Formalising Lie algebras

Oliver Nash

August 6, 2021

Abstract

Lie algebras are an important class of algebras arising throughout mathematics and physics. We report on the formalisation of Lie algebras in Lean's Mathlib library [16]. Particular emphasis is paid to the construction of the exceptional Lie algebras. Thanks to this construction, it is possible to state the classification theorem for finite-dimensional semisimple Lie algebras over an algebraically closed field of characteristic zero.

1 Introduction

1.1 Skew-symmetric matrices

At least as far back at the 19$^{\text{th}}$ Century, it was observed that if A is a skew-symmetric matrix and $\epsilon \in \mathbb{R}$ is small then $I + \epsilon A$ is almost a rotation. Indeed since $A^t = -A$, we have

$$(I + \epsilon A)^t (I + \epsilon A) = I - \epsilon^2 A^2.$$

That is, the inverse of $I + \epsilon A$ is its transpose, if we neglect terms order ϵ^2.

Better yet, the exponential $e^{\epsilon A}$ is truly a rotation (no terms neglected) and for another such matrix B, the Baker–Campbell–Hausdorff formula quantifies how the composition of the rotations $e^{\epsilon A}$, $e^{\epsilon B}$ behaves in terms of skew-symmetric matrices:

$$e^{\epsilon A} e^{\epsilon B} = e^{\epsilon(A+B) + \frac{\epsilon^2}{2}[A,B] + O(\epsilon^3)}. \tag{1.1}$$

The term $[A, B]$ appearing in (1.1) is defined as:

$$[A, B] = AB - BA \tag{1.2}$$

and is an instance of a Lie bracket. It defines a natural product: if A and B are skew-symmetric then[1] so is $[A, B]$.

The Lie bracket is skew-commutative:

$$[A, B] = -[B, A],$$

and in general non-associative, but by way of compensation it satisfies the Jacobi identity:

$$[A, [B, C]] + [B, [C, A]] + [C, [A, B]] = 0. \tag{1.3}$$

1.2 Abstract Lie algebras and their ubiquity

Recognising skew-symmetric matrices merely as an example, one can consider the study of abstract Lie algebras. These are modules carrying a bilinear, skew-commutative product which satisfies the Jacobi identity (1.3).

The study of abstract Lie algebras was initiated by Lie and independently by Killing more than 140 years ago [13], [10, 11, 12], [8] and the subject now pervades much of modern mathematics and physics.

In classical physics, both linear and angular momentum are best understood as taking values in the dual of a Lie algebra. More importantly, recognising that the space of classical observables forms a Lie algebra[2] under the Poisson bracket is an important step in quantisation. Furthermore, in particle physics, elementary particles such as the electron are essentially basis vectors of irreducible representations of Lie groups [3] and thus of Lie algebras.

In number theory, automorphic forms, central objects of study in the Langlands programme, satisfy a differential equation defined in terms of a reductive Lie algebra. In differential geometry, the tangent bundle is special amongst vector bundles because its sections carry a natural Lie algebra structure; moreover with just this structure one can define the De Rham cohomology, thus connecting with algebraic topology. In Riemannian geometry and gauge theory, the curvature 2-form takes values in a Lie algebra. In symplectic geometry, the moment map takes values in the dual of a Lie algebra.

Of course the lists above hardly scratch the surface. The unifying theme is that a great many types of symmetry are naturally Lie groups or algebraic groups, and thus have associated Lie algebras which are essential for their study. Understanding symmetry thus requires understanding Lie algebras and for this reason the classification of semisimple Lie algebras is rightly regarded as a landmark result of early 20$^{\text{th}}$ Century mathematics.

[1] If this is the first time you have seen this then check it: it's a fun calculation.

[2] In fact a Poisson algebra.

