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 since it depends on the integration of two complex system. On one hand, we have the automatic theorem provers (henceforth ATP) and their specification logic. These tools are usually classified in at least one of the following categories. A SAT solver (e.g. `zChaff` [29] and `MiniSat` [13]) to prove unsatisfiability of CNF formulas, a QBF solver (e.g. `GhostQ` [26] and `DepQBF` [27]) to prove satisfiability and invalidity of quantified Boolean formulas, a SMT solver (e.g `CVC4` [3], `veriT` [9], and `Z3` [11]) to prove unsatisfiability of formulas from first-order logic with theories, and a prover for validity of formulas from first-order logic with equality (e.g. `E` [36], `leanCoP` [32], `Metis` [21], `SPASS` [43] and `Vampire` [34]), high-order logic (e.g. `Leo-II` [4] and `Satallax` [10]) or intuitionistic logic (e.g. `ileanCoP` [32], `JProver` [35], and `Gandalf` [40]), among others.

On the other hand, we have the proof checkers, interactive theorem provers (henceforth ITP) or proof assistants (e.g. `Agda` [41], `Coq` [42], `Isabelle` [33], and `HOL4` [31]). The ITP tools provide us the logic framework to check and validate the reply of the ATPs, since they allow us to define the formal language for the problems like operators, logic variables, axioms, and theorems.

A proof reconstruction tool provides such an integration in one direction, the bridge between ATPs to ITPs. This is mostly a translation of the reply delivered by the prover into the formalism of the proof assistant. Because the formalism of the source (the proof generated by the ATP) is not necessarily the same logic in the target, the reconstruction turns out in a "reverse engineering" task. Then, reconstructing a proof involves a deep understanding of the algorithms in the ATP and the specification logic in the ITP.

What we need from the ATP tools is a proof object in a consistent format to work with, that is, a full script describing step-by-step with exhaustive details and without ambiguities, the table of the derivation to get the actual proof. For problems in classical propositional logic (henceforth CPL), from a list of at least forty[1] ATPs, just a few provers are able to deliver proofs (e.g. `CVC4` [3], `SPASS`, and `Waldmeister` [18]) and a little bit less reply with a proof in a file format like TSTP [39] (e.g. `E`, `Metis`, `Vampire`, and `Z3`), LFSC [37], or the SMT-LIB [8] format.

Many approaches have been proposed and some tools have been implemented for proof reconstruction in the last decades. These programs are relevant not only because it helps to spread their usage but they also increase the confidence of their users about their algorithms and their correctness (see, for example, bugs in ATPs [25], [8], and [15]). We mention some tools in the following.

Related Work.

`Sledgehammer` is a tool for `Isabelle` proof assistant [33] that provides a full integration of automatic theorem provers including ATPs (see, for example, [28], [6] and [15]) and SMT solvers (see, for example, [22], [7], [6], and [15]) with `Isabelle/HOL` [30], the specialization of `Isabelle` for higher-order logic. Sultana, Benzmüller, and Paulson [13] integrates `Leo-II` and `Satallax`, two theorem provers for high-order logic with `Isabelle/HOL` proposing a modular proof reconstruction work flow.

`SMTCoq` [2, 14] is a tool for the `Coq` proof assistant [42] which provides a certified checker for proof witness coming from the SMT solver `veriT` [9] and adds a new tactic named verit, that calls `veriT` on any `Coq` goal. In [5], given a fixed but arbitrary first-order signature, the authors transform a proof produced by the first-order automatic theorem prover `Bliksem` [12] in a `Coq` proof term.

Hurd [20] integrates the first-order resolution prover `Gandalf` with `HOL` [31], a high-order theorem prover, following a LCF model implementing the tactic `GANDALF_TAC`. The SMT solver `CVC4` was integrated with `HOL Light`, a version of `HOL` but with a simpler logic core. `PRocH` tool by Kaliszyk and Urban [23] reconstruct proofs from different ATPs to `HOL Light`, replaying the detailed inference steps from the ATPs with internal inference methods implemented in the ITP.

`Waldmeister` is an automatic theorem prover for unit equational logic [18]. Foster and Struth [16] integrate `Waldmeister` into `Agda` [41]. This integration requires a proof reconstruction step but authors' approach is restricted to pure equational logic –also called identity theory [19]– that is, first-order logic with equality but no other predicate symbols and no functions symbols [1].

Kanso and Setzer [24] integrate as SAT solvers (`iProver`, `E`, and `Z3`) in `Agda` using an approach that they call *oracle and reflection.*

In this paper, we describe the integration of `Metis` prover with the proof assistant `Agda`. 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

[1] ATPs available from the web service `SystemOnTPTP` of the TPTP World.

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 [21].

2.1 Input and Output Language

The `TPTP` language –which includes the First-Order Form (FOF) and Clause Normal Form (CNF) formats [38] – is de facto input standard language and the `TSTP` language is de facto output standard language [39].

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 $\bot$ since `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 recollects all hypothesis required in order to prove the goal. At the end, the final hypothesis represent the subgoals.

