Towards an Incremental Dataset of Proofs

HANNELI C. A. TAVANTE, McGill University

In this work, we describe an approach for a data-centered user study for proof assistant tools, targeting jsCoq. While we were not yet able to obtain an initial dataset from students enrolled in a Programming Languages course for Fall 2021, we report several hypothesis that could be validated upon the analysis of this data. Up to this point, there are no records of user studies involving large amounts of data for none of the existing tools. An analysis centered on user data could improve the overall usability of these interfaces by revealing issues with their design. In the educational field, the investigation could also help lecturers and staff to understand the students' struggles and issues when learning Coq.

Additional Key Words and Phrases: Coq, user-study, data collection, proof assistants

1 INTRODUCTION

Multiple studies proposing user interfaces for proof assistants have been published in the last decades. In each edition of *User Interfaces for Theorem Provers* (UITP), it is possible to find a wide range of experiments for both educational and professional purposes.

For the Coq proof assistant, CoqIDE[1] and ProofGeneral [Aspinall 2000] are among the most widely adopted interfaces. More recent attempts consider options with support for Web interfaces, such as jsCoq [Gallego Arias et al. 2017]. The immediate benefit of online environments like jsCoq is that the user does not need to install any tool in their local machine. Hence, they are able to try out the Coq without any setup overhead.

Several projects listed on jsCoq Github page [2] seem to benefit from its portability. In particular, several educational events (Summer/Winter schools, tutorials, workshops) simply embedded the required code to be used in a particular demonstration on jsCoq. For instance, it becomes clear that as an online environment, jsCoq has a lot of potentials to be adopted as an educational resource [Warren et al. 2014].

There are multiple other tools and plugins for Coq (VSCoq [3] for Visual Studio Code, Coqtail for Vim [4] and even a Jupyter Notebook style interface [5]). Other integrated development environments or extensions were proposed (such as CoqPIE [Roe and Smith 2016] and Company-Coq [Pit-Claudel and Courtieu 2016]), broadening options for the end user.

2 USER STUDY WITH LARGE DATASETS

Up to this point, there is no record of a large data-centered user study for any of the tools previously mentioned. For instance, in prior work, Knobelsdorf et al. [2017] performed surveys with small groups of students to answer the following question: "What kind of problems and issues do students run into when working with Coq, especially usability issues?". Ringer et al. [2020] investigates the development process of users using proof assistants, but it targets experienced users and also reports the results of a small data set. Whereas qualitative data is a perfectly valid approach, it would be interesting to have large datasets to observe how new learners of Coq are performing,

[1]https://coq.inria.fr/refman/practical-tools/coqide.html
[2]https://github.com/jscoq/jscoq
[3]https://github.com/coq-community/vscoq
[4]https://github.com/whonore/Coqtail
[5]https://github.com/EugeneLoy/coq_jupyter

HATRA '21, October 17–22, 2021, Chicago, USA
2021. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

and more importantly, which types of issues they have when using Coq. An analysis based on big data could also reveal crucial insights for experienced users at the same time.

One way to obtain such a data set would be via jsCoq. Contrary to similar attempts for data massive data collection in IDEs (see the project Blackbox [Brown et al. 2018], for reference), jsCoq is an online tool, and an asynchronous integration with node.js server redirecting the browser data to a database (for example, MongoDB [6]) would be enough to generate data entries. Figure 1 shows an overview of the architecture for collecting the data.

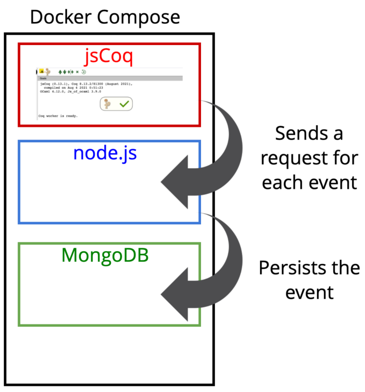

Fig. 1. Architecture of data collection for jsCoq.

A working setup, ready for collecting anonymized data, can be downloaded from https://github.com/galois1/jscoq/tree/data-collection-basic. The main *README.md* file contains a section with build instructions.

At this point, the following events generate new records in the database:

- Starting a new session;
- Any click on the arrows of the main panel (see Figure 2). Each button generates a different label on the persisted record (namely, "Up", "Down", "To Cursor", "Interrupt Worker", "Reset Worker");
- Stopping a session.

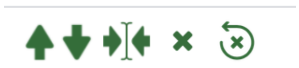

Fig. 2. Buttons in the main control panel that generate a new record in the dataset.

Each record contains a snapshot of the user text (from the scratchpad), a label to indicate which event happened (start or stop of the worker, or an event from the main panel), the output of the Coq worker (for example, the current goals), the raw log message from jsCoq and a timestamp. This setting would enable a full reconstruction of the user timeline and their sequence of steps when using jsCoq. Each record is persisted as a JSON document in MongoDB.

