Towards Proof-Reconstruction of Problems in Classical Propositional Logic in Agda

Jonathan Prieto-Cubides and Andrés Sicard-Ramírez

EAFIT University, School of Sciences and Engineering,
Medellín, Colombia
{jprieto9,asr}@eafit.edu.co

Abstract. ...

Keywords: Proof reconstruction, Agda, Automatic theorem prover, Classical Propositional Logic, Metis

1 Introduction

Proof reconstruction is a hard labor, it integrates two complex systems. On one hand, we have the automatic theorem provers (henceforth ATPs) that depending on the specification logic and background theories exist many alternatives. They are SAT solvers (e.g. `zChaff` [21] and `MiniSat` [10]) to prove unsatisfiability of CNF formulas, QBF solvers (e.g. `GhostQ` [18] and `DepQBF` [19]) to prove satisfiability and invalidity of quantified Boolean formulas, SMT solvers (e.g `CVC4` [3], `veriT` [7], `Z3` [8]) to prove unsatisfiability of formulas from first-order logic with theories, and finally, the ATPs to prove validity of formulas from first-order logic with equality (e.g. `E` [28], `Gandalf` [31], `leanCoP` [24], `Metis` [16], `SPASS` [34] and `Vampire` [26]), high-order logic (e.g. `Leo-II`, `Satallax`), intuitonistic logic (e.g. `ileanCoP` [24], `JProver` [27], `Gandalf` [31]), among others.

For proof reconstruction, we are interested in ATP tools that can deliver proof objects in a consistent format to work with, that is, a full script describing step-by-step the table of the derivation to get the actual proof. Unfortunately, just a few of the ATPs generate in their outputs a proof object.

For problems in classical propositional logic (henceforth CPL), from a list of at least forty[1] ATPs, just a few deliver proofs (e.g. `CVC4` [3], `SPASS` [34], `Waldmeister` [14]) and a little bit less proofs in file format like TSTP (e.g. `E` [28], `Metis` [16], `Vampire` [26], `Z3`).

On the other hand, we have the proof checkers, Interactive Theorem Provers or proof assistants (e.g. `Agda` [32], `Coq` [33], `Isabelle` [25], `HOL4` [23]). These programs allow us to define the formal language for the problems like operators, logic variables, axioms, and theorems. They can assist us to check and validate the proof script delivered by the provers. Because the formalism in the source (the proof generated by the ATP) is not necessarily the same formalism in the

[1] ATPs available in the website `SystemOnTPTP` from the TPTP World

target (proof reconstructed in the proof assistant), then the reconstruction of a proof involves a deep understanding around the deduction process done by the ATP, for translating that deduction later into the proof assistant.

Many approaches have been proposed and some tools have been implemented for proof reconstruction in the last decades, we mention some of them in the following.

Related Work.

`Sledgehammer` is a tool for `Isabelle` proof assistant [25] that provides a full integration of automatic theorem provers including ATPs (see, for example, [20], [5] and [12]) and SMT solvers (see, for example, [5], [6] and [12]) with `Isabelle/HOL` [22], the specialization of `Isabelle` for Higher-Order Logic.

`Waldmeister` is an automatic theorem prover for Unit Equational Logic [14]. Foster and Struth [13] integrate `Waldmeister` into `Agda` [32]. This integration requires a proof reconstruction step but authors' approach is restricted to Pure Equational Logic –also called Identity Theory [15]– that is, First-Order Logic with Equality but no other predicate symbols and no functions symbols [1].

An integration between SMT Solvers and `Agda` in [17] use a *oracle and reflection* approach to give certificates for propositional problems.

`SMTCoq` (see [2] and [11] for more details) is a tool for the `Coq` proof assistant [33] which provides a certified checker for proof witness coming from the SMT solver `veriT` [7] and adds a new tactic named verit, that calls `veriT` on any `Coq` goal.

Given a fixed but arbitrary First-Order signature, in [4] transform a proof produced by the first-order automatic theorem prover `Bliksem` [9] in a `Coq` proof term.

We structure the paper as follows. In section 2, we briefly introduce the `Metis` Prover. In section 3, we present our approach to reconstruct proofs deliver by `Metis` in `Agda`. In section 4, we present a complete example of a proof reconstructed with our tool for a CPL problem. In section 4, we discuss some limitations and conclusions, for ending up with the future work.

2 Metis: Language and Proof Terms

`Metis` is an automatic theorem prover for First-order Logic with Equality [16].

2.1 Input and Output Language

The `TPTP` language –which includes the First-Order Form (FOF) and Clause Normal Form (CNF) formats [29] – is de facto input standard language and the `TSTP` language is de facto output standard language [30].

