Obtaining combinatorial structures associated with low-dimensional Leibniz algebras

Manuel Ceballos, Juan Núñez
University of Seville (Spain)

Ángel F. Tenorio
Pablo de Olavide University (Spain)

mceballos@us.es

Abstract

In this paper, we analyze the relation between Leibniz algebras and combinatorial structures. More concretely, we study the properties to be satisfied by (pseudo)digraphs so that they are associated with low-dimensional Leibniz algebras. We present some results related to this association and show an algorithmic method to obtain them, which has been implemented with Maple.

Keywords

Pseudodigraph, Combinatorial structure, Leibniz algebra, Structure Theory, Algorithm

1 Introduction

Leibniz algebras were introduced at the beginning of the 1990s by J.-L. Loday [3]. They are a particular case of non-associative algebras and provide a non-commutative generalization of Lie algebras. There exists extensive research on these algebras due to their many applications in Engineering, Physics and Applied Mathematics. However, some aspects of Leibniz algebras remain unknown. In fact, the classification of nilpotent and solvable algebras is still an open problem.

Graph Theory is also very important and useful due to its many uses as a tool for other subjects. Our main goal is to extend the study and analysis of the relations between Graph Theory and Lie algebras proposed in [1, 2], but this time to the case of Leibniz algebras.

2 Preliminaries

We show some preliminary concepts on Leibniz algebras, bearing in mind that the reader can consult [3] as an introductory paper.

Definition 1 *A* Leibniz algebra $\mathcal{L}$ *over a field* $\mathbb{K}$ *is a vector space with a second inner bilinear composition law* $[\cdot,\cdot]$*, which verifies the so-called Leibniz identity*

$$[[X,Y],Z]-[[X,Z],Y]-[X,[Y,Z]]=0,\ \forall\, X,Y,Z\in\mathcal{L}$$

¿From now on, we will denote $L(X,Y,Z)=[[X,Y],Z]-[[X,Z],Y]-[X,[Y,Z]]$.

If, in addition, is verified that $[X,X]=0$*, for all* $X\in\mathcal{L}$*, the Leibniz algebra is also a* Lie algebra*. In this case, it is satisfied that* $[X,Y]=-[Y,X]$ *and the Leibniz identity is equivalent to the Jacobi identity.*

Definition 2 *Given a basis* $\{e_i\}_{i=1}^n$ *of an* n*-dimensional Leibniz algebra* $\mathcal{L}$*, its* structure constants *are defined by* $[e_i,e_j]=\sum_{h=1}^n c_{i,j}^h e_h$*, for* $1\le i,j\le n$.

Definition 3 *The* derived *and* central series *of a finite-dimensional Leibniz algebra* $\mathcal{L}$ *are*

$$\mathcal{L}_1=\mathcal{L},\ \mathcal{L}_2=[\mathcal{L},\mathcal{L}],\ \dots,\ \mathcal{L}_k=[\mathcal{L}_{k-1},\mathcal{L}_{k-1}],\ \dots \quad\text{and}\quad \mathcal{L}^1=\mathcal{L},\ \mathcal{L}^2=[\mathcal{L},\mathcal{L}],\ \dots,\ \mathcal{L}^k=[\mathcal{L}^{k-1},\mathcal{L}],\ \dots$$

So, $\mathcal{L}$ *is called* $(m-1)$*-step* solvable *(resp.* nilpotent*) if there exists* $m\in\mathbb{N}$ *such that* $\mathcal{L}_m=\{0\}$ *and* $\mathcal{L}_{m-1}\neq\{0\}$ *(resp.* $\mathcal{L}^m=\{0\}$ *and* $\mathcal{L}^{m-1}\neq\{0\}$*).*

3 Associating combinatorial structures with Leibniz algebras

Let $\mathcal{L}$ be a n-dimensional Leibniz algebra with basis $\mathcal{B} = \{e_i\}_{i=1}^n$. Its structure constants correspond to $[e_i, e_j] = \sum_{h=1}^n c_{i,j}^h e_h$ and, hence, the pair $(\mathcal{L}, \mathcal{B})$ is associated with a combinatorial structure by the following procedure

a) For each $e_i \in \mathcal{B}$, we draw a vertex i.

b) For every vertex i verifying $[e_i, e_i] \neq 0$, we draw a loop such that its weight is an n-tuple given by $(c_{i,i}^1, c_{i,i}^2, \ldots, c_{i,i}^n)$.

c) Given two vertices i, j verifying $(c_{i,j}^j, c_{j,i}^j) \neq (0, 0)$, we draw a directed edge from vertex i to j whose weight is given by the pair $(c_{i,j}^j, c_{j,i}^j)$.

