Rules, types and the transcendence of second order logic

Paolo Pistone

As soon as formulas of the form "exists X A" or "Nat->... " are involved, incompleteness tells us that logical consequence transcends provability within a given formal system. This is due to the comprehension rule (i.e. the existence introduction rule) which violates the subformula property and whose treatment, if we follow Quine's well known account, demands for reference to a set-theoretic universe.

The aim of this talk is to argue for a different understanding of second order logic: building on some ideas by Prawitz, Martin-Löf and Girard, it is argued that a better grasp of the functional content of the comprehension rule comes from the consideration of inference rules independently of logical correctness; the situation is analogous to that of computation, whose proper functional description imposes to consider non terminating algorithms. A general equational theory of (possibly incorrect) inference is thus presented.

Suppose that a lambda term t represents a certain recursive function f, how can we know if the function f is actually a total function? Since it is impossible to verify that the function is defined for every argument, as this would require to perform infinite computations, one is forced to look for a (finite) argument, i.e. a proof. From the standpoint of the Curry-Howard correspondence between proofs and typed lambda terms, this amounts to trying to type t as Nat->Nat: in other words, logical typing replaces infinite verifications with a finite operation. The goal will be to characterize this finite (i.e. recursive) aspect of typing.

It is well known that the typing conditions for a lambda-term can be expressed, in all generality, through a system of recursive type equations. It will be shown that the types, if any, which solve such systems can be characterized by a set of introduction/elimination rules; that is to say, one no more focuses on what one can prove by means of a certain package of rules (provided by a formal system), but rather on what the rules needed to prove a certain formula must be like, at the level of their functional description. In this context, we can interpret unrestricted comprehension roughly as stating that every set of introduction/elimination rules (satisfying basic constraints) defines a logical type, or that any type equation system has a solution. How is it possible, then, to discern the "good" types from the paradoxical ones which will naturally arise in such a "naïve" type system?

Matters as to the legitimacy of comprehension instances are translated into the question whether logical harmony (i.e. cut-elimination) obtains between

a set of rules. By the way, the justification of logical rules by harmony runs inexorably into a form of epistemic ("pragmatic", in Dummett's terminology) circularity: instances of the comprehension axiom of set-theory must indeed be used to justify the harmony of the comprehension rule. Even worse, since, by applying the comprehension rule, every proof can be trivially transformed into a cut-free one, the entire matter of justifying comprehension by harmony looks like an empty shell. That's why insisting on the functional content of type inference seems more promising than focusing on the endless purpose of justifying rules (by set-theory, at last): accordingly, the notion of proof arising from the ideas above embodies the problematic (i.e., indefinitely questionable) condition of second order reasoning. It is conjectured that, by admitting unrestricted comprehension over types, the thesis that logical consequence transcends (problematic) provability can be challenged: for instance, can such a "naïve" type system, though globally inconsistent, contain enough consistent subsystems to type all total recursive functions?