LiePRing
—
A GAP4 Package

Version 1.4

by

Michael Vaughan-Lee
and
Bettina Eick

Contents

1 Introduction

A Lie p-ring is a nilpotent Lie ring of p-power order for a prime p. This package contains some algorithms for Lie p-rings and it gives access to the database of Lie p-rings of order at most p^7 as determined by Mike Newman, Eamonn O'Brien and Michael Vaughan-Lee, see [NOVL03] and [OVL05].

For this purpose, this package introduces a datastructure for Lie p-rings. For a fixed prime p, this datastructure is quite similar to power-conjugate presentations for finite p-groups. The datastructure introduced here also allows to define Lie p-rings in which the presentation contains indeterminates. In particular, the prime p is allowed to be an indeterminate. Such Lie p-rings are called *generic*; they are described in more detail in Chapter 2 of this manual.

The package then defines for each $n \in \{1, \ldots, 7\}$ a (finite) list of generic presentations of Lie p-rings. For each prime $p \geq 5$, each of the generic Lie p-rings gives rise to a family of Lie p-rings over the considered prime p by specialising the indeterminates to a certain list of values. The resulting lists of Lie p-rings provides a complete and irredundant set of isomorphism type representatives of the Lie p-rings of order p^n. The generic Lie p-rings of p-class at most 2 can also be considered for the prime $p = 3$ and yield a list of isomorphism type representatives for the Lie p-rings of order 3^n and class at most 2. (The p-class of a Lie ring is the length of the lower exponent-p central series.)

The Lazard correspondence induces a one-to-one correspondence between the Lie p-rings of order p^n and class less than p and the p-groups of order p^n and class less than p. This package provides a function to evaluate this correspondence; this function has been implemented by Willem de Graaf. It uses the Liering package [CdG10].

The Lazard correspondence has been used to check the correctness of the database of Lie p-rings: for various small primes it has been checked that the Lie p-rings of this database define non-isomorphic finite p-groups.

2 Lie p-rings

A Lie ring L is an additive abelian group with a multiplication that is alternating, bilinear and satisfies the Jacobi identity. The multiplication in a Lie ring is often denoted with brakets $[g,h]$, however, in GAP and also in this manual multiplication is denoted by $g \cdot h$.

Let L be a Lie p-ring and thus a nilpotent Lie ring of order p^n. Then L has a central series $L = L_1 \geq \ldots \geq L_n \geq \{0\}$ with quotients of order p. Choose $l_i \in L_i \setminus L_{i+1}$ for $1 \leq i \leq n$. Then $(l_1, \ldots, l_n)$ is a generating set of L satisfying that $p \cdot l_i \in L_{i+1}$ and $l_i \cdot l_j \in L_{i+1}$ for $1 \leq j < i \leq n$. We also call such a generating sequence a *basis* for L and we say that L has *dimension* n.

Given a basis $(l_1, \ldots, l_n)$ for a Lie p-ring L, there exist coefficients $c_{i,j,k} \in \{0, \ldots, p-1\}$ so that the following relations hold in L for $1 \leq j < i \leq n$:

$$l_i \cdot l_j = \sum_{k=i+1}^{n} c_{i,j,k} l_k,$$

$$pl_i = \sum_{k=i+1}^{n} c_{i,i,k} l_k.$$

These relations define the Lie p-ring L. This package contains the definition of a datastructure *LiePRing* that allows to define Lie p-rings via structure constants $c_{i,j,k}$.

2.1 Ordinary Lie p-rings

In an ordinary Lie p-ring, the prime p is an integer and the structure constants $c_{i,j,k}$ are elements in $\{0, \ldots, p-1\}$. The following example takes the 9th Lie p-ring from the database of Lie p-rings of order 5^4 and does some elementary computations with it.

```
gap> L := LiePRingsByLibrary(4, 5)[9];
<Lie ring of dimension 4 over prime 5>
gap> l := BasisOfLiePRing(L);
[ l1, l2, l3, l4 ]
gap> l[1]*l[2];
0
gap> 5*l[1];
l3
```

