Can we Resolve the Tension between Constructive Type Theorists and Classical Mathematicians?

Paul B. Jackson
pbj@dcs.ed.ac.uk

1 Introduction

Constructive type theories (CTTs) are advocated as a foundation for mathematics which replaces classical logic and set theory. Significant work has gone into building interactive theorem-proving systems based on CTTs [dB80, C+86, Jac95, AGNvS94, LP92, CCF+95] and it seems desirable to involve the projects currently around such systems in any future QED venture. However, mathematics based on CTTs is rather different from the usual classical mathematics taught in schools and universities. QED must support this classical mathematics if it is to have any success. How then is cooperation possible?

2 CTT-based mathematics

All CTT-based mathematics has a computational reading. For example, a theorem of form:

$$\forall x \in A . \exists y \in B . P_{x,y}$$

can be read as saying that given any x in set A, there is a method of *computing* a y in set B satisfying the predicate $P_{x,y}$. Further, a CTT proof of such a theorem precisely specifies one such method. In general, theorems in CTT-based mathematics can be considered as specifications for programs and proofs as guides for the automatic construction of programs. This program synthesis paradigm has been one of the major factors stimulating recent interest in CTTs.

CTT-based mathematics so far has been largely modelled on the school of constructive mathematics first developed by Bishop [BB85, MRR88]. A feature of the Bishop school that increases its acceptability to classical mathematicians is that every theorem also has a reading as a classical theorem. This is not the case with other schools of constructive mathematics.

CTT-based mathematics has a finer grain than classical mathematics. Many distinctions are made that offer alternative computational readings. These distinctions are at a very basic level: for example, different readings are frequently be given for equivalence and subtype relations. It is a challenge to decide which alternatives to adopt and to keep the number of alternatives considered to a reasonable size.

Formalization forces definite choices to be made on alternatives where in the texts the need for a decision is glossed over or delayed. It also frequently turns out that functions need extra arguments that provide computational information. A formal development must include these arguments, though they also are often glossed over in the texts.

Current CTTs are somewhat awkward. In nearly all, the notion of *type* is not nearly as versatile as that of *set* in set theories. For example, equality of types is usually not extensional and a principle of comprehension is usually lacking. Also, many CTTs are regarded as being too complex to be acceptable as foundational theories.

Examples of formalization of mathematics in CTTs include the intermediate value theorem and some basic abstract algebra [For93, Jac95].

For the above reasons, formalizing Bishop-style mathematics in CTTs seems to be a significantly slower and more uncertain process than formalizing classical mathematics.

3 Opportunities for Cooperation

3.1 Libraries

Sharing of libraries on elementary concrete topics such as integers and finite sequences (lists) might be possible since there CTT-based and classical mathematics are similar. However, sharing of libraries on more abstract topics would be much more problematic.

Importing of classical developments into a CTT setting would be all but impossible because all the essential distinctions would be missing.

Importing a CTT-based development into a classical setting is possible, though the classical mathematician is likely to consider all the extra distinctions as irrelevant clutter. More pragmatically, the need to import might not be there, since the quantity and sophistication of formalized classical mathematics is likely to be much greater than that of CTT-based mathematics, both for reasons given in the previous section and simply because the corpus of formalizable classical mathematics is far far larger.

3.2 Systems Development

Opportunities are brightest here. There are many engineering challenges common to any system intended for helping to develop mathematics. For example, in the areas of user interfaces, mathematical databases, and automated reasoning.

3.3 Classical Mathematicians using CTT-based Systems

It has been shown consistent to extend CTTs with oracles in order to create classical type theories [How91]. A CTT-based system with such an extended CTT could be used by a classical mathematician to develop classical mathematics, though work is still needed to demonstrate that the extended type theory would have a versatility approaching that of set theory.

3.4 Constructive Type Theorists using Classical Systems

It might be possible to persuade the architects of CTT-based systems to consider seeking in a classical system some of the advantages they attribute to CTTs.

For example, type theories are claimed to provide a more structured language than set theory for mathematics, closer to that used in normal mathematical practice. However the Mizar project [Rud92] has demonstrated how such advantages can be gained by layering a type system on top of a classical set theoretic foundation.