1.3 Lie algebras in Lean

Here is the definition of a Lie algebra in Lean's Mathlib [16] library:

```
class lie_ring (L : Type v)
  extends add_comm_group L, has_bracket L L :=
(add_lie     : ∀ (x y z : L), [x + y,z] = [x,z] + [y,z])
(lie_add     : ∀ (x y z : L), [x,y + z] = [x,y] + [x,z])
(lie_self    : ∀ (x : L), [x,x] = 0)
(leibniz_lie : ∀ (x y z : L), [x,[y,z]] = [[x,y],z] + [y,[x,z]])

class lie_algebra (R : Type u) (L : Type v)
  [comm_ring R] [lie_ring L] extends module R L :=
(lie_smul : ∀ (t : R) (x y : L), [x,t • y] = t • [x,y])
```

The skew-commutative property follows from the `lie_self` axiom and the Jacobi identity is equivalent to the `leibniz_lie` axiom[3].

There exist computer algebra systems such as GAP and MAGMA as well as a Mathematica package available at http://katlas.org, that are capable of performing calculations involving Lie algebras. However to the best of our knowledge, there is no previous work formalising the theory of Lie algebras.

As of August 2021, Mathlib contains over 6,000 lines of code about Lie algebras and their representations, broadly following Bourbaki [5, 6, 7]. Material covered includes:

- Lie algebras and Lie modules
- Morphisms and equivalences of Lie algebras and Lie modules
- Lie subalgebras, Lie submodules, Lie ideals, and quotients
- Extension and restriction of scalars
- Direct sums of Lie modules and Lie algebras
- Tensor product of Lie modules
- Lie ideal operations, the lower central series, the derived series, and derived length
- Nilpotent, solvable, simple, semisimple Lie algebras, the radical, and the centre of a Lie algebra
- Cartan subalgebras
- Weight spaces of a Lie module, and thus root spaces of a Lie algebra

[3] We comment on the choice of axioms in section 2.

- The universal enveloping algebra (and its universal property)
- The free Lie algebra (and its universal property)
- Definition of the classical Lie algebras
- Definition of the exceptional Lie algebras

The final item is worth highlighting. There is no easy route to the definition of the exceptional Lie algebras (section 5) and it is an important milestone since it allows us to state the classification of semisimple Lie algebras (section 6). A *proof* of this classification within Mathlib would be a significant undertaking but now looks achievable.

2 Design choices: Leibniz vs. Jacobi

The choice of the axiom `leibniz_lie` in the definition exhibited in section 1.3 deserves explanation, if only because it serves as a simple example of the sorts of choices that repeatedly came up in the course of formalisation.

In the presence of the other Lie algebra axioms, each of the following are equivalent:

$$[x,[y,z]]+[y,[z,x]]+[z,[x,y]]=0, \tag{2.1}$$

$$[[x,y],z]=[x,[y,z]]-[y,[x,z]], \tag{2.2}$$

$$[x,[y,z]]=[[x,y],z]+[y,[x,z]]. \tag{2.3}$$

Note that (2.3) is the axiom `leibniz_lie` and that given $x : L$, if we define $D_x : L \to L$ by:

$$D_x y = [x,y]$$

then (2.3) says that D_x satisfies the Leibniz product rule:

$$D_x[y,z]=[D_x y,z]+[y,D_x z],$$

i.e., it says that D_x is a derivation.

One might think that the Jacobi identity (2.1) is the best choice since it looks the most symmetric; in fact it is the worst choice. This becomes clear when one considers that we also need a theory of Lie modules.

Informally, if a Lie algebra L acts linearly on a module M, and if we denote the action of $x : L$ on $m : M$ by $[x,m] : M$ then this action turns M into a Lie module for L iff:

$$[x,[y,m]]=[[x,y],m]+[y,[x,m]],$$

for all $x, y : L$ and $m : M$.

Now consider the case $L = M$ and observe that any Lie algebra is thus a module over itself. This so-called adjoint action is extremely important in Lie theory. Observe also that if we replace $z : L$ in equations (2.1) – (2.3) by $m : M$ then the terms $[y, [z, x]]$, $[z, [x, y]]$ do not make sense since there is no action of M on L. Thus (2.1) can never be used as an axiom for Lie modules.

