Company-Coq: Taking Proof General one step closer to a real IDE

A tutorial on using Proof General and its new extension to write proofs more efficiently

Clément Pit-Claudel
MIT CSAIL
cpitcla@mit.edu

Pierre Courtieu
CNAM, Lab. Cédric
pierre.courtieu@cnam.fr

Abstract

`Company-Coq` is a new Emacs package that extends Proof General with a contextual auto-completion engine for Coq proofs and many additional facilities to make writing proofs easier and more efficient. Beyond fuzzy auto-completion of tactics, options, module names, and local definitions, `company-coq` offers offline in-editor documentation, convenient snippets, and multiple other Coq-specific IDE features. The system will be presented at CoqPL 2016, focusing on a live demo with an emphasis on writing proofs in Emacs more efficiently, and a discussion of desirable features of proof-oriented development environments.

Categories and Subject Descriptors D.2.6 [*Software Engineering*]: Programming Environments—Integrated environments

Keywords IDE, documentation, proof engineering, user experience

Introduction

Users of the Coq Proof Assistant [3] are roughly divided between two interactive development environments[1]: *Proof General*, an extension of Emacs written by David Aspinall [1], and *CoqIDE*, a Coq-specific development environment written from scratch by members of the Coq team and generally touted as more beginner-friendly (mostly due to Proof General's dependence on Emacs). Both are powerful tools for writing proofs, and significantly improve the experience of proof authors when compared to Coq's simple read-eval-print loop. Yet these tools do not offer advanced features typically found in IDEs, such as in-editor documentation or context-sensitive completion. In addition, when advanced features are in fact available (Proof General, for example, does supports snippets and improved display of mathematics), they tend to lack discoverability: users do not explore the menus and miss convenient features that would make them more efficient.

[1] There also exist Coq interfaces for vim and Eclipse, though their use does not seem very widespread.

CoqPL 2016, January 23, 2016, St. Petersburg, Florida, United States.
http://dx.doi.org/10.5281/zenodo.44331

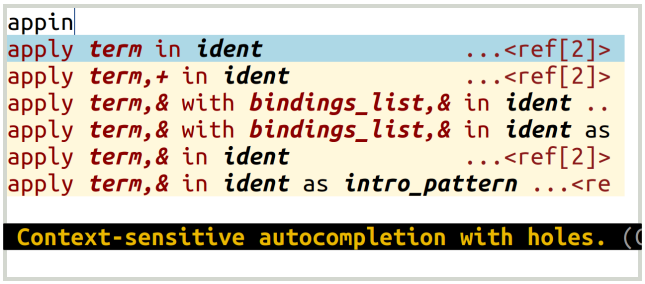
`Company-Coq` is a new Emacs package that attempts to fix some of these limitations: we extend Proof General with many advanced IDE features (fuzzy completion, various Coq-specific snippets, and in-editor documentation for most of Coq's 2000-odd tactics, options, and errors), and solve the discoverability issue by taking an "all-on" approach: the default distribution has most features automatically enabled[2]. In addition, `company-coq` comes with a comprehensive tutorial that showcases most of its features. Since it is not part of the core Proof General, `company-coq` can serve as a convenient experimentation area for new features and development directions, before they are merged into other IDEs.

Workshop description

The presentation will consist of a quick run through `company-coq`'s features, and a tutorial on using these and other Proof General features to write proofs more efficiently with emphasis on the lemma extraction feature. The workshop will also be a good occasion to showcase some features that Proof General inherits from Emacs, to discuss how much of this work could be used to enhance other Coq interfaces, and to ponder about desirable features for proof-oriented development environments.

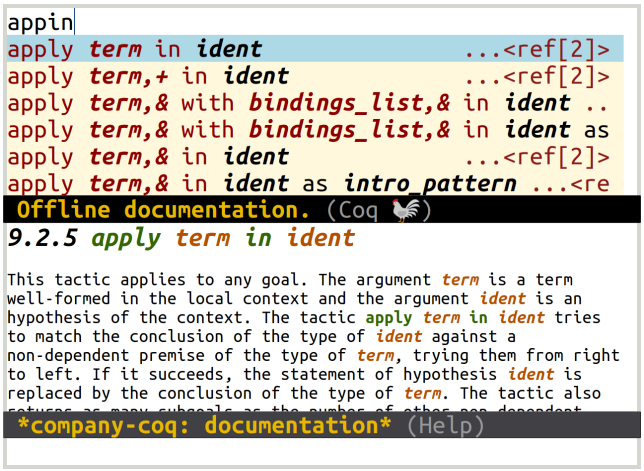
Overview of some features of `company-coq`

Context-sensitive autocompletion with holes `Company-Coq` implements a number of backends for the *CompleteAnything* Emacs package (`company`). Typing `appin` therefore suggests variants of the apply tactic (bold text indicates holes):