2.2 Proof Terms

`Metis`' proof terms encode natural deduction proofs. Its deduction system uses six simple inference rules and it proves conjectures by refutation.

$$\frac{}{C}\ \text{axiom} \qquad \frac{}{L \vee \neg L}\ \text{assume } L \qquad \frac{}{t = t}\ \text{refl } t \qquad \frac{C}{\sigma\, C}\ \text{subst } \sigma$$

$$\frac{}{\neg(L[p] = t) \vee \neg L \vee L[p \mapsto t]}\ \text{equality } L\ p\ t \qquad \frac{L \vee C \qquad \neg L \vee D}{C \vee D}\ \text{resolve } L$$

Metis proofs are directed acyclic graphs (henceforth DAG), refutations trees. Each node stands for an application of an inference rule and the leaves in the tree represent formulas in the given problem. Each node is labeled with an axiom or an inference rule name (e.g. `resolve`). Each edge links a premise with one conclusion. All proof graphs have in their root the conclusion si nce Metis uses refutation in each deduction step.

2.3 Proof Rules

Using `Metis` to prove CPL problems, we found that their TSTP derivations showed six inference rules, `canonicalize`, `conjunct`, `negate`, `simplify`, `strip` and `resolve`.

The `canonicalize` rule transforms a formula to one of its normal form, CNF, NNF, and DNF. This rule also performs simple simplification in the process, applying for instance some of the following theorems assuming commutative properties.

$$\frac{P \vee \bot}{P} \quad \frac{P \vee \top}{P} \quad \frac{P \vee \neg P}{\top} \quad \frac{P \wedge \bot}{\bot} \quad \frac{P \wedge \top}{P} \quad \frac{P \wedge \neg P}{\bot}$$

The `strip` rule extracts a subgoal from a goal. The splitting process takes a goal and recursively recolects all hypotesis required in order to prove the goal. At the end, the final hypotesis represent the subgoals.

3 Proof Reconstruction in Agda

As a proof checker, we choose the proof-assistant Agda without a clear reason in mind rather than its syntax language shares similarity with Haskell, the programming language of our proof-reconstruction tool, Athena.

3.1 LCF-Style Theorem Proving

3.2 The Translation Method

3.3 Reconstruction Work-flow

Explain in a diagram like we did in the slides for the AIM ...

3.4 Emulation of Inference Rules in Agda

3.5 Implementation

...

3.6 Examples

...

4 Conclusions

`simplify` and `canonicalize` coverage. Proof-reconstruction can be done in Agda from the Metis' proofs. ...

Future Work. First-Order Logic support.

Acknowledgments. We thank EAFIT University in Medellín, Colombia for funding support.

References

1. Appel, K.I.: Horn Sentences in Identity Theory. The Journal of Symbolic Logic 24(4), 306–310 (1959), `http://www.jstor.org/stable/2963901`
2. Armand, M., Faure, G., Grégoire, B., Keller, C., Théry, L., Werner, B.: A Modular Integration of SAT/SMT Solvers to Coq through Proof Witnesses, pp. 135–150. Springer Berlin Heidelberg, Berlin, Heidelberg (2011), `https://doi.org/10.1007/978-3-642-25379-9{\_}12`
3. Barrett, C., Conway, C.L., Deters, M., Hadarean, L., Jovanović, D., King, T., Reynolds, A., Tinelli, C.: CVC4. In: Gopalakrishnan, G., Qadeer, S. (eds.) Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings, pp. 171–177. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
4. Bezem, M., Hendriks, D., De Nivelle, H.: Automated Proof Construction in Type Theory Using Resolution. Journal of Automated Reasoning 29(3), 253–275 (2002)
5. Blanchette, J.C., Böhme, S., Paulson, L.C.: Extending Sledgehammer with SMT Solvers. Journal of Automated Reasoning 51(1), 109–128 (jun 2013), `https://doi.org/10.1007/s10817-013-9278-5`
6. Böhme, S., Weber, T.: Fast LCF-Style Proof Reconstruction for Z3, pp. 179–194. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
7. Bouton, T., de Oliveira, D., Déharbe, D., Fontaine, P.: veriT: An Open, Trustable and Efficient SMT-Solver, pp. 151–156. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
8. De Moura, L., Bjørner, N.: Z3: An efficient SMT Solver. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 4963 LNCS, pp. 337–340 (2008)

