The Coq Proof Script Visualiser (coq-psv)*

Mario Frank

University of Potsdam, Institute of Computer Science, Potsdam, Germany
mario.frank@uni-potsdam.de, orcid:0000-0001-8888-7475

The Coq proof assistant [2] is used in various scientific, industrial, and teaching contexts and enables scientists, teachers, and students to exchange formalisations in an easy way. But this exchange is limited to the vernacular files and the respective LaTeX representation generated by coqdoc [3], for example. Also, those representations merely contain the definitions and proof scripts but not the information about the specific proof steps, i.e. the current goals and hypotheses. Thus, in order to fully understand the specific proofs, including the way how hypotheses change, the recipient needs to acquire a compatible version of Coq. This can be cumbersome as the installation of the specific Coq version and necessary additional tools is not always straight-forward due to operating system and Opam version differences. To our knowledge, there currently is no tool that is able to represent Coq proofs on paper including all used tactics, goals and hypotheses in a readable and printable manner as tree representations usually do not scale well for long proofs.

Applications. The Coq Proof Script Visualiser aims at supporting the development, teaching, and review process of proofs by exporting the proof scripts together with the current proof state for each proof step. As all information about the proof is then at hand, readers merely need to know the semantics of the used tactics to understand the proof. Proof developers can benefit from additional warnings given by coq-psv when superfluous structuring is used. Reviewers can retrace the proof more easily without the need to install Coq. In teaching, the generated output can be easily modified to be used as a cloze where students have to fill in the matching tactic or introduced hypothesis, for example. And most notably, the exported proof representations can be used for archiving purposes and enriching the output of coqdoc.

Approach. Given a Coq file, coq-psv processes it using the Coq parsing functionality and data structures. Although the output of coq-psv is linear, it internally generates a labelled proof tree for each proof in the file as using a tree structure simplifies the analysis and augmenting of the proof information. The label itself contains the name and type of the proven statement (e.g. theorem or lemma) together with the statement itself and the information, how the proof was completed (Qed or Admitted). The tree consists of nodes that contain the used tactic or command, and the resulting hypotheses and goals. While the extraction of the label content is easy, building the tree nodes is much more complex. This is due to the numerous ways in Coq to branch in a proof situation, which includes bullets, brackets and the (deprecated) Focus command. After extracting the labelled trees, they are translated to LaTeX tables according to the given command line options. We will give insight into the way we handle different branching options and why we do so in the talk.

Versatile Use. The coq-psv supports many command line options that define the output behaviour. It is able to process both single Coq files and complete Coq projects. Furthermore, it is possible to generate either complete LaTeX documents or only the LaTeX tables. Also, all proofs from a vernacular file can be exported either to one single file or to separate files per proof. The latter simplifies the integration into other LaTeX documents. There are two different table layouts: The first one mimics the goal and hypothesis visualisation of CoqIDE while the second one represents the goals and hypotheses in sequent style. Also, it is possible to hide goals and hypotheses in the step in which they are created if all of them are handled by bullets. This can reduce the size of the output significantly. Hiding hypotheses that remain unchanged until the end of the (sub-)proof and marking them as invariant

Next step in Coq	Proof situation
Proof.	$\forall\ L\ K : List\ \mathbb{N},\ list\_in\_strorder\ L \to list\_in\_strorder\ K \to L \subseteq K \to K \subseteq L \to L = \{List\ \mathbb{N}\}\ K$
prove_by_induction.	Hidden 2 goal(s)
$+_{1/2}$	$\forall\ K : List\ \mathbb{N},\ list\_in\_strorder\ [] \to list\_in\_strorder\ K \to [] \subseteq K \to K \subseteq [] \to [] = \{List\ \mathbb{N}\}\ K$
prove_all_imp_star.	K : $List\ \mathbb{N}$ H : $list\_in\_strorder\ []$ H0 : $list\_in\_strorder\ K$ H1 : $[] \subseteq K$ H2 : $K \subseteq []$ --- $[] = \{List\ \mathbb{N}\}\ K$
exception H2.	$H2 : K = \{List\ \mathbb{N}\}\ []$ --- $[] = \{List\ \mathbb{N}\}\ K$

Figure 1: Excerpt of a proof visualisation

*Special thanks to Sebastian Böhne who gave the idea and concept to the tool.

on introduction is also supported, which again condenses the size of the tables (see Figure 1). Finally, the layout of the tables can be modified after the generation. This can be done by overriding the default LaTeX commands used in the generated files before importing them. For example the size of the table itself and of specific table columns can be set to a desired value without having to modify the table itself.

