Type theory for education

Johan Commelin, Paige Randall North, Jim Portegies

In this project, we will train a PhD student in the formalization of mathematics in computer proof assistants (with expertise provided by Johan Commelin, UU), the mathematical foundations of proof assistants (with expertise provided by Paige Randall North, UU), and the practical design of proof assistants for mathematical education (with expertise provided by Jim Portegies, TU/e). The end result of the student's PhD research will be significant improvements to systems such as Lean-verbose and Waterproof, the latter a proof assistant for mathematical education pioneered by Portegies. Waterproof is currently being tested in the course Analysis 1 at TU/e. We aim to develop a version of Waterproof that will be used to improve student outcomes at TU/e, UU, and beyond.

Background on proof assistants for education, Lean-verbose and Waterproof

University students often struggle with learning how to prove mathematical statements [27, 21]. To ameliorate this, many educators have turned to the use of proof assistants [24, 22, 13, 20]. A proof assistant, such as Lean [9] and Coq [8], is software in which users interactively build proofs with assistance and feedback from the computer. This assistance includes some degree of automation of the proof, often similar to what a human reader of a proof can accept as obvious. This feedback includes certification that a proof is finished and correct, or otherwise information about what remains to be done. Proof assistants in mathematics education thus constitute a kind of automated, personal 'tutor' for each student.

Despite these benefits of using proof assistants, they are not yet perfectly suitable for use in mathematics education. That is because the act of writing a proof in a proof assistant and the resulting proof can seem very different, especially to a student, from the act of writing on paper and the resulting pen-and-paper proof [14, 4]. Other issues are the steep learning curve of proof assistants [5], the sometimes outdated user interfaces interacting with proof assistants, and the fact that feedback from proof assistants is often geared towards experts and mostly concerns correctness of proof steps, while students need other types of feedback as well.

To address these issues and make proof assistants more effective for education, Portegies has begun developing *Waterproof* [31, 30], a combination of a modern editor and a library for the proof assistant Coq [8], that allows for the use of controlled natural language in proof scripts, similar to Massot's Lean-verbose for the proof assistant Lean [18]. Indeed, to make proof assistants more effective for education, Waterproof has the following design principle strongly at its core:

Writing a proof in Waterproof should be as close as possible to writing a proof on paper, both in terms of end result and in terms of process.

Such a proof assistant will not only be useful for education, but also for the wider adoption of proof assistants by mathematicians in general. Indeed, experience with Waterproof shows that the Waterproof proof scripts can be understood by mathematicians with no training in proof assistants.

Since the beginning of its development, Waterproof has been used in the course Analysis 1 at the TU/e. We have observed that even in their handwritten proofs, students use the structuring trained by Waterproof.

Current challenges

Although we see this improvement in the structure of student proofs after they have worked with Waterproof, challenges remain. **(1)** Students and teachers have mentioned the desire to have the Waterproof definitions, notations and use of concepts as close as possible to the way they are used in lectures. This is challenging because the foundations of the mathematics being used is different: mathematics education is predominantly built on a foundation of set theory, while modern proof assistants are built on type theory. **(2)** It is currently difficult for teachers to start using Waterproof for a new course, which limits its scalability. Despite the existence of large libraries of formalized mathematical theory, direct exposure to these libraries would both be too challenging to students and would violate Waterproof's core design principle. **(3)** In our evaluations (e.g. [6]), students report that the feedback from the computer is often unclear and confusing, and that the automation system is sometimes too weak, sometimes too strong, and sometimes unpredictable (for students) in what it can and cannot solve.

Supervising team

In order to solve these challenges, one requires a deep understanding of the underlying type theory, the ability to use it creatively to find out-of-the-box solutions, didactical and subject knowledge of the topics to be taught, and expertise in efficiently formalizing proofs in mathematics. This is why in this project we will combine the expertise of Johan Commelin, with extensive experience in the formalization of mathematics, that of Paige Randall North, with deep understanding of type theory, and that of Jim Portegies who has significant experience in using this type theory for and in education.

