A Decentralized Proof-Term Library for Coq

Michael Nahas*

Coq Workshop 2013 – Proposal

1 Proposal

For Coq Workshop 2013, I propose a discussion on the design of a proof library. By storing proof-terms in a distributed fashion, Coq could provide long-term large-scale storage while also enabling small-scale collaboration. This document lays out a rough rationale and design, with a list of open issues for discussion.

2 Design Alternatives

I want to build a library of every math proof ever written. In my opinion, Coq is currently the best system for writing those proofs.

A library must store Coq's proofs as either proof scripts or proof terms. For large-scale long-term storage, I think proof terms are the obvious choice, since they are simpler and are produced by more stable and better checked code. Also, not using proof scripts frees Coq's developers to experiment with the user interface.

Organization of a library can be done in either a centralized or distributed manner. A centralized approach, such as the one taken by the Mizar Math Library[6], has a gate-keeper who decides what gets into the library. A major benefit of the centralized approach is that it can maintain a unified structure and style to the library. However, this requires a lot of preparation by contributors, verification by the gate-keeper, and delays. This discourages sharing and collaboration. A distributed approach would be preferred.

3 What Might a Design Look Like?

A library must be able to (1) add proofs, (2) find proofs, and (3) assign unique names to proofs. In addition, it must ensure the reliability of its contents and work efficiently.

In a fully decentralized approach, there would be multiple libraries and each would be just a collection of proof terms and inductive types. A user could add a new proof (or type) at any time. A common way to add proofs would be to copy the entire contents of one library into another.

Finding a proof could be done by searching on a term's type or on a human-readable name. A distributed approach to human-readable names is to let each user have their own "view" of the library; the same proof may go by different names to different users.

*View*s map users' human-readable names to the library's unique internal identifiers. To generate unique internal names in a distributed fashion, we suggest borrowing git's[2] technique: using the hash number of the contents. Possibly, the hash number of its type concatenated with the hash number of the proof term. (Many lookups will be by type, rather than proof term.)

A library should not store false proofs. Every proof term submitted by a user will have to be type checked. However, when a library adds all the proofs of another library, there may not be enough CPU to recheck every proof term. The solution to this problem requires trust, which means using cryptography.

*michael@nahas.com, affiliated with Radboud University Nijmegen, NL

Whenever a proof term is type checked, we must sign a certificate for that proof term with a private key. When copying proof terms from another library, we must also copy the certificates and validate them with a public key. The management of public and private keys will be an annoying, but necessary, task.

The certificate can also serve another purpose. When creating it, we can include the Coq version number, OS version, hardware identifier, etc.. If any of these dependencies are suspected of subverting the type checking, the certificate can be ignored and the proof term can be either removed from the library or rechecked locally. Random rechecking of proof terms can be used to improve reliability and detect fraud.

An efficient library does not store duplicates. The distributed approach to this is to convert each object to a canonical form. This would require ordering non-dependent arguments, choosing "a=b" over "b=a", etc..

For efficient merging of libraries, proof terms should be independent of each other. This can be achieved by changing every theorem into a tautology, so that dependencies become arguments. This approach comes with a cost: turning a tautology back into the theorem requires recursively descending through proof terms to determine if any fully connected subset rests on the axioms. It also comes with an unexpected benefit: CPU-intensive proofs can be partitioned into many tautologies, type checked using multiple machines, and then "stitched back together" at the end.

An important part of this design are the views that keep track of the human-readable names. They are mixture of namespaces from programming and branches from source control. The library will require tools for managing views. Collaborators will want to synchronize views, so that they use the same names for the same types and theorems. Users many want different views for different projects. Views will also be used to connect proofs to external systems, such as wikis, which might be used for documentation.

4 Discussion Topics

This proposal is for a discussion. The rough design has interesting discussion topics of:

- how users collaborate
- how to search a collection of proof terms
- how to manage trust
- tracking systematic errors
- canonical forms for types and terms
- tautologies for proof-relevant theorems

References

[1] A. Asperti, et al., *Towards a Library of Formal Mathematics*, Panel session of the 13th International Conference on Theorem Proving in Higher Order Logics, 2000.

[2] S. Chacon, *Pro Git*, `http://git-scm.com/book`, 2009.

[3] The Coq Development Team, *Reference Manual: Version 8.4*, `http://coq.inria.fr/refman/`, 2012.

[4] G. Klein, T. Nipkow, and L. Paulson, Eds., *Archive of Formal Proof*, `http://afp.sourceforge.net/`, 2004.

[5] M. Nahas, et al., *Parity Volume Set Specification 2.0*, `http://parchive.sourceforge.net/docs/specifications/parity-volume-spec/article-spec.html`, 2003.

[6] Mizar Library Committee, Eds., *Mizar Mathematics Library*, `http://www.mizar.org/library/`, 1997.

[7] J. Urban, *Content-based encoding of mathematical and code libraries*, Proceedings of the ITP 2011 Workshop on Mathematical Wikis, CEUR Workshop Proceedings Vol. 767, 2011, pages 49-53.