d) Given three vertices $i < j < k$ such that $(c_{i,j}^k, c_{j,i}^k, c_{j,k}^i, c_{k,j}^i, c_{i,k}^j, c_{k,i}^j) \neq (0, 0, 0, 0, 0, 0)$, we draw a full triangle ijk such that the edges ij, jk and ik have weights $(c_{i,j}^k, c_{j,i}^k)$, $(c_{j,k}^i, c_{k,j}^i)$ and $(c_{i,k}^j, c_{k,i}^j)$, respectively. Moreover,

  d1) we use a discontinuous line (named *ghost edge*) for edges with weight $(0, 0)$.

  d2) If two triangles ijk and ijl satisfy $(c_{i,j}^k, c_{j,i}^k) = (c_{i,j}^l, c_{j,i}^l)$, draw only one edge between vertices i and j shared by both triangles.

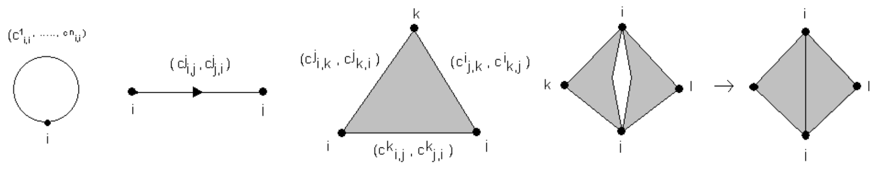

Figure 1: Loop, directed edge, full triangle and two triangles sharing an edge.

4 Leibniz algebras and (pseudo)digraphs

In this section, we study the structure of digraphs associated with low-dimensional Leibniz algebras. For each case, we will study the type of Leibniz algebra according to the solvability of this algebra. To be associated with a (pseudo)digraph G, a given Leibniz algebra $\mathcal{L}$ with basis $\mathcal{B} = \{e_i\}_{i=1}^n$ has the following law

$$[e_i, e_j] = c_{i,j}^i e_i + c_{i,j}^j e_j, \ 1 \leq i \neq j \leq n; \quad [e_k, e_k] = \sum_{h=1}^n c_{k,k}^h e_h \tag{1}$$

since these brackets avoid the appearance of full triangles in G.

Proposition 1 *Every digraph admitting some configuration of [1, Fig. 9] as a subdigraph is not associated with any Leibniz algebra.*

Proposition 2 *The abelian Leibinz algebra is the only one of dimension 1, associated with a digraph.*

Proposition 3 *Let $\mathcal{L}$ be a 2-dimensional Leibniz algebra associated with a connected pseudodigraph G. Then, the configuration d) shown in Figure 2 is forbidden in G. In fact, G must present one of the remaining configurations in that figure. Moreover, it is verified that*

- *Configurations a) and c) are associated with 2-step solvable non-nilpotent Leibniz algebras.*

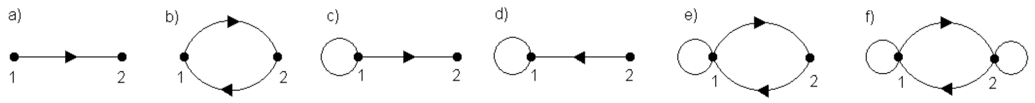

Figure 2: Pseudodigraphs with two vertices and associated with Leibniz algebras.

- *Configuration b) is always associated with 2-step solvable non-nilpotent Lie algebras (i.e. only commutative Leibniz algebras).*
- *Configurations e) and f) are associated with a 2-step nilpotent Leibniz algebras.*

Example 1 *Let $\mathcal{L}$ be the Leibniz algebra with brackets $[e_2, e_1] = e_2$ associated with Configuration a). In this case, $\mathcal{L}_2 = \mathcal{L}^2 = \langle e_2 \rangle$, whereas $\mathcal{L}^i = \mathcal{L}_2$ and $\mathcal{L}_i = 0$, for all $i \geq 3$. Therefore, $\mathcal{L}$ is 2-step solvable, non-nilpotent.*

Example 2 *We consider the Leibniz algebra $\mathcal{L}$ with law $[e_1, e_1] = -e_1 - e_2$, $[e_1, e_2] = e_1 + e_2$ associated with Configuration e). In this case, $\mathcal{L}^2 = \mathcal{L}_2 = \langle e_1 + e_2 \rangle$ and $\mathcal{L}_3 = \mathcal{L}^3 = 0$. Hence, $\mathcal{L}$ is 2-step nilpotent.*