By choosing to define Lie algebras using the `leibniz_lie` axiom (2.3) we thus obtain a theory where Lie algebras and Lie modules *definitionally* satisfy the same axiom. This is a desirable convenience that we exploit. For example, here is the code that defines the adjoint action:

```
instance lie_ring_self_module : lie_ring_module L L :=
{ .. (infer_instance : lie_ring L) }
```

One might ask why to choose (2.3) over (2.2). This is of lesser importance but (2.3) is still the better choice. This is because (2.2) requires using subtraction which is often a secondary operation defined via addition and inverses. This means that when constructing Lie algebras downstream, it is likely there will be more direct proof of (2.3).

On the other hand, as a simplification lemma (rather than a definition) (2.2) is excellent since it can be used to push Lie brackets right-most in nested expressions. Indeed the following `simp` lemma establishes this as the normal form in Mathlib:

```
@[simp] lemma lie_lie : [[x,y],m] = [x,[y,m]] - [y,[x,m]] := ...
```

And of course we could never register (2.3) as `simp` lemma since we would get a `simp` loop: the term $[y, [x, m]]$ on the right hand side of (2.3) is of the same form as the term $[x, [y, m]]$ on the left.

Notwithstanding the words above, the choice of axiom here is of minor importance. However there are many such choices and en mass they accumulate to have non-trivial impact.

3 Case study: the radical is solvable

Whenever possible, we strove to work at the greatest reasonable level of generality. At times the unified nature of Mathlib made it possible to establish results at a level of generality beyond that of the standard references, including Bourbaki. A good example is the basic result that finite-dimensional Lie algebras possess a maximal solvable ideal.

Like groups, Lie algebras admit a notion of being solvable. For the purposes of this discussion, the precise meaning is unimportant. What is

important is that if I, J are ideals of a Lie algebra and if, regarding them as Lie algebras in their own right, they are both solvable, then their sum $I + J$ is solvable. Mathlib knows this fact. Indeed here is the statement and proof for a Lie algebra L over a commutative ring R:

```
instance is_solvable_add {I J : lie_ideal R L}
  [hI : is_solvable R I] [hJ : is_solvable R J] :
  is_solvable R ↥(I + J) :=
begin
  tactic.unfreeze_local_instances,
  obtain ⟨k, hk⟩ := hI,
  obtain ⟨l, hl⟩ := hJ,
  exact ⟨⟨k+l, lie_ideal.derived_series_add_eq_bot hk hl⟩⟩,
end
```

The (solvable) radical of a Lie algebra is the sum of all solvable ideals, or more precisely, the supremum of the subset of solvable ideals in the complete lattice of ideals of a Lie algebra. Here is the definition in Mathlib:

```
def radical := Sup { I : lie_ideal R L | is_solvable R I }
```

It is clear that if R is a field and L is finite-dimensional then the radical itself is finite-dimensional and can thus be represented as a sum of finitely-many solvable ideals. By iterating `is_solvable_add` we thus see the radical is solvable. This was the greatest level of generality in which this fact was established in any reference the author could find.

However it is not necessary to make such strong assumptions. Indeed the result is true over any commutative ring as long as L is Noetherian, as can be seen from the following proof in Mathlib, in which R is any commutative ring:

```
instance radical_is_solvable [is_noetherian R L] :
  is_solvable R (radical R L) :=
begin
  have h := lie_submodule.well_founded_of_noetherian R L L,
  rw ←complete_lattice.is_sup_closed_compact_iff_well_founded at h,
  refine h { I : lie_ideal R L | is_solvable R I } _ _,
  { use ⊥, exact lie_algebra.is_solvable_bot R L, },
  { intros I J hI hJ,
    apply lie_algebra.is_solvable_add R L;
    [exact hI, exact hJ], },
end
```

The key lemma is `complete_lattice.is_sup_closed_compact_iff_well_founded` which the author added to Mathlib's lattice theory library for the purposes of proving `radical_is_solvable`. This addition was only possible because Mathlib already contained a superb lattice theory library

and numerous key results about well-founded relations. Furthermore the lemma `lie_submodule.well_founded_of_noetherian` ultimately depends upon results which were originally introduced to Mathlib for the purposes of formalising results about Noetherian modules over commutative rings with a view toward algebraic geometry.

