

# ***Algorithm to find S-related Lie algebras***

*ACAT 2017*

**Carlos Inostroza**

***Universidad de Concepción,  
Concepción - Chile***

***Collaboration with : I. Kondrashuk, N. Merino and F. Nadal***

***Based on : arXiv:1703.04036, and a work to be submitted soon***

# Contents

- 1) Description of the Java library**
- 2) Algorithm to find S-related Lie algebras**
- 3) Applications**
- 4) Final remarks**



# Contents

- 1) Description of the Java library**
- 2) Algorithm to find S-related Lie algebras
- 3) Applications
- 4) Final remarks

# Description of the Java library

As a preliminary step, we have used the program *gen.f* of Ref. [2] to generate the files *sem.2*, *sem.3*, *sem.4*, *sem.5* and *sem.6*, which contain all the semigroups up to order 6.

[2]

HANDBOOK OF FINITE SEMIGROUP PROGRAMS	
John A Hildebrant	
TABLE OF CONTENTS	
TABLE OF CONTENTS . . . . .	1
FORWARD . . . . .	2
SEMIGROUP GENERATING PROGRAM	
 <i>gen.f</i> . . . . .	3
COMMUTATIVE SEMIGROUP	
SORTING PROGRAM	
 <i>com.f</i> . . . . .	10

For example, for  $n=2$  the program *com.f* gives:

$S^1_{(2)}$	$\lambda_1$	$\lambda_2$		$S^2_{(2)}$	$\lambda_1$	$\lambda_2$		$S^4_{(2)}$	$\lambda_1$	$\lambda_2$
$\lambda_1$	$\lambda_1$	$\lambda_1$	,	$\lambda_1$	$\lambda_1$	$\lambda_1$	,	$\lambda_1$	$\lambda_1$	$\lambda_2$
$\lambda_2$	$\lambda_1$	$\lambda_1$		$\lambda_2$	$\lambda_1$	$\lambda_2$		$\lambda_2$	$\lambda_2$	$\lambda_1$

**Table 2.** List of semigroups of the third rank.

$S_{(3)}^1$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$S_{(3)}^2$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$S_{(3)}^3$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$S_{(3)}^6$	$\lambda_1$	$\lambda_2$	$\lambda_3$
$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$
$\lambda_2$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_2$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_2$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_2$	$\lambda_1$	$\lambda_1$	$\lambda_2$
$\lambda_3$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_3$	$\lambda_1$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_1$	$\lambda_1$	$\lambda_3$	$\lambda_3$	$\lambda_1$	$\lambda_2$	$\lambda_3$
$S_{(3)}^7$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$S_{(3)}^9$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$S_{(3)}^{10}$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$S_{(3)}^{12}$	$\lambda_1$	$\lambda_2$	$\lambda_3$
$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$
$\lambda_2$	$\lambda_1$	$\lambda_2$	$\lambda_1$	$\lambda_2$	$\lambda_1$	$\lambda_2$	$\lambda_2$	$\lambda_2$	$\lambda_1$	$\lambda_2$	$\lambda_2$	$\lambda_2$	$\lambda_1$	$\lambda_2$	$\lambda_3$
$\lambda_3$	$\lambda_1$	$\lambda_1$	$\lambda_3$	$\lambda_3$	$\lambda_1$	$\lambda_2$	$\lambda_2$	$\lambda_3$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_3$	$\lambda_1$	$\lambda_3$	$\lambda_2$
$S_{(3)}^{15}$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$S_{(3)}^{16}$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$S_{(3)}^{17}$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$S_{(3)}^{18}$	$\lambda_1$	$\lambda_2$	$\lambda_3$
$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_3$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_3$	$\lambda_1$	$\lambda_1$	$\lambda_2$	$\lambda_2$	$\lambda_1$	$\lambda_1$	$\lambda_2$	$\lambda_3$
$\lambda_2$	$\lambda_1$	$\lambda_1$	$\lambda_3$	$\lambda_2$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_2$	$\lambda_2$	$\lambda_1$	$\lambda_1$	$\lambda_2$	$\lambda_2$	$\lambda_3$	$\lambda_1$
$\lambda_3$	$\lambda_3$	$\lambda_3$	$\lambda_1$	$\lambda_3$	$\lambda_3$	$\lambda_3$	$\lambda_1$	$\lambda_3$	$\lambda_2$	$\lambda_1$	$\lambda_1$	$\lambda_3$	$\lambda_3$	$\lambda_1$	$\lambda_2$

# About installation instructions

To use the Java library that we present in this work it is necessary to download the linear algebra package `jama.jar` from [3].

