Proof-Pattern Search in Coq/SSReflect*

Jónathan Heras and Ekaterina Komendantskaya

School of Computing, University of Dundee, UK
{jonathanheras,katya}@computing.dundee.ac.uk

Abstract. ML4PG is an extension of the Proof General interface of Coq, allowing the user to invoke machine-learning algorithms and find proof similarities in Coq/SSReflect libraries. In this talk, we will show the recent ML4PG features in action, using examples from the standard SSReflect library and HoTT library. We will compare ML4PG with traditional Coq searching tools and dependency graphs.
Keywords: Coq/SSReflect, Proof-Patterns, Recurrent Clustering.

Development of Coq has led to the creation of big libraries and varied infrastructures for formal mathematical proofs. For instance, there are approximately 4200 definitions and 15000 theorems in the formalisation of the Feit-Thompson theorem. The growing size and sophistication of libraries make maintenance and re-use of the methods contained in them harder, and require new tools for library analysis and search.

For analysing existing Coq libraries, *Dependency Graphs* [7] are a useful tool. There are actually two types of dependency graphs: (*i*) graphs showing dependency of a Coq theorem to all the auxiliary results that were used to prove it; and (*ii*) graphs showing the relations between libraries.

For searching, Coq/SSReflect provides comprehensive search mechanisms: `Search` , `SearchAbout`, `SearchPattern` and `SearchRewrite`. In addition, SSReflect implements its own version of the `Search` command [2] – SSReflect's `Search` gathers the functionality of the 4 Coq's search commands. The Whelp platform [1] is a web search engine for mathematical knowledge formalised in Coq, which features 3 functionalities: `Match` (similar to Coq's `Search` command), `Hint` (that finds all the theorems which can be applied to derive the current goal) and `Elim` (that retrieves all the eliminators of a given type).

The ML4PG ("Machine-Learning for Proof General") tool [5,6] was created to complement the functionalities of the above two groups of methods with statistical machine-learning techniques. In this talk, we will use several SSReflect libraries [2], as well as the HoTT library [8], to illustrate the use of the new version of ML4PG compared to the above-mentioned tools.

For standard searching, the user provides a search pattern e.g. using commands of the form "`Search "distr" in bigop`" or "`Search _ (_ * (\big[_/_]_(_ <- _| _)_))`", where `bigop` is a library, `"distr"` is a pattern in a lemma name, and `_ (_ * (\big[_/_]_(_ <- _| _)_))` is a pattern for search.

* The work was supported by EPSRC grants EP/J014222/1 and EP/K031864/1.

The situation is more complicated if the user does not know the right pattern to search for, and just wishes to know whether there exist proofs or definitions that are similar to his current development, in order to extrapolate them.

ML4PG uses *recurrent clustering* [4] to detect proof patterns in Coq/SSReflect libraries. A proof pattern is now understood differently – as a correlation of several proof features. Proof features that ML4PG collects include term-tree structures, types, tactics, (types and shapes of) tactic arguments, and the number of generated subgoals, to name a few. Overall, 300 features are analysed for every term (definition, type declaration, lemma statement, and so on); and 85 features are analysed for every 5 proof steps when proofs are data-mined. Moreover, very much like in dependency graphs, the gathered feature statistic takes into consideration mutual dependencies of various lemmas, types and definitions; by using previous results of clustering auxiliary terms, types and proofs when extracting features for new terms, types and proofs.

The resulting machine-learning tool works differently compared to traditional searching or dependency graphs. It is no longer deterministic, due to its statistical character; it finds proof *similarities*, not dependencies or exact matching of symbols; and it finds patterns beyond concrete notation. For example, if someone re-defines the same or a similar notion using different notation or literally different but structurally similar types, ML4PG can easily group such definitions together, as many structural features will correlate. Comparing to dependency graphs that show all the existing dependencies of a (lemma) statement, ML4PG helps to "post-process" such dependencies and discriminate between "important" and "unimportant" dependencies, relative to whether they play a role in forming a proof pattern as compared to other Coq objects in the given libraries.

Finally, the graphical output of ML4PG is *similarity graphs*, showing either a correlation of proof features (for a single proof cluster); or groups of similar definitions or lemma statements (for a whole library); see [3].

References

1. A. Asperti et al. A Content Based Mathematical Search Engine: Whelp. In *TYPES'04*, volume 3839 of *LNCS*, pages 17–32, 2006.
2. G. Gonthier and A. Mahboubi. An Introduction to Small Scale Reflection. *Journal of Formalized Reasoning*, 3(2):95–152, 2010.
3. J. Heras and E. Komendantskaya. HoTT Formalisation in Coq: Dependency Graphs and ML4PG. A technical note, 2014. `http://arxiv.org/abs/1403.2531`.
4. J. Heras and E. Komendantskaya. Proof Pattern Search in Coq/SSReflect, 2014. `http://arxiv.org/abs/1402.0081`.
5. J. Heras and E. Komendantskaya. Recycling Proof Patterns in Coq: Case Studies. *Journal Mathematics in Computer Science, accepted*, 2014.
6. E. Komendantskaya et al. Machine Learning for Proof General: interfacing interfaces. *Electronic Proceedings in Theoretical Computer Science*, 118:15–41, 2013.
7. A. Pacalet and Y. Bertot. The dpdgraph tool, 2009–2013. `https://anne.pacalet.fr/dev/dpdgraph/`.
8. The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013.