle(SmallBrace(8,2));
<brace of size 4>
```

4.3.10 Annihilator (for IsSkewbrace)

▷ Annihilator(*obj*) (attribute)

Returns: the annihilator of *obj*

The socle of a skew brace A is the ideal $\ker\lambda \cap Z(A,+) \cap Z(A,\circ)$.

Example
```
gap> Annihilator(SmallSkewbrace(8,12));
<brace of size 2>
gap> Annihilator(SmallSkewbrace(4,2));
<brace of size 2>
gap> Annihilator(SmallSkewbrace(8,14));
<brace of size 4>
```

4.4 Mutipermutation skew braces

4.4.1 SocleSeries (for IsSkewbrace)

▷ SocleSeries(*obj*) (operation)

Returns: the socle series of *obj*

The socle series of a skew brace A is defined recursively as $A_1 = A$ and $A_{n+1} = A_n/\mathrm{Soc}(A_n)$, see [SV18].

4.4.2 MultipermutationLevel (for IsSkewbrace)

▷ `MultipermutationLevel(obj)` (attribute)

Returns: the multipermutation level of the skew brace `obj`

The multipermutation level of a skew brace A is defined as the smallest positive integer n such that the n-th term A_n of the socle series has only one element, see Definition 5.17 of [SV18].

Example
```
gap> br := SmallBrace(8,20);;
gap> SocleSeries(br);
[ <brace of size 8>, <brace of size 1> ]
gap> MultipermutationLevel(br);
2
```

4.4.3 IsMultipermutation (for IsSkewbrace)

▷ `IsMultipermutation(obj)` (property)

Returns: `true` if the skew brace `obj` has finite multipermutation level and `false` otherwise

4.4.4 Fix (for IsSkewbrace)

▷ `Fix(obj)` (attribute)

Returns: the left ideal $\{x \in A : \lambda_a(x) = x \, \forall a \in A\}$ of the skew brace A.

Example
```
gap> br := SmallSkewbrace(6,1);;
gap> IsTrivialSkewbrace(br);
true
gap> Fix(br);
[ <()>, <(1,2,3)(4,5,6)>, <(1,3,2)(4,6,5)>, <(1,4)(2,6)(3,5)>,
  <(1,5)(2,4)(3,6)>, <(1,6)(2,5)(3,4)> ]
```

4.4.5 KernelOfLambda (for IsSkewbrace)

▷ `KernelOfLambda(obj)` (attribute)

Returns: the kernel of the map λ as a subset of elements of the skew brace `obj`.

Example
```
gap> br := SmallBrace(6,1);;
gap> KernelOfLambda(br);
[ <()>, <(1,2,3)(4,5,6)>, <(1,3,2)(4,6,5)> ]
```

4.4.6 Quotient (for IsSkewbrace, IsSkewbrace)

▷ `Quotient(obj, ideal)` (operation)

Returns: the quotient `obj` by `ideal`

Example
```
gap> br := SmallBrace(8,10);;
gap> ideals := Ideals(br);;
gap> Quotient(br, ideals[3]);
<brace of size 4>
gap> br/ideals[3];
<brace of size 4>
```

4.5 Prime and semiprime ideals

4.5.1 IsPrimeBrace (for IsSkewbrace)

▷ `IsPrimeBrace(obj)` (property)

Returns: `true` if the skew brace `obj` is prime

A skew brace A is said to be prime if for all non-zero ideals I and J one has $I*J \neq 0$

Example
```
gap> IsPrimeBrace(SmallBrace(24,12));
false
gap> IsPrimeBrace(SmallBrace(24,94));
true
```

4.5.2 IsPrimeIdeal (for IsSkewbrace and IsIdealInParent)

▷ `IsPrimeIdeal(obj)` (property)

Returns: `true` if the ideal `obj` is prime

An ideal I of a skew brace A is said to be prime if A/I is a prime skew brace.

Example
```
gap> br := SmallBrace(24,94);
<brace of size 24>
gap> IsPrimeBrace(br);
true
gap> Ideals(br);;
gap> IsPrimeIdeal(last[2]);
true
```

4.5.3 PrimeIdeals (for IsSkewbrace)

▷ `PrimeIdeals(obj)` (attribute)

Returns: the list of prime ideals of the skew brace `obj`

Example
```
gap> Length(PrimeIdeals(SmallBrace(24,94)));
2
```

4.5.4 IsSemiprime (for IsSkewbrace)

▷ `IsSemiprime(obj)` (attribute)

Returns: `true` if the skew brace `obj` is semiprime

An ideal I of a skew brace A is said to be semiprime if A/I is a semiprime skew brace.