## ***JAMA : A Java Matrix Package***

[\[ Background \]](#) ..... [\[ The Package \]](#) ..... [\[ Request for Comments \]](#) ..... [\[ Authors \]](#) ..... [\[ Related Links & Libraries \]](#)

---

### Background

JAMA is a basic linear algebra package for Java. It provides user-level classes for constructing and manipulating real, dense matrices. It is meant to provide sufficient functionality for routine problems, packaged in a way that is natural and understandable to non-experts. It is intended to serve as *the* standard matrix class for Java, and will be proposed as such to the [Java Grande Forum](#) and then to [Sun](#). A straightforward public-domain reference implementation has been developed by the [MathWorks](#) and [NIST](#) as a strawman for such a class. We are releasing this version in order to obtain public comment. There is no guarantee that future versions of JAMA will be compatible with this one.

A sibling matrix package, [Jampack](#), has also been developed at NIST and the University of Maryland. The two packages arose from the need to evaluate alternate designs for the implementation of matrices in Java. JAMA is based on a single matrix class within a strictly object-oriented framework. Jampack uses a more open approach that lends itself to extension by the user. As it turns out, for the casual user the packages differ principally in the syntax of the matrix operations. We hope you will take the time to look at Jampack along with JAMA. There is much to be learned from both packages.

**Capabilities.** JAMA is comprised of six Java classes: `Matrix`, `CholeskyDecomposition`, `LUDecomposition`, `QRDecomposition`, `SingularValueDecomposition` and `EigenvalueDecomposition`.

The `Matrix` class provides the fundamental operations of numerical linear algebra. Various constructors create `Matrices` from two dimensional arrays of double

[3] <http://math.nist.gov/javanumerics/jama/>

# About installation instructions

As explained in our article [4], the library we have constructed is available in [www.github.com](http://www.github.com) and, to use it, one must download the following files from [5]:

File	Brief description
<i>data.zip</i>	Contains the the files <i>sem.n</i>
<i>sexpansion.jar</i>	Is the library itself
<i>examples.zip</i>	Example programs listed in Appendix A and explained in Section 5
<i>Output_examples.zip</i>	Output samples of the example programs

Also the the full documentation of the library is available in [6], on our article [4] we have explained only the most important methods of these classes (see Sections 3-4).

[4] C. Inostroza, I. Kondrashuk, N. Merino, and F. Nadal, “A Java library to perform S-expansions of Lie algebras,” arXiv:1703.04036 [cs.MS].

[5] <https://github.com/SemigroupExp/Sexpansion/releases/tag/v1.0.0>

[6] <https://github.com/SemigroupExp/Sexpansion/wiki>

GitHub, Inc. [US] | <https://github.com/SemigroupExp/Sexpansion/releases/tag/v1.0.0>

Features Business Explore Marketplace Pricing This repository Search Sign in or Sign up

SemigroupExp / Sexpansion

Watch 1 Star 3 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights

Releases Tags

Latest release






v1.0.0  
8682192

## Initial release of the S-Expansion Library

cinostroza released this on 11 Nov 2016 · 18 commits to master since this release

Release v1.0.0 of the sexpansion Library, it also includes 45 example programs and semigroup data files.

## Downloads

 <a href="#">data.zip</a>	148 KB
 <a href="#">examples.zip</a>	46.5 KB
 <a href="#">Output_examples.zip</a>	2.32 MB
 <a href="#">ReadMe.pdf</a>	8.02 MB
 <a href="#">sexpansion.jar</a>	57.6 KB

[7] R. Caroca, I. Kondrashuk, N. Merino, and F. Nadal, “Bianchi spaces and their three-dimensional isometries as S-expansions of two-dimensional isometries,” J.Phys. A46 (2013) 225201, arXiv:1104.3541 [math-ph].

[8] L. Andrianopoli, N. Merino, F. Nadal, and M. Trigiante, “General properties of the expansion methods of Lie algebras,” J.Phys. A46 (2013) 365204, arXiv:1308.4832 [gr-qc].



