On doing a math[s] project with “Aristotle”

For the SFLean Meetup, July 6, 2026

Gerhard R. Paseman (with lots of help)

Archive version 2026.07.07

Preliminaries:

Disclaimers

- nothing new; just a little polemical
- things are changing: some particulars will be outdated soon
- this is my opinion; speaking with authority is pure posture
- there will be errors — questions and corrections welcome! (Even during the presentation!)

Introduce Aristotle from Harmonic

aristotle.harmonic.fun 26.07.07

Some Lean code

(generated by Aristotle, released under Apache 2.0 license)

(See accompanying files. This slide set and the Notes PDF
are generated by Gerhard Paseman.
All other files were generated by Aristotle
during the month of July 2026.
26.07.07)

Why Lean code?

- want some validation of informal argument
- want a formalization to be established and improved upon
- want a record of “pattern of proof” to be reused

How will this code be used?

Primarily to say “These results have been formalized in Lean and partially verified by the author” .

Secondarily to have something for others to build upon, if they so choose. The proof certificates/Lean code are not as important as the pattern and the intuition conveyed by an in-formalization of the formal proof.

What else?

Explain.md — an attempt by Aristotle to explain what it did and why and how. Lists resources used (Mathlib lemmata, references to machine-accessible and human-viewable literature)

Summary.md — an Aristotle generated summary used to hit the high points of the results being formalized and some of the reasons why the results should hold

Applications.md — an Aristotle generated list of applications and potential uses of these results in the literature.

Slides.tex — Aristotle generated version of the summary suitable for conference presentation.

What isn't here?

- A deep dive into the literature, covering the background behind the results
- An record of the motivations for the result and its potential applications
- A companion-style document with intermediate results and simple consequences for the student to consider
- A programmatic-style interface allowing reuse of the results by other entities
- A laundry list of related but not necessarily equivalent statements from the literature, to which future work may address ties to the present result

Of course, this is a lot to expect of a researcher over a short time frame. No one in their right mind would ask for all this. Right? (Polemics)

Mathematics as Software Engineering

Rightness of mind set aside, similar tasks are done in organizations making large-scale platforms involving software development.

Current tools allow some development and planning on the individual desktop. Aristotle is capable of (maybe ideal for?) handling the requirements mentioned.

You can become the professor/research director. You will need to develop the skills to plan, manage, direct, evaluate, and present.

The days of just researching and writing a paper are numbered. Instead of doing a math paper, do a math[s] project. You decide whether [s] stands for software or system.

Not all roses

In addition to learning to be a project manager, there will be the following concerns (partial list):

- Misformalization: need expertise to verify the formalization does what is needed. (Use experts, test cases, and adversarial techniques to see how bulletproof the formalization is)
- Inadvertant plagiarism: The text generators need material to generate text; sometimes plagiarism can occur. (Use current software doing plagiarism detection, be explicit as to who generated what and how, and encourage human review and reporting; follow up on correction)
- Faulty perspective: machine generated analogies have a use, but may mislead. Be prepared to have a competing perspective or way of presenting things that helps validate/refute the work

Takeaways

- The tools exist now to help with all of these.
- These tools are still “learning” how to do them well. You must help them.
- The proof artifacts are important to establish existence (“there is a correct argument”).
- The methods and strategies of proof are more important. These should be well-designed to be reused. (Don’t forget testing!)
- The communication and understanding of these patterns is equally important, and more important than the artifacts themselves.
- Make it easy to establish relevance and connections.

Acknowledgments

Thanks to the personnel at Mox for the resources and the venue.

Thanks to the SFLean group for motivating and reviewing this presentation. (Special thanks to Rado Kirov for his leadership and encouragement.)

Thanks to Harmonic for the creation, maintenance, and continuing enhancement of Aristotle.

(aristotle.harmonic.fun 26.07.07)

Brought to you by:

Gerhard “Ask Me About System Design” Paseman, 2026.07.06.