Related work. There are already some approaches to representing proof scripts in a more readable and independent way (see [5, 7, 8, 9, 11, 10], for example). But except for [5] and [11], all of them represent the proofs only as trees or diagrams. While trees and diagrams can give a good overview about the structure of a proof, printing them in a readable way can be problematic for long proofs. Although the approach described in [5] supports both a tree and a Fitch style representation, the latter only shows the current step and it does not seem to be possible to export the complete proof as LaTeX file, for example. Also, only ProofWeb [5], Proviola [11], ProofTree [10] and Traf [7] are targeting Coq and the two latter only work in a live session with Coq and, for example, ProofGeneral [1]. Furthermore, Traf uses ProofTree and both seem to have been discontinued for more than two years. according to the project pages [10, 12]. Proviola is able to generate an animated version of a proof script giving good insight into the proving process, but it does not focus on printing the output on paper. Anther project makes Coq proofs available in the browser (see [6]) which makes the local installation of Coq unnecessary and improves usability as documents can be shared. But it does not seem to be possible to generate a printable version with all necessary information as coq-psv does.

Current state and future work. The tool should be currently seen as proof of concept and supports Coq 8.10. It is implemented in OCaml and can be installed from our Opam repository as described on the coq-psv project page [4]. There are currently some minor issues in the vertical alignment of the used tactic. But even non-trivial proofs can be visualised decently as can be seen in some examples on the project page. It is possible to generate LaTeX and PDF files. Obviously, there are many options for future work. For example, one could hide tactics like clear, rename and move and give them a special treatment instead: for instance, marking cleared hypotheses, annotating identifier changes with the substitution pattern and just change the position of the displayed hypotheses. Then, a support for modules in files should be incorporated. Furthermore, supporting especially HTML output is a main aspect of future work. We aim at integrating such an improved coq-psv into coqdoc. Additionally, coq-psv could be extended to improve the proof scripts themselves. For instance, structure improvements can be suggested. Especially, unused or irrelevant hypotheses might be detected and automatically deleted, if desired. This is not an easy task since it may change the names of the hypotheses appearing later on. Finally, one could give the option to conduct an automatic reordering of the lines in a proof script.

References

[1] David Aspinall. Proof General: A Generic Tool for Proof Development. In Susanne Graf and Michael Schwartzbach, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 38–43. Springer, 2000.

[2] The Coq proof assistant. http://coq.inria.fr. (Last visited 18.06.2020).

[3] The coqdoc wiki page. https://github.com/coq/coq/wiki/Coqdoc. (Last visited 18.06.2020).

[4] Coq Proof Script Visualiser project page. https://www.cs.uni-potsdam.de/coq-psv. (Last visited 18.06.2020).

[5] Maxim Hendriks, Cezary Kaliszyk, Femke Van Raamsdonk, and Freek Wiedijk. Teaching Logic Using a State-of-the-Art Proof Assistant. *Acta Didactica Napocensia*, 3(2):35–48, 2010.

[6] The JSCoq project page. https://jscoq.github.io/. (Last visited 18.06.2020).

[7] Hideyuki Kawabata, Yuta Tanaka, Mai Kimura, and Tetsuo Hironaka. Traf: A Graphical Proof Tree Viewer Cooperating with Coq Through Proof General. In Sukyoung Ryu, editor, *Programming Languages and Systems*, pages 157–165, Cham, 2018. Springer International Publishing.

[8] Tomer Libal, Martin Riener, and Mikheil Rukhaia. Advanced Proof Viewing in ProofTool. In Christoph Benzmüller and Bruno Woltzenlogel Paleo, editors, *Proceedings Eleventh Workshop on User Interfaces for Theorem Provers (UITP 2014)*, volume 167 of *EPTCS*, pages 35–47, 2014.

[9] Jorge Pais and Álvaro Tasistro. Proof Assistant Based on Didactic Considerations. *Journal of Universal Computer Science*, 19(11):1570–1596, 2013.

[10] The ProofTree project page. https://askra.de/software/prooftree/. (Last visited 18.06.2020).

[11] Carst Tankink, Herman Geuvers, James McKinna, and Freek Wiedijk. Proviola: A tool for proof re-animation. *CoRR*, abs/1005.2672, 2010.

[12] The Traf project page. https://github.com/hide-kawabata/traf. (Last visited 18.06.2020).