3 Proof Reconstruction in Agda

As a proof checker, we choose the proof-assistant `Agda`.

Such TPTP proofs produced by ATPs on the type-annotated input are the starting point for the HOL proof reconstruction.

3.1 LCF-Style Theorem Proving

The propositional formulas are represented using the `Prop` data type.

```
data Prop : Set where
  Var                   : Fin n → Prop
  ⊤                     : Prop
  ⊥                     : Prop
  _∧_ _∨_ _⇒_ _⇔_       : (φ ψ : Prop) → Prop
  ¬_                    : (φ : Prop) → Prop
```

The theorems in CPL are represented using an abstract data type to implement a natural deduction calculus. Theorems represent the *sequents* $\Gamma \vdash \phi$, where Γ is a set of hypothesis –using for the implementation the list data type from the `Agda` standard library version– and ϕ is the sequent's conclusion.

```
data _⊢_ : (Γ : Ctxt) (φ : Prop) → Set where
  assume   : ∀ {Γ} → (φ : Prop)       → Γ , φ ⊢ φ
  axiom    : ∀ {Γ} → (φ : Prop)       → φ ∈ Γ
                                      → Γ ⊢ φ
  weaken   : ∀ {Γ} {φ} → (ψ : Prop)   → Γ ⊢ φ
                                      → Γ , ψ ⊢ φ
  weaken₂  : ∀ {Γ} {φ} → (ψ : Prop)   → Γ ⊢ φ
                                      → ψ :: Γ ⊢ φ
  ⊤-intro  : ∀ {Γ}                    → Γ ⊢ ⊤
  ⊥-elim   : ∀ {Γ} → (φ : Prop)       → Γ ⊢ ⊥
                                      → Γ ⊢ φ
  ¬-intro  : ∀ {Γ} {φ}                → Γ , φ ⊢ ⊥
                                      → Γ ⊢ ¬ φ
  ¬-elim   : ∀ {Γ} {φ}                → Γ ⊢ ¬ φ → Γ ⊢ φ
                                      → Γ ⊢ ⊥
  ∧-intro  : ∀ {Γ} {φ ψ}              → Γ ⊢ φ → Γ ⊢ ψ
                                      → Γ ⊢ φ ∧ ψ
  ∧-proj₁  : ∀ {Γ} {φ ψ}              → Γ ⊢ φ ∧ ψ
                                      → Γ ⊢ φ
  ∧-proj₂  : ∀ {Γ} {φ ψ}              → Γ ⊢ φ ∧ ψ
                                      → Γ ⊢ ψ
  ∨-intro₁ : ∀ {Γ} {φ} → (ψ : Prop)   → Γ ⊢ φ
```

$$
\begin{array}{ll}
 & \rightarrow\ \Gamma \vdash \phi \vee \psi \\
\vee\text{-intro}_2\ :\ \forall\,\{\Gamma\}\,\{\psi\}\ \rightarrow\ (\phi\ :\ \mathsf{Prop}) & \rightarrow\ \Gamma \vdash \psi \\
 & \rightarrow\ \Gamma \vdash \phi \vee \psi \\
\vee\text{-elim}\ :\ \forall\,\{\Gamma\}\,\{\phi\ \psi\ \chi\} & \rightarrow\ \Gamma\ ,\ \phi \vdash \chi \\
 & \rightarrow\ \Gamma\ ,\ \psi \vdash \chi \\
 & \rightarrow\ \Gamma\ ,\ \phi \vee \psi \vdash \chi \\
\Rightarrow\text{-intro}\ :\ \forall\,\{\Gamma\}\,\{\phi\ \psi\} & \rightarrow\ \Gamma\ ,\ \phi \vdash \psi \\
 & \rightarrow\ \Gamma \vdash \phi \Rightarrow \psi \\
\Rightarrow\text{-elim}\ :\ \forall\,\{\Gamma\}\,\{\phi\ \psi\} & \rightarrow\ \Gamma \vdash \phi \Rightarrow \psi\ \rightarrow\ \Gamma \vdash \phi \\
 & \rightarrow\ \Gamma \vdash \psi \\
\Leftrightarrow\text{-intro}\ :\ \forall\,\{\Gamma\}\,\{\phi\ \psi\} & \rightarrow\ \Gamma\ ,\ \phi \vdash \psi \\
 & \rightarrow\ \Gamma\ ,\ \psi \vdash \phi \\
 & \rightarrow\ \Gamma \vdash \phi \Leftrightarrow \psi \\
\Leftrightarrow\text{-elim}_1\ :\ \forall\,\{\Gamma\}\,\{\phi\ \psi\} & \rightarrow\ \Gamma \vdash \phi\ \rightarrow\ \Gamma \vdash \phi \Leftrightarrow \psi \\
 & \rightarrow\ \Gamma \vdash \psi \\
\Leftrightarrow\text{-elim}_2\ :\ \forall\,\{\Gamma\}\,\{\phi\ \psi\} & \rightarrow\ \Gamma \vdash \psi\ \rightarrow\ \Gamma \vdash \phi \Leftrightarrow \psi \\
 & \rightarrow\ \Gamma \vdash \phi
\end{array}
$$

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 of Medellín, Colombia for funding support. This is part of first author's Master thesis in Applied Mathematics, written under the supervision of Andrés Sicard-Ramírez for the Logic and Computation Research Group at the EAFIT University.

