Coq Platform docs: A Compilation of Short Interactive Tutorials and How-To Guides for Coq

Thomas Lamiaux[1], Pierre Rousselin[2], and Théo Zimmermann[3]

[1] ENS Paris-Saclay `thomas.lamiaux@ens-paris-saclay.fr`
[2] LAGA, Université Sorbonne Paris Nord `rousselin@math.univ-paris13.fr`
[3] LTCI, Télécom Paris, Polytechnic Institute of Paris `theo.zimmermann@telecom-paris.fr`

1 Motivation

Having a proper, clean and accessible documentation is one of the keys to the success of software. There are different forms of documentation: abstract and detailed documentation like the reference manual [8], course-shaped documentation like Coq'Art [2] or Software Foundations [5], or short action-oriented documentation. We focus on the latter. Short action-oriented documentation provides users with practical information on specific features of Coq, so that users can learn and discover new features by themselves or consult it when they fail to use them efficiently.

At this point, Coq has only minimal and scattered action-oriented documentation about specific features or topics. The reference manual is by design not learning-oriented and not action-oriented, and it would be a mistake to try to bend it that way. Books like Coq'Art or Software Foundations provide nice pedagogical explanations but target specific audiences, and are rather meant to be read from cover to cover. Moreover, they are not well suited for learning about specific features, to discover horizontally, and are not easy to keep updated.

Yet, short action-oriented documentation has many interests:

- Having an easy to access documentation, accessible through a nice centralized online interface is of utmost importance to engage new users and keep current users. We cannot expect users to have to dig on their own through the reference manual, books, or GitHub repositories of ecosystem packages to learn how to use or get information while working about a specific feature. Moreover, these sources may not contain the basic answers they are looking for, due to their nature.
- Not having such a documentation prevents people from actually discovering and learning by themselves new amazing features, as well as the richness of our ecosystem [1]. Indeed, many features and packages are still currently under-documented, and when existing, the documentation is often scattered out, making it hard to discover a feature if one is not already an expert. A symptom of that is the trouble that students are currently facing to find answers or discover new functionalities by themselves, even sometimes about basic features.
- Even advanced users need help to keep up to date with recent developments inside of Coq (`SProp`, universe polymorphism, Ltac2, ...) or around it (MetaCoq, coq-elpi, Hierarchy Builder, ...).
- Writing proper documentation forces us to explain the different aspects of a feature clearly, step by step, and without relying on recipes to get the code working, thus to understand it better. Therefore, we hope that by writing the documentation, we will clarify the use of many features, and potentially discover or shed light on bugs or weaknesses. Actually,

writing tutorials for Equations has already revealed different issues with the main tactic funelim and bugs involving rewrite.

- Most users are currently unaware of the extent of what has been formalised and is available in Coq. There are many libraries, and it is not easy to know which library to use, or to know on which axioms they rely or their compatibilities. This is obviously not just a documentation issue, but having a clearer documentation of what we have and where would help.

2 Description of the project

This project aims to create an online compilation of short and interactive tutorials and how-to guides for Coq and the Coq Platform [7, 4]. Each core functionality and plugin of Coq and the Coq Platform should have (short) pedagogical tutorials and/or how-to guides demonstrating how to use this functionality, with practical examples. They should further be available online through an interactive interface, most likely using jsCoq [3], and a non-interactive interface (for reading on mobile devices).

Tutorials and how-to guides serve different purposes and are complementary. Tutorials guide a user during learning in discovering specific aspects of a feature like “Notations in Coq”, by going through (simple) predetermined examples, and introducing notions gradually. In contrast, how-to guides are use-case-oriented and guide users through real life problems and their inherent complexity, like “How to define functions by well-founded recursion and reason about them”.

This form of documentation is complementary to other kinds of documentation and has several advantages:

- Tutorials should enable users to learn and discover specific features on their own, modularly, and according to their needs.
- How-to guides should provide users practical answers to practical problems that they can refer to when working.
- By nature, the documentation should be mostly horizontal, which should:
  - make it easy to navigate and to find specific information,
  - prevent users from having to read a bunch of documentation to be able to read a specific tutorial,
  - make it possible to build it gradually, making new tutorials and how-to guides available as we progress
  - allow differentiated learning: depending on your background or objective you can navigate the documentation differently, potentially reading different tutorials.
- It will enable us to showcase all that is possible in Coq’s ecosystem.

At the moment, we use a GitHub repository that people can check out to discover the project, and a dedicated Zulip stream to discuss the project. A few tutorials are already available on the repository. They have proven useful by providing practical answers that are otherwise hard to find without asking directly to an expert. A first online interactive interface based on JsCoq and coqdoc is currently being developed and will be available soon. In the future, we hope to support a more standard and expressive format, possibly using Alectryon [6].

To further discuss the project, we have written a Coq Enhancement Proposal. We hope that, eventually, this documentation will become part of the website that will be created when renaming Coq to *the Rocq Prover*.

References

[1] Andrew W Appel. Coq's vibrant ecosystem for verification engineering (invited talk). In *Proceedings of the 11th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 2–11, 2022.

[2] Yves Bertot and Pierre Castéran. *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions*. Springer Science & Business Media, 2013.

[3] Emilio Jesús Gallego Arias, Benoît Pin, and Pierre Jouvelot. jsCoq: Towards Hybrid Theorem Proving Interfaces. In S. Autexier and P. Quaresma, editors, *Proc. of the 12th Workshop on User Interfaces for Theorem Provers (UITP 2016)*, volume 239 of *Electronic Proceedings in Theoretical Computer Science*, pages 15–27. Open Publishing Association, 2017. https://doi.org/10.4204/EPTCS.239.2.

[4] Karl Palmskog, Enrico Tassi, and Théo Zimmermann. Reliably reproducing machine-checked proofs with the Coq Platform. In *RRRR 2022-Workshop on Reproducibility and Replication of Research Results*, 2022.

[5] Benjamin C. Pierce, Arthur Azevedo de Amorim, Chris Casinghino, Marco Gaboardi, Michael Greenberg, Cătălin Hriţcu, Vilhelm Sjöberg, and Brent Yorgey. *Logical Foundations*, volume 1 of *Software Foundations*. Electronic textbook, 2023. Version 6.5, http://softwarefoundations.cis.upenn.edu.

[6] Clément Pit-Claudel. Untangling mechanized proofs. In *Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering*, pages 155–174, 2020.

[7] Michael Soegtrop and contributors. The Coq Platform. https://github.com/coq/platform, 2019–2024.

[8] The Coq Development Team. The Coq reference manual, version 8.19.0. https://coq.inria.fr/doc/V8.19.0/refman/, 1999–2024.