2.2 Generic Lie p-rings

In a generic Lie p-ring, the structure constants are allowed to be polynomials in a finite set of indeterminates. In particular, the prime p may not be a fixed integer, but an indeterminate. The following examples takes the 9th Lie p-ring from the database of Lie p-rings of order p^4 and does some elementary computations with it.

```
gap> L := LiePRingsByLibrary(4)[9];
<Lie ring of dimension 4 over prime p>
gap> l := BasisOfLiePRing(L);
[ l1, l2, l3, l4 ]
gap> p := PrimeOfLiePRing(L);
p
gap> p*l[1];
l3
gap> l[1]*l[2];
0
```

A generic Lie p-ring thus defines a family of Lie p-rings by evaluating the prime p and by evaluating the other parameters. It is generally assumed that p is evaluated to a prime and w is evaluated to a primitive root of the field of p elements. The following functions allow to evaluate indeterminates in values.

1 ► `SpecialisePrimeOfLiePRing(L, P)`

takes a generic Lie p-ring L and specialises its prime p (an indeterminate) to the value P. It also specialises the indeterminate w to a primitive root of $GF(p)$ if w occurs in the presentation of L.

The following example shows a generic Lie p-ring with the parameter x in the relations. This parameter x is not evaluated together with the prime.

```
gap> L := LiePRingsByLibrary(6)[14];
<Lie ring of dimension 6 over prime p with parameters [ x ]>
gap> ViewPCPresentation(L);
p*l1 = l4
p*l2 = x*l6
p*l4 = l5
[l2,l1] = l3
[l3,l1] = l5
[l3,l2] = l6
gap> K := SpecialisePrimeOfLiePRing(L, 5);
<Lie ring of dimension 6 over prime 5 with parameters [ x ]>
gap> ViewPCPresentation(K);
5*l1 = l4
5*l2 = x*l6
5*l4 = l5
[l2,l1] = l3
[l3,l1] = l5
[l3,l2] = l6
```

The following example shows a generic Lie p-ring with the parameter w in the relations. As w is evaluated to a primitive root of $GF(p)$, it is evaluated together with the prime.

```
gap> L := LiePRingsByLibrary(6)[19];
<Lie ring of dimension 6 over prime p with parameters [ w ]>
gap> ViewPCPresentation(L);
p*l1 = l4
p*l2 = w*l5
p*l4 = l6
[l2,l1] = l3
[l3,l1] = l5
gap> K := SpecialisePrimeOfLiePRing(L, 17);
<Lie ring of dimension 6 over prime 17>
gap> ViewPCPresentation(K);
17*l1 = l4
17*l2 = 3*l5
17*l4 = l6
[l2,l1] = l3
[l3,l1] = l5
```

2 ► `SpecialiseLiePRing(L, P, para, vals)`

takes a generic Lie p-ring L and specialises its prime p as above and also specialises the indeterminates in *para* to the values *vals*.

```
gap> L := LiePRingsByLibrary(6)[14];
<Lie ring of dimension 6 over prime p with parameters [ x ]>
gap> ViewPCPresentation(L);
p*l1 = l4
p*l2 = x*l6
p*l4 = l5
[l2,l1] = l3
[l3,l1] = l5
[l3,l2] = l6
gap> para := ParametersOfLiePRing(L);
[ x ]
gap> SpecialiseLiePRing(L, 29, para, [0]);
<Lie ring of dimension 6 over prime 29>
gap> ViewPCPresentation(last);
29*l1 = l4
29*l4 = l5
[l2,l1] = l3
[l3,l1] = l5
[l3,l2] = l6
```

The following example shows that it is possible to specialise some of the parameters only. Again, note that w is always specialised together with p.

```
gap> L := LiePRingsByLibrary(6)[267];
<Lie ring of dimension 6 over prime p with parameters [ w, x, y, z, t ]>
gap> ViewPCPresentation(L);
p*l1 = t*l5 + x*l6
p*l2 = y*l5 + z*l6
[l2,l1] = l4
[l3,l1] = l6
[l3,l2] = w*l5
[l4,l1] = l5
```

```
[l4,l2] = l6
gap> x := Indeterminate(Integers, "x");
x
gap> SpecialiseLiePRing(L, 29, [x], [0]);
<Lie ring of dimension 6 over prime 29 with parameters [ t, z, y ]>
gap> ViewPCPresentation(last);
29*l1 = t*l5
29*l2 = y*l5 + z*l6
[l2,l1] = l4
[l3,l1] = l6
[l3,l2] = 2*l5
[l4,l1] = l5
[l4,l2] = l6
```

