JFP **13** (2): 257–260, March 2003.
DOI: 10.1017/S0956796802004549 Printed in the United Kingdom

Special issue on 'Logical frameworks and metalanguages'

GÉRARD HUET
INRIA-Rocquencourt, France

There is both a great unity and a great diversity in presentations of logic. The diversity is staggering indeed – propositional logic, first-order logic, higher-order logic belong to one classification; linear logic, intuitionistic logic, classical logic, modal and temporal logics belong to another one. Logical deduction may be presented as a Hilbert style of combinators, as a natural deduction system, as sequent calculus, as proof nets of one variety or other, etc. Logic, originally a field of philosophy, turned into algebra with Boole, and more generally into meta-mathematics with Frege and Heyting. Professional logicians such as Gödel and later Tarski studied mathematical models, consistency and completeness, computability and complexity issues, set theory and foundations, etc. Logic became a very technical area of mathematical research in the last half century, with fine-grained analysis of expressiveness of subtheories of arithmetic or set theory, detailed analysis of well-foundedness through ordinal notations, logical complexity, etc. Meanwhile, computer modelling developed a need for concrete uses of logic, first for the design of computer circuits, then more widely for increasing the reliability of sofware through the use of formal specifications and proofs of correctness of computer programs. This gave rise to more exotic logics, such as dynamic logic, Hoare-style logic of axiomatic semantics, logics of partial values (such as Scott's denotational semantics and Plotkin's domain theory) or of partial terms (such as Feferman's free logic), etc. The first actual attempts at mechanisation of logical reasoning through the resolution principle (automated theorem proving) had been disappointing, but their shortcomings gave rise to a considerable body of research, developing detailed knowledge about equational reasoning through canonical simplification (rewriting theory) and proofs by induction (following Boyer and Moore successful integration of primitive recursive arithmetic within the LISP programming language). The special case of Horn clauses gave rise to a new paradigm of non-deterministic programming, called Logic Programming, developing later into Constraint Programming, blurring further the scope of logic. In order to study knowledge acquisition, researchers in artificial intelligence and computational linguistics studied exotic versions of modal logics such as Montague intentional logic, epistemic logic, dynamic logic or hybrid logic. Some others tried to capture common sense, and modeled the revision of beliefs with so-called non-monotonic logics. For the careful crafstmen of mathematical logic, this was the final outrage, and Girard gave his anathema to such "montres à moutardes".

Behind this flurry of varieties of logic, however, lies a common heritage of methods and structures. Furthermore, the problematics of the formalisation of mathematical

objects, and most notably constructive mathematics, gave a unifying point of view in which the structures of logical reasoning (i.e. proof theory, pioneered by Gentzen and developed by Kleene, Kreisel, and more recently, Girard) were naturally integrated within some version of Church's typed λ-calculus. Dependent product, first studied by Howard, gave the final link to the correspondance between typed computational systems and proof structures, now known as the Curry-Howard isomorphism, yielding one of the most productive paradigms of theoretical informatics. A pioneer of this effort was N. de Bruijn, who worked on a family of formalisms, called Automath, back in the 1960s. The underlying idea is to exploit this Curry–Howard correspondance, which models logical propositions as types, and formal proofs as well-typed terms in some computational algebra. For instance, Church's simply typed λ-calculus is isomorphic to natural deduction structures for minimal propositional logic. Later in the 1970s, Per Martin-Löf started a program of design of constructive mathematics along similar lines, and a research group from Göteborg's University followed his ideas to advocate the use of Type Theory as a Programming Methodology paradigm. In the USA, Bob Constable at Cornell University set to adapt this framework to various logics of programming in the systems PRL and NuPRL, influenced by the LCF effort from Robin Milner's team in University of Edinburgh. The convergence of these various trends was apparent in the beginning of the 1980s, and a crystallisation of these ideas occurred at the Conference on Types organised in Sophia-Antipolis by Gilles Kahn in 1984. I presented myself at this meeting a synthesis of Automath with the higher-order λ-calculi F and $F\omega$ of J. Y. Girard within a Calculus of Constructions, a joint work with Thierry Coquand which was finalised and analysed in his thesis in 1985.

Now a coherent methodology was taking place for using typed λ-calculus as the common vehicle for meta-mathematical representations. Details of basic sorts and closure operations were still the object of heated debate (with esoteric discussions on mysteries surrounding elusive notions such as predicativity), and there were minor differences of treatment of framework equality (such as postulating or not extensionality principes such as η-conversion). However, such disagreements were really secondary issues compared to the methodological progress achieved by this type-theoretic unification, allowing the factorisation of the justifications of meta-theoretic properties, as well as the sharing of a lot of the implementation burden. A central issue remained the strength of the meta-theoretic framework; while N. de Bruijn advocated the use of weak formalisms, with low complexity of decision procedures, others claimed that stronger frameworks were useful, because they had a superior abstraction power, and consequently were amenable to lighter space requirements and could exhibit stronger modularity constructs. A typical weak formalism, close to the core of Automath, was proposed as the Edinburgh Logical Framework by Harper, Honsell and Plotkin at *LICS'87*. By then the Logical Framework methodology was recognised as such and the first computer implementations of generic proof systems were soon to appear, such as Isabelle, designed at Cambridge University by Larry Paulson. Such a proof assistant could be parameterised by an axiomatic description of the intended specific logic, while all the formula and context management procedures used generic search, matching and unification modules.

