Inhabiting Negative Types

Stéphane Lengrand
CNRS - École Polytechnique, France.
`lengrand@lix.polytechnique.fr`

20th June 2008

Abstract

In this talk I will challenge the view, which I among many have often put forward, that classical logic is symmetrical. The view is perhaps strengthened by the intuitions of boolean algebra, dualising as negation, De Morgan's laws, and is reflected in many proof theoretic formalisms or semantics.

Here we consider invertibility and η-conversion / equivalence of computational behaviours, in an approach *à la* Curry-Howard where formulae are types and proofs are objects with computational behaviours, inhabiting types. Indeed, the Law of Excluded Middle $A \vee A^{\perp}$ is far from being symmetrical : When analysing the computational behaviour of proofs, inhabitants of one formula (of *positive* polarity) *act*, while those of its negation / De Morgan dual (of *negative* polarity) *react*.

The canonical *action* is offering a value (with a potentially non-deterministic choice of value), but logical completeness requires erasure (weakening) or duplication (contraction) as proper actions as well.

On the other hand, inhabiting a negative type is a question that can have many answers. Standard presentations of classical logic use invertible introduction rules, making weakening and contraction on negative formulae redundant features. Noam Zeilberger has recently suggested to inhabit negative types with functions of the meta-level, mapping counter-proofs to proofs of absurdity. (Note that delegating *reactions* to the meta-level potentially integrates erasure or duplication of counter-proofs, according to the function space of the meta-level.) We connect this to the theory of *realisability* where inhabitants of negative types are terms that behave well when facing inhabitants of their duals, thus forming function spaces. At the heart of such a connection we find the questions of finiteness of function views (how deep can functions perform case analysis), computability, and decidability of type-checking.

We will then discuss how the polarity of atomic types fits in this, especially in the framework of second-order and higher-order logic.