Goals of this project

The overarching goal of the PhD project is:

To redesign the type-theoretic representation of the mathematical theory in the Waterproof library, to make it more suitable for use in education, while at the same time allowing for the use of existing libraries with formalized mathematics.

This overaching goal breaks down into the following interdependent subprojects.

Subproject 1. Design Waterproof theory interface with the use in Mathematics education at its core.

First, we hope to add features to the underlying type theory and Waterproof itself that make the type theory behave more similarly

to set theory. One difference that is currently the most pressing is in the use of subsets. In set theory an element can be and is a member of many different sets, but in type theory the analogous statement is not true: the type of a term is unique. For students, a natural number should also be an integer, but in type theory the natural numbers and integers are kept separate and the natural inclusion $i : \mathbb{N} \to \mathbb{Z}$ is explicitly used instead (see also [2]). This calls for a better type-theoretic representation of subsets in Waterproof. There is some support to suppress explicit mentions of inclusions like I (called coercive subtyping) in the underlying proof assistant, Coq, but this often fails. There is much foundational work to improve this (e.g. [15, 16, 17]), but it is difficult to implement in Coq because of the wide-ranging needs of Coq users. We will build on this work, and improve coercive subtyping in Waterproof which has an audience with more specific needs.

A proof assistant, such as Coq, can treat different definitions of the same object as the same, and it can perform implicit coercions, as discussed above, for objects other than subtypes. In particular, in a proof that is expected to contain several computational steps, Coq can do the computation without any input from the user. Coq has been designed to assist users in writing proofs, but in education we often want students to explicitly write that two things are the same or do a computation. For instance, in an earlier project with the University of Marburg, the requirement came up that simple computational proofs with matrices should be possible. We would like to design Waterproof so that teachers can force users to explicitly call on definitions or alternative characterizations when using certain concepts. This requires a different approach to definitions, making them opaque to the underlying proof assistant, similar to the system of *locking* in Coq. We will follow recent work such as [11].

Subproject 2. Improve scalability: connect with existing libraries
The development of a large library of formalized mathematics comes with several design goals that are in tension with each other: generality of the material, readability of the proofs, robustness/maintainability of the formalization, and the speed with which the library can be verified. As one of the maintainers of `mathlib`, Commelin has extensive experience with these design goals and the trade-offs between them. In this project, with its focus on education, we will study how to create derivative libraries that focus on improved readability, and possibly trade off some generality and speed.

In this subproject, we will develop tools to aid in creating derivative libraries on top of large existing libraries, while at the same time furthering the design of didactical libraries such as Lean-verbose and Waterproof so that they can connect with the derivative libraries in a way optimized for education. As part of this work, we will develop a tool that can translate between formal tactic scripts and proofs in a controlled natural language. Important prior work in this respect is the mechanized informalization project by Massot and Miller [19], which translates formal Lean proofs to LaTeX proofs in natural mathematical English. Recent efforts in the group of Benjamin Pierce on deautomation are also likely to be helpful. Another building block is the tactic combinator `says` in Lean, that allows to write two versions of a proof, that can have different properties in terms of speed or readability, etc. Configuration flags determine which version of the proof is used.

Subproject 3. Improve automation
As mentioned before, currently the automation is sometimes too weak, sometimes too strong, and towards students it is unpredictable. Moreover, students can currently extract little more information from the feedback from the automation other than that it fails. Part of the reason is that the current automation heavily relies on the reflection-based `lia` and `lra` tactics [3], which are powerful but intransparent and for some problems really not suited. Another reason very much related to Subproject 2 is that the automation system is not yet geared to the implementation of some of the large libraries of formalized mathematics.