3 ► `LiePValues(K)`

if K is obtained by specialising, then this attribute is set and contains the parameters that have been specialised and their values.

```
gap>  L := LiePRingsByLibrary(6)[14];
<Lie ring of dimension 6 over prime p with parameters [ x ]>
gap>  K := SpecialisePrimeOfLiePRing(L, 5);
<Lie ring of dimension 6 over prime 5 with parameters [ x ]>
gap> LiePValues(K);
[ [ p, w ], [ 5, 2 ] ]
```

2.3 Creation of Lie p-rings

Lie p-rings can be created from certain table containing the structure constants.

1 ► `LiePRingBySCTable(SC)`
► `LiePRingBySCTableNC(SC)`

creates a Lie p-ring datastructure from *SC*. The input *SC* should be a record with entries *dim*, *prime*, *tab* and possibly *param*. The NC version assumes that the Jacobi identity is satisfied by *SC* and the other version checks this. The entry *tab* is a list of lists. This list defines $l_i \cdot l_j$ for $j < i$ via the entry at position $1 + \ldots + j - 1 + i$ and it defines $p \cdot l_i$ via the entry at position $1 + \ldots + i$. If an entry in this list is not bound, then it is assumed to be the empty list.

2 ► `CheckIsLiePRing(L)`

this function assumes that L has been defined via *LiePRingBySCTableNC* and it checks the Jacobi identity for the multiplication in L.

```
gap> p := Indeterminate(Integers,"p");;
gap> w := Indeterminate(Integers,"w");;
gap> x := Indeterminate(Integers,"x");;
gap> y := Indeterminate(Integers,"y");;
gap> z := Indeterminate(Integers,"z");;
gap> t := Indeterminate(Integers,"t");;
gap> SC := rec( dim := 6, param := [w,x,y,z,t], prime := p,
>               tab := [ [ 5, t, 6, x ], [ 4, 1 ], [ 5, y, 6, z ],
>                        [ 6, 1 ], [ 5, w ], [  ], [ 5, 1 ], [ 6, 1 ] ] );;
gap> L := LiePRingBySCTable(SC);
<Lie ring of dimension 6 over prime p with parameters [ w, x, y, z, t ]>
```

```
gap> ViewPCPresentation(L);
p*l1 = t*l5 + x*l6
p*l2 = y*l5 + z*l6
[l2,l1] = l4
[l3,l1] = l6
[l3,l2] = w*l5
[l4,l1] = l5
[l4,l2] = l6
```

2.4 Subrings of Lie p-rings

Let L be a Lie p-ring defined via structure constants. Then each subring U of L is a Lie ring and it has p-power order as well. Hence it is also a Lie p-ring and thus it has a basis $(u_1, \ldots, u_m)$. Suppose that L has order p^n and let $(l_1, \ldots, l_n)$ denote its natural basis corresponding to its defining structure constants. Then each u_i can be expressed as a linear combination $u_i = m_{i,1}l_1 + \ldots + m_{i,n}l_n$ with $m_{i,j} \in \{0, \ldots, p-1\}$. Let $M = (m_{i,j})$ denote the matrix of coefficients. Then we say that $(u_1, \ldots, u_m)$ is *induced* if M is in upper triangular form. We say that $(u_1, \ldots, u_m)$ is *canonical* if M is in upper echelon form; that is, it is upper triangular, each row in M has leading entry 1 and there are 0's above each leading entry.

