The Blueprint For Formalizing Geometric Algebra in Lean

Eric Wieser, Utensil Song

November 29, 2023

Introduction

The goal of this document is to provide a detailed account of the formalization of Geometric Algebra (GA) a.k.a. Clifford Algebra [Hestenes and Sobczyk(1984)] in the Lean 4 theorem prover and programming language [Moura and Ullrich(2021), de Moura et al.(2015), Ullrich(2023)] and using its Mathematical Library Mathlib [The mathlib Community(2020)].

The web version of this blueprint is available here.

1 Preliminaries

This section introduces the algebraic environment of Clifford Algebra, covering vector spaces, groups, algebras, representations, modules, multilinear algebras, quadratic forms, filtrations and graded algebras.

The material in this section should be familiar to the reader, but it is worth reading through it to become familiar with the notation and terminology that is used, as well as their counterparts in Lean, which usually require some additional treatment, both mathematically and technically (probably applicable to other formal proof verification systems).

Details can be found in the references in corresponding section, or you may hover a definition/theorem, then click on "L∃∀N" for the Lean 4 code.

In this section, we follow [Jadczyk(2019)], with supplements from [Garling(2011), Chen(2016)], and modifications to match the counterparts in Lean's Mathlib .

Remark 1.0.1 — We unify the informal mathematical language for a definition to:

Let A be a `concept` A. A `concept` B is a set/pair/triple/tuple $(B, \mathtt{op}, ...)$, satisfying:

1. B is a `concept` C over A under `op` .
2. `formula` for all `elements in` B (`property`).
3. for each `element` in `concept` A there exists `element` such that `formula` for all `elements in concept` B.
4. op is called `op name`, for all `elements in` B, we have
   (i) `formula`
   (ii) `formula`

   (`property`).

By default, A is a set, `op` is a binary operation on A.

1.1 Basics: from groups to modules

Definition 1.1.1 (Group). A group is a pair $(G, *)$, satisfying:

1. $(a * b) * c = a * (b * c)$ for all $a, b, c \in G$ (associativity).
2. there exists $1 \in G$ such that $1 * a = a * 1 = a$ for all $a \in G$.
3. for each $a \in G$ there exists $a^{-1} \in G$ such that $a * a^{-1} = a^{-1} * a = 1$.

Remark 1.1.2 — It then follows that e, the identity element, is unique, and that for each $g \in G$ the inverse g^{-1} is unique.

A group G is abelian, or commutative, if $g * h = h * g$ for all $g, h \in G$.

Remark 1.1.3 — In literatures, the binary operation are usually denoted by juxtaposition, and is understood to be a mapping $(g, h) \mapsto g * h$ from $G \times G$ to G.

Mathlib uses a slightly different way to encode this, $G \to G \to G$ is understood to be $G \to (G \to G)$, that sends $g \in G$ to a mapping that sends $h \in G$ to $g * h \in G$.

Further more, a mathimatical construct is represented by a ”type”, as Lean has a dependent type theory foundation, see [Carneiro(2019)] and [Ullrich(2023), section 3.2].

It can be denoted multiplicatively as $*$ in Group or additively as $+$ in AddGroup, where e will be denoted by 1 or 0, respectively.

Sometimes we use notations with subscripts (e.g. $*_G$, 1_G) to indicate where they are.

We will use the corresponding notation in Mathlib for future operations without further explanation.

Definition 1.1.4 (Monoid). A monoid is a pair $(R, *)$, satisfying:

1. $(a * b) * c = a * (b * c)$ for all $a, b, c \in R$ (associativity).
2. there exists an element $1 \in R$ such that $1 * a = a * 1 = a$ for all $a \in R$

   i.e. 1 is the multiplicative identity (neutral element).

Definition 1.1.5 (Ring). A ring is a triple $(R, +, *)$, satisfying:

1. R is a commutative group under $+$.
2. R is a monoid under $*$.
3. for all $a, b, c \in R$, we have

   (i) $a * (b + c) = a * b + a * c$

   (ii) $(a + b) * c = a * c + b * c$

   (left and right distributivity over $+$).

Remark 1.1.6 — In applications to Clifford algebras R will be always assumed to be commutative.

