

Coq Modulo Theory

Pierre-Yves Strub
INRIA - Tsinghua University
pierre-yves@strub.nu

Qian Wang
Tsinghua University
q-w04@mails.tsinghua.edu.cn

Theorem provers like COQ [3] based on the Curry-Howard isomorphism enjoy a mechanism which incorporates computations within deductions. This allows replacing the proof of a proposition by the proof of an equivalent proposition obtained from the former thanks to possibly complex computations. Adding more power to this mechanism leads to a calculus which is more expressive (more terms are typable), which provides more automation (more deduction steps are hidden in computations) and, most importantly, eases the use of dependent data types in proof development.

COQ was initially based on the Calculus of Constructions (CC) of Coquand and Huet [4], which is an impredicative type theory incorporating polymorphism, dependent types and type constructors. At that time, computations were restricted to β -reduction: other forms of computations were encoded as deductions.

The Calculus of Inductive Constructions of Coquand and Paulin [5, 11] introduced inductive types and their associated elimination rules. CIC permits, for example, a definition of Peano natural numbers based on the two constructors 0 and S along with the ability to define addition by induction: $x + 0 \rightarrow x$ and $x + S(y) \rightarrow S(x + y)$. The calculus can then identify the expressions $x + S(S(y))$ and $S(x + y) + S(0)$, but fails in identifying $x + S(y)$ and $S(x) + y$. This forbids users to easily define functions on dependent data-types (like the `reverse` function on dependent lists) as the types `list(n + 1)` and `list(1 + n)` will not be convertible to each other.

In the 90's, new attempts to incorporate user-defined computations as rewrite rules were carried out. This resulted in the definition of the Calculus of Algebraic Constructions [1]. By introducing the correct rewriting rules, the calculus is able to identify terms like $x + S(y)$ and $S(x) + y$. Although quite powerful (CAC captures CIC [2]), this paradigm does not yet fulfill all needs, as it fails to identify open terms like $x + y$ and $y + x$.

Further steps in the direction of integrating first-order theories into the Calculus of Constructions are Stehr's Open Calculus of Constructions (OCC) [8] and Oury's Exten-

sional Calculus of Constructions (ECC) [7]. OCC allows the use of an arbitrary equational theory in conversion. ECC can be seen as a particular case of OCC in which all provable equalities can be used in conversion. Unfortunately, strong normalization and decidability of type checking are lost in ECC and OCC.

In [9], we defined the Calculus of Inductive Congruence Constructions (CCIC), which was our first answer to the problem of incorporating validity entailment of an equational first-order theory \mathcal{T} in the computation mechanism of the Calculus of Inductive Constructions. This calculus partially solved the problem but contained too many ad-hoc restrictions to be implementable in practice. Besides, the procedure could not be invoked to solve goals occurring below eliminators.

1. Achievements

Our objective is to incorporate in the Calculus of Inductive Constructions, a general mechanism invoking a decision procedure for equality in some first-order theory \mathcal{T} , for solving conversion-goals of the form $\Gamma \Rightarrow U = V$. This mechanism should be simple enough to be implementable in practice.

Our theoretical contribution is the definition of a new version of the Calculus of Congruent Inductive Constructions, called COQMT. COQMT integrates in its computational part the entailment relation - via decision procedures - of a first order theory \mathcal{T} over equalities. A major technical innovation of COQMT lies in the computational mechanism: goals are sent to the decision procedure together with a set of user hypotheses available from the current context Γ of the proof - a mechanism we call *equations extraction*.

Not only is the new formulation much simpler, but is also more general: in particular, we succeeded to remove the ad-hoc restrictions of CCIC [9]. In particular, conversion now contains the entire equational theory \mathcal{T} , allowing its use below eliminators.

In CCIC, meta-theoretical results were obtained via a complex sequence of definitions and lemmas. By drastically simplifying the definition of CCIC, we were able to

¹This work was partly supported by the ANR ANR-08-BLAN-0326-01

carry out the meta-theoretical study of COQMT in a much simpler way than in [9]: subject reduction, strong normalization of reduction, logical consistency and decidability of type-checking are all satisfied.

Further, we have formalized CoqMT (without the equations extraction mechanism) in COQ, and already carried out part of its formal meta-theoretical study.