1 ► `LiePSubring(L, gens)`

returns the subring of L generated by *gens*. This function computes a canonical basis for the subring. Note that this function also works for generic Lie p-rings L, but there may be strange effects in this case. The following example shows that.

```
gap> L := LiePRingsByLibrary(6)[100];
<Lie ring of dimension 6 over prime p>
gap> l := BasisOfLiePRing(L);
[ l1, l2, l3, l4, l5, l6 ]
gap> U := LiePSubring(L, [5*l[1]]);
WARNING: Multiplying by 1/5
<Lie ring of dimension 3 over prime p>
gap> BasisOfLiePRing(U);
[ l1, l4, l6 ]
gap>
gap> K := SpecialisePrimeOfLiePRing(L, 5);
<Lie ring of dimension 6 over prime 5>
gap> b := BasisOfLiePRing(K);
[ l1, l2, l3, l4, l5, l6 ]
gap> LiePSubring(K, [5*b[1]]);
<Lie ring of dimension 2 over prime 5>
gap> BasisOfLiePRing(last);
[ l4, l6 ]
gap>
gap> K := SpecialisePrimeOfLiePRing(L, 7);
<Lie ring of dimension 6 over prime 7>
gap> b := BasisOfLiePRing(K);
[ l1, l2, l3, l4, l5, l6 ]
gap> U := LiePSubring(L, [5*b[1]]);
<Lie ring of dimension 1 over prime p>
gap> BasisOfLiePRing(U);
[ l1 + 2*l4 ]
```

2 ► `LiePIdeal(L, gens)`

return the ideal of L generated by *gens*. This function computes a an induced basis for the ideal.

```
gap> LiePIdeal(L, [l[1]]);
<Lie ring of dimension 5 over prime p>
gap> BasisOfLiePRing(last);
[ l1, l3, l4, l5, l6 ]
```

3 ► `LiePQuotient(L, U)`

return a Lie p-ring isomorphic to L/U where U must be an ideal of L. This function requires that L is an ordinary Lie p-ring.

```
gap> LiePIdeal(K, [b[1]]);
<Lie ring of dimension 5 over prime 5>
gap> LiePIdeal(K, [b[2]]);
<Lie ring of dimension 4 over prime 5>
gap> LiePQuotient(K,last);
<Lie ring of dimension 2 over prime 5>
```

2.5 Elementary functions for Lie p-rings and their subrings

The following functions work for ordinary and generic Lie p-rings L and their subrings.

1 ► `PrimeOfLiePRing(L)`

returns the underlying prime. This can either be an integer or an indeterminate.

2 ► `BasisOfLiePRing(L)`

returns a basis for L.

3 ► `DimensionOfLiePRing(L)`

returns n where L has order p^n.

4 ► `ParametersOfLiePRing(L)`

returns the list of indeterminates involved in L. If L is a subring of a Lie p-ring defined by structure constants, then the parameters of the parent are returned.

5 ► `ViewPCPresentation(L)`

prints the presentation for L with respect to its basis.

2.6 Series for Lie p-rings and their subrings

1 ► `LiePLowerCentralSeries(L)`

returns the lower central series of L.

2 ► `LiePLowerPCentralSeries(L)`

returns the lower exponent-p central series of L.

3 ► `LiePDerivedSeries(L)`

returns the derived series of L.

4 ► `LiePMinimalGeneratingSet(L)`

returns a minimal generating set of L.

2.7 The Lazard correspondence

The following function has been implemented by Willem de Graaf. It uses the Baker-Campbell-Hausdorff formula as described in [CdGVL12] and it is based on the Liering package [CdG10].

1 ► `PGroupByLiePRing(L)`

returns the p-group G obtained from L via the Lazard correspondence. This function requires that L is an ordinary Lie p-ring with $cl(L) < p$.

3 The Database

In this chapter we describe functions to access the database of Lie p-rings of dimension at most 7. Throughout, we assume that $dim \in \{1, \ldots, 7\}$ and P is a prime with $P \neq 2$.

3.1 Numbers of Lie p-rings

1 ▶ `NumberOfLiePRings( dim )`

returns the number of generic Lie p-rings in the database of the considered dimension. This is available for $dim \leq 7$.

```
gap> List([1..7], x -> NumberOfLiePRings(x));
[ 1, 2, 5, 15, 75, 542, 4773 ]
```

2 ▶ `NumberOfLiePRings( dim, P )`

returns the number of isomorphism types of Lie p-rings of order $P^d im$ in the database. If $P \geq 5$, then this is the number of all isomorphism types of Lie p-rings of order $P^d im$ and if $P = 3$ then this is the number of all isomorphism types of Lie p-rings of p-class at most 2. If $P \geq 7$, then this number coincides with NumberSmallGroups(dim^P).