# Description of the Java library

Our library is composed of the following *classes* (to see the source code unzip the file *sexpansion.jar*):

1. Semigroup.java
2. SetS.java
3. Selector.java
4. SelectorReduced.java
5. SelectorResonant.java
6. SelectorResonantReduced.java
7. StructureConstantSet.java
8. StructureConstantSetExpanded.java
9. StructureConstantSetExpandedReduced.java
10. StructureConstantSetExpandedResonant.java
11. StructureConstantSetExpandedResonantReduced.java

# Contents

- 1) Description of the Java library
- 2) Algorithm to find S-related Lie algebras**
- 3) Applications
- 4) Final remarks

# General algorithm to find S-related algebras

- Perform the S-expansion with an unknown semigroup
- Relate dimensions of the two algebras we want to connect (impose conditions like existence of a zero element, resonant decomposition)
- Extract information from the Lie bracket of the expanded algebra

Then some part of the of the multiplication table are fixed and we get a **template** that must be filled.

For a template of order  $n$  where  $x$  independent component has been fixed, there are  $n^{n(n+1)/2-x}$  different forms to fill it in a commutative way.

This method will be fully described in the manuscript [9] that we are preparing, but I will describe the main results in the following section of applications.

[9] C. Inostroza, I. Kondrashuk, N. Merino, and F. Nadal, “Algorithm to find S-related Lie algebras,” will be submitted soon.

# Contents

- 1) Description of the Java library
- 2) Algorithm to find S-related Lie algebras
- 3) Applications**
- 4) Final remarks

In Ref. [7] it has been shown that some Bianchi algebras can be obtained as S-expansion of 2-dimensional isometries with semigroups that do not belong to the family SE(N).

**Principal Idea:** can be related 2 and 3-dimensional isometries?

Considering the two set of algebras:  $[X_1, X_2] = 0$  and  
 $[X_1, X_2] = X_1$

Group	Algebra
→ type I	$[X_1, X_2] = [X_1, X_3] = [X_2, X_3] = 0$
→ type II	$[X_1, X_2] = [X_1, X_3] = 0, [X_2, X_3] = X_1$
→ type III	$[X_1, X_2] = [X_2, X_3] = 0, [X_1, X_3] = X_1$
type IV	$[X_1, X_2] = 0, [X_1, X_3] = X_1, [X_2, X_3] = X_1 + X_2$
→ type V	$[X_1, X_2] = 0, [X_1, X_3] = X_1, [X_2, X_3] = X_2$
type VI	$[X_1, X_2] = 0, [X_1, X_3] = X_1, [X_2, X_3] = hX_2,$ where $h \neq 0, 1$
type VII <sub>1</sub>	$[X_1, X_2] = 0, [X_1, X_3] = X_2, [X_2, X_3] = -X_1$
type VII <sub>2</sub>	$[X_1, X_2] = 0, [X_1, X_3] = X_2, [X_2, X_3] = -X_1 + hX_2,$ where $h \neq 0$ ( $0 < h < 2$ ).
type VIII	$[X_1, X_2] = X_1, [X_1, X_3] = 2X_2, [X_2, X_3] = X_3$
type IX	$[X_1, X_2] = X_3, [X_2, X_3] = X_1, [X_3, X_1] = X_2$

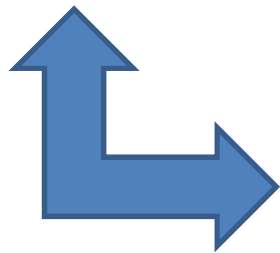
[7] R. Caroca, I. Kondrashuk, N. Merino and F. Nadal, *J. Phys. A: Math. Theor.* 46 (2013) 225201 (24pp), arXiv: 1104.3541

## Example: Bianchi type II algebra

$T_{II}$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_4$
$\lambda_1$		$\lambda_3$		$\lambda_4$
$\lambda_2$	$\lambda_3$		$\lambda_4$	$\lambda_4$
$\lambda_3$		$\lambda_4$		$\lambda_4$
$\lambda_4$	$\lambda_4$	$\lambda_4$	$\lambda_4$	$\lambda_4$

where  $\begin{cases} S_0 = \{\lambda_2, \lambda_4\}, \\ S_1 = \{\lambda_1, \lambda_3, \lambda_4\}, \end{cases}$

Method  
"fill\_Template"



$S_{II}^1$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_4$
$\lambda_1$	$\lambda_4$	$\lambda_3$	$\lambda_4$	$\lambda_4$
$\lambda_2$	$\lambda_3$	$\lambda_4$	$\lambda_4$	$\lambda_4$
$\lambda_3$	$\lambda_4$	$\lambda_4$	$\lambda_4$	$\lambda_4$
$\lambda_4$	$\lambda_4$	$\lambda_4$	$\lambda_4$	$\lambda_4$

