ITP Automation in Practice: A User Study on Tactician

Lasse Blaauwbroek[1]*and Herman Geuvers[2]

[1] Czech Institute for Informatics, Robotics and Cybernetics, Czech Republic
Radboud University Nijmegen, the Netherlands
lasse@blaauwbroek.eu
[2] Radboud University Nijmegen, the Netherlands
Technical University Eidhoven, the Netherlands
herman@cs.ru.nl

We present the initial results of a study on the real-world usage patterns of the Tactician automation plugin for Coq. Telemetry was collected from students of two courses on type theory and proof assistants (specifically Coq).

Many new automation techniques for interactive theorem proving through machine learning and other methods have been presented in recent times. These techniques are usually experimentally evaluated by attempting to automatically re-prove existing mathematical developments. This is an important metric to judge automation by. However, due to the interactive nature of proof assistants, this does not tell the whole story. In practice, automation tools are used in a feedback loop, where users continually refine lemmas, theorems and their proofs, until they "go through". We wish to know in what way different types of automation tools can assist in this. Existing work on this is hard to come by, and often consists of (valuable) user testimonies, such as for Sledgehammer [3]. The only study with hard data we are aware of is REPLica [4], which aims to study the general usage patterns of Coq users. Because many members of our community are the creators, users and teachers of their tools, we have an opportunity to obtain more direct and scientific evidence by collecting telemetry from real-life usage situations.

Here, we present initial work in this direction, where we instrument the Tactician [2, 1] plugin for Coq while used in a pedagogical setting. Tactician is a machine learning plugin that adds two new functions to Coq: The `Suggest` command provides a list of tactics to the users that might advance the current proof, and the `search` tactic attempts to autonomously finish the current proof. We test Tactician over the run of two introductory Master level courses on type theory and proof assistants (specifically Coq). Since these courses serve as a first introduction to proof assistants, our study mainly concerns the ability of Tactician to aid in a pedagogical setting. During the courses, we provided the students with a distribution of Coq with a special version of Tactician installed that collects data about their usage of the two commands described above and automatically transmits it to a remove server. All data was collected anonymously and it was made clear to students that participation was entirely voluntary and would not have any bearing on their course grade. In addition to automated data collection, we also asked students to complete a short survey about their experience at the end of the course. To introduce students to the automation, an simple tutorial Coq file was used that showcased the new commands. In addition, some attention was given on the subject during lectures.

Each time a student executed either the `Suggest` command or `search` tactic, data was collected, including the date and time, an anonymous user identifier, the operating system, the size

*This work was supported by the European Regional Development Fund under the project AI&Reasoning (reg. no. CZ.02.1.01/0.0/0.0/15_003/0000466)

of Tactician's learned model and the proof state the command was executed on. For `Suggest` we additionally recorded the suggestions presented to the users. For `search` we recorded the amount of time it was run by the users, whether or not it was successful, the amount of inference steps performed, a witness for the found proof, and the trace through the search space of that proof.

Results Due to space limits we will only present some initial results and conclusions here. For both courses together, a total of 74 students were enrolled. The final exam was attended by 60 students, and 56 students turned in a project (in one course students were allowed to do their projects in pairs, which 14 pairs did). Telemetry was received at least once from 29 students. Most of these students were part of the second course. During the first course there were some hiccups with the software and the way it was introduced to students that caused adoption to be low.

After filtering out testing telemetry, telemetry generated by teachers, and telemetry originating from the tutorial file, we obtained data from 1605 unique usages of the commands. Of these, 1100 originated from `Suggest` and 505 from `search`. Of the searches, 225 (48.9%) were solved, while 225 (44.6%) were aborted. In the remaining 33 cases, the entire search space was exhausted. Adoption of the automation by students was highly irregular. Below, we show usage counts from individual students from high to low.

total:	404	222	156	134	125	110	73	56	56	43	26	26	18	17	16	16	...
`Suggest:`	403	222	134	56	37	35	34	19	16	16	15	14	13	13	11	10	...
`search:`	121	95	88	39	37	37	20	16	15	9	8	7	4	2	1	1	...

The `Suggest` command saw quite some usage, at least by some students. Our survey indicates that it is especially useful in the beginning, when students need to be reminded of the available tactics often. It is easier to run this command than to look tactics up in Coq's manual. However, for educational purposes, this comes with a caveat: To increase students understanding of proof assistants, it is desirable for them to understand the most basic building blocks (tactics) of a proof (`intro`, `apply`, `rewrite`,..). However, suggestions will often come in the form of more high-level tactics that are not suitable for pedagogical reasons. For this reason, the decision was made to introduce `Suggest` only after this initial learning phase was complete.

Adoption of the `search` command was much lower, which is somewhat surprising given that it solved the students goal almost half of the time. We think that two factors may have contributed to this: (1) Due to the Covid pandemic, all teaching and tutoring happened remotely which resulted in less contact between students and teachers and therefore fewer opportunities to expose students to the automation. (2) Our survey indicates that several students (understandably) mistook the `search` tactic for Coq's built-in `Search` command (which helps users find lemmas). This once again gives credence to the famous quote *"there are only two hard things in Computer Science: cache invalidation and naming things."* As a result of this new insight we are in the market for a new name for `search`.

The median amount of time it took for `search` solve a proof was 0.03 seconds with a maximum of 115 seconds. In case of failure, students waited to abort `search` for a median of 14 seconds and a maximum of 940 seconds. This indicates that the time automation has to do its work is limited, but should be enough to pick most of the low-hanging fruit. The number of tactics executed in the current lemma before `search` was executed was a median of 2 for in case `search` was successful and 3 when aborted. One might expect to see a longer trace in case of success (because arguably the remaining proof would be easier), but we are unable to observe this here.

References

[1] Lasse Blaauwbroek, Josef Urban, and Herman Geuvers. Tactic learning and proving for the coq proof assistant. In Elvira Albert and Laura Kovács, editors, *LPAR 2020: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Alicante, Spain, May 22-27, 2020*, volume 73 of *EPiC Series in Computing*, pages 138–150. EasyChair, 2020.

[2] Lasse Blaauwbroek, Josef Urban, and Herman Geuvers. The tactician - A seamless, interactive tactic learner and prover for coq. In Christoph Benzmüller and Bruce R. Miller, editors, *Intelligent Computer Mathematics - 13th International Conference, CICM 2020, Bertinoro, Italy, July 26-31, 2020, Proceedings*, volume 12236 of *Lecture Notes in Computer Science*, pages 271–277. Springer, 2020.

[3] Lawrence C. Paulson. Three years of experience with sledgehammer, a practical link between automatic and interactive theorem provers. In Renate A. Schmidt, Stephan Schulz, and Boris Konev, editors, *Proceedings of the 2nd Workshop on Practical Aspects of Automated Reasoning, PAAR-2010, Edinburgh, Scotland, UK, July 14, 2010*, volume 9 of *EPiC Series in Computing*, pages 1–10. EasyChair, 2010.

[4] Talia Ringer, Alex Sanchez-Stern, Dan Grossman, and Sorin Lerner. Replica: REPL instrumentation for coq analysis. In Jasmin Blanchette and Catalin Hritcu, editors, *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, LA, USA, January 20-21, 2020*, pages 99–113. ACM, 2020.