3.2 Accessing Lie p-rings

1 ▶ `LiePRingsByLibrary( dim )`
▶ `LiePRingsByLibrary( dim, gen, cl )`

returns the generic Lie p-rings of dimension dim in the database. The second form returns the Lie p-rings of minimal generator number gen and p-class cl only.

2 ▶ `LiePRingsByLibrary( dim, P )`
▶ `LiePRingsByLibrary( dim, P, gen, cl )`

returns isomorphism type representatives of ordinary Lie p-rings of dimension dim for the prime P. The second form returns the Lie p-rings of minimal generator number gen and p-class cl only. The function assumes $P \geq 3$ and for $P = 3$ there are only the Lie p-rings of p-class at most 2 available.

The first example yields the generic Lie p-rings of dimension 4.

```
gap> LiePRingsByLibrary(4);
[ <Lie ring of dimension 4 over prime p>,
  <Lie ring of dimension 4 over prime p>,
  <Lie ring of dimension 4 over prime p>,
  <Lie ring of dimension 4 over prime p>,
  <Lie ring of dimension 4 over prime p>,
  <Lie ring of dimension 4 over prime p>,
  <Lie ring of dimension 4 over prime p>,
  <Lie ring of dimension 4 over prime p with parameters [ w ]>,
  <Lie ring of dimension 4 over prime p>,
```

```
  <Lie ring of dimension 4 over prime p>,
  <Lie ring of dimension 4 over prime p>,
  <Lie ring of dimension 4 over prime p>,
  <Lie ring of dimension 4 over prime p>,
  <Lie ring of dimension 4 over prime p>,
  <Lie ring of dimension 4 over prime p> ]
```

The next example yields the isomorphism type representatives of Lie p-rings of dimension 3 for the prime 5.

```
gap> LiePRingsByLibrary(3, 5);
[ <Lie ring of dimension 3 over prime 5>,
  <Lie ring of dimension 3 over prime 5>,
  <Lie ring of dimension 3 over prime 5>,
  <Lie ring of dimension 3 over prime 5>,
  <Lie ring of dimension 3 over prime 5> ]
```

The following example extracts the generic Lie p-rings of dimension 5 with minimal generator number 2 and p-class 4.

```
gap> LiePRingsByLibrary(5, 2, 4);
[ <Lie ring of dimension 5 over prime p>,
  <Lie ring of dimension 5 over prime p>,
  <Lie ring of dimension 5 over prime p>,
  <Lie ring of dimension 5 over prime p with parameters [ w ]>,
  <Lie ring of dimension 5 over prime p with parameters [ w ]>,
  <Lie ring of dimension 5 over prime p>,
  <Lie ring of dimension 5 over prime p>,
  <Lie ring of dimension 5 over prime p with parameters [ w ]>,
  <Lie ring of dimension 5 over prime p with parameters [ w ]>,
  <Lie ring of dimension 5 over prime p with parameters [ w ]>,
  <Lie ring of dimension 5 over prime p>,
  <Lie ring of dimension 5 over prime p with parameters [ w ]>,
  <Lie ring of dimension 5 over prime p with parameters [ w ]>,
  <Lie ring of dimension 5 over prime p>,
  <Lie ring of dimension 5 over prime p> ]
```

Finally, we determine the isomorphism type representatives of Lie p-rings of dimension 5, minimal generator number 2 and p-class 4 for the prime 7.

```
gap> LiePRingsByLibrary(5, 7, 2, 4);
[ <Lie ring of dimension 5 over prime 7>,
  <Lie ring of dimension 5 over prime 7>,
  <Lie ring of dimension 5 over prime 7>,
  <Lie ring of dimension 5 over prime 7>,
  <Lie ring of dimension 5 over prime 7>,
  <Lie ring of dimension 5 over prime 7>,
  <Lie ring of dimension 5 over prime 7>,
  <Lie ring of dimension 5 over prime 7>,
  <Lie ring of dimension 5 over prime 7>,
  <Lie ring of dimension 5 over prime 7>,
  <Lie ring of dimension 5 over prime 7>,
  <Lie ring of dimension 5 over prime 7>,
  <Lie ring of dimension 5 over prime 7> ]
```

3.3 Searching the database

The functions in the previous section give access to the database. In this section we give some more detailed information.