Example
```
gap> br := DirectProductSkewbraces(SmallSkewbrace(12,22),SmallSkewbrace(12,22));;
gap> IsSemiprime(br);
true
```

4.5.5 IsSemiprimeIdeal (for IsSkewbrace and IsIdealInParent)

▷ `IsSemiprimeIdeal(obj)` (attribute)

Returns: `true` if the ideal `obj` is semiprime

Example
```
gap> SemiprimeIdeals(SmallSkewbrace(12,24));
[ <skew brace of size 12> ]
gap> IsSemiprimeIdeal(last[1]);
true
```

4.5.6 SemiprimeIdeals (for IsSkewbrace)

▷ `SemiprimeIdeals(obj)` (attribute)

Returns: the list of semiprime ideals of the skew brace `obj`

Example
```
gap> SemiprimeIdeals(SmallSkewbrace(12,24));
[ <skew brace of size 12> ]
gap> Length(SemiprimeIdeals(SmallSkewbrace(12,22)));
2
```

4.5.7 BaerRadical (for IsSkewbrace)

▷ `BaerRadical(obj)` (attribute)

Returns: the Baer radical of the skew brace `obj`

Example
```
gap> br := SmallSkewbrace(6,2);;
gap> BaerRadical(br);
<skew brace of size 6>
```

4.5.8 IsBaer (for IsSkewbrace)

▷ `IsBaer(obj)` (property)

Returns: `true` if the skew brace `obj` is ia Baer radical skew brace.

A skew brace A is said to be Baer radical if $A = B(A)$, where $B(A)$ is the Baer radical of A (i.e., the intersection of all prime ideals of A).

Example
```
gap> br := SmallSkewbrace(6,2);;
gap> IsBaer(br);
true
```

4.5.9 WedderburnRadical (for IsSkewbrace)

▷ `WedderburnRadical(obj)` (attribute)

Returns: the Wedderburn radical of the skew brace `obj`

The Wedderburn radical of a skew brace is the intersection of all its prime ideals

Example
```
gap> br := SmallSkewbrace(6,2);;
gap> WedderburnRadical(br);
<brace of size 3>
```

4.5.10 SolvableSeries (for IsSkewbrace)

▷ `SolvableSeries(obj)` (attribute)

Returns: a list with the solvable series of the skew brace *obj*

The solvable series of a skew brace A is defined recursively as $A_1 = A$ and $A_{n+1} = A_n * A_n$ for $n \geq 1$, where $a * b = \lambda_a(b) - b$

Example

```
gap> br := SmallSkewbrace(8,20);;
gap> IsSolvable(br);
true
gap> SolvableSeries(br);
[ <skew brace of size 8>, <brace of size 2>, <brace of size 1> ]
gap> br := SmallSkewbrace(12,23);;
gap> IsSolvable(br);
false
```

4.5.11 IsMinimalIdeal (for IsSkewbrace and IsIdealInParent)

▷ `IsMinimalIdeal(obj, ideal)` (property)

Returns: *true* if *ideal* is a minimal ideal of *obj* An ideal I of A is said to be *minimal* if does not contain any other ideal of A. To check if an ideal I of A is minimal, one computes the ideals of I and keep only those that are simple as a skew brace.

4.5.12 MinimalIdeals (for IsSkewbrace)

▷ `MinimalIdeals(obj)` (attribute)

Returns: a list of minimal ideals of the skew brace *obj*

References

[GV17] L. Guarnieri and L. Vendramin. Skew braces and the Yang–Baxter equation. *Math. Comp.*, 86(307):2519–2534, 2017. 3, 14

[Rum07] Wolfgang Rump. Braces, radical rings, and the quantum Yang-Baxter equation. *J. Algebra*, 307(1):153–170, 2007. 6

[SV18] Agata Smoktunowicz and Leandro Vendramin. On skew braces (with an appendix by N. Byott and L. Vendramin). *J. Comb. Algebra*, 2(1):47–86, 2018. 14, 18, 19

Index