Proof Reconstruction

Alejandro Gómez Londoño

Research Proposal

Supervisor: Andrés Sicard-Ramírez

EAFIT University

Medellín

Proof Reconstruction

1 Statement of the Problem

Automated theorem provers (ATP) and proof assistants has been around for decades [3, 5]. 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* [1] allow the users to write proofs in a interactive manner from within a proof assistant, but with the option of sending parts of a proof to an ATP.

Hammers by themselves aren't ATPs nor poof assistants, they act more like a plugin that sits on top of the proof assistant and allows the communication with various ATPs for proof automation, hammer-like tools typically consists on three mayor components [1]:

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 ATPs input syntax (usually TPTP [9]) this translation in most cases involves the mapping between the proof assistants logic and the ATPs logic.
3. *Proof reconstruction module*: processes the proof returned by the ATP reconstructing it in the proof assistants syntax/logic.

Agda [7] as a proof assistant lacks of a hammer-like tool, but programs like Apia [8] which allows to prove first-order theorems from within Agda using ATPs are closing 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 tool like Apia it could provide a fully functional hammer for Agda.

2 Objectives

2.1 General Objective

Translate the output from an ATP-generated proof into idiomatic Agda code.

2.2 Specific Objectives

- Select a suitable ATP (which supports TSTP output format) to translate from.
- Parse the output proof of the selected ATP.
- Build an Agda library that implements the logical kernel of the ATP.
- 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* [2] a tool that sits on top of the Isabelle proof assistant [6] and allows the translation/reconstruction of proofs to/from multiple ATPs. Another similar example for the Agda proof assistant is a work by Simon Foster and Georg Struth that proposes the integration of Agda with *Waldmeister* [4] a theorem prover for equational logic. 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 interesting developments like the Apia tool [8], which allows to prove first-order theorems from within Agda translating the formulae to TSTP 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 [8] (and any *translation module* in general) to "close the loop" and act as a true hammer for Agda. This would be a big step forward for the Agda community and thus significantly relevant.

6 Methodology

This project will be developed in parallel between the authors, in order to expand the scope to a multiple ATP and SMT systems. 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 Andrés Sicard-Ramírez.

7 Intellectual Property

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

8 Schedule

Weeks	Date	Activity
1 - 2	January 26 - February 6	Preliminary steps: • Clarify the problem. • Select an ATP. • Determine a development environments.
3 - 6	February 10 - March 6	Parser implementation
7 - TBA	March 9 - TBA	Further development: • AST to DAG translation • Typed-DAG construction • Proof term reconstruction in Agda

References

[1] Jasmin C. Blanchette, Cezary Kaliszyk, and Lawrence C. Paulson. Hammering towards QED. *Draft version*, 2014.

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

[3] Martin Davis. Chapter 1 - The early history of automated deduction: Dedicated to the memory of Hao Wang. In *Handbook of Automated Reasoning*, pages 3 – 15. North-Holland, 2001.

[4] Simon Foster and Georg Struth. Integrating an automated theorem prover into Agda. In Mihaela Bobaru, Klaus Havelund, GerardJ. Holzmann, and Rajeev Joshi, editors, *NASA Formal Methods*, Lecture Notes in Computer Science, pages 116–130. Springer Berlin Heidelberg, 2011.

[5] H Geuvers. Proof assistants: History, ideas and future. *Sadhana*, 34(1):3–25, 2009.

[6] Tobias Nipkow, Markus Wenzel, and Lawrence C Paulson. *Isabelle/HOL: A Proof Assistant for Higher-order Logic*. Springer-Verlag, Berlin, Heidelberg, 2002.

[7] Ulf Norell. *Towards a practical programming language based on dependent type theory*. PhD thesis, Chalmers University of Technology and Göteborg University, September 2007.

[8] Andrés Sicard-Ramírez. *Reasoning about Functional Programs by Combining Interactive and Automatic Proofs*. PhD thesis, University of the Republic, Uruguay, 2014. Unpublished doctoral dissertion.

[9] G Sutcliffe. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0. *Journal of Automated Reasoning*, 43(4):337–362, 2009.