We will focus on the automation for proving mathematical equalities and inequalities. We will specify what type of (in)equalities the automation should be able to prove, design a tunable algorithm to prove such (in)equalities that can give feedback on its functioning and prove that the algorithm works. In writing our automation, we will build on work by Patiachvili in an earlier internship in which he wrote an OCaml plugin to make the automation more flexible and transparent [23], `mathlib`'s `rewrite_search` tactic, Lean's `calc` blocks, and investigate to what extent we can still incorporate small and large scale reflection techniques [1], such as those in the `mathcomp algebra-tactics` library [25].

Embedding of the project
The research is timely, because more and more mathematicians start to value the verification of their proofs by a computer. In the past five years, several high profile projects took place. Peter Scholze initiated the Liquid Tensor Experiment [26], a challenge that Commelin took up and brought to a successful conclusion [7, 12]. Timothy Gowers launched a project on automated theorem proving [10]. Very recently, Terence Tao has formally verified two of his recent preprints together with several Lean experts [28, 29]. There is currently a large amount of expertise in this emerging area in the Netherlands (Dahmen and Mahboubi at VU, Cockx and Ahrens at TU Delft, Wiedijk and Geuvers at RU, Portegies at TU/e, North and Commelin at UU). We want to use this project to take advantage of and increase the particular strength that the Netherlands has compared to other countries in this domain. In particular, we want to ensure that UU and TU/e take a leadership role in this domain. This project will further develop this expertise already present here, in particular strengthening and expanding it in the direction of type theory for education, and, as a side-effect, type theory for the wider adoption of the formal proof verification as an useful and approachable tool for mathematicians.

Outlook
The proposed PhD project covers the most urgent and important next steps for proof assistants for education. To keep the scope of this project focused, we have refrained from explicitly mentioning the use of generative AI. For the subsequent steps, which will need to involve generative AI and didactical evaluation of the developed tools, we will involve a larger team in an NWO XL proposal. It is our expectation that in the coming decade, mathematics education and research will change drastically through the combination of generative AI and proof assistants. With this research we hope to contribute to the best possible outcome of these

developments.

References

[1] S.F. Allen, R.L. Constable, D.J. Howe, and W.E. Aitken. The semantics of reflected proof. In *[1990] Proceedings. Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 95–105, 1990.

[2] Yves Bertot. The challenges of using type theory to teach mathematics. Talk at ThEdu'23 (12th International Workshop on Theorem proving components for Educational software), 2023.

[3] Frédéric Besson. Fast reflexive arithmetic tactics the linear case and beyond. In *Types for Proofs and Programs: International Workshop, TYPES 2006, Nottingham, UK, April 18-21, 2006, Revised Selected Papers*, pages 48–62. Springer, 2007.

[4] Jasmin Blanchette, Jeremy Avigad, Julien Narboux, Heather Macbeth, Gihan Marasingha, and Patrick Massot. Panel: Teaching with proof assistants. Lean Together 2021, Jan 2021.

[5] Sebastian Böhne and Christoph Kreitz. Learning how to Prove: From the Coq Proof Assistant to Textbook Style. In Pedro Quaresma and Walther Neuper, editors, Proceedings 6th International Workshop on *Theorem proving components for Educational software,* Gothenburg, Sweden, 6 Aug 2017, volume 267 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–18. Open Publishing Association, 2018.

[6] Dorina Bor. Improving the user experience of an Intelligent Tutoring System for proving mathematical statements, Februari 2023.

[7] Davide Castelvecchi. Mathematicians welcome computer-assisted proof in 'grand unification' theory. *Nature*, 595:18–19, June 2021. https://doi.org/10.1038/d41586-021-01627-2.

[8] The Coq Development Team. The Coq Proof Assistant, January 2022.

[9] Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The Lean Theorem Prover (System Description). In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25*, pages 378–388, Cham, 2015. Springer International Publishing.

[10] Timothy Gowers. Human-oriented automatic theorem proving. https://wtgowers.github.io/human-style-atp.

[11] Daniel Gratzer, Jonathan Sterling, Carlo Angiuli, Thierry Coquand, and Lars Birkedal. Controlling unfolding in type theory, 2022.