We have implemented COQMT, a new version of COQ based on our work. As of today, this implementation allows the use of decision procedures in the conversion rule. The *equations extraction* mechanism will be released very soon.

We developed in COQMT the theory of dependent lists as a paradigmatic example. In COQMT, dependent lists definitions are just those of non-dependent lists which can be found in Coq. Likewise, proofs of all basic properties of dependent lists in CoqMT follow from the ones of non-dependent lists in Coq. The full (commented) development can be found in the source release of COQMT (*test-suite/dp/**).

All these developments are available at on the first author website ². An extended abstract of the first author results can be found in [10].

2. Work in progress

When formalizing abstract mathematical structures and properties (abstract algebras, lattices, categories, ...), it is often necessary to use constructions which are not compatible with impredicativity. This problem can be addressed by introducing a cumulative hierarchy of predicative universes, and then define the needed constructions in this hierarchy. Indeed, the current version of COQ has such a type hierarchy, which is similar to the one of [6], except for the impredicative universe of propositions situated at the bottom of the hierarchy.

Although the current implementation of COQMT- as an extension of COQ - makes use of this type hierarchy, we did not study the meta-theory of such a calculus. We have strong reasons to believe that the introduction of decision procedures, as done in COQMT, is compatible with the introduction of a type hierarchy. We have recently started this study as a part of the PhD work of the second author, which aims at studying the metatheory of the Coq and CoqMT implementations.

3. Perspectives

The extraction mechanism under implementation in COQMT will allow much more than described in Sections 1 and 2: using extracted equations below eliminators and within the reduction mechanism will be possible.

In the theoretical model, however, reduction modulo \mathcal{T} cannot use extracted equations, since otherwise potential user's inconsistencies could propagate and compromise the logical consistency of the calculus. A work-around would be to only use, during reduction, a subset of extracted equations consistent by construction. Although the idea is quite straightforward, the many dependencies between conversion, reduction modulo and consistency of extracted equations make it difficult to get a definition which meets our requirement of simplicity.

Further, we plan to extend the extraction mechanism (currently restricted to equations) to arbitrary first-order propositions, possibly using other predicates (like the ordering on natural numbers) or universal quantification.

Finally, we plan to define a more general calculus where decision procedures are seen as heuristics tagging the terms of the calculus with non-intrusive conversion markers. Such a calculus would capture COQMT, allow the use of incomplete decision procedures, and lead to a more elegant implementation of COQMT.

Acknowledgment: we thank Jean-Pierre Jouannaud for useful discussions.

References

- [1] F. Blanqui. Definitions by rewriting in the calculus of constructions. *Mathematical Structures in Computer Science*, 15(1):37–92, 2005. Journal version of LICS'01.
- [2] F. Blanqui. Inductive types in the calculus of algebraic constructions. *Fundamenta Informaticae*, 65(1-2):61–86, 2005. Journal version of TLCA'03.
- [3] Coq-Development-Team. *The Coq Proof Assistant Reference Manual - Version 8.2*. INRIA, INRIA Rocquencourt, France, 2009. At URL <http://coq.inria.fr/>.
- [4] T. Coquand and G. Huet. The Calculus of Constructions. *Information and Computation*, 76(2-3):95–120, 1988.
- [5] T. Coquand and C. Paulin-Mohring. Inductively defined types. In Martin-Löf and G. Mints, editors, *Colog'88, International Conference on Computer Logic*, volume 417 of *LNCS*, pages 50–66. Springer-Verlag, 1990.
- [6] P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.
- [7] N. Oury. Extensionality in the calculus of constructions. In *Proceedings 18th TPHOL, Oxford, UK. LNCS 3603*, pages 278–293, 2005.
- [8] M. Stehr. The Open Calculus of Constructions: An equational type theory with dependent types for programming, specification, and interactive theorem proving (part I and II). *Fundamenta Informaticae*, 68, 2005.
- [9] P.-Y. Strub. *Type Theory and Decision Procedures*. PhD thesis, École Polytechnique, July 2008.
- [10] P.-Y. Strub. Coq modulo theories. In *Proceedings 19th CSL 2007, Brno, Czech Republic*, 2010. To appear.
- [11] B. Werner. *Une Théorie des Constructions Inductives*. PhD thesis, University of Paris VII, 1994.

²<http://strub.nu/research/coqmt/>