Customizations of Deduction Systems

Claude Kirchner
INRIA Bordeaux - Sud-Ouest

Proof engineering is a main concern of industrial actors and a major scientific challenge. For instance, a "simple" automotive cruise control software consists of more than one hundred thousand lines of code and, of course, one prefers to have its main functionalities, if not all, formally proved. On the other side, the design and in depth study of appropriate logics and deduction systems is a quite active research area. At the cross of informatics, logic and mathematics, the notion of proofs has a long and fruitful history which now becomes even richer with a century of research and experience in its formalization. In this context, proof engineering becomes crucial and relies on semi-interactive designs where human interactions are unavoidable. Moreover, to be well designed, proofs have to be well understood and built. As the size of critical software increases dramatically, proof engineering should scale-up and this is in itself a difficult challenge.

The engineering of proofs is often done today using proof assistants like Coq [The04], Isabelle [Pau94], PVS [ORS92], HOL [HOL93], Mizar [Rud92], B and large libraries of automated strategies and formalized theories ease this task.

In this context one has to deal with at least two main difficulties. First, proof engineering should scale-up as the theories describing the context become huge and may consist of thousand of axioms and definitions, some of them being quite sophisticated. Second, the proof assistant needs to provide the user with appropriate ways to understand and to guide the proof construction. Both concerns are currently tackled by making libraries available, by providing specific tactics, tacticals or strategies (see typically `coq.inria.fr`), by integration rewriting [BJO02] and decision procedures [NKK02, Alv00, MQP06] safely into the proof assistants, or by interfacing first-order automated theorem provers with proof assistants like [BHdN02] or like the use of Zenon in Focal [Pre05].

Indeed, these approaches raise the question of structuring the theories of interest. For instance one would like to identify the subtheory of lists or of naturals to apply specific decision procedures, *e.g.* [KRRT06] and of course finding a good modular structure is one of the first steps in an engineering process.

To deal with these crucial questions, we have proposed in [BHK07] a foundational framework making use of three complementary dimensions. First, as pioneered by *deduction modulo*, the computational axioms should be identified. Typically the definition of addition on naturals ought to be embedded into a congruence modulo which deduction is performed [DHK03]. In this case, the deduction rules

like the one of natural deduction or of the sequent calculus are not modified but they are applied modulo a congruence embedding part of the theory. Second, we are proposing a complementary approach where *new deduction rules* are inferred from part of the theory in a correct, systematic and complete way. Third, the rest of the theory will be used as the context on which all the standard and new deduction rules will act, possibly modulo some congruence.

Technically, a theory is split in three parts $\mathcal{T}h = \mathcal{T}h_1 \cup \mathcal{T}h_2 \cup \mathcal{T}h_3$ and instead of seeking for a proof of $\mathcal{T}h_1 \cup \mathcal{T}h_2 \cup \mathcal{T}h_3 \vdash \varphi$, we are building a proof of $\mathcal{T}h_3 \vdash^{+\mathcal{T}h_2}_{\sim_{\mathcal{T}h_1}} \varphi$, *i.e.* we use the theory $\mathcal{T}h_3$ to prove φ using the extended deduction system modulo the congruence $\sim_{\mathcal{T}h_1}$. We assume that the propositions in $\mathcal{T}h_2$ are all proposition rewrite rules, *i.e.* are of the form $\forall \overline{x}.(P \Leftrightarrow \varphi)$, where P is atomic.

This approach has interesting consequences like admitting unbounded proof size speed-up [Bur07] as well as the capability to represent elaborated logical families like Pure Type System [Bur08]. Based on collaborations with and results of Paul Brauner, Guillaume Burel, Gilles Dowek, Clément Houtmann, we will develop the main approach and results of Superdeduction Modulo and show how this contributes to the modular design of complex proofs. The consequence of these principles results in the foundation of a new generation of proof assistants for which we have a first downloadable prototype, lemuridæ (`rho.loria.fr`). In particular proofs in higher-order logic, mathematical induction, equational logic become in these settings quite convenient and natural.

References

[Alv00] C. Alvarado. Reflection for rewriting in the calculus of inductive constructions. In *Proceedings of TYPES 2000*, Durham, United Kingdom, December 2000.

[BHdN02] M. Bezem, D. Hendriks, and H. de Nivelle. Automated proof construction in type theory using resolution. *Journal of Automated Reasoning*, 29(3-4):253–275, 2002.

[BHK07] P. Brauner, C. Houtmann, and C. Kirchner. Principles of superdeduction. In *Proceedings of LICS*, July 2007.

[BJO02] F. Blanqui, J.-P. Jouannaud, and M. Okada. Inductive Data Type Systems. *Theoretical Computer Science*, 272(1-2):41–68, 2002.

[Bur07] G. Burel. Unbounded proof-length speed-up in deduction modulo. Technical report, INRIA Lorraine, 2007. Available at `http://hal.inria.fr/inria-00138195`.

[Bur08] G. Burel. Superdeduction as a logical framework. In F. Pfenning, editor, *Proceedings of LICS'08*, Pittsburgh, PA, USA, jun 2008.