Thesis abstract: “Design and development of a tool based on Coq to write and format mathematical proofs”

Théo Zimmermann   Hugo Herbelin (supervisor)

Coq is an interactive proof assistant relying on a foundation language which is both a logical framework and a strongly-typed programming language. It has recently increased in popularity thanks to two ACM prizes and some significant proof developments by George Gonthier and his team. Foundational mathematicians have started to be really interested in Coq, in particular in the links between type theory and homotopy theory. Yet, it is still a tool that requires some “hacker” skills to use.

A general aim of my PhD thesis is to progress along the path of making Coq easier to use for any mathematician. In particular, I wish to make it attractive by transforming it into a tool to write and format mathematical proofs.

Mathematical writing

There has already been a lot of work in this direction, for various proof assistants. We can cite in particular Mizar and its Journal of Formalized Mathematics, and Martijn Oostdijk’s and Yann Coscoy’s PhD theses.

I plan to create a tool to transform proof scripts through a number of steps, each adding constraints to the form the proof script should take. Coq’s tactic language is indeed very flexible and the proof scripts can take many forms and styles. We will target a style that is as close as possible to informal mathematical writing so that the last step, which will be to transform a proof script into English (and LaTeX), is as straight-forward as possible. Preliminary experiments have shown that it is indeed possible to produce such a “mathematical” style in Coq, using its tactic language, unmodified. Additionally, we want to make the last translation (from Coq to English) reversible, so that the generated articles can be re-interpreted and checked by Coq.

Technically, this tool will be based on Arnaud Spiwack’s infrastructure to instrument proof script execution, which is available in Coq 8.5. This will allow transforming the scripts while they are being written, thus providing useful feedback to the mathematician.

On a theoretical level, one of the goals is to understand how transformations between various proof styles maintain the ability to reconstruct the implicit information (information which can be reconstructed by the unification algorithm or some decision or semi-decision procedure).

In this thesis, we restrict ourselves by targeting only arithmetical proofs. However, we also want to make the tool easily extensible, so that other domains can be added. I will evaluate the quality of the tool by using it on some examples of formalization projects and promote its testing by enthusiastic early adopters.

CICM being a community in which this project perfectly fits, I will try to attend each of its meetings where I will strive to publish and present my progress.

Internship work continued: Theorem transfer

I had previously started working on ideas which make proof formalization closer to pen-and-paper mathematics during an internship with Hugo Herbelin, now my PhD supervisor. The main goal of the internship was to help reason modulo isomorphism. The work I did then was to devise and implement an algorithm to automatically transfer theorems between two related types, given the right kind of user-provided declarations.

I generalized upon the work of Matthieu Sozeau, on generalized rewriting, and of Cyril Cohen, et al., on type refinements, and obtained a set of inference rules which allow the transfer to take place. I presented this work at CICM 2015 in the work-in-progress track. The anonymous referees then pointed me to other works on the topic, in the context of Isabelle, of which I had not yet been aware. The ideas behind Isabelle's Transfer package, which are described in great detail in Ondřej Kunčar's PhD thesis, are very close to my own, which, while validating the path I had taken, also make it harder to publish my own work as novel.

Since the conference, I have finished implementing a prototype of my transfer method. The implementation is a small library[1], relying on Coq's type class mechanism. It is able to transfer all sorts of theorems, including theorems about predicates, such as induction principles. Following my work and Jérémie Koenig's binary logical relation library, Matthieu Sozeau is now, with my help, extending once more the rewriting capabilities of Coq to include rewriting with heterogeneous binary relations (relations between elements of two distinct types). I foresee that theorem transfer will then become a special case of generalized rewriting, benefiting at the same time from a more robust implementation.

[1]This library is available at https://github.com/Zimmi48/transfer.