We thank Joe Leslie-Hurd for supporting comments about `Metis`. We also acknowledge the work done by Alejandro Gómez-Londoño [17] that deserves as the basis code for our parsing module for TSTP files. And last but not least, we gratefully acknowledge Andreas Abel and Chalmers University of Gothenburg, Sweden for inviting us to be part of the Agda Implementors' Meeting XXV where we presented part of this paper.

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. Benzmüller, C., Paulson, L.C., Theiss, F., Fietzke, A.: LEO-II - A cooperative automatic theorem prover for classical higher-order logic (system description). In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 5195 LNAI, pp. 162–170 (2008)
5. Bezem, M., Hendriks, D., De Nivelle, H.: Automated Proof Construction in Type Theory Using Resolution. Journal of Automated Reasoning 29(3), 253–275 (2002)
6. Blanchette, J., 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`
7. Böhme, S., Weber, T.: Fast LCF-Style Proof Reconstruction for Z3, pp. 179–194. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
8. Böhme, S., Weber, T.: Designing Proof Formats: A User's Perspective - Experience Report. In: First International Workshop on Proof eXchange for Theorem Proving - PxTP 2011 (2011), `http://hal.inria.fr/hal-00677244`
9. 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)
10. Brown, C.E.: Satallax: An automatic higher-order prover. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 7364 LNAI, pp. 111–117 (2012)
11. 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)

12. De Nivelle, H.: Bliksem 1.10 User Manual, http://www.ii.uni.wroc.pl/~nivelle/software/bliksem.
13. 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
14. 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
15. 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
16. Foster, S., Struth, G.: Integrating an Automated Theorem Prover into Agda, pp. 116–130. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
17. Gómez-Londono, A.: Proof Reconstruction: Parsing Proofs. Tech. rep., EAFIT University (2015), http://repository.eafit.edu.co/handle/10784/5484
18. 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
19. Humberstone, L.: The Connectives. MIT Press (2011)
20. Hurd, J.: Integrating Gandalf and HOL. pp. 311–321. Springer, Berlin, Heidelberg (1999), http://link.springer.com/10.1007/3-540-48256-3_21
21. 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), http://www.gilith.com/research/papers
22. Hurlin, C., Chaieb, A., Fontaine, P., Merz, S., Weber, T.: Practical Proof Reconstruction for First-Order Logic and Set-Theoretical Constructions. In: Dixon, L., Johansson, M. (eds.) Proceedings of the Isabelle Workshop 2007. pp. 2–13. Bremen, Germany (2007)
23. Kaliszyk, C., Urban, J.: PRocH: Proof Reconstruction for HOL Light, pp. 267–274. Springer Berlin Heidelberg, Berlin, Heidelberg (2013), https://doi.org/10.1007/978-3-642-38574-2_18
24. Kanso, K., Setzer, A.: A Light-weight Integration Of Automated And Interactive Theorem Proving. Mathematical Structures in Computer Science 26(1), 129–153 (2016), http://cs.swansea.ac.uk/~cskarim/agda/
25. Keller, C.: A Matter of Trust: Skeptical Communication Between Coq and External Provers (2013), https://hal.archives-ouvertes.fr/pastel-00838322/
26. Klieber, W.: Formal Verification Using Quantified Boolean Formulas (QBF). Ph.D. thesis, Carnegie Mellon University (2014), http://www.cs.cmu.edu/~wklieber/thesis.pdf
27. 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
28. Meng, J., Quigley, C., Paulson, L.C.: Automation for Interactive Proof: First Prototype. Information and computation 204(10), 1575–1596 (2006)
29. 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)
30. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-order Logic, vol. 2283. Springer Science & Business Media (2002)

31. Norrish, M., Slind, K.: The HOL system description (2007)
32. 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`
33. Paulson, L.C.: Isabelle: A Generic Theorem Prover, vol. 828. Springer Science & Business Media (1994)
34. 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_26`
35. 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`
36. Schulz, S.: E – A Brainiac Theorem Prover. Journal of AI Communications 15(2/3), 111–126 (2002)
37. Stump, A., Oe, D.: Towards an SMT proof format. In: Proceedings of the Joint Workshops of the 6th International Workshop on Satisfiability Modulo Theories and 1st International Workshop on Bit-Precise Reasoning - SMT '08/BPR '08. p. 27. ACM Press, New York, New York, USA (2008), `http://portal.acm.org/citation.cfm?doid=1512464.1512470`
38. 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`
39. 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)
40. 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`
41. Team, T.A.D.: Agda 2.4.2.3, version 8. edn. (2015), `http://wiki.portal.chalmers.se/agda/pmwiki.php`
42. Team, T.C.D.: The Coq Proof Assistant. Reference Manual, version 8. edn. (2015), `https://coq.inria.fr/distrib/current/files/Reference-Manual.pdf`
43. 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)