Example 3 *Let $\mathcal{L}$ be the Leibniz algebra with brackets $[e_1, e_1] = [e_2, e_2] = -e_1 - e_2, [e_1, e_2] = [e_2, e_1] = e_1 + e_2$, associated with Configuration f). For this algebra, $\mathcal{L}_2 = \langle e_1 + e_2 \rangle$, $\mathcal{L}_3 = \mathcal{L}^3 = \{0\}$. So, $\mathcal{L}$ is 2-step nilpotent.*

Proposition 4 *Let $\mathcal{L}$ be a 3-dimensional Leibniz algebra associated with a connected pseudograph G including some loop. Then, G must present one of the configurations in Figure 3 up to permutation of labels. Any other pseudodigraph is forbidden in G.*

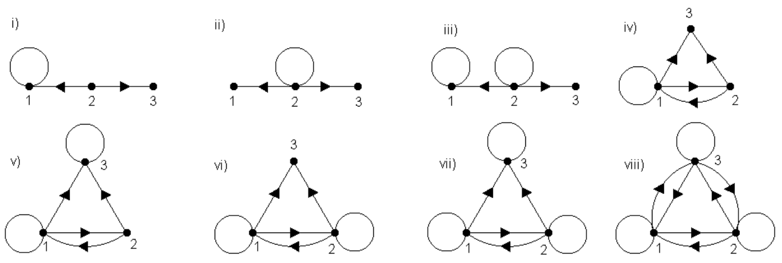

Figure 3: Pseudodigraphs with three vertices, associated with Leibniz algebras, not being Lie algebras.

Proposition 5 *Let G be a pseudodigraph formed by the first configuration of [1, Fig. 15] with loops. Then G is associated with a solvable non-nilpotent Leibniz algebra.*

Proposition 6 *Let G be a pseudodigraph formed by the second configuration of [1, Fig. 15] with loops. Then, G is associated with a Leibniz algebra if and only if G has a loop on each vertex incident with a double edge.*

5 Implementation and complexityfor the Leibniz identity

Now, we show the algorithmic method that we have used in the previous section to evaluate the Leibniz identity in order to find out the allowed and forbidden configurations and the restrictions over the weights of the edges. Regarding this, we have implemented our algorithm using the symbolic computation package MAPLE, working the implementation in version 12 or higher. To do this, we will use the libraries `linalg`, `combinat`, `GraphTheory` and `Maplets[Elements]` to activate commands related to Linear and Combinatorial Algebra, Graph Theory and the last one to display a message so that the user introduces the required input in the first subprocedure. So, we start considering a vector space $\mathcal{L}$ with basis $\mathcal{B}$ and the type of brackets expressed in (1) and give the following steps:

1. Computing the bracket product between two arbitrary basis vectors in $\mathcal{B}$.

   This first subprocedure is called `law` and computes the bracket between two arbitrary basis vectors in $\mathcal{B}$. It receives the subindexes of two basis vectors in $\mathcal{B}$. A conditional sentence is introduced to determine each non-zero bracket. The user has to complete the implementation depending on the law of $\mathcal{L}$, so we have added a sentence at the beginning of the implementation, reminding of this fact. Before running any other sentence, we restart all the variables by using the command `restart`. Moreover, we save the value of variable `dim` (the dimension) with the command `assign`.

```
> maplet:=Maplet(AlertDialog("Don't forget to introduce non-zero brackets and the dimension in
subprocedure law",'onapprove'=Shutdown("Continue"),'oncancel'=Shutdown("Aborted"))):
> Maplets[Display](maplet):
> assign(dim,...):
> law:=proc(i,j)
>   if (i,j)=... then ...;
>   elif ....
>   else 0; end if;
> end proc;
```

2. Evaluating the bracket between two vectors expressed as a linear combination of vectors from basis $\mathcal{B}$.

   We implement the subprocedure called `bracket` to compute the product between two arbitrary vectors of $\mathcal{L}$, which are expressed as linear combinations of the vectors in $\mathcal{B}$. The subprocedure `law` is called in the implementation.

```
> bracket:=proc(u,v,n)
>   local exp; exp:=0;
>   for i from 1 to n do
>      for j from 1 to n do
>         exp:=exp + coeff(u,e[i])*coeff(v,e[j])*law(i,j);
>      end do;
>   end do;
>   exp;
> end proc:
```