1 ► `LiePRingsInFamily( L, P )`

takes as input a generic Lie p-ring L from the database and a prime P and returns all Lie p-rings determined by L and P up to isomorphism. This function returns fail if the generic Lie p-ring does not exist for the special prime P; this may be due to the conditions on the prime or (if $P = 3$) to the p-class of the Lie p-ring.

```
gap> L := LiePRingsByLibrary(7)[118];
<Lie ring of dimension 7 over prime p with parameters [ x, y ]>
gap> LibraryConditions(L);
[ "all x,y, y~-y", "p=1 mod 4" ]
gap> LiePRingsInFamily(L,3);
fail
gap> Length(LiePRingsInFamily(L,5));
15
gap> LiePRingsInFamily(L, 7);
fail
gap> Length(LiePRingsInFamily(L,13));
91
gap> 13^2;
169
```

The following example shows how to determine all Lie p-rings of dimension 5 and p-class 4 over the prime 29 up to isomorphism.

```
gap> L := LiePRingsByLibrary(5);;
gap> L := Filtered(L, x -> PClassOfLiePRing(x)=4);
[ <Lie ring of dimension 5 over prime p>,
  <Lie ring of dimension 5 over prime p>,
  <Lie ring of dimension 5 over prime p>,
  <Lie ring of dimension 5 over prime p with parameters [ w ]>,
  <Lie ring of dimension 5 over prime p with parameters [ w ]>,
  <Lie ring of dimension 5 over prime p>,
  <Lie ring of dimension 5 over prime p>,
  <Lie ring of dimension 5 over prime p with parameters [ w ]>,
  <Lie ring of dimension 5 over prime p with parameters [ w ]>,
  <Lie ring of dimension 5 over prime p with parameters [ w ]>,
  <Lie ring of dimension 5 over prime p>,
  <Lie ring of dimension 5 over prime p with parameters [ w ]>,
  <Lie ring of dimension 5 over prime p with parameters [ w ]>,
  <Lie ring of dimension 5 over prime p>,
  <Lie ring of dimension 5 over prime p> ]
gap> K := List(L, x-> LiePRingsInFamily(x, 29));
[ [ <Lie ring of dimension 5 over prime 29> ],
  [ <Lie ring of dimension 5 over prime 29> ],
  [ <Lie ring of dimension 5 over prime 29> ], fail, fail,
  [ <Lie ring of dimension 5 over prime 29> ],
  [ <Lie ring of dimension 5 over prime 29> ],
  [ <Lie ring of dimension 5 over prime 29> ],
  [ <Lie ring of dimension 5 over prime 29> ],
```

```
  [ <Lie ring of dimension 5 over prime 29> ],
  [ <Lie ring of dimension 5 over prime 29> ], fail, fail,
  [ <Lie ring of dimension 5 over prime 29> ],
  [ <Lie ring of dimension 5 over prime 29> ] ]
gap> K := Filtered(Flat(K), x -> x<>fail);
[ <Lie ring of dimension 5 over prime 29>,
  <Lie ring of dimension 5 over prime 29>,
  <Lie ring of dimension 5 over prime 29>,
  <Lie ring of dimension 5 over prime 29>,
  <Lie ring of dimension 5 over prime 29>,
  <Lie ring of dimension 5 over prime 29>,
  <Lie ring of dimension 5 over prime 29>,
  <Lie ring of dimension 5 over prime 29>,
  <Lie ring of dimension 5 over prime 29>,
  <Lie ring of dimension 5 over prime 29>,
  <Lie ring of dimension 5 over prime 29> ]
```

3.4 More details

Let L be a Lie p-ring from the database. Then the following additional attributes are available.

1 ► `LibraryName(L)`

returns a string with the name of L in the database. See p567.pdf for further background.

