Components of a Hammer for Type Theory: Coq Goal Translation and Reconstruction

Łukasz Czajka and Cezary Kaliszyk
University of Innsbruck, Austria
{lukasz.czajka,cezary.kaliszyk}@uibk.ac.at

Abstract

Proof assistants based on the Calculus of Constructions lack powerful general purpose automation. In this talk we present an extension of the various proof advice components to type theory: (i) a translation of a significant part of the Coq logic into the format of automated proof systems; (ii) improvements of the encoding for the goals practically arising in Coq proofs; as well as (iii) a reconstruction mechanism based on a Ben-Yelles-type algorithm combined with limited rewriting, congruence closure and a first-order generalization of the left rules of Dyckhoff's system LJT.

Formalizing proofs in interactive theorem provers based on the Calculus of Inductive Constructions and its extensions is a daunting task. For systems based on simpler foundations robust general purpose procedures are able to discharge many easier goals completely automatically [2], providing strong proof advice, which reduces human labor [7]. Providing such proof advice requires an encoding of type theory judgements and proof goals into the logics and formats of automated theorem provers (ATPs), as well as a reconstruction mechanism able to build type theory derivations based on ATP-found proofs. For higher-order logic both of these proof advice components have been studied thoroughly. Powerful encodings that are able to automatically derive properties of types, such as monotonicity, as part of the translation have been studied theoretically and implemented in the recent versions of proof advice tools HOL(y)Hammer [9] and Sledgehammer [10]. The built-in HOL automation is able to reconstruct the majority of the automatically found proofs using either internal proof search [8] or source-level reconstruction. The internal proof search mechanisms provided in Coq, such as the `firstorder` tactic [3], have been insufficient for this purpose so far. Matita's ordered-paramodulation [1] is able to reconstruct many goals with up to two or three premises, and the congruence-closure based internal automation techniques in Lean [5] are also promising.

In this talk, we present our recently developed proof advice components for type theory and systems based on it. We first introduce an encoding of the Calculus of Inductive Constructions, including the additional logical constructions introduced by the Coq system, in untyped first-order logic. Subsequently, we improve the encoding for the kind of goals that practically arise in Coq standard library formal proofs. Finally, we present a reconstruction mechanism based on a Ben-Yelles-type procedure combined with a first-order generalization of the left rules of Dyckhoff's LJT, congruence closure and heuristic rewriting.

The first part of the talk, partly based on [4], introduces an encoding of (a close approximation of) the Calculus of Inductive Constructions in untyped first-order logic. The encoding should be a practical one, which implies that its general theoretical soundness is not the main focus, instead it is important that the encoding is precise enough to provide useful information about the necessary proof dependencies or instantiations. For the sake of efficiency, terms of type Prop are encoded directly as FOL formulas using a function $\mathscr{F}$. Terms that have type Type but not Prop are encoded using a function $\mathscr{G}$ as guards which essentially specify what it means for an object to have the given type. For instance, $\forall f : \tau.\varphi$ where $\tau = \Pi x : \alpha.\beta$ is translated to $\forall f.\mathscr{G}(\tau, f) \to \mathscr{F}(\varphi)$ where $\mathscr{G}(\tau, f) = \forall x.\mathscr{G}(\alpha, x) \to \mathscr{G}(\beta, fx)$. So $\mathscr{G}(\tau, f)$ says that an object f has type $\tau = \Pi x : \alpha.\beta$ if for any object x of type α, the application fx has type β.

The encoding has been evaluated on the theorems proved in the Coq standard library, experimentally confirming that it is efficient and precise enough to build an automated reasoning-based proof advice system for Coq. In particular, state-of-the-art classical ATPs are able to reprove above 30% of the human-written proofs, and the dependencies used in the automatically found proofs are indeed those needed by the user to prove the theorems in Coq. We will discuss ideas how this number can be improved by adapting the techniques available in other proof assistants to type theory in the talk. In order to ultimately confirm the adequacy of the encoding, it is necessary to reconstruct the proofs in Coq.

