Review of: "Reification, Curry-Howard Correspondence, and Didactical Consequences"

Andrew Powell[1]

1 Imperial College London

Potential competing interests: No potential competing interests to declare.

This is a nice, well-written article that shows understanding of the literature. The idea is that mathematical objects are the reification of mathematical constructions, where the constructions can be thought of as proofs or as computer programs. This is a very reasonable idea, but personally, I would prefer to say that mathematics studies functions that can be constructed (or “witnessed”) by means of proofs or computer programs. I acknowledge that the view that mathematical objects are functions is not the orthodox view in philosophy, due mainly to the “meaning is use” school of philosophy, who think that if people talk about mathematical functions as objects, then they must be such.

The duality of proof and programs is known as the Curry-Howard Correspondence or Isomorphism. What it really says is that for intuitionistic formal systems at least, the typed lambda calculus with sufficiently rich types forms a functional programming language. The article does a good job of spelling out the correspondence. However, I would suggest a discussion of dependent types, because you cannot do quantification of predicates without them. The other difficult area in the typed lambda calculus is negation. In the intuitionistic typed lambda calculus, negation is proving that a type is empty, or otherwise that the proposition corresponding to the type implies the falsum or is absurd. Even in the intuitionistic case, there is an underlying reliance on a notion of proof. This is plausible, but negation is much messier if you introduce classical negation (usually via allowing double negation elimination), and the fact that classical negation corresponds to a global context switch or jump (and jump back) is equally as messy.

Despite these difficulties, the typed lambda calculus can be powerful enough to express proofs. An issue which the article does not address fully is the role of the infinite in mathematical constructions. The article assumes that it is sufficient to use error bars and symbolic manipulation, but of course many mathematical functions (or objects) are infinite, real numbers, for example. It is of course true that humans use error bars and symbolic manipulation too in proofs and computations, but the problem becomes apparent when you go beyond finite types (usually the natural functions, N, N→N, N→(N→N), (N→N)→N, etc.) to transfinite types. Reductions to a normal form are finite processes, and with transfinite types, the processes are in general infinite. There are ways around the infinities (which amount to carrying out infinitely many steps in one step), but the power of the reduction process is thereby limited.

Finally, types and the substitional reading of quantification are nicer than sets and infinite conjunctions or disjunctions of propositions from a teaching perspective, but you might consider whether types are sets in disguise, particularly since non-extensional types require a theory of intensionality to unpick.