Another advantage claimed is that CTTs provide a natural environment for program verification and synthesis, since CTTs have a built-in programming language and a program synthesis paradigm. However, the built-in languages are rather simple purely-functional languages. CTTs have no advantages when it comes to dealing with more sophisticated languages with imperative, concurrent or parallel features. Analogous synthesis paradigms can be explored in a classical setting if for example programs are represented syntactically and logic variables are used to stand in for program sections that remain to be synthesized. With such an approach both the programming language and the synthesis mechanism would be more open to exploration since neither would be be hard wired.

4 Conclusions

- *No*, the fundamental tension between constructive type theorists and classical mathematicians is not resolvable: the mathematics that each are interested in is just too different.
- *Yes*, many aspects of the tension between constructive type theorists and classical mathematicians are resolvable: in particular, there do seem to be a number of opportunities for productive collaboration, especially in the engineering of interactive theorem-proving systems.

5 Discussion

Here I've reconstituted a few of the comments made at the end of the talk from some rather sketchy notes that I took down. Hopefully no one's views are misrepresented. The comments are being checked with their ascribed authors at the moment.

- (*Holmes*) "One of the major features of CTTs is the role of proof objects". In proof theory, these proof objects can be more compact and convenient to work with than proof trees.
- (*Kapur*) "Present day constructivist often cite 19th century mathematics as being in their tradition, but this mathematics also fits in perfectly well with classical mathematics". Kapur also emphasized the size of the CTT community and expressed the need to have them involved in any QED venture.
- (*Trybulec*) "Too much of the QED manifesto was devoted to constructive concerns. There *is* a koine on which nearly all mathematicians agree. The QED project should try to take the simplest possible approach. The problems are hard enough as it is".
- (*McCarthy*) "Hilbert complained about being driven out of Cantor's paradise. Well, set theories like ZF allow us to be driven out the minimal amount".
- (*Trybulec*) "A development of Heyting Algebras was carried out in the classical system Mizar". Trybulec showed this to a colleague who thought the development contained a new result.

References

[AGNvS94] Thorsten Altenkirch, Veronica Gaspes, Bengt Nordström, and Björn von Sydow. *A User's Guide to ALF.* Chalmers University of Technology, Sweden, May 1994. Available on the WWW `file://ftp.cs.chalmers.se/pub/users/alti/alf.ps.Z`.

[BB85] Errett Bishop and Douglas Bridges. *Constructive Analysis.* Springer-Verlag, 1985.

[C+86] Robert Constable et al. *Implementing Mathematics with The Nuprl Development System.* Prentice-Hall, NJ, 1986.

[CCF+95] C. Cornes, J. Courant, J. Filliatre, G. Huet, P. Manoury, C. Munoz, C. Murthy, C. Parent, C. Paulin-Mohring, A. Saibi, and B. Werner. The Coq proof assistant reference manual: Version 5.10. Technical Report 0177, INRIA, July 1995. Available from `ftp.inria.fr`.

[dB80] N. G. de Bruijn. A survey of the project AUTOMATH. In J. P. Seldin and J. R. Hindley, editors, *Essays in Combinatory Logic, Lambda Calculus, and Formalism*, pages 589–606. Academic Press, 1980.

[For93] Max B. Forester. Formalizing constructive real analysis. Technical Report TR93-1382, Computer Science Dept., Cornell University, Ithaca, NY, 1993.

[How91] Douglas J. Howe. On computational open-endedness in Martin-Löf's type theory. In *Proceedings of Sixth Symposium on Logic in Computer Science*, pages 162–172. IEEE Computer Society, 1991.

[Jac95] Paul B. Jackson. *Enhancing the Nuprl Proof Development System and Applying it to Computational Abstract Algebra.* PhD thesis, Cornell University, January 1995. Available as Cornell Computer Science Technical Report TR95-1509 from `http://cs-tr.cs.cornell.edu`.

[LP92] Zhaohui Luo and Robert Pollack. Lego proof development system: User's manual. Technical Report ECS-LFCS-92-211, LFCS, University of Edinburgh, May 1992. Available from `http://www.dcs.ed.ac.uk/lfcsreps/`.

[MRR88] Ray Mines, Fred Richman, and Wim Ruitenburg. *A Course in constructive Algebra.* Universitext. Springer-Verlag, 1988.

[Rud92] P. Rudnicki. An overview of the Mizar project. In *1992 Workshop on Types for Proofs and Programs*, Bastad, 1992. Chalmers University of Technology.