Coqoon/PIDE : Asynchronous Coq proof development in Eclipse

Alexander Faithfull and Jesper Bengtson

IT University of Copenhagen

At the 6th Coq Workshop in Vienna, we presented Coqoon[1], an IDE for Coq with sophisticated features like syntax highlighting, context-aware search functionality driven by a rich model of Coq code, and an integrated build system with automatic dependency resolution. Coqoon is one part of a larger effort to build a platform for using Coq to reason about Java programs, based on Mehnert's work on Kopitiam [2].

Up until recently, Coqoon behaved similarly to Proof General or CoqIDE with `coqtop` keeping track of the current proof state and Coqoon highlighting the script that had been parsed to reach that state. Any changes in the marked region would result in `coqtop` backtracking to the point that the change was made and the code between these points would have to be replayed before the developer could continue working from the original point.

The Isabelle proof assistant started moving away from this way of developing proofs around seven years ago when Wenzel introduced the PIDE-protocol [4]. This protocol provides an API that lets development environments communicate asynchronously with the proof assistant which makes them work more like standard software IDEs – changes to the code can be made anywhere, only code that is dependent on those changes must be recompiled, and all errors (regardless of their location) are reported back to the user. This type of functionality has up until now only been enjoyed by Isabelle users, but recent advances has made it possible to make it available in Coq IDEs as well.

PIDE for Coq 2014 saw two major advances in the Coq backend to facilitate asynchronous proof development in Coq.

The first was a state transaction machine (STM) by Barras and Tassi, added in Coq 8.5, which provides the necessary infrastructure required to communicate assynchronously with `coqtop`. The second was by Tankink who used the STM to port the PIDE-protocol to Coq. As a proof of concept Tankink also created a prototype Coq editor [3] using PIDE.

Working with Tankink, we have added support for the PIDE protocol to Coqoon. The PIDE editor automatically re-evaluates commands and their dependencies whenever they change.

Support for non-Coqoon users To make it possible for non-Coqoon users to work with Coqoon projects, the Coqoon build system used to finish building a project by writing out a snapshot of its dependency tree as a Makefile with rules that simulated Coqoon's own behaviour. However, these Makefiles only worked reliably for simple projects—when projects depended on other projects, or on external developments, the resulting Makefile was totally unportable. Worse yet, the build system's constant attempts to regenerate this file often led to version control merge conflicts.

Recent versions of Coqoon adopt a different approach. The builder, instead of producing a project-specific Makefile, now adds a *generic* configure script to every project; when run, this script parses the Coqoon project configuration and produces a machine-specific Makefile from it, prompting the user if they need to provide a path for a dependency that cannot be resolved automatically.

Embedding Coq proofs Since Coqoon is an Eclipse plugin it is possible to create hooks from existing Eclipse projects into Coq projects. We have started working on one such approach where we combine the standard Java development environment of Eclipse with Coqoon. This allows us to provide a new type of editor, aware of the interleavings in the documents that it displays. This awareness allows the editor to provide both Coq- and Java-specific services, like syntax highlighting or code completion for both Java code and embedded Coq proofs. Ultimately, this will make it possible to write Java programs and proofs of their correctness in the same IDE – and, by taking advantage of the asynchrony of PIDE, with very similar workflows.

Conclusion Coqoon continues to develop, and the new support for asynchronous proof editing makes it behave more like an IDE than ever before.

More information about Coqoon, including installation instructions and links to both precompiled Eclipse plugins and the source code, is available from `https://coqoon.github.io/`.

References

1. Alexander John Faithfull and Jesper Bengtson. Coqoon: towards a modern IDE for Coq. In *6th Coq Workshop*, Vienna, Austria, July 2014. URL: `https://coqoon.github.io/docs/coq6.pdf`.
2. Hannes Mehnert. Kopitiam: Modular incremental interactive full functional static verification of java code. In *NASA Formal Methods*, pages 518–524, 2011. URL: `http://dx.doi.org/10.1007/978-3-642-20398-5_42`.
3. Carst Tankink. Asynchronous interaction for Coq. In *6th Coq Workshop*, Vienna, Austria, July 2014.
4. Makarius Wenzel. Isabelle/jedit — a prover IDE within the PIDE framework. *CoRR*, abs/1207.3441, 2012. URL: `http://arxiv.org/abs/1207.3441`.