Different people with different aims in different corners of Mathlib are enabling each other to push boundaries into new territory.

4 General non-associative algebra and the free Lie algebra

As part of this work we formalised a construction of the free Lie algebra on a type X with coefficients in a commutative ring R. Here is the statement of the universal property (i.e., left adjointness) as stated in Mathlib with respect to a Lie algebra L:

```
def lift : (X → L) ≃ (free_lie_algebra R X →ₗ[R] L) :=
```

This definition, and the proof of its universality, was hard-won and is worth comment.

The construction is to take a quotient of the free non-unital, non-associative algebra[4] on X with coefficients in R. We thus needed to define `free_non_unital_non_assoc_algebra` and prove its universal property:

```
def lift : (X → A) ≃
  non_unital_alg_hom R (free_non_unital_non_assoc_algebra R X) A :=
```

In the above, A is a general non-unital, non-associative algebra and `non_unital_alg_hom` is the type of morphisms of such algebras.

Establishing the above result while adhering to the standards of Mathlib was not straightforward. The problem was that Mathlib's theories of rings and algebras were entirely specialised to the unital, associative setting. To handle this without fragmenting the algebraic hierarchy, it was necessary to insert new classes, notably `non_unital_non_assoc_semiring`, low down in the hierarchy. In a library as large as Mathlib, such changes are significant undertakings.

Eric Wieser generously took on this challenge[5] and also showed how to encode a general non-unital, non-associative algebra:

[4]Strictly speaking we should say 'not-necessarily-unital' and 'not-necessarily-associative' but it is common and easier to say simply 'non-unital' and 'non-associative'.

[5]This pull request shows what was required after all other preparatory work had been completed.

```
variables {R A : Type*}
  [comm_ring R] [non_unital_non_assoc_semiring A]
  [module R A] [is_scalar_tower R A A] [smul_comm_class R A A]
```

After Wieser's work (see also [19]) it was essentially straightforward to construct `free_non_unital_non_assoc_algebra` by relaxing the associativity assumption in the existing monoid algebra construction and proving the corresponding universal property with respect to a magma M:

```
def lift_magma [has_mul M] :
  mul_hom M A ≃ non_unital_alg_hom R (monoid_algebra R M) A :=
```

With this in hand, the author was able to make the key definition:

```
def free_non_unital_non_assoc_algebra :=
monoid_algebra R (free_magma X)
```

and the corresponding universal property followed trivially.

Finally the free Lie algebra was constructed as a quotient using the following relation:

```
local notation `lib` := free_non_unital_non_assoc_algebra

inductive rel : lib R X → lib R X → Prop
| lie_self (a : lib R X) : rel (a*a) 0
| leibniz_lie (a b c : lib R X) :
    rel (a*(b*c)) (((a*b)*c) + (b*(a*c)))
| smul (t : R) (a b : lib R X) : rel a b → rel (t·a) (t·b)
| add_right (a b c : lib R X) : rel a b → rel (a+c) (b+c)
| mul_left (a b c : lib R X) : rel b c → rel (a*b) (a*c)
| mul_right (a b c : lib R X) : rel a b → rel (a*c) (b*c)
```

and its universal property followed easily.

It should be noted that the use of `inductive` above was necessary because Mathlib does not yet contain a theory of ideals and their quotients for general non-associative algebras. Filling this gap would improve the construction even further, though the benefit would be slight.

We should say that it would have been easy to establish what we needed without any of the above work by ignoring most Mathlib's of algebra library and taking a quotient of an inductively-defined type with a constructor for every term of X as well as separate constructors corresponding to the scalar action, additive law, and Lie bracket. We rejected this low-level approach because it would require a significant quantity of single-use code, because it would be harder to maintain, because the alternative approach was a opportunity to start developing a general theory of non-associative rings and algebras, and because this is very unlikely to be the approach that a mathematician would take in informal mathematics.

