Formalization of Linear Algebra in Lean

Clea Bergsman [1] Katherine Buesing [2] Sahan Wijetunga [3] Supervisor: Dr. George McNinch [4]

[1]Bowdoin College [2]University of Minnesota, Twin Cities [3]University of California, Los Angeles [4]Tufts University

Introduction

Lean is a programming language that allows mathematicians to prove theorems and enables correct, maintainable, and formally verified code. Lean relies on dependent type theory, which means that every expression has a type, like $\mathbb{N}$ or $\mathbb{Q} \to \mathbb{R}$. Therefore, when introducing a new item, we have to specify its type, or include enough information to have it inferred from context. Mathematical statements in lean are of the type "Prop" and can be proved or used to prove more complex propositions in Lean. Lean's reliance on type theory removes inferences from traditional proofs by ensuring that theorems and properties apply to the specific type being used in the proof.

What is Formalization

Formalization is the process of proving theorems in Lean and is an increasing trend in mathematics. Formalizing involves a translation between two mathematical languages: traditional mathematical proofs and formalized proof codes. Formalization relies on Mathlib, the library of formalized proofs in Lean. The open-source nature of Mathlib facilitates mathematical collaboration between professionals on formalization, mathematicians, and graduate and undergraduate students. Mathlib also allows formalized proofs to be maintained and built upon by others.

Acknowledgments

We would like to thank the National Science Foundation for their support under REU Site grant DMS-2349058. We would also like to thank the Tufts Mathematics Department for their support, specifically Fulton Gonzalez, Kasso Okoudjou, Todd Quinto, and Banafsheh Akbari, for their mentorship and guidance.

Results

Our work mainly builds on mathlib's formalization of bilinear forms. Some results are listed below. We formalized various constructions, and proved theorems grounding them. This allowed us to prove independent classification results that are both interesting in their own right and a crucial tool in developing representation theory (among other things).

- Defined a notion of isomorphism of vector spaces with bilinear forms, and proves (defines) the natural structures associated to it being an equivalence relation
- Proved anisotropic, nondegenerate, symmetric, alternating forms are preserved under isomorphism
- Defined Hyperbolic Spaces, which allowed us to prove alternating forms of equal finite dimension are isomorphic and nondegenerate symmetric bilinear forms of equal finite dimension over an algebraically closed field are isomorphic.
- Proved Cassels-Pfister's Theorem, which yields as a corollary that a sum of n squares in $F(X)$ which lies in $F[X]$ is a sum of n squares in $F[X]$.
- Formalized degree constructions over $M[X] = R[X] \otimes_R M$ and compatibility with quadratic/bilinear form extensions
- Proved that the orthogonal complement of a subspace is nondegenerate, and that linear independence and bases are preserved under disjoint unions for transverse subspaces
- Proved any reflexive bilinear form is alternating or symmetric and vice versa

To view more of our work:

Scan the QR code in the top left corner or go to the following link to view our full GitHub repository with our Lean proofs.

https://github.com/gmcninch-tufts/VERSEIM-2025

Future of the Project

We have formalized various results regarding linear algebra and quadratic forms, including alternating, symmetric, and nondegenerate bilinear forms, as well as hyperbolic and transverse subspaces. Incorporating these results into Mathlib will allow additional results to be formalized by the Lean community. Additionally, further results can be formalized using these theorems we have proved. Specifically, further results in formalizing classification results for fields (for example, finite or algebraically closed), as well as describing all non-degenerate symmetric bilinear forms β on V up to isometry.

What Have We Learned

Because Lean is a language based on dependent type theory, the way we think about objects in Lean is different compared to human mathematics. For example, in human mathematics, a basis of a vector space V is both simply a basis, while also being a set of vectors. In Lean, a basis could be a basis object, a set of vectors that are linearly independent and span the space, or a function taking an index set to a set of vectors that act as a basis. These are all ways that we can choose to represent a basis, but the way that we interact with these different objects in Lean varies by its type. This forces us to really think about the way that we define mathematical objects and the minute differences between them.

References

1. Avigad, J., de Moura, L., Kong, S., & Ullrich, S. Theorem Proving in Lean 4. https://leanprover.github.io/theoremprovinginlean4/
2. Avigad, J., & Massot, P. (2020). Mathematics in Lean. https://leanprover-community.github.io/mathematicsinlean/index.html
3. Browning, T., & Lutz, P. (2022). Formalizing Galois Theory. Experimental Mathematics, 31(2), 413–424.