In the second part of the talk, we report on an early work-in-progress on proof reconstruction. We evaluate the Coq internal reconstruction mechanisms including `tauto` and `firstorder` [3] on the original proof dependencies and on the ATP found proofs, which are in certain cases more precise. In particular `firstorder` seems insufficient for finding proofs for problems created using the advice obtained from the ATP runs. This is partly caused by the fact that it does not fully axiomatize equality, but even on problems which require only purely logical first-order reasoning its running time is often unacceptable.

The formulas that we attempt to reprove usually belong to fragments of intuitionistic logic low in the Mints hierarchy [11]. Most of proved theorems follow by combining a few known lemmas. The formulas are small and in practice often belong to the negative fragment of intuitionistic logic. This raises a possibility of devising an automated proof procedure optimized for this fragment and for the utilization of the advice obtained from the ATP runs. We implemented a preliminary version of a Ben-Yelles-type procedure (essentially `eauto`-type proof search with lightweight looping check) augmented with a first-order generalization of the left rules of Dyckhoff's system LJT [6], the use of the `congruence` tactic, and heuristic rewriting using equational hypotheses. An experimental evaluation on the problems originating from ATP proofs of lemmas in the Coq standard library shows that in our setting this algorithm tends to perform significantly better than the available Coq's tactics. Our tactic manages to reconstruct above 90% of the reproved theorems. However, it needs to be remarked that if we utilize the advice obtained from ATP runs then about 50% of the the reproved theorems follow by a combination of hypothesis simplification, the tactics `intuition`, `auto`, `easy`, `congruence` and a few simple heuristics. The reconstruction success rate of the `firstorder` tactic combined with various heuristics is about 70% if generic axioms for equality are added to the context.

References

[1] A. Asperti and E. Tassi. Higher order proof reconstruction from paramodulation-based refutations: The unit equality case. In M. Kauers, M. Kerber, R. Miner, and W. Windsteiger, editors, *Mathematical Knowledge Management (MKM 2007)*, volume 4573 of *LNCS*, pages 146–160. Springer, 2007.

[2] J. C. Blanchette, C. Kaliszyk, L. C. Paulson, and J. Urban. Hammering towards QED. *J. Formalized Reasoning*, 9(1):101–148, 2016.

[3] P. Corbineau. First-order reasoning in the calculus of inductive constructions. In S. Berardi, M. Coppo, and F. Damiani, editors, *Types for Proofs and Programs (TYPES 2003)*, volume 3085 of *LNCS*, pages 162–177. Springer, 2003.

[4] Ł. Czajka and C. Kaliszyk. Encoding the calculus of constructions in FOL for a hammer for Coq. Submitted, `http://cl-informatik.uibk.ac.at/cek/submitted/lcck-coqencode.pdf`, 2016.

[5] L. M. de Moura. Dependent type practice (invited talk). In J. Avigad and A. Chlipala, editors, *Conference on Certified Programs and Proofs (CPP 2016)*, page 2. ACM, 2016.

[6] R. Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *J. Symb. Log.*, 57(3):795–807, 1992.

[7] T. Hales. Developments in formal proofs. *Séminaire Bourbaki*, 1086, 2013–2014. abs/1408.6474.

[8] J. Hurd. First-order proof tactics in higher-order logic theorem provers. In M. Archer, B. D. Vito, and C. Muñoz, editors, *Design and Application of Strategies/Tactics in Higher Order Logics (STRATA 2003)*, number NASA/CP-2003-212448 in NASA Technical Reports, pages 56–68, 2003.

[9] C. Kaliszyk and J. Urban. Learning-assisted automated reasoning with Flyspeck. *J. Autom. Reasoning*, 53(2):173–213, 2014.

[10] L. C. Paulson and J. Blanchette. Three years of experience with Sledgehammer, a practical link between automated and interactive theorem provers. In G. Sutcliffe, S. Schulz, and E. Ternovska, editors, *International Workshop on the Implementation of Logics (IWIL 2010)*, volume 2 of *EPiC*, pages 1–10. EasyChair, 2012.

[11] A. Schubert, P. Urzyczyn, and K. Zdanowski. On the Mints hierarchy in first-order intuitionistic logic. In A. M. Pitts, editor, *Foundations of Software Science and Computation Structures (FoSSaCS 2015)*, volume 9034 of *Lecture Notes in Computer Science*, pages 451–465. Springer, 2015.