Current Trends in Logical Frameworks and Metalanguages

David Basin
Universität Freiburg, Institut für Informatik, Freiburg Germany

Amy Felty
School of Information Technology and Engineering, University of Ottawa, Canada

1. Introduction

Logical frameworks and metalanguages are intended as a common substrate for representing and implementing a wide variety of logics and formal systems. Their definition and implementation has been the focus of considerable work over the last couple decades. At the heart of this work is a quest for generality: A logical framework provides a basis for capturing uniformities across deductive systems and support for implementing and reasoning about particular systems. Similarly a metalanguage supports formalizing and using languages.

The workshop on Logical Frameworks and Metalanguages (LFM'99) was held as part of the colloquium on Principles, Logics, and Implementations of High-Level Programming Languages (PLI'99) in Paris in September 1999. This workshop brought together designers, implementers, and practitioners to discuss the development of logical frameworks and their use in metareasoning and programming.

This special issue of the Journal of Automated Reasoning contains three papers on the above topic, solicited in an open call-for-papers after the workshop.

2. The Articles

In the first article"An Implementation of LF with Coercive Subtyping and Universes", Paul Callaghan and Zhaohui Luo present an implementation of the logical framework "Plastic". This framework is based on Luo's metalogic, the Typed LF, which is a typed version of Martin-Löf's logical framework, for which type checking is decidable. The authors present an extension of the Typed LF with inductive types, Tarski-style type universes, and type coercions and they also present machine support for, and applications of, these extensions. Overall, the result is a logical framework that is both semantically clear and practical to use.

The second article focuses on a particular representation problem and a general solution for it. Contexts are expressions containing holes in which other expressions can be inserted; they are often used, for example, in modeling programs and program environments, operational semantics, and in natural language representations. In "A Calculus of Lambda Calculus Contexts", Mirna Bognar and Roel de Vrijer examine the problem of providing a metalanguage for representing contexts and provide a calculus λc for this purpose. Their calculus is an extension of the lambda calculus with facilities for representing contexts and context-related operations such as filling the holes of a context by expressions or by contexts, and formalizing the interaction between a context and the expressions that fill its holes. The resulting formalism is sufficiently expressive to formalize the various different uses of contexts, has an elegant metatheory, and is ultimately translatable to the standard lambda calculus.

The third article, "Studies of a Theory of Specifications with Built-in Program Extraction" by Paula Severi and Nora Szasz, examines the problem of separating programs from their correctness proofs in constructive type theories. Based on Martin-Löf's Type Theory, they present a particular type theory called the Theory of Specifications. In this theory, as in Martin-Löf's, proof rules simultaneously construct programs with their correctness proofs. For example, a proof of a formula $\forall x : A. \exists y : B. P(x, y)$ builds an implementation f where for any $x \in A$, $f(x)$ evaluates to a pair whose first projection is an element of B such that $P(x, f(x))$ holds, and the second projection is a proof of this fact. The use of a single function f mixes the computation of these two parts of the pair. The authors' contribution is to show that, for their theory, there is a confluent and normalizing computational relation that performs the task of separating programs from proofs. As a consequence, every implementation computes to a pair composed of a program and a proof of its correctness, and so the program extraction procedure is immediate at that point.