[12] Kevin Hartnett. Proof assistant makes jump to big-league math. *Quanta Magazine*, July 2021. https://www.quantamagazine.org/lean-computer-program-confirms-peter-scholze-proof-20210728/.

[13] Martin Henz and Aquinas Hobor. Teaching Experience: Logic and Formal Methods with Coq. In Jean-Pierre Jouannaud and Zhong Shao, editors, *Certified Programs and Proofs*, pages 199–215, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[14] Maria Knobelsdorf, Christiane Frede, Sebastian Böhne, and Christoph Kreitz. Theorem Provers as a Learning Tool in Theory of Computation. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*, ICER '17, page 83–92, New York, NY, USA, 2017. Association for Computing Machinery.

[15] Théo Laurent, Meven Lennon-Bertrand, and Kenji Maillard. Definitional functoriality for dependent (sub)types, 2023.

[16] Georgiana Elena Lungu and Zhaohui Luo. On Subtyping in Type Theories with Canonical Objects. In Silvia Ghilezan, Herman Geuvers, and Jelena Ivetic, editors, *22nd International Conference on Types for Proofs and Programs (TYPES 2016)*, volume 97 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:31, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[17] Harry Maclean and Zhaohui Luo. Subtype Universes. In Ugo de'Liguoro, Stefano Berardi, and Thorsten Altenkirch, editors, *26th International Conference on Types for Proofs and Programs (TYPES 2020)*, volume 188 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 9:1–9:16, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[18] Patrick Massot. Lean verbose, 2021. https://github.com/PatrickMassot/lean-verbose.

[19] Patrick Massot. Formal mathematics for mathematicians and mathematics students, 2023. https://www.youtube.com/watch?v=tp_h3vzk0bo.

[20] Hendriks Maxim, Cezary Kaliszyk, Femke van Raamsdonk, and Freek Wiedijk. Teaching logic using a state-of-art proof assistant. *Acta Didactica Napocensia*, 3(2):35–48, June 2010.

[21] Robert C. Moore. Making the transition to formal proof. *Educational Studies in Mathematics*, 27(3):249–266, Oct 1994.

[22] Tobias Nipkow. Teaching Semantics with a Proof Assistant: No More LSD Trip Proofs. In Viktor Kuncak and Andrey Rybalchenko, editors, *Verification, Model Checking, and Abstract Interpretation*, pages 24–38, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[23] Balthazar Patiachvili. Improvement of the automation in the coq-waterproof library. Technical report, École normale supérieure Paris-Saclay, 2023.

[24] Krzysztof Retel and Anna Zalewska. Mizar as a Tool for Teaching Mathematics. *Mechanized Mathematics and Its Applications*, 4, Special Issue on 30 Years of Mizar:35–42, March 2005.

[25] Kazuhiko Sakaguchi. Reflexive Tactics for Algebra, Revisited. In June Andronick and Leonardo de Moura, editors, *13th International Conference on Interactive Theorem Proving (ITP 2022)*, volume 237 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 29:1–29:22, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[26] Peter Scholze. Liquid tensor experiment. *Exp. Math.*, 31(2):349–354, 2022.

[27] Annie Selden and John Selden. *Overcoming Students' Difficulties in Learning to Understand and Construct Proofs*, page 95–110. MAA Notes. Mathematical Association of America, 2008.

[28] Terence Tao, 2023. `https://mathstodon.xyz/@tao/111360218854032278`.

[29] Terence Tao, 2023. `https://mathstodon.xyz/@tao/111526765350663641`.

[30] Jelle Wemmenhove, Thijs Beurskens, Sean McCarren, Jan Moraal, David Tuin, and Jim Portegies. Waterproof: educational software for learning how to write mathematical proofs. *arXiv preprint arXiv:2211.13513*, 2022.

[31] Jelle Wemmenhove and Jim Portegies. Waterproof. `https://github.com/impermeable/waterproof-vscode`.