We should also say that we rejected an approach that constructs the free Lie algebra as the smallest Lie subalgebra of the free unital, associative algebra containing the generating type X. This can be expressed in Lean as:

```
lie_subalgebra.lie_span R
  (free_algebra R X) (set.range (free_algebra.ι R))
```

This approach is mathematically appealing but the proof that this construction satisfies the universal property appears to need a powerful version of the Poincaré-Birkhoff-Witt theorem (see [14] as well as [5] I §2.7, §3.1).

5 The exceptional Lie algebras

We assume for now that the coefficients R form an algebraically closed field of characteristic zero. The work under discussion does not make this assumption but it will simplify the discussion here if we do.

There are numerous beautiful ways to construct the five exceptional Lie algebras $\mathfrak{g}_2, \mathfrak{f}_4, \mathfrak{e}_6, \mathfrak{e}_7, \mathfrak{e}_8$ (e.g., [1], [17], [18], [9]) but the most useful construction from the point of view of proving the classification theorem (see section 6) is an approach due to Serre [15]. This approach takes a square matrix of integers as input and yields a Lie algebra. When the matrix is the Cartan matrix of a semisimple Lie algebra, we recover the corresponding Lie algebra, together with a splitting Cartan subalgebra.

If A is an $l \times l$ Cartan matrix, the corresponding Lie algebra is defined to be the quotient of the free Lie algebra on $3l$ generators: $H_1, H_2, \dots H_l$, $E_1, E_2, \dots, E_l$, $F_1, F_2, \dots, F_l$ by the following relations:

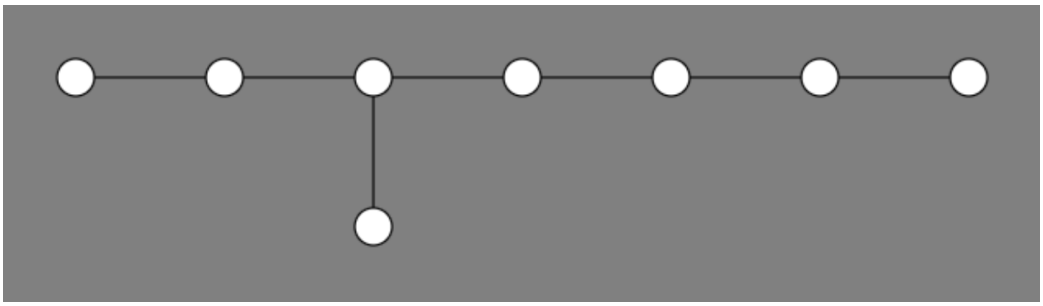
$$
\begin{aligned}
[H_i, H_j] &= 0 \\
[E_i, F_i] &= H_i \\
[E_i, F_j] &= 0 \quad \text{if } i \neq j \\
[H_i, E_j] &= A_{ij} E_j \\
[H_i, F_j] &= -A_{ij} F_j \\
ad(E_i)^{1-A_{ij}}(E_j) &= 0 \quad \text{if } i \neq j \\
ad(F_i)^{1-A_{ij}}(F_j) &= 0 \quad \text{if } i \neq j
\end{aligned}
$$

Thanks to the construction of the free Lie algebra described in section 4, it was easy to implement Serre's construction in Mathlib and thus to define the exceptional Lie algebras. For example, here is Mathlib's definition of $\mathfrak{f}_4$:

```
def cartan_matrix.F₄ : matrix (fin 4) (fin 4) ℤ :=
![![ 2, -1,  0,  0],
  ![-1,  2, -2,  0],
  ![ 0, -1,  2, -1],
  ![ 0,  0, -1,  2]]

abbreviation f₄ := cartan_matrix.F₄.to_lie_algebra R
```

What's more, thanks to Ed Ayers's Lean Widgets [2], it was easy to generate the Dynkin diagram corresponding to a Cartan matrix. For example, here is a screenshot from the author's proof-of-concept widget, written in Lean, which reads Mathlib's definition of the E_8 Cartan matrix and renders the corresponding Dynkin diagram:



Finally, we should confess that we have yet to prove almost anything about the exceptional Lie algebras. The path is clear but much work remains until we can prove key facts such as their simplicity, or for example:

```
def dimension_g₂ : Prop := finrank (g₂ ℂ) = 14
```

