Back to the Future: Explicit Logic for Computer Science

Sergei Artemov*

We will speak about three traditions in Logic:

• *Classical*, usually associated with Frege, Hilbert, Gödel, Tarski, and others;

• *Intuitionistic*, founded by Brouwer, Heyting, Kolmogorov, Gödel, Kleene, and others;

• *Explicit*, which we trace back to Skolem, Curry, Gödel, Church, and others.

The classical tradition in logic based on quantifiers $\forall$ and $\exists$ essentially reflected the 19th century mathematician's way of representing dependencies between entities. A sentence $\forall x \exists y A(x, y)$, though specifying a certain relation between x and y, did not mean that the latter is a function of the former, let alone a computable one. The Intuitionistic approach provided a principal shift toward the effective functional reading of the mathematician's quantifiers. A new, nonTarskian semantics had been suggested by Kleene: realizability that revealed a computational content of logical derivations. In a decent intuitionstic system, a proof of $\forall x \exists y A(x, y)$ yields a program f that computes $y = f(x)$.

Explicit tradition makes the ultimate step by using representative systems of functions instead of quantifiers from the very beginning. Since the work of Skolem, 1920, it has been known that the classical logic can be adequately recast in this way. Church in 1936 showed that even the very basic system of function definition and function application is capable of emulating any computable procedure. However, despite this impressive start, the explicit tradition remained a Cinderella of the mathematical logic for decades. Now things have changed: due to its very explicitness, this third tradition became the one most closely connected with Computer Science.

In this talk we will show how switching from quantifiers to explicit functional language helps problem solving in both theoretical logic and its applications. A discovery of a natural system of self-referential proof terms, *proof polynomials*, was essential in the solution to an open problem of Gödel concerning formalization of provability. Proof polynomials considerably extend the Curry-Howard isomorphism and lead to a joint calculus of propositions and proofs which unifies several previously unrelated areas. It changes our conception of the appropriate syntax and semantics for reasoning about knowledge, functional programming languages, formalized deduction and verification.

*The Graduate Center of the City University of New York, 365 Fifth Avenue, New York, NY, 10016, U.S.A. sartemov@gc.cuny.edu; Moscow University