At this time, there is a technical limitation related to the anonymization of the user. We identify each user by recording their browser user agent. A work in progress feature consists of creating

[6] https://www.mongodb.com/

proper session tokens, which would be more reliable to keep track of users. The other limitation related to identification is that we cannot locate a returning user in case they terminate a session. This feature would be achieved upon the implementation of a login mechanism.

The data collection strategy previously described would also enable institutions to deal with their own privacy restrictions, consent forms, ethics approval, and anonymization mechanisms according to their norms, as opposed of relying on a centralized database in an external institution. A similar approach has been described for the LearnOCaml platform ([Canou et al. 2017]) by Ceci et al. [2021].

3 DATA ANALYSIS AND HYPOTHESIS

The previously described setup would enable an incremental dataset of proofs. The persisted timestamps for each record allow a full reconstruction of the user steps when writing a proof in Coq. It is difficult to derive conclusions in the data analysis if the motives for a collected sequence of records are unknown. However, in an educational environment, namely a course or workshop, it is possible to investigate the user interaction, experience and progression when writing their programs. The analysis described in the upcoming subsections assumes a group of users in a beginner, or maximum at intermediate level of familiarity with Coq.

3.1 A distinction between usability and learning curve

It would be interesting to identify what are the actual limitations of the interface versus a possible learning curve from a beginner. To identify sequences of actions that could reveal a usability issue, we suggest a particular type of exercise, where some sort of template with the required proof steps and necessary tactics are described. For example, ensure one activity that is nearly trivial for a user of any level: `(* Exercise: Prove the addition for natural numbers is commutative. You may need to use the following tactics: induction, rewrite, solve *)`. After recording and analyzing the sequence of events upon completion of the required proof, we may determine if the user had issues with the interface given the number of actions they needed to complete the simple assignment. If the event sequence contains multiple clicks without any changes in the structure of the proof, that might indicate a poor user experience with the commands panel or shortcuts.

The same exercise might reveal issues in the learning process. If the sequence of events for completing the previously described problem holds multiple code changes and brings up misleading tactics, then the user might be still adapting to Coq's syntax and to the theoretical foundations of the course.

3.2 Automated guidance for classes

The data analysis can also be a key for offering automated support for learners. For a given exercise, it would be possible to analyze and compare the current user sequence with data from other users who completed the same question in previous editions of the course. After a certain number of unsuccessful attempts, the platform could make suggestions if the user seems to be stuck. This feature would require some degree of real-time data analysis.

A complete dataset from students could also enable answers for the following questions: "What is the average number of steps a user takes to complete this exercise?", "What are the proofs very few students complete?", "What are similar misleading event sequences we obtain for difficult proofs?", "Which tactics seem to be inadvertently used in a given exercise?".

The data analysis of these events would also enable the identification of sequences of tactics that can be simplified. Once redundant steps are detected, the environment could inform the user to refactor their code and obtain a better structure for the final proof. These features would serve as a style checker for the user. Arguably, building those in real-time might be challenging; a large number of records of the same theorem could make it possible.

3.3 Reconstructing a sequence of events to detect errors in the platform

Since we record the full sequence of events to complete a proof, it would be feasible to run all the steps offline and reproduce reported errors. The data collection, in this case, would serve as a reproducibility asset for the development team.

3.4 Support for the machine learning community

Projects such as CoqGym [Yang and Deng 2019][7] rely on datasets of proofs extracted from the Coq Package Index[8]. The datasets generated with the tool described in this work, on the other hand, come from users at different levels of expertise. The outcomes could support the extension of similar existing machine learning projects and could incentivize new ones.

4 DISCUSSION AND FUTURE WORK

This work aims to show the architecture for data collection in jsCoq (still a work in progress during Summer 2021), as well as some possible hypotheses to be tested. In addition, it investigates ways to provide students with a better user experience of Coq tools. The results of the data analysis can guide the path for interface enhancements and possibly lower the entry bar to the topic of proof assistants.

A number of related ideas and questions may deserve a place for discussion:

- Is it possible to derive lessons from other existing environments for theorem provers, such as Lean [9]? The Lean community has a detailed introductory tutorial, which seems efficient for both beginners in the field, and also for people with prior experience in proofs, but new to Lean.
- Would it be valid to try a cross-institution effort in the Coq community to improve these tools? The Blackbox project relied on multiple universities for collecting data and analyzing.
- What other research questions could one ask with such a dataset, besides interface and educational-related questions?
- What about the usability of offline tools? Would it make sense to have a Language Server Protocol (LSP) for Coq? There seems to be an open issue on the official project [10].

Scripts for cleaning up and analyzing the sequences of events will be an essential part of future work. We plan to make these assets publicly available.