6 Stating the classification

An important milestone, passed in the course of this work, was teaching Lean the statements of the classification of semisimple Lie algebras. Modulo some boilerplate to establish notation, the statements are:

```
variables (K L : Type*)

/-- Let K be an algebraically closed field of characteristic 0. -/
variables [field K] [is_alg_closed K] [char_zero K]

/-- Let L be a finite-dimensional Lie algebra over K. -/
variables [lie_ring L] [lie_algebra K L] [finite_dimensional K L]

def simple_classification : Prop :=
  is_simple K L ↔
```

```
  ((L ≃ₗ[K] g₂ K) ∨
   (L ≃ₗ[K] f₄ K) ∨
   (L ≃ₗ[K] e₆ K) ∨
   (L ≃ₗ[K] e₇ K) ∨
   (L ≃ₗ[K] e₈ K) ∨
   (∃ l, (L ≃ₗ[K] sl l K) ∧ 1 < l) ∨
   (∃ l, (L ≃ₗ[K] sp l K) ∧ 2 < l) ∨
   (∃ l, (L ≃ₗ[K] so l K) ∧ 4 < l ∧ l ≠ 6))

def semisimple_classification : Prop :=
  is_semisimple K L ↔
  ∃ n (I : fin n → lie_ideal K L),
    (L ≃ₗ[K] (⊕ i, I i)) ∧ ∀ i, is_simple K (I i)
```

Note that the `simple_classification` contains several of the "exceptional isomorphisms". E.g., $\mathfrak{so}(6)$ is simple so it must be isomorphic to one of the other algebras on the list. For dimensional reasons, this has to be $\mathfrak{sl}(4)$. Likewise for the other cases excluded.

Note also that if we pursue a proof of the classification, we will probably restate `simple_classification` in terms of algebras constructed from Cartan matrices of types A, B, C, D rather than the models $\mathfrak{sl}$, $\mathfrak{sp}$, $\mathfrak{so}$ defined in terms of matrices.

7 Final words

7.1 Trivial proofs should be trivial

When building a library the size of Mathlib, one must constantly try to be mindful of how one's work will scale as more is built upon it. One metric for the health of a particular area of the library is how much effort one is forced to put into proving trivialities. We share an example of what this looks like when things go well.

Given a type X and a commutative ring R one can use this data to build the the free unital, associative algebra $A(R, X)$. However, there is another way to build a unital, associative algebra from this data: one first builds the free Lie algebra $L(R, X)$ and then takes its universal enveloping algebra $U(L(R, X))$. A simple diagram chase reveals that these are the same, in particular:

$$U(L(R, X)) \simeq A(R, X).$$

Mathlib knows this fact; here is the proof (using some notational shortcuts for readability):

```
def universal_enveloping_equiv_free_algebra :
  universal_enveloping_algebra R (free_lie_algebra R X) ≃ₐ[R]
  free_algebra R X :=
alg_equiv.of_alg_hom
  (liftu R $ liftl R $ ιa R) (lifta R $ (ιu R) ∘ (ιl R))
  (by { ext, simp, })
  (by { ext, simp, })
```

The point of the above is the two lines that read `(by { ext, simp, })`. This is the Lean code discharging the proof obligations which correspond to the informal mathematician's diagram chase. It is encouraging that they are trivial applications of standard tactics.

7.2 The Lie algebra of a Lie group

Far away in a different corner of Mathlib, Sébastien Gouëzel has developed a theory of differentiable manifolds. Building on top of this, Nicolò Cavalleri, under the supervision of Anthony Bordg, has defined Lie groups and has used it to construct the Lie algebra associated to a Lie group [4].

7.3 Proof of classification

With the statement of the classification theorem formalised, it is tempting to consider formalising a proof. Several key concepts such as Cartan subalgebras, weight spaces, and root spaces have also been formalised. The evidence so far is that formalising a proof of the classification would be non-trivial but is absolutely within reach.