Definition 1.1.7 (Division ring). A division ring is a ring $(R, +, *)$, satisfying:

1. R contains at least 2 elements.

2. for all $a \neq 0$ in R, there exists a multiplicative inverse $a^{-1} \in R$ such that

$$a * a^{-1} = a^{-1} * a = 1$$

Definition 1.1.8 (Module). Let R be a commutative ring. A module over R (in short R-module) is a pair $(M, \bullet)$, satisfying:

1. M is a group under $+$.
2. $\bullet : R \to M \to M$ is called scalar multiplication, for every $a, b \in R$, $x, y \in M$, we have
   (i) $a \bullet (x + y) = a \bullet x + b \bullet y$
   (ii) $(a + b) \bullet x = a \bullet x + b \bullet x$
   (iii) $a * (b \bullet x) = (a * b) \bullet x$
   (iv) $1_R \bullet x = x$

Remark 1.1.9 — The notation of scalar multiplication is generalized as heterogeneous scalar multiplication HMul in Mathlib :

$$\bullet : \alpha \to \beta \to \gamma$$

where α, β, γ are different types.

Definition 1.1.10 (Vector space). If R is a division ring, then a module M over R is called a vector space.

Remark 1.1.11 — For generality, Mathlib uses Module throughout for vector spaces, particularly, for a vector space V, it's usually declared as

```
variable [DivisionRing K] [AddCommGroup V] [Module K V]
```

for definitions/theorems about it, and most of them can be found under Mathlib.LinearAlgebra e.g. LinearIndependent.

Remark 1.1.12 — A submodule N of M is a module N such that every element of N is also an element of M.

If M is a vector space then N is called a subspace.

1.2 Algebras

Definition 1.2.1 (Ring homomorphism). Let $(\alpha, +_\alpha, *_\alpha)$, and $(\beta, +_\beta, *_\beta)$ be rings.

A ring homomorphism from α to β is a function $f : \alpha \to_{+*} \beta$ such that

(i) $f(x +_\alpha y) = f(x) +_\beta f(y)$ for each $x, y \in \alpha$.

(ii) $f(x *_\alpha y) = f(x) *_\beta f(y)$ for each $x, y \in \alpha$.

(iii) $f(1_\alpha) = 1_\beta$.

Definition 1.2.2 (Algebra). Let R be a commutative ring. An algebra A over R is a pair $(A, \bullet)$, satisfying:

1. A is a ring under $*$.
2. there exists a ring homomorphism from R to A, denoted $\iota : R \to_{+*} A$.
3. $\bullet : R \to M \to M$ is a scalar multiplication
4. for every $r \in R$, $x \in A$, we have

   (i) $r * x = x * r$ (commutativity between R and A)

   (ii) $r \bullet x = r * x$ (definition of scalar multiplication)

where we omitted that the ring homomorphism ι is applied to r before each multiplication, while they are explicitly written as `toRingHom(r)` in Mathlib .

Remark 1.2.3 — The definition above (adopted in Mathlib) is more general than the definition in literature:

Let R be a commutative ring. An algebra A over R is a pair $(M, *)$, satisfying:

1. A is a module M over R under $+$ and $\bullet$.
2. A is a ring under $*$.
3. For $x, y \in A, a \in R$, we have

$$a \bullet (x * y) = (a \bullet x) * y = x * (a \bullet y)$$

See Implementation notes in Algebra for details.

Remark 1.2.4 — What's simply called algebra is actually associative algebra with identity, a.k.a. associative unital algebra. See the red herring principle for more about such phenomenon for naming, particularly the example of (possibly) nonassociative algebra.

Definition 1.2.5 (Quotient of non-commutative ring). Let R be a non-commutative ring, r an arbitrary equivalence relation on R. The ring quotient of R by r is by the strengthen equivalence relation of r such that for all a, b, c in R:

1. $a + c \sim b + c$ if $a \sim b$
2. $a * c \sim b * c$ if $a \sim b$
3. $a * b \sim a * c$ if $b \sim c$

i.e. the equivalence relation is compatible with the ring operations $+$ and $*$.

Remark 1.2.6 — As ideals haven't been formalized for the non-commutative case, Mathlib uses RingQuot in places where the quotient of non-commutative rings by ideal is needed.

The universal properties of the quotient are proven, and should be used instead of the definition that is subject to change.