$S_{II}^2$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_4$
$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_4$	$\lambda_4$
$\lambda_2$	$\lambda_3$	$\lambda_4$	$\lambda_4$	$\lambda_4$
$\lambda_3$	$\lambda_4$	$\lambda_4$	$\lambda_4$	$\lambda_4$
$\lambda_4$	$\lambda_4$	$\lambda_4$	$\lambda_4$	$\lambda_4$

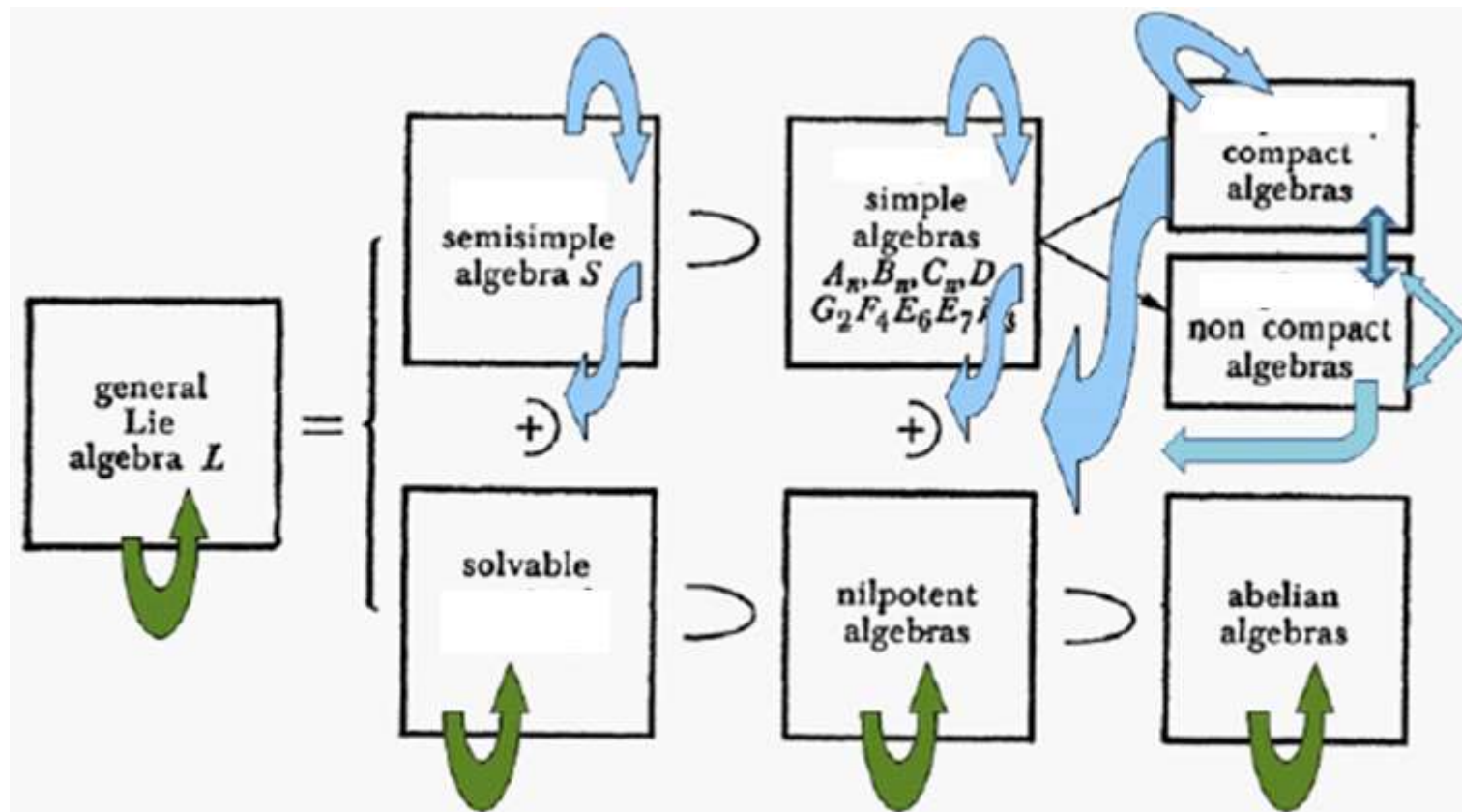
Isomorphic to	Isomorphism
$S_{II}^1 \iff S_{(4)}^{10}$	$(\lambda_4 \ \lambda_3 \ \lambda_1 \ \lambda_2)$
$S_{II}^2 \iff S_{(4)}^{12}$	$(\lambda_4 \ \lambda_3 \ \lambda_2 \ \lambda_1)$