9. De Nivelle, H.: Bliksem 1.10 User Manual, http://www.ii.uni.wroc.pl/{~}nivelle/software/bliksem.
10. Eén, N., Sörensson, N.: An Extensible SAT-solver. pp. 502–518. Springer, Berlin, Heidelberg (2004), http://link.springer.com/10.1007/978-3-540-24605-3{_}37
11. Ekici, B., Mebsout, A., Tinelli, C., Keller, C., Katz, G.: SMTCoq: A plug-in for integrating SMT solvers into Coq. stanford.edu (2017), http://web.stanford.edu/{~}guyk/pub/CAV2017{_}C.pdf
12. Fleury, M.&., Blanchette, J.: Translation of Proofs Provided by External Provers More Automatic Prover Support for Isabelle: Two Higher-Order Provers and a SMT Solver pp. 2–0 (2014), http://perso.eleves.ens-rennes.fr/{~}mfleur01/documents/Fleury{_}internship2014.pdf
13. Foster, S., Struth, G.: Integrating an Automated Theorem Prover into Agda, pp. 116–130. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
14. Hillenbrand, T., Buch, A., Vogt, R., Löchner, B.: WALDMEISTER - High-Performance Equational Deduction. Journal of Automated Reasoning 18(2), 265–270 (1997), https://doi.org/10.1023/A:1005872405899
15. Humberstone, L.: The Connectives. MIT Press (2011)
16. Hurd, J.: First-order Proof Tactics In Higher-order Logic Theorem Provers. Design and Application of Strategies/Tactics in Higher Order Logics, number NASA/CP-2003-212448 in NASA Technical Reports pp. 56–68 (2003)
17. Kanso, K., Setzer, A.: A Light-weight Integration Of Automated And Interactive Theorem Proving. Mathematical Structures in Computer Science 26(1), 129–153 (2016)
18. Klieber, W.: Formal Verification Using Quantified Boolean Formulas (QBF). Ph.D. thesis, Carnegie Mellon University (2014), http://www.cs.cmu.edu/{~}wklieber/thesis.pdf
19. Lonsing, F., Egly, U.: DepQBF 6.0: A Search-Based QBF Solver Beyond Traditional QCDCL. In: International Conference on Automated Deduction. Springer, Gotenburg (2017), https://arxiv.org/pdf/1702.08256.pdf
20. Meng, J., Quigley, C., Paulson, L.C.: Automation for Interactive Proof: First Prototype. Information and computation 204(10), 1575–1596 (2006)
21. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: engineering an efficient SAT solver. In: Proceedings of the 38th Design Automation Conference (DAC 2001). pp. 530–535 (2001)
22. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-order Logic, vol. 2283. Springer Science & Business Media (2002)
23. Norrish, M., Slind, K.: The HOL system description (2007)
24. Otten, J.: leanCoP 2.0 and ileanCoP 1.2: High Performance Lean Theorem Proving in Classical and Intuitionistic Logic (System Descriptions). In: Automated Reasoning, pp. 283–291. Springer Berlin Heidelberg, Berlin, Heidelberg (2008), http://link.springer.com/10.1007/978-3-540-71070-7{_}23
25. Paulson, L.C.: Isabelle: A Generic Theorem Prover, vol. 828. Springer Science & Business Media (1994)
26. Riazanov, A., Voronkov, A.: Vampire. In: Automated Deduction — CADE-16: 16th International Conference on Automated Deduction Trento, Italy, July 7–10, 1999 Proceedings, pp. 292–296. Springer Berlin Heidelberg, Berlin, Heidelberg (1999), https://doi.org/10.1007/3-540-48660-7{_}26https://doi.org/10.1007/3-540-48660-7{\%}7B{_}{\%}7D26

27. Schmitt, S., Lorigo, L., Kreitz, C., Nogin, A.: JProver: Integrating Connection-Based Theorem Proving into Interactive Proof Assistants. pp. 421–426. Springer, Berlin, Heidelberg (2001), http://link.springer.com/10.1007/3-540-45744-5{_}34
28. Schulz, S.: E – A Brainiac Theorem Prover. Journal of AI Communications 15(2/3), 111–126 (2002)
29. Sutcliffe, G.: The TPTP Problem Library and Associated Infrastructure. Journal of Automated Reasoning 43(4), 337 (jul 2009), https://doi.org/10.1007/s10817-009-9143-8
30. Sutcliffe, G., Zimmer, J., Schulz, S.: TSTP data-exchange formats for automated theorem proving tools. Distributed Constraint Problem Solving and Reasoning in Multi-Agent Systems 112, 201–215 (2004)
31. Tammet, T.: Gandalf. Journal of Automated Reasoning 18(2), 199–204 (1997), http://www.scopus.com/inward/record.url?eid=2-s2.0-0031108576{\&}partnerID=tZOtx3y1
32. Team, T.A.D.: Agda 2.4.2.3, version 8. edn. (2015), http://wiki.portal.chalmers.se/agda/pmwiki.php
33. Team, T.C.D.: The Coq Proof Assistant. Reference Manual, version 8. edn. (2015)
34. Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P.: SPASS version 3.5. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 5663 LNAI, pp. 140–145 (2009)