Definition 1.2.7 (Free algebra). Let X be an arbitrary set. An free R-algebra A on X (or "generated by X ") is the ring quotient of the following inductively constructed set A_{pre}

1. for all x in X, there exists a map $X \to A_{pre}$.
2. for all r in R, there exists a map $R \to A_{pre}$.
3. for all a, b in A_{pre}, $a + b$ is in A_{pre}.
4. for all a, b in A_{pre}, $a * b$ is in A_{pre}.

by that equivalence relation that makes A an R-algebra, namely:

1. there exists a ring homomorphism from R to A_{pre}, denoted $R \to_{+*} A_{pre}$.
2. A is a commutative group under $+$.
3. A is a monoid under $*$.
4. left and right distributivity under $*$ over $+$.
5. $0 * a \sim a * 0 \sim 0$.
6. for all a, b, c in A, if $a \sim b$, we have

   (i) $a + c \sim b + c$
   (ii) $c + a \sim c + b$
   (iii) $a * c \sim b * c$
   (iv) $c * a \sim c * b$

   (compatibility with the ring operations $+$ and $*$)

Remark 1.2.8 — What we defined here is the free (associative, unital) R-algebra on X, it can be denoted $R\langle X\rangle$, expressing that it's freely generated by R and X, where X is the set of generators.

Definition 1.2.9 (Linear map). Let R, S be rings, M an R-module, N an S-module. A linear map from M to N is a function $f : M \to_l N$ over a ring homomorphism $\sigma : R \to_{+*} S$, satisfying:

1. $f(x + y) = f(x) + f(y)$ for all $x, y \in M$.
2. $f(r \bullet x) = \sigma(r) \bullet f(x)$ for all $r \in R$, $x \in M$.

Definition 1.2.10 (Tensor algebra). Let A be a free R-algebra generated by module M, let $\iota : M \to A$ denote the map from M to A.

An tensor algebra over M (or "of M ") T is the ring quotient of the free R-algebra generated by M, by the equivalence relation satisfying:

1. for all a, b in M, $\iota(a + b) \sim \iota(a) + \iota(b)$.
2. for all r in R, a in M, $\iota(r \bullet a) \sim r * \iota(a)$.

i.e. making the inclusion of M into an R-linear map.

Remark 1.2.11 — The definition above is equivalent to the following definition in literature:

Let M be a module over R. An algebra T is called a tensor algebra over M (or "of M ") if it satisfies the following universal property

1. T is an algebra containing M as a submodule, and it is generated by M,
2. Every linear mapping λ of M into an algebra A over R, can be extended to a homomorphism θ of T into A.

1.3 Forms

Definition 1.3.1 (Bilinear form). Let R be a ring, M an R-module. An bilinear form B over M is a map $B : M \to M \to R$, satisfying:

1. $B(x + y, z) = B(x, z) + B(y, z)$
2. $B(x, y + z) = B(x, y) + B(x, z)$
3. $B(a \bullet x, y) = a * B(x, y)$
4. $B(x, a \bullet y) = a * B(x, y)$

for all $a \in R, x, y, z \in M$.

Definition 1.3.2 (Quadratic form). Let R be a commutative ring, M a R-module. An quadratic form Q over M is a map $Q : M \to R$, satisfying:

1. $Q(a \bullet x) = a * a * Q(x)$ for all $a \in R, x \in M$.
2. there exists a companion bilinear form $B : M \to M \to R$, such that $Q(x + y) = Q(x) + Q(y) + B(x, y)$

Remark 1.3.3 — This notion generalizes to commutative semirings using the approach in [Izhakian et al.(2016)].

2 Foundations

2.1 Clifford algebras - definition

Definition 2.1.1 (Clifford algebra). Let M be a module over a commutative ring R, equipped with a quadratic form $Q : M \to R$.

Let $\iota : M \to_{l[R]} T(M)$ be the canonical R-linear map for the tensor algebra $T(M)$.

Let $\iota_a : R \to_{+*} T(M)$ be the canonical map from R to $T(M)$, as a ring homomorphism.

A Clifford algebra over M, denoted $\mathcal{C}\ell(M)$, is the ring quotient of the tensor algebra $T(M)$ by the equivalence relation satisfying:

$\forall m \in M, \iota(m)^2 \sim \iota_a(Q(m))$.

Remark 2.1.2 — In literatures, M is often written V, and the quotient is taken by the two-sided ideal I_Q generated from the set $\{v \otimes v - Q(v) \mid v \in V\}$.

As of writing, Mathlib does not have direct support for two-sided ideals, but it does support the equivalent operation of taking the ring quotient by a suitable closure of a relation like $v \otimes v \sim Q(v)$.

Hence the definition above.

Remark 2.1.3 — This definition and what follows in Mathlib are based on [Grinberg(2016)], and in turn based on [Bourbaki(2007)] which is in French and never translated to English.

The most informative English reference on the latter is [Jadczyk(2019)], which has an updated exposition in [Jadczyk(2023)].

Example 2.1.4 (Clifford algebra over a vector space)

Let V be a vector space $\mathbb{R}^n$ over $\mathbb{R}$, equipped with a quadratic form Q.

