Proof Reconstruction: Translating Proofs

Alejandro Gómez Londoño

Mathematical Engineering
Research Practise 2
Research Proposal

Supervisor: Andrés Sicard-Ramírez

EAFIT University

Medellín

August 18, 2015

Proof Reconstruction

1 Statement of the Problem

Automated theorem provers (ATP) and proof assistants has been around for decades (Davis 2001; Geuvers 2009). Despite the obvious differences between the two, both approaches share a fundamental goal which is to aid humans with complex proofs in an automatic or interactive manner, is maybe due to this relationship that in recent years various tools have been developed using a mixture of both systems. Such tools, some times referred to as *hammers* (Blanchette et al. 2014a) allow the users to write proofs in an interactive manner from within a proof assistant, but with the option of sending sub-proofs to an ATP.

Hammers by themselves aren't ATP nor poof assistants, they act more like a plugin that sits on top of the proof assistant and allow the communication with various ATP for proof automation. Hammer-like tools typically consists on three mayor components (Blanchette et al. 2014a):

1. *Premise selector*: it gathers relevant theorems from the available libraries that can help with the current proof.
2. *Translation module*: takes the premises and the goal and translates them into the ATP input syntax (a common syntax to represent first order ATP problems is TPTP (Sutcliffe 2009)), this translation in most cases involves mapping a subset of the proof assistant logic into the ATP logic.
3. *Proof reconstruction module*: processes the proof returned by the ATP reconstructing it in the proof assistants syntax/logic.

As shown in Figure 1, hammers act as a bridge between the interactive and the automated process, constituting a more complete system capable of a more smooth interaction with logical reasoning process.

Agda (Norell 2007; Agda Team 2015) as a proof assistant lacks of a hammer-like tool, but programs like Apia (Sicard-Ramírez 2014; Bove et al. 2012) which allows to prove first-order theorems from within Agda using ATPs are closing

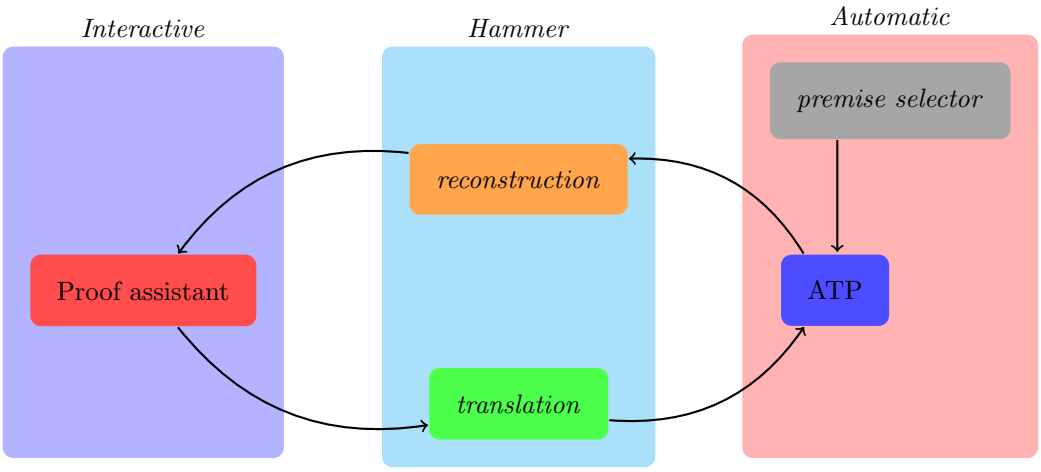

Figure 1: Architecture of the various components involved in a hammer-like tool

this gap. Unfortunately Apia only works as a *translation module* and as a front-end for the ATPs. Some further development has to be done to achieve an Agda-hammer tool and one of the missing pieces in this enterprise is a *proof reconstruction module*, this would allow proofs to be verified from within Agda and jointly with a tool like Apia it could provide a fully functional hammer for Agda. Our long term goal is then to build a *reconstruction module* for Agda in order to fill this gap.

2 Objectives

2.1 General Objective

Translate into idiomatic Agda code the AST[1] resulting from the parsing of an ATP-generated proof.

2.2 Specific Objectives

- Translate to Agda code the Haskell AST data type.
- Build an Agda library that implements the logical kernel of the ATP.

[1] Abstract syntax tree.

- Reconstruct the proof in Agda using the aforementioned library.

