Coq à la Tarski: a predicative calculus of constructions with explicit subtyping

Ali Assaf[1,2]

[1] INRIA Paris-Rocquencourt, Paris, France
[2] École Polytechnique, Paris, France

The predicative Calculus of Inductive Constructions (pCIC), the theory behind the Coq proof system, contains an infinite hierarchy of predicative universes

$$Type_0 \in Type_1 \in Type_2 \in \dots$$

and an impredicative universe $Prop$ for propositions, together with an implicit cumulativity relation

$$Prop \subseteq Type_0 \subseteq Type_1 \subseteq Type_2 \subseteq \dots .$$

This gives rise to a subtyping relation $\leq$ which is used in the subsumption rule

$$\frac{\Gamma \vdash M : A \qquad A \leq B}{\Gamma \vdash M : B}.$$

Subtyping in Coq is implicit, and is handled by the kernel. An attempt to simplify the theory would be to make subtyping explicit, by inserting explicit coercions such as

$$c_{0,1} : Type_0 \to Type_1$$

and rely on a kernel that only uses the classic conversion rule

$$\frac{\Gamma \vdash M : A \qquad A \equiv B}{\Gamma \vdash M : B}.$$

However, because of dependent types, coercions change the shape of the types and therefore interfere with type checking.

We present a formulation of the predicative calculus of constructions using Tarski-style universes [4] where subtyping is explicit. Other such systems have been proposed in the past [5, 2, 3]. However, they do not preserve equality: a term in the original Coq system can have many non-equivalent representations in the new system, which breaks typing. As a result, these systems lose some of the expressivity of Russell-style universes with implicit subtyping, and are therefore incomplete.

Our system fully preserves equality. By adding aditional equations between terms, we ensure that every well-typed term in the original system has a unique canonical representation in our system. To our knowledge, this is the first time such work has been done for the full predicative calculus of constructions. We will also show how to orient the equations into reduction rules. This work can be used as a basis for embedding Coq in a logical framework like the λΠ-calculus modulo [1], implemented in Dedukti [6].

References

[1] Denis Cousineau and Gilles Dowek. Embedding pure type systems in the lambda-pi-calculus modulo. In Simona Ronchi Della Rocca, editor, *TLCA*, volume 4583 of *Lecture Notes in Computer Science*, pages 102–117. Springer, 2007.

[2] Zhaohui Luo. *Computation and Reasoning: A Type Theory for Computer Science*. Oxford University Press, Inc., New York, NY, USA, 1994.

[3] Zhaohui Luo. Notes on universes in type theory. Lecture notes for a talk at Institute for Advanced Study, Princeton (URL: `http://www.cs.rhul.ac.uk/home/zhaohui/universes.pdf`), 2012.

[4] Per Martin-Lof and Giovanni Sambin. *Intuitionistic type theory.* Bibliopolis Naples, 1984.

[5] Erik Palmgren. On universes in type theory. In *Twenty Five Years of Constructive Type Theory.* Oxford University Press, 1998.

[6] Ronan Saillard. Dedukti: a universal proof checker. In *Foundation of Mathematics for Computer-Aided Formalization Workshop*, 2013.