A Development Process for Coq Projects Permitting Invalid Proofs

Hendrik Tews, Kernkonzept GmbH, Germany

July 31, 2024

This note describes a development process for Coq projects that permits invalid proofs in the main development line while still keeping control over the project. The note reviews the existing tool support and presents new Proof-General features that enable such a process. The described process has recently been introduced at Kernkonzept.

1 Advantages and Challenges when for permitting invalid proofs

For good reasons, IDEs for Coq and the Coq tool chain itself focus on correctness. During development in a Coq project, one is therefore forced to work on the first invalid proof or definition. However, abandoning the correctness requirement and permitting incomplete or invalid proofs in the main development line for certain periods in the life cycle of a project has the following advantages.

- The developer can focus on the most interesting or challenging proof.
- A team can work in parallel on different proofs.
- Pull requests can be merged into the main line before all proofs have been fixed, making collaboration and review much easier.
- Depending on deadlines and available resources, one can focus on features, neglecting proofs until there is time again to update all proofs. This is especially important in a commercial context, where certain customers may only require features, unfortunately without caring about proofs.

Invalid proofs in the main development line are a kind of technical dept. To control this dept and to keep the project alive the following requirements seem sensible.

1. Continuous integration should ensure that the main development line is always in a state where new features can be added to the project and were one can work on any of the invalid proofs.
2. The proofs that become invalid in a pull request should be highlighted, such that the reviewer can judge the impact of the changes.
3. There should be a summary about the proof state of the project, generated by continuous integration, that contains the list of currently invalid proofs.

2 Available and needed tool support

Although not developed for this purpose, there seems to be enough tool support for requirement 1. Vos compilation (compiled interfaces), asynchronous processing in

CoqIDE and VsCoq, and the omit proofs feature of Proof General can be used to work on a project that contains invalid proofs. This seems sufficient for requirement 1, although the described tool support is limited to invalid opaque proofs only.

There is currently no support in the Coq command-line tools for requirement 3, because the compile check using `-vok` exits on the first invalid proof, see also feature wish #11479. To cope with requirement 3, a new feature for generating proof-status statistics (command `proof-check-report`) was recently implemented in Proof General. While this works, it is far from optimal because one needs to start Proof General in batch mode, which then processes the file in its usual command loop with the known efficiency restrictions.

Requirement 2 seems essential to avoid the situation that a lot of proofs become invalid without anybody taking notice. However, it is not clear how to realize this requirement when using a version-control system such as Git, where review typically focuses on source code only. One possible solution is to build lists of newly failing and newly passing proofs in continuous integration, based on Proof General's feature for proof-status statistics presented above. However, at Kernkonzept we were interested in a process where failing proofs are annotated with `FAIL` comments such that they stand out during code review and where continuous integration can check the correctness of these annotations. We therefore recently implemented the command `proof-check-annotate` in Proof General, which inserts `FAIL` comments at failing proofs and deletes such comments on passing proofs. This annotation feature suffers from the same problems as the proof-status statistics feature.

3 Current realization at Kernkonzept

At Kernkonzept we permit invalid proofs in the main development line. Before submitting a pull request, developers must annotate each failing proof with a `FAIL` comment and delete such comments on successful proofs, using a `make` target on the basis of Proof General's proof-status annotation feature. This way new failing proofs stand out during code review. We implemented a mandatory test for each pull request that checks that vos-compilation is successful and that updating all `FAIL` comments does not change anything (in the sense of `git diff`). Pull requests for which standard batch compilation (producing `vo` files) is successful receive a bonus point. There is a make target producing a proof-state summary for the project. A corresponding dashboard in our continuous integration environment is planned.

4 Conclusion

A development process that permits invalid proofs in the main development line in a Coq project simplifies team collaboration and development. Vos compilation and the possibility to ignore failing opaque proofs in current Coq IDEs are essential for such a process. Until recently, appropriate tool support to control the technical depth accumulated by failing proofs was missing. This note introduces recent Proof-General extensions that close this gap. A `coqc` feature that produces a pass/fail list for all opaque proofs more efficiently than Proof General would complete the tool support for permitting invalid proofs and could replace Proof General batch-mode runs in the future.

Acknowledgments The work presented in this note was partially supported by the German government through the Versecloud research project.