3 Literature Review

As stated before, the development of ATPs and proof assistants dates from decades ago, but in comparison, the mixture of this two approaches (as *hammers*) is relatively new, nonetheless some exponents of this trend have been developed in the later years. Perhaps the best precedent in this category is Sledgehammer (Blanchette et al. 2014b), a tool that sits on top of the Isabelle/HOL proof assistant (Nipkow et al. 2002) and allows the translation/reconstruction of proofs to/from multiple ATPs. Another similar example for the Agda proof assistant is a work by Foster and Struth that proposes the integration of Agda with Waldmeister (Foster et al. 2011), a theorem prover for equational logic (Buch et al. 1996). The usefulness and convenience of the hammers can truly improve the way proof assistants work, taking most of the boilerplate and tediousness of proofs out of the way, an thus allowing to get more work done faster.

Currently Agda unlike Isabelle lacks of a true *hammer*, but this is an issue that is being addressed by the aforementioned work by Foster and Struth, and by some developments like the Apia tool, which allows to prove first-order theorems from within Agda translating the formulae to TPTP and then sending it to multiple ATPs.

4 Scope

The scope of this project is to implement a *proof reconstruction module* for the Agda proof assistant, along with documentation, tests and packaging, for a proper distribution and usability, no further work is stated in this project.

5 Justification

This project aims to improve the current state of automatic theorem proving in the Agda proof assistant, to do so we will implement a *proof reconstruction*

module this will allow tools like Apia (and any *translation module* in general) to "close the loop" and act as a true hammer for Agda. This would beg a big step forward for the Agda community and thus significantly relevant.

6 Methodology

The proposed schedule will be followed as much as possible, nevertheless changes or additions may occur down the road. This activities are going to be weekly monitored, guided and complemented by the supervisor.

7 Intellectual Property

The present research is property of Alejandro Gómez-Londoño, and Andrés Sicard-Ramírez as authors.

8 Schedule

Weeks	**Date**	**Activity**
1 - TBA	July 21 - TBA	• AST to DAG[2] translation • Typed-DAG construction • Proof term reconstruction in Agda

References

Agda Team (2015). *The Agda wiki*. URL: `http://wiki.portal.chalmers.se/agda/pmwiki.php`.

Blanchette, Jasmin C., Cezary Kaliszyk, and Lawrence C. Paulson (2014a). "Hammering towards QED". English. In: *Draft version*.

Blanchette, Jasmin Christian and Lawrence C. Paulson (2014b). *Hammering Away. A User's Guide to Sledgehammer for Isabelle/HOL*. Institut für Informatik, Technische Universität München.

[2]Directed acyclic graph.

Bove, Ana, Peter Dybjer, and Andrés Sicard-Ramírez (2012). "Combining Interactive and Automatic Reasoning in First Order Theories of Functional Programs". In: *Foundations of Software Science and Computational Structures*. Springer Berlin Heidelberg, pp. 104–118.

Buch, Arnim, Thomas Hillenbrand, and Roland Fettig (1996). "WALDMEISTER: High Performance Equational Theorem Proving". In: *Proceedings of the International Symposium on Design and Implementation of Symbolic Computation Systems*. DISCO '96. London, UK, UK: Springer-Verlag, pp. 63–64.

Davis, Martin (2001). "Chapter 1 - The Early History of Automated Deduction: Dedicated to the memory of Hao Wang". In: *Handbook of Automated Reasoning*. North-Holland.

Foster, Simon and Georg Struth (2011). "Integrating an Automated Theorem Prover into Agda". English. In: *NASA Formal Methods*. Ed. by Mihaela Bobaru et al. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 116–130.

Geuvers, H (2009). "Proof assistants: History, ideas and future". English. In: *Sadhana* 34.1, pp. 3–25.

Nipkow, Tobias, Markus Wenzel, and Lawrence C Paulson (2002). *Isabelle/HOL: A Proof Assistant for Higher-order Logic*. Berlin, Heidelberg: Springer-Verlag. ISBN: 3-540-43376-7.

Norell, Ulf (2007). "Towards a practical programming language based on dependent type theory". PhD thesis. Chalmers University of Technology and Göteborg University.

Sicard-Ramírez, Andrés (2014). "Reasoning about Functional Programs by Combining Interactive and Automatic Proofs". PhD thesis. Uruguay: University of the Republic.

Sutcliffe, G (2009). "The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0". In: *Journal of Automated Reasoning* 43.4, pp. 337–362.