The Evolution of Proof Assistant Math Repositories

Mahsa Bazzaz
Northeastern University
Boston, Massachusetts, USA

Minsung Cho
Northeastern University
Boston, Massachusetts, USA

Gwen Lincroft
Northeastern University
Boston, Massachusetts, USA

1 INTRODUCTION

Proof assistants (also known as interactive theorem provers) are programming languages with the primary purpose of providing computational meaning to mathematical proof through code. The first proof assistants, developed over fifty years ago, were designed to help reason about the correctness of software and hardware through mathematical abstraction. However, as proof assistants became more expressive and user-friendly, they have gained traction in numerous fields, including formal methods, artificial intelligence, and computer science education TODO: add citation..

However, over the past 20 years, proof assistants garnered attention from communities beyond computer science. In particular, the mathematics community took interests in proof assistants with their potential to bridge software development and mathematical proof. This potential has been realized through multiple open-source, well-documented libraries of formalized proofs as programs. Although there have been many such libraries over time, the predominant ones today are written in three languages: Lean, Coq, and Isabelle/HOL.

Lean has been the focal point of math library development in the past five years, gaining the attention of prominent mathematicians by demonstrating its ability to prove research-level mathematics TODO: add citation.. Its math library, known as `mathlib` , has been open-sourced on GitHub under an Apache license. It has extremely rigid conventions for contributing, including guidelines on style, naming, commit messages, and pull request labeling.

Coq is one of the most widely-used proof assistants in computer science today. However, its math library is not centralized like `mathlib` . Instead, it offers a *loosely federated* list of open-source GitHub repositories. Many of these repositories are under the same GitHub organizations, for example `math-comp` or `coq-community`, but not all of them are and their interdependence is not obvious.

Isabelle/HOL is another major proof assistant that offers an extremely strong general automation, which the other proof assistants lack. Although its math library is open source, it is not hosted natively on GitHub and external contribution is not done through GitHub's standard means. In particular, contribution is vetted behind closed doors by academics and is not done through a standard pull request-merge system. Furthermore, there is no centralized repository á la `mathlib` and most are hosted on the website *Archive of Formal Proofs*, where the proofs lack version control but give a detailed account of authors, dependencies, and proof techniques.

It is clear that each of these proof assistants and their math libraries have different software development practices. However, they all attempt to demonstrate the power of proof assistants by formalizing known mathematics through code. This project aims to compare and contrast the evolution of these three main repositories through a software engineering lens. In particular, for this project we have two main research questions:

(1) What mathematical theorems did different formalized mathematics libraries prove? How has this developed over time? How have the proofs changed?
(2) How has the popularity of contributing to theorem provers over time changed? Are there any factors we need to take into account to evaluate proof assistant software different than traditional software?

2 PROJECT RESULTS

Methodology

To investigate research problem 1, we used Freek Wiedijk's *list of 100 theorems*TODO: add citation., an online list of "top 100" mathematical theorems that proof assistants to attempt to formalize.

We first gathered exactly what proportion of the 100 theorems have been formalized fully in the respective proof assistants. Then, we used `Octokit` to collect data of relevant commits to the file containing each theorem. For Lean, we collected data only from the *mathlib* repository, where all the proofs were held. For Coq, we looked at 30 different repositories where the proofs were held. For Isabelle, we either looked at the *mirror-afp-2022* repository or had to take data directly from the *Archive of Formal Proofs* website.

Using the data, we selected the following information:

- when each theorem was proven,
- which theorems required the most commits
- and what type of commits–code refactoring, dependency updating, or simply writing new supporting definitions and lemmas–were typically made.

To investigate research problem 2, we again used `Octokit` to take data on pull requests and issues. However, this was only doable on Lean and Coq, as Isabelle does not have any pull requests or issues. Using the data, we selected the following information:

- the proportion of pull requests merged to the main or master branch, closed, and still open,
- the proportion of issues opened or closed,
- the rate of growth of the repositories, measured as

$$\frac{\#\text{ of commits} + \#\text{ of opened PRs} + \#\text{ of opened issues}}{\text{month}} \tag{1}$$

- the general type of pull request and issues seen in math repositories, such as suggestions, bugfixes, or new additions.

With this, we present our results.

Results for research problem 1

We first observed that Lean is the focal point of math library development in the past five years, Isabelle actually has the most theorems proved on the 100 Theorems list, with 87. Coq follows with 79, and Lean is actually the least populated with 76. Thus, in objective number of theorems proved, Lean is actually not the dominant proof assistant for formalized math.

However, the pace at which Lean's userbase is contributing to its codebase in the past 5 years far outpaces that of Coq or Isabelle.

We see in Figure 1 that in the past five years Lean has significantly more commits to its main repositories than both Coq and Isabelle. Before that Isabelle had a commanding lead, although its activity seems to have diminished with the emergence of Lean. However, Isabelle's frequent activity in the decades preceding Lean explains its 87 theorems proven out of the 100 theorems list.

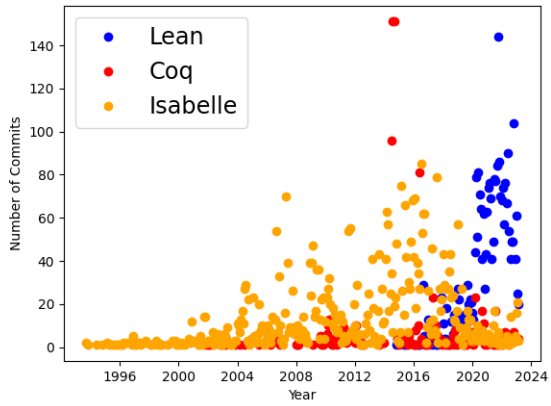

Figure 1: Number of commits per year for Lean, Coq, and Isabelle.

3 PROJECT CHALLENGES

Limits in project scope

The original goal of this project was to investigate more proof assistants than Lean, Coq, and Isabelle. However, this proved challenging as many proof assistants predate GitHub and modern software engineering practice, including style and version control. In particular, we observed that some prominent theorems according to the 100 Theorems list, such as Mizar and Metamath, only host proofs on the proof assistant website itself. However, we do believe that scraping the websites for relevant statistics about the proofs is worthwhile; we leave it to future work beyond the time frame of this project.

An additional original goal of the project was to investigate the use cases of each proof assistant over time by examining what type of projects on GitHub were written in the language. However, this was difficult to do considering the time constraint. Not only does the GitHub rate limit affect the speed at which we can gather information about individual projects, but it would also be time-consuming to parse the information and categorize it into appropriate metrics for use case. As such, we also delegate this goal to future work.