Lastly, data collection itself is an interactive process. After a batch of analysis, we may be able to detect other pieces of information to be collected. This work presented a draft of the initial model we aim to test, but the described record structure may change.

REFERENCES

David Aspinall. 2000. Proof General: A Generic Tool for Proof Development. In *Tools and Algorithms for Construction and Analysis of Systems, 6th International Conference, TACAS 2000, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS 2000, Berlin, Germany, March 25 - April 2, 2000, Proceedings (Lecture Notes in Computer Science, Vol. 1785)*, Susanne Graf and Michael I. Schwartzbach (Eds.). Springer, 38–42. https://doi.org/10.1007/3-540-46419-0_3

[7]https://github.com/princeton-vl/CoqGym
[8]https://coq.inria.fr/packages
[9]https://leanprover-community.github.io/index.html
[10]https://github.com/ejgallego/coq-serapi/issues/26

Neil C. C. Brown, Amjad AlTadmri, Sue Sentance, and Michael Kölling. 2018. Blackbox, Five Years On: An Evaluation of a Large-scale Programming Data Collection Project. In *Proceedings of the 2018 ACM Conference on International Computing Education Research, ICER 2018, Espoo, Finland, August 13-15, 2018*, Lauri Malmi, Ari Korhonen, Robert McCartney, and Andrew Petersen (Eds.). ACM, 196–204. https://doi.org/10.1145/3230977.3230991

Benjamin Canou, Roberto Di Cosmo, and Grégoire Henry. 2017. Scaling up functional programming education: under the hood of the OCaml MOOC. *Proc. ACM Program. Lang.* 1, ICFP (2017), 4:1–4:25. https://doi.org/10.1145/3110248

Alana Ceci, Hanneli C. A. Tavante, Brigitte Pientka, and Xujie Si. 2021. Data Collection for the Learn-OCaml Programming Platform: Modelling How Students Develop Typed Functional Programs. In *SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education, Virtual Event, USA, March 13-20, 2021*, Mark Sherriff, Laurence D. Merkle, Pamela A. Cutter, Alvaro E. Monge, and Judithe Sheard (Eds.). ACM, 1341. https://doi.org/10.1145/3408877.3439579

Emilio Jesús Gallego Arias, Benoît Pin, and Pierre Jouvelot. 2017. jsCoq: Towards Hybrid Theorem Proving Interfaces. In *Proceedings of the 12th Workshop on User Interfaces for Theorem Provers, Coimbra, Portugal, 2nd July 2016 (Electronic Proceedings in Theoretical Computer Science, Vol. 239)*, Serge Autexier and Pedro Quaresma (Eds.). Open Publishing Association, 15–27. https://doi.org/10.4204/EPTCS.239.2

Maria Knobelsdorf, Christiane Frede, Sebastian Böhne, and Christoph Kreitz. 2017. Theorem Provers as a Learning Tool in Theory of Computation. In *Proceedings of the 2017 ACM Conference on International Computing Education Research, ICER 2017, Tacoma, WA, USA, August 18-20, 2017*, Josh Tenenberg, Donald Chinn, Judy Sheard, and Lauri Malmi (Eds.). ACM, 83–92. https://doi.org/10.1145/3105726.3106184

Clément Pit-Claudel and Pierre Courtieu. 2016. Company-Coq: Taking Proof General one step closer to a real IDE. In *CoqPL'16: The Second International Workshop on Coq for PL.* https://doi.org/10.5281/zenodo.44331

Talia Ringer, Alex Sanchez-Stern, Dan Grossman, and Sorin Lerner. 2020. REPLica: REPL instrumentation for Coq analysis. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, LA, USA, January 20-21, 2020*, Jasmin Blanchette and Catalin Hritcu (Eds.). ACM, 99–113. https://doi.org/10.1145/3372885.3373823

Kenneth Roe and Scott F. Smith. 2016. CoqPIE: An IDE Aimed at Improving Proof Development Productivity - (Rough Diamond). In *Interactive Theorem Proving - 7th International Conference, ITP 2016, Nancy, France, August 22-25, 2016, Proceedings (Lecture Notes in Computer Science, Vol. 9807)*, Jasmin Christian Blanchette and Stephan Merz (Eds.). Springer, 491–499. https://doi.org/10.1007/978-3-319-43144-4_32

Joe Warren, Scott Rixner, John Greiner, and Stephen Wong. 2014. Facilitating human interaction in an online programming course. In *The 45th ACM Technical Symposium on Computer Science Education, SIGCSE 2014, Atlanta, GA, USA, March 5-8, 2014*, J. D. Dougherty, Kris Nagel, Adrienne Decker, and Kurt Eiselt (Eds.). ACM, 665–670. https://doi.org/10.1145/2538862.2538893

Kaiyu Yang and Jia Deng. 2019. Learning to Prove Theorems via Interacting with Proof Assistants. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 6984–6994. http://proceedings.mlr.press/v97/yang19a.html