Extensional Set Equality in the Calculus of Constructions: Extended Abstract*

Jonathan P. Seldin
Department of Mathematics and Computer Science
University of Lethbridge
Lethbridge, Alberta, Canada
jonathan.seldin@uleth.ca
http://home.uleth.ca/~jonathan.seldin

June 29, 2000

In [4, §11], I developed a significant part of set theory in the calculus of constructions by starting with a universe $U : \mathsf{Prop}$ and defining Set_U to be $U \to \mathsf{Prop}$, so that for $A : \mathsf{Set}_U$ and $a : U$, $u \in A$ becomes Au. This followed a suggestion of Huet in [2, Ch 12] and [3].

Berardi (personal communication) has informed me that set theory has not been implemented this way in Coq because contradictions follow from the axiom of extensionality. More precisely, contradictions follow from

$$(\forall A : \mathsf{Prop})(\forall B : \mathsf{Prop})((A \leftrightarrow B) \supset A =_{\mathsf{Prop}} B).$$

(See [1, Theorem 5.4.21]. Here, the equality involved, $A =_{\mathsf{Prop}} B$ is defined to be $(\forall z : \mathsf{Prop} \to \mathsf{Prop})(zA \supset zB)$.) For this reason, apparently in Coq set theory has been implemented with a new type for sets. None of the consistency results of [4] apply to this implementation.

It is hardly surprising that contradictions follow from the axiom of extensionality or from the above formula, for the equality $=_U$, which is Leibniz' equality, is *intensional* rather than extensional. The language of the calculus of constructions is rich enough to provide predicates that can distinguish different equivalent propositions. In fact, as I pointed out in [4, p. 94], if an equivalence relation $=_{\mathrm{ex}}$ between sets is defined so that

$$A =_{\mathrm{ex}} B \quad \equiv \quad (\forall x : U)(x \in A \leftrightarrow x \in B),$$

*This work was supported in part by grant RGP-23391-98 from the Natural Sciences and Engineering Research Council of Canada.

then the axiom of extensionality,

$$(\forall A : \mathsf{Set}_U)(\forall B : \mathsf{Set}_U)((\forall x : U)(x \in A \leftrightarrow x \in B) \supset A =_{\mathrm{ex}} B),$$

is trivially provable, but if $=_{\mathrm{ex}}$ is replaced by $=_{\mathsf{Set}_U}$ then it is false. Here $A =_{\mathrm{ex}} B$ should be taken as an abbreviation for $\mathsf{SetQ}UAB$, where SetQ is defined by

$$\mathsf{SetQ} \equiv \lambda U : \mathsf{Prop} \,.\, \lambda A : \mathsf{Set}_U \,.\, \lambda B : \mathsf{Set}_U \,.\, (\forall x : U)(x \in A \leftrightarrow x \in B).$$

In ordinary axiomatic set theory, set equality behaves like Leibniz equality in that equals may always be replaced by equals. This is clearly untrue in the calculus of constructions with SetQ. However, there is a class of properties with respect to which equals under SetQ can be replaced by equals. This is the class of *extensional properties*, defined as follows:

$$\begin{aligned}\mathsf{Ext} \equiv\ & \lambda U : \mathsf{Prop} \,.\, \lambda P : \mathsf{Set}_U \to \mathsf{Prop} \,. \\ & (\forall A : \mathsf{Set}_U)(\forall B : \mathsf{Set}_U)(A =_{\mathrm{ex}} B \supset PA \supset PB).\end{aligned}$$

The class of extensional properties can be shown to parallel the definition of well formed formulas in first-order axiomatic set theory. These formulas are built up from atomic formulas of the form $x = y$ and $x \in y$ by the ordinary connectives in quantifiers. It is possible to prove the following properties of Ext, which parallel the definition of well-formed formulas:

1. $(\forall A : \mathsf{Set}_U \to \mathsf{Prop})(\mathsf{Ext}U(\mathsf{SetQ}UA))$
2. $(\forall A : \mathsf{Set}_U \to \mathsf{Prop})(\mathsf{Ext}U(\lambda B : \mathsf{Set}_U \,.\, \mathsf{SetQ}UBA))$.
3. $(\forall A : \mathsf{Set}_U \to \mathsf{Prop})(\mathsf{Ext}U(\lambda x : U \,.\, Ax))$.
4. $(\forall A : \mathsf{Set}_U \to \mathsf{Prop})(\forall B : \mathsf{Set}_U \to \mathsf{Prop})(\mathsf{Ext}UA \supset \mathsf{Ext}UB \supset \mathsf{Ext}U(\lambda x : U \,.\, Ax \wedge Bx)$.
5. $(\forall A : \mathsf{Set}_U \to \mathsf{Prop})(\forall B : \mathsf{Set}_U \to \mathsf{Prop})(\mathsf{Ext}UA \supset \mathsf{Ext}UB \supset \mathsf{Ext}U(\lambda x : U \,.\, Ax \vee Bx)$.
6. $(\forall A : \mathsf{Set}_U \to \mathsf{Prop})(\forall B : \mathsf{Set}_U \to \mathsf{Prop})(\mathsf{Ext}UA \supset \mathsf{Ext}UB \supset \mathsf{Ext}U(\lambda x : U \,.\, Ax \supset Bx)$.
7. $(\forall A : \mathsf{Prop})(\forall B : A \to (\mathsf{Set}_U \to \mathsf{Prop})(\forall u : A)(\mathsf{Ext}U(Bu) \supset \mathsf{Ext}U(\lambda x : \mathsf{Set}_U \,.\, (\forall u : A)(Bux))$.
8. $(\forall A : \mathsf{Prop})(\forall B : A \to (\mathsf{Set}_U \to \mathsf{Prop})(\forall u : A)(\mathsf{Ext}U(Bu) \supset \mathsf{Ext}U(\lambda x : \mathsf{Set}_U \,.\, (\exists u : A)(Bux))$.

(Note that in the formula $x \in A$, the type of x is U, and $\lambda x : U \,.\, x \in A$ does not have the right type to satisfy Ext.)

As pointed out in [4, Remark 17], while we can prove most of the axioms of constructive set theory, we cannot prove the axioms of power set and $\in$-induction. Furthermore, we cannot expect to get all of set theory this way, because the consistency of this implementation follows from the strong normalization theorem for the calculus of constructions. If a complete set theory satisfying these axioms is desired, then the implementation given here will not do. But the need for a different (and, presumably, stronger implementation does not follow from any problem with the axiom of extensionality.

References

[1] H. P. Barendregt. Lambda calculi with types. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 117–309. Oxford University Press, 1992.

[2] Gérard Huet. Formal structures for computation and deduction. Course Notes, Carnegie-Mellon University, First Edition, May 1986.

[3] Gérard Huet. Induction principles formalized in the calculus of constructions. In Hartmut Ehrig, Robert Kowalski, Giorgio Levi, and Ugo Montanari, editors, *TAPSOFT '87: Proceedings of the International Joint Conference on Theory and Practice of Software Development, Pisa, Italy, March 23–27, 1987. Volume 1: Advanced Seminar on Foundations of Innovative Software Development I and Colloquium on Trees in Algebra and Programming (CAAP '87)*, volume 249 of *Lecture Notes in Computer Science*, pages 276–286, Berlin, 1987. Springer-Verlag.

[4] Jonathan P. Seldin. On the proof theory of Coquand's calculus of constructions. *Annals of Pure and Applied Logic*, 83:23–101, 1997.