Acknowledgements

It is a pleasure to ackowledge the significant help the author received from numerous members of the thriving Mathlib community. Almost all of the maintainers were of direct assistance at some point. Special thanks are owed to Johan Commelin and Eric Wieser for their astonishing appetite to review pull requests (and excellent suggestions) as well as to Scott Morrison for providing the motivation to take up this work and for writing the `noncomm_ring` tactic. I am also grateful to Kevin Buzzard for frequent advice, guidance, and encouragement.

References

[1] J. F. Adams. *Lectures on exceptional Lie groups*. Chicago Lectures in Mathematics. University of Chicago Press, Chicago, IL, 1996. With a foreword by J. Peter May, Edited by Zafer Mahmud and Mamoru Mimura.

[2] Edward Ayers. Widgets: interactive output in VSCode. In *Lean Together 2021, January 4–7, 2021*. url, 2021.

[3] John Baez and John Huerta. The algebra of grand unified theories. *Bull. Amer. Math. Soc. (N.S.)*, 47(3):483–552, 2010.

[4] Anthony Bordg and Nicolò Cavalleri. Elements of Differential Geometry in Lean A Report for Mathematicians. In *14th Conference on Intelligent Computer Mathematics CICM 2021, Timisoara, Romania, July 26–31, 2021*. url, 2021.

[5] Nicolas Bourbaki. *Lie groups and Lie algebras. Chapters 1–3*. Elements of Mathematics (Berlin). Springer-Verlag, Berlin, 1998. Translated from the French, Reprint of the 1989 English translation.

[6] Nicolas Bourbaki. *Lie groups and Lie algebras. Chapters 4–6*. Elements of Mathematics (Berlin). Springer-Verlag, Berlin, 2002. Translated from the 1968 French original by Andrew Pressley.

[7] Nicolas Bourbaki. *Lie groups and Lie algebras. Chapters 7–9*. Elements of Mathematics (Berlin). Springer-Verlag, Berlin, 2005. Translated from the 1975 and 1982 French originals by Andrew Pressley.

[8] A. J. Coleman. The greatest mathematical paper of all time. *Math. Intelligencer*, 11(3):29–38, 1989.

[9] José Figueroa-O'Farrill. A geometric construction of the exceptional Lie algebras F_4 and E_8. *Comm. Math. Phys.*, 283(3):663–674, 2008.

[10] Wilhelm Killing. Die Zusammensetzung der stetigen endlichen Transformations-gruppen. *Math. Ann.*, 31(2):252–290, 1888.

[11] Wilhelm Killing. Die Zusammensetzung der stetigen endlichen Transformationsgruppen. *Math. Ann.*, 33(1):1–48, 1888.

[12] Wilhelm Killing. Die Zusammensetzung der stetigen endlichen Transformations-gruppen. *Math. Ann.*, 34(1):57–122, 1889.

[13] Sophus Lie. Theorie der Transformationsgruppen I. *Math. Ann.*, 16(4):441–528, 1880.

[14] Mathoverflow. Is the natural map from the free Lie algebra to the free associative algebra injective? url, 2021.

[15] Jean-Pierre Serre. *Complex semisimple Lie algebras*. Springer-Verlag, New York, 1987. Translated from the French by G. A. Jones.

[16] The mathlib community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, LA, USA, January 20-21, 2020*, pages 367–381, 2020.

[17] J. Tits. Algèbres alternatives, algèbres de Jordan et algèbres de Lie exceptionnelles. I. Construction. *Nederl. Akad. Wetensch. Proc. Ser. A 69 = Indag. Math.*, 28:223–237, 1966.

[18] E. B. Vinberg. Construction of the exceptional simple Lie algebras. In *Lie groups and invariant theory*, volume 213 of *Amer. Math. Soc. Transl. Ser. 2*, pages 241–242. Amer. Math. Soc., Providence, RI, 2005. Translated from Trudy Sem. Vekt. Tenz. Anal. **1**3 (1966), 7–9.

[19] Eric Wieser. Scalar actions in Lean's mathlib. In *14th Conference on Intelligent Computer Mathematics CICM 2021, Timisoara, Romania, July 26–31, 2021*. to appear, 2021.