Since $\mathbb{R}$ is a commutative ring and V is a module, definition 2.1.1 of Clifford algebra applies.

2.1.1 Involutions

2.2 Structure of Clifford algebras

2.3 Classifying Clifford algebras

2.4 Representing Clifford algebras

2.5 Spin

3 Geometric Algebra

3.1 Axioms

3.2 Contents

This section would contain what's in Section "The contents of a geometric algebra" in [Chisolm(2012)], e.g. r-blades, r-vectors, before we can discuss anything about the GA operations.

That means we need to first formalize the counter parts in Clifford Algebra, e.g. Lipschitz Group, Spin Group, and Z-filteration in Clifford Algebra.

I'll take a stab at porting Jiale Miao's mathlib#16040 which seems to be a more principled attempt than versors in lean-ga except for the part involving Z-filteration which is still worth porting, possibly with ideas from the prototype here.

First step is to port it with mathport, then fill in the only sorry `invertible_of_invertible_ι`.

Definition 3.2.1 (Lipschitz group). TODO

Definition 3.2.2 (Spin group). TODO

3.3 Operations and properties

4 Concrete algebras - definition

4.1 CGA

4.2 PGA

4.3 STA

5 Applications

5.1 Geometry

6 Dependency graph

Figure 1: Dependency graph

Legends:

- `Boxes`: definitions
- `Ellipses`: theorems and lemmas
- `Blue border`: the statement of this result is ready to be formalized; all prerequisites are done
- `Blue background`: the proof of this result is ready to be formalized; all prerequisites are done
- `Green border`: the statement of this result is formalized
- `Green background`: the proof of this result is formalized

References

[Bourbaki(2007)] Nicolas Bourbaki. 2007. Éléments de mathématique. Algèbre. Chapitre 9 (reprint of the 1959 original ed.). Berlin: Springer.

[Carneiro(2019)] Mario Carneiro. 2019. The type theory of Lean. preparation (https://github.com/digama0/lean-type-theory/releases) (2019).

[Chen(2016)] Evan Chen. 2016. An infinitely large napkin.

[Chisolm(2012)] Eric Chisolm. 2012. Geometric Algebra. (2012). http://arxiv.org/abs/1205.5935

[de Moura et al.(2015)] Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. 2015. The Lean Theorem Prover (System Description). In Automated Deduction - CADE-25, Amy P. Felty and Aart Middeldorp (Eds.). Lecture Notes in Computer Science, Vol. 9195. Springer International Publishing, Cham, 378–388. https://doi.org/10.1007/978-3-319-21401-6_26

[Garling(2011)] David JH Garling. 2011. Clifford algebras: an introduction. Vol. 78. Cambridge University Press.

[Grinberg(2016)] Darij Grinberg. 2016. The Clifford algebra and the Chevalley map - a computational approach. https://www.cip.ifi.lmu.de/~grinberg/algebra/chevalley.pdf

[Hestenes and Sobczyk(1984)] David Hestenes and Garret Sobczyk. 1984. Clifford Algebra to Geometric Calculus: A Unified Language for Mathematics and Physics. Vol. 5. Springer Science & Business Media. https://www.springer.com/gp/book/9789027716736

[Izhakian et al.(2016)] Zur Izhakian, Manfred Knebusch, and Louis Rowen. 2016. Supertropical quadratic forms I. Journal of Pure and Applied Algebra 220, 1 (2016), 61–93. https://doi.org/10.1016/j.jpaa.2015.05.043

[Jadczyk(2019)] Arkadiusz Jadczyk. 2019. Notes on Clifford Algebras. (2019).

[Jadczyk(2023)] Arkadiusz Jadczyk. 2023. On the bundle of Clifford algebras over the space of quadratic forms. Advances in Applied Clifford Algebras 33, 2 (2023), 15.

[Moura and Ullrich(2021)] Leonardo de Moura and Sebastian Ullrich. 2021. The lean 4 theorem prover and programming language. In Automated Deduction–CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings 28. Springer, 625–635.

[The mathlib Community(2020)] The mathlib Community. 2020. The Lean Mathematical Library. In Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (New Orleans, LA, USA, 2020-01-20) (CPP 2020). Association for Computing Machinery, 367–381. https://doi.org/10.1145/3372885.3373824

[Ullrich(2023)] Sebastian Andreas Ullrich. 2023. An Extensible Theorem Proving Frontend. Ph. D. Dissertation. Dissertation, Karlsruhe, Karlsruher Institut für Technologie (KIT), 2023.