$S_{(4)}^{10}$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_4$
$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$
$\lambda_2$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$
$\lambda_3$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_2$
$\lambda_4$	$\lambda_1$	$\lambda_1$	$\lambda_2$	$\lambda_1$

$S_{(4)}^{12}$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_4$
$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$
$\lambda_2$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_1$
$\lambda_3$	$\lambda_1$	$\lambda_1$	$\lambda_1$	$\lambda_2$
$\lambda_4$	$\lambda_1$	$\lambda_1$	$\lambda_2$	$\lambda_3$

Algebra	Semigroup used
Type I	many semigroups (see section 4.3 of Ref. [26])
Type II	$S_{(4)}^{10}, S_{(4)}^{12}$
Type III	$S_{(4)}^{13}, S_{(4)}^{28}, S_{(4)}^{42}, S_{(4)}^{43}, S_{(4)}^{44}, S_{(4)}^{45}$ and $S_{(4)}^{64}$
Type V	$S_{(4)}^{42}$

In particular, in [8] there were found some criteria, related to the preservation of some properties of the Lie algebra, that allows to answer that question:



[8] L. Andrianopoli, N. Merino, F. Nadal, M. Trigiante, General properties of the S-expansion method, *J. Phys. A: Math. Theor.* 46 (2013) 365204 (33pp) [arXiv:1308.4832](https://arxiv.org/abs/1308.4832)



## All possible expansions with semigroups of order 3:

<i>Reduction of the resonant subalgebra</i>		1,2,3,6,12
<i>Reduced algebra</i>		1,2,3,6,12,7,9,10
<i>Resonant subalgebra</i>	15,16,17,1,2,3,6,12	
<i>Expanded algebra</i>	15,16,17,1,2,3,6,12,7,9,10,18	

## Classification of all possible expansions that can be done with semigroups up to order 6:

		$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$
# $\mathcal{G}_S$		3	12	58	325	2,143
	#pss	2	5	16	51	201
# $\mathcal{G}_{S,\text{red}}$		2	8	39	226	1,538
	#pss	1	3	9	34	135
# $\mathcal{G}_{S,R}$		1	8	48	299	2,059
	#r	1	9	124	1,653	25,512
	#pss	1	1	4	7	23
# $\mathcal{G}_{S,R,\text{red}}$		0	5	32	204	1,465
	#r	0	6	92	1,295	20,680
	#pss	0	1	1	6	12

**#r** : Total number of different resonances

**#pss** : Number of semigroups expansions preserving semisimplicity

# Contents

- 1) Description of the Java library
- 2) Algorithm to find S-related Lie algebras
- 3) Applications
- 4) Final remarks**

## Final remarks

- As mentioned in [9], the full classification of Lie algebras is known only for the semisimple branch.
- And since the 50' there were many works using contraction methods to see if they may help to fix the classification of solvable Lie algebras. The answer was negative and remains as a big problem in mathematics.
- In [9] it was mentioned that would be interesting to know whether S-expansions may help better to fit that missing classification.
- Therefore, apart from **new physical applications**, the programs proposed in this work could be useful in solving this mathematical problem.

[9] Nesterenko M. 2012 S-expansions of three dimensional Lie algebras  
*arXiv:1212.1820*

## Final remarks

Also, it would be very interesting to extend our library to study other kind of resonances like, e.g.,

$$S = S_0 \cup S_1 \cup S_2$$

$$S_0 \times S_0 \subset S_0, \quad S_0 \times S_1 \subset S_1, \quad S_0 \times S_2 \subset S_2$$

$$S_1 \times S_1 \subset S_0 \cap S_2, \quad S_1 \times S_2 \subset S_1, \quad S_2 \times S_2 \subset S_0 \cap S_2$$

This type of decomposition would be very useful to study expansions of Lie super algebras (used in the context of *supergravity* and *string theory*), which have the following subspace structure,

$$\mathcal{G} = V_0 \oplus V_1 \oplus V_2$$

$$[V_0, V_0] \subset V_0, \quad [V_0, V_1] \subset V_1, \quad [V_0, V_2] \subset V_2$$

$$[V_1, V_1] \subset V_0 \oplus V_2, \quad [V_1, V_2] \subset V_1, \quad [V_2, V_2] \subset V_0 \oplus V_2$$

## Final remarks

- Apart from the mentioned applications in gravity theories and Lie group theory, it would be interesting analyze if this tool could have also applications in other physical theories related with symmetries.

**MANY THANKS!**