3. Imposing the Leibniz identity and solving the corresponding system of equations.

   Next, we show the implementation of the main procedure called `Leibniz`, which checks if the vector space $\mathcal{L}$ is or not a Leibniz algebra. This procedure receives as input the dimension n of the vector space $\mathcal{L}$ and returns the solution of a system of equations obtained from imposing the Leibniz identity in $\mathcal{L}$. If the system has no solution, then we can conclude that the vector space $\mathcal{L}$ is not a Leibniz algebra. Otherwise, we will obtain the conditions over the structure constants $c_{i,j}^k$ so that $\mathcal{L}$ is a Leibniz algebra.

```
> Leibniz:=proc(n)
>   local L,M,N,P;
>   L:=[];M:=[];N:=[];P:=[];
>   for i from 1 to n do
>     L:=[op(L),i,i,i];
>   end do;
>   M:=permute(L,3);
>   for j from 1 to nops(M) do
>     eq[j]:=bracket(bracket(e[M[j][1]],e[M[j][2]],n),e[M[j][3]],n)-
>            bracket(bracket(e[M[j][1]],e[M[j][3]],n),e[M[j][2]],n)-
>            bracket(e[M[j][1]],bracket(e[M[j][2]],e[M[j][3]],n),n);
>   end do;
>   N:=[seq(eq[k], k=1..nops(M))];
>   for k from 1 to nops(N) do
>      for h from 1 to n do
>        P:=[op(P),coeff(N[k],e[h])=0];
>      end do;
>   end do;
>   solve(P);
> end proc:
```

Example 4 *Now, we show an example with the configuration i) from Figure 3. We consider the 3-dimensional vector space $\mathcal{L}$ with brackets*

$$[e_1, e_1] = \sum_{i=1}^{3} c_{1,1}^{i} e_i;\ [e_j, e_2] = c_{j,2}^{j} e_j,\ [e_2, e_j] = c_{2,j}^{j} e_j,\ \text{for}\, j = 1, 3$$

First, we have to complete the implementation of the subprocedure `law` *as follows*

```
> maplet:=Maplet(AlertDialog("Don't forget to introduce non-zero brackets and the dimension in
subprocedure law",'onapprove'=Shutdown("Continue"),'oncancel'=Shutdown("Aborted"))):
> Maplets[Display](maplet):
> assign(dim,3):
> law:=proc(i,j)
>   if (i,j)=(1,1) then c111*e[1]+c112*e[2]+c113*e[3];
>   elif (i,j)=(1,2) then c121*e[1];
>   elif (i,j)=(2,1) then c211*e[1];
>   elif (i,j)=(2,3) then c233*e[3];
>   elif (i,j)=(3,2) then c323*e[3];
>     else 0;
>   end if;
> end proc:
```

After that, we must run the subprocedure `bracket` *and the procedure* `Leibniz`*. Now, we evaluate this main procedure over the variable* `dim`

```
> Leibniz(dim);
>      {c111=0,c112=0,c113=c113,c121=-c211,c211=c211,c233=0,c323=-2*c211}
```

So, we obtain those restrictions for the weights of the edges in configuration i) from Figure 3.

Next, we compute the complexity of the algorithm. To do so, we consider the number of operations carried out in the worst case. We use the big O notation to express the complexity. To recall the big O notation, the reader can consult [4]: given two functions $f, g : \mathbb{R} \to \mathbb{R}$, we could say that $f(x) = O(g(x))$ if and only if there exist $M \in \mathbb{R}^+$ and $x_0 \in \mathbb{R}$ such that $|f(x)| < M \cdot |g(x)|$, for all $x > x_0$.

We denote by $N_i(n)$ the number of operations when considering the step i. This function depends on the dimension n of the Lie algebra. Table 1 shows the number of computations and the complexity of each step, as well as indicating the name of the procedure corresponding to each step.

Table 1: Complexity and number of operations.

Step	Procedure	Complexity	Operations
1	`law`	$O(n^2)$	$N_1(n) = O\left(\frac{n(n-1)}{2}\right)$
2	`bracket`	$O(n^4)$	$N_2(n) = \sum_{i=1}^{n}\sum_{j=1}^{n} N_1(n)$
3	`Leibniz`	$O(n^7)$	$N_3(n) = O(n) + O(n^3) + \sum_{i=1}^{n^3} N_2(n) + \sum_{j=1}^{n^3}\sum_{k=1}^{n} 1$

References

[1] A. Carriazo, L.M. Fernández, J. Núñez, Combinatorial structures associated with Lie algebras of finite dimension, Linear Algebra Appl. 389 (2004), 43–61.

[2] J. Cáceres, M. Ceballos, J. Núñez, M.L. Puertas, A.F. Tenorio, Combinatorial structures of three vertices and Lie algebras, Int. J. Computer Math. 89:13–14 (2012), 1879–1900.

[3] J.L. Loday, Une version non commutative des algèbres de Lie: les algèbres de Leibniz, Enseign. Math. (2), 39 (1993), pp. 269–293.

[4] H.S. Wilf, Algorithms and Complexity, Prentice Hall, Englewood Cliffs, 1986.