based on Coq. After five years using SF in the classroom, I have come to the conclusion that Coq is not the best vehicle for this purpose, as too much of the course needs to focus on learning tactics for proof derivation, to the cost of learning programming language theory. Accordingly, I have written a new textbook, Programming Language Foundations in Agda (PLFA). PLFA covers much of the same ground as SF, although it is not a slavish imitation.

What did I learn from writing PLFA? First, that it is possible. One might expect that without proof tactics that the proofs become too long, but in fact proofs in PLFA are about the same length as those in SF. Proofs in Coq require an interactive environment to be understood, while proofs in Agda can be read on the page. Second, that constructive proofs of preservation and progress give immediate rise to a prototype evaluator. This fact is obvious in retrospect but it is not exploited in SF (which instead provides a separate normalise tactic) nor can I find it in the literature. Third, that using raw terms with a separate typing relation is far less perspicuous than using inherently-typed terms. SF uses the former presentation, while PLFA presents both; the former uses about 1.6 as many lines of Agda code as the latter, roughly the golden ratio.

The textbook is written as a literate Agda script, and can be found here: https://plfa.inf.ed.ac.uk"

Project proposal; your AI for Math project idea

Which category does your project fall under? *

Field-Building (e.g. courses, textbooks, workshops, developer community support)

Write a few sentences about your proposed project. How will it advance the state of AI-for-mathematics? *

I wish to write a new version of Programming Language Foundations, this time based on the Lean proof assistant. The new book will be based closely on my existing textbook based on Agda, hence: Programming Language Foundations in Agda (PLFA) --> Programming Language Foundations in Lean (PLFL). The previous textbook is used in several courses (see https://plfa.inf.ed.ac.uk for a list) and many people have told me they learned Agda from reading it on their own. Lean has introductory texts aimed at mathematicians, but the goal here is to provide one aimed at computer scientists, broadening the user community. The book will focus on Lean as a proof assistant. A chapter or two will be devoted to how artificial intelligence (e.g., LLMs/copilot) can aid in constructing proofs. I have already secured an invitation to visit the Hoskinson Center for Formal Mathematics (https://www.cmu.edu/hoskinson/)---a centre of expertise in Lean---for one month in 2025, which will help me link to the Lean community.

If you have multiple applicants, explain what each person's role on the project will be.

By submitting this application, you consent to the terms of the grant agreement, including use of your personal data, as outlined in our Privacy Policy. You can withdraw your consent at any time by contacting privacy@renphil.org. *

Clear form

Submit