2 ► `ShortPresentation(L)`

returns a string exhibiting a short presentation of L.

3 ► `LibraryConditions(L)`

returns the conditions on L. This is a list of two strings. The first string exhibits the conditions on the parameters of L, the second shows the conditions on primes.

4 ► `MinimalGeneratorNumberOfLiePRing(L)`

returns the minimial generator number of L.

5 ► `PClassOfLiePRing(L)`

returns the p-class of L.

```
gap> L := LiePRingsByLibrary(7)[118];
<Lie ring of dimension 7 over prime p with parameters [ x, y ]>
gap> LibraryName(L);
"7.118"
gap> LibraryConditions(L);
[ "all x,y, y~-y", "p=1 mod 4" ]
```

All of the information liested in this section is inherited when L is specialised.

```
gap> L := LiePRingsByLibrary(7)[118];
<Lie ring of dimension 7 over prime p with parameters [ x, y ]>
gap> K := SpecialiseLiePRing(L, 5, ParametersOfLiePRing(L), [0,0]);
<Lie ring of dimension 7 over prime 5>
gap> LibraryName(K);
"7.118"
gap> LibraryConditions(K);
[ "all x,y, y~-y", "p=1 mod 4" ]
```

The following example shows how to find a Lie p-ring with a given name in the database.

```
gap> L := LiePRingsByLibrary(7);;
gap> Filtered(L, x -> LibraryName(x) = "7.1010")[1];
<Lie ring of dimension 7 over prime p>
```

3.5 Special functions for dimension 7

The database of Lie p-rings of dimension 7 is very large and it may be time-consuming (or even impossible due to storage problems) to generate all Lie p-rings of dimension 7 for a given prime P.

Thus there are some special functions available that can be used to access a particular set of Lie p-rings of dimension 7 only. In particular, it is possible to consider the descendants of a single Lie p-ring of smaller dimension by itself. The Lie p-rings of this type are all stored in one file of the library. Thus, equivalently, it is possible to access the Lie p-rings in one single file only.

The table LIE_TABLE contains a list of all possible files together with the number of Lie p-rings generated by their corresponding Lie p-rings.

1 ► `LiePRingsDim7ByFile( nr )`

returns the generic Lie p-rings in file number *nr*.

2 ► `LiePRingsDim7ByFile( nr, P )`

returns the isomorphism types of Lie p-rings in file number *nr* for the prime P.

```
gap> LIE_TABLE[100];
[ "3gen/gapdec6.139", 1/2*p+g3+3/2 ]
gap> LiePRingsDim7ByFile(100);
[ <Lie ring of dimension 7 over prime p>,
  <Lie ring of dimension 7 over prime p>,
  <Lie ring of dimension 7 over prime p with parameters [ w ]>,
  <Lie ring of dimension 7 over prime p with parameters [ w ]>,
  <Lie ring of dimension 7 over prime p with parameters [ x ]> ]
gap> LiePRingsDim7ByFile(100, 7);
[ <Lie ring of dimension 7 over prime 7>,
  <Lie ring of dimension 7 over prime 7>,
  <Lie ring of dimension 7 over prime 7>,
  <Lie ring of dimension 7 over prime 7>,
  <Lie ring of dimension 7 over prime 7>,
  <Lie ring of dimension 7 over prime 7>,
  <Lie ring of dimension 7 over prime 7>,
  <Lie ring of dimension 7 over prime 7> ]
```

Bibliography

[CdG10] Serena Cicalò and Willem A. de Graaf. *Liering*, 2010. A GAP 4 package.

[CdGVL12] Serena Cicalò, Willem A. de Graaf, and Michael Vaughan-Lee. An effective version of the Lazard correspondence. *J. Algebra*, 352:430–450, 2012.

[NOVL03] Mike F. Newman, Eamonn A. O'Brien, and Michael R. Vaughan-Lee. Groups and nilpotent Lie rings whose order is the sixth power of a prime. *J. Alg.*, 278:383 – 401, 2003.

[OVL05] E. A. O'Brien and M. R. Vaughan-Lee. The groups with order p^7 for odd prime p. *J. Algebra*, 292(1):243–258, 2005.