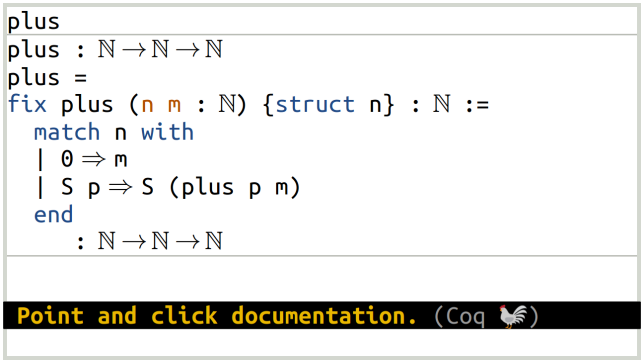
[2] For example, although Proof General does support enhanced displaying of mathematics (non-destructively displaying `fun (n m: nat) => forall p, p <> n -> p >= m` as λ `(n m:` $\mathbb{N}$`)` $\Rightarrow \forall$ `p, p` $\neq$ `n` $\rightarrow$ `p` $\geq$ `m`), few users seem to know about this feature. `Company-Coq`, on the other hand, enables a similar feature by default, and most users seem pleased with it.

Offline documentation Part of the development of `company-coq` involved cross-referencing tactics and options from the user manual; this allows `company-coq` to display documentation for most completion entries, without querying INRIA's website[3]:



Lemma extraction At any point in a proof, users may choose to extract the current goal, including some hypotheses, to a separate lemma. Instead of painstakingly copy-pasting bits of the proof context, `company-coq` offers a convenient interface to pick hypotheses and generate the statement of the extracted lemma[4].

Point and click documentation Clicking on an identifier while pressing the control key opens an inline documentation window (which disappears when the mouse button is released):



Snippets `Company-Coq` connects with `YASnippet` to make certain common Coq patterns quicker to write. For example, to write

```
match goal with
| [ H: ?a /\ ?b |- ?a ] => destruct H; assumption
end
```

the user only needs to type the following commands:

`m g w` ↵

Alt+⇧+↵ ⇥ `?a /\ ?b` ⇥ `?a` ⇥

`destr` ↵ `H; ass` ↵

[3] The list of tactic templates, and the associated documentation, is extracted from the annotated manual into an index and a set of data files that could be useful to other editors.

[4] The statement is produced by generalizing the hypotheses that the goal mentions and the ones that the user selected, using a small Ltac script.

The key here is the Alt+⇧+↵ shortcut, which introduces the

```
| [ H: _ |- _ ] => _
```

pattern, leaving holes that the user can navigate between with `TAB` in place of each `_`.

Automatic named introduction Scripts that depend on the names of hypotheses can often be made more robust by choosing names explicitly: `company-coq` leverages an existing feature of Proof General to let the user type

`intros!` ↵

to create an invocation of the `intros` tactic that explicitly mentions all introduced variables.

New features of Proof General

A number of new Proof General features are also useful for proof development:

Automatic indentation of bulleted proofs Coq proofs can be structured with *bullets* and curly brackets, making proof structure more readily apparent, and helping with maintenance. Thanks to `SMIE` [2], Proof General implements an indentation routine based on structuring commands which makes proofs easier to format.

Automatic recompilation at `Require` When developing a proof, one has to deal with several inter-dependent files. Proof General, thanks to a contribution of Hendrik Tews, can transparently and recursively recompile dependencies as it reaches `Require` commands, launching the required compilation jobs in the background.

Conclusion

`Company-Coq` has been well received by the community[5], and we expect many of its features to find their way into other Coq development environments[6]; we hope that the CoqPL workshop will be a good venue to discuss it, and more generally to discuss the development of new editor features enhancing the experience of authors of Coq programs and proofs.

References

[1] D. Aspinall. Proof General: A Generic Tool for Proof Development. In S. Graf and M. Schwartzbach, editors, *Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2000*, volume 1785 of *Lecture Notes in Computer Science*, pages 38–43. Springer Berlin Heidelberg, 2000. ISBN 978-3-540-67282-1. . URL http://dx.doi.org/10.1007/3-540-46419-0_3.

[2] S. Monnier. SMIE: Simple Minded Indentation Engine, 2010. URL https://www.gnu.org/software/emacs/manual/html_node/elisp/SMIE.html.

[3] The Coq Development Team. *The Coq Proof Assistant Reference Manual*. LogiCal Project, 2004. URL http://coq.inria.fr. Version 8.0.

[5] At this time, `company-coq` has accumulated 800 downloads.

[6] Indeed, the architecture of company-coq should be amenable to such cross-pollination: none of its features are Emacs-specific, and the collection of pre-processing scripts that it relies on to offer documentation, completion and snippets is freely available (along with their output).