The year 1989 was a turning point for the constitution of a research community around this paradigm, grouping proof theorists from mathematical logic with computer scientists both from the denotational semantics and λ-calculus traditions, and from more applied backgrounds such as interactive structure editors. A dozen of European research teams joined their expertise to collaborate in a Basic Research Action funded within the European Esprit program, called "Logical Frameworks" in the period 1989–1992. Significant conceptual progress was achieved, such as the notion of Pure Type Systems, which allowed Henk Barendregt and Stefano Berardi to abstract from peculiarities of Automath, Edinburgh LF, System F, Calculus of Constructions and Martin Löf's Constructive Type Theory into a Cube making explicit the various instances of a generic notion of Pure Type System. In the same way that pure λ-calculus had been the vehicular language for denotational semantics, with non-applicative control structures modeled through higher-order continuations and monads, typed λ-calculus became the vehicular language for proof theory, even for apparently non-constructive frameworks such as classical logic, as Michel Parigot showed with his μ-calculus theory. On the practical side, numerous proof assistants were designed, implemented, and mutually cross-fertilised, such as LEGO at University of Edinburgh, Alf at Chalmers University in Göteborg, and Coq at INRIA in Rocquencourt. Isabelle became a European endeavour through the involvement of Tobias Nipkow from TUM in Munich. Yves Bertot at INRIA in Sophia-Antipolis adapted the Centaur structure editor to become a proof editor for Coq in the CtCoq system, and experimented with sophisticated interaction paradigms such as proof by pointing. In the USA, a notable contribution to this area was the ELF system designed by Frank Pfenning at Carnegie-Mellon University, as an adaptation of Dale Miller's λ-PROLOG to Edinburgh LF. Two books, *Logical Frameworks* and *Logical Environments*, which I co-edited with Gordon Plotkin at Cambridge University Press, published important contributions from this joint effort. The LF European project was renewed in a second stage Basic Research Action called "Types" with more teams joining in the period 1992–1995. This effort is still active as a Research Network. I take this opportunity to thank the European Commission Research Funding Agency for its continuous support of our research.

By 1995 the field had matured into a well-understood area generating its own research problematics. Linear logic proved to be a fundamental tool to investigate geometrical symetries in proof nets through the Geometry of Interaction, and J. Y. Girard developed new models such as Coherent Spaces. A fine-grained analysis of substitution yielded a new theory of explicit substitutions. Researchers moved to investigate definitional congruences significantly different from simple β-conversion from λ-calculus, notably with primitive inductive types, which cross-fertilised with term-rewriting research. Proof assistants started to be applied to significant industrial applications, such as certification of security in smartcard architectures. In the *International Conference on Logic in Computer Science* (LICS) at Santa Barbara in June 2000, Joëlle Despeyroux and Robert Harper organised a Workshop on Logical Frameworks and Meta-languages, which brought together researchers in this area. The current special issue of this journal is a followup to this workshop,

and it presents five representative papers giving the state of the art on several of its aspects.

Gilles Barthe, Venanzio Capretta and Olivier Pons investigate in "Setoids in type theory" the vexing problem of implementing set theory within type theory, which lacks general quotients. They consider various possible axiomatisations of types presented with a congruence (setoids), and study their interaction with various versions of choice principles, from the points of view of expressiveness and consistency.

A recurring theme of logical frameworks is the formalisation of binding constructs. The implicit scoping modification induced by binding operations prevents them from accommodation into the usual apparatus within free algebras (abstract syntax). Researchers such as Joëlle Deypeyroux have proposed notions of higher-order abstract syntax for their representation. This raises questions of possible over-generative power, and the proper status of inductive reasoning in the presence of such constructions. Christine Röckl et Daniel Hirschkoff tackle these questions in their paper "A Fully Adequate Shallow Embedding of the π-Calculus in Isabelle/HOL with Mechanized Syntax Analysis". They describe a mechanisable proposal for higher order abstract syntax on a significant case study for π-calculus, an important distributed process calculus.

Zhaohui Luo presents in "PAL^+: a lambda-free logical framework" a proposal for a weak framework in the spirit of de Bruijn's PAL, the root formalism of the Automath family. In PAL one may define combinators, but there is no general λ-calculus operator like in the richer Automath languages. Luo gives a full meta-theoretic analysis of PAL^+, and he concludes that it provides an attractive alternative to stronger logical frameworks.

Michael Levin and Benjamin Pierce present "TinkerType: A Language for Playing with Formal Systems". In TinkerType, a formal system is presented by the interaction of clauses with features. This contribution presents the concepts of TinkerType, its implementation, and give an extensive presentation of axiomatisations in this framework of various type theories presented in the literature, with a number of features such as subtypes and computation rules.

Daria Walukiewicz-Chrzaszcz presents in "Termination of rewriting in the Calculus of Constructions" an extension of the Calculus of Constructions with rewrite rules, preserving the termination of computations in the presence of rewriting steps. It is a remarkable application to type theory of sophisticated concepts from term rewriting theory, which opens the possibility of general computation within a proof system.

These five papers are representative of recent trends in logical frameworks research, showing that this paradigm is still at the center of important developments.

https://doi.org/10.1017/S0956796802004549 Published online by Cambridge University Press