Automated Theorem Proving in a Simple Meta-Logic for LF

Carsten Schürmann and Frank Pfenning *

Carnegie Mellon University
School of Computer Science
carsten@cs.cmu.edu, fp@cs.cmu.edu

Abstract. Higher-order representation techniques allow elegant encodings of logics and programming languages in the logical framework LF, but unfortunately they are fundamentally incompatible with induction principles needed to reason about them. In this paper we develop a meta-logic \( \mathcal{M}_2 \) which allows inductive reasoning over LF encodings, and describe its implementation in Twelf, a special-purpose automated theorem prover for properties of logics and programming languages. We have used Twelf to automatically prove a number of non-trivial theorems, including type preservation for Mini-ML and the deduction theorem for intuitionistic propositional logic.

1 Introduction

The logical framework LF [HHP93] has been designed as a meta-language for representing deductive systems which are common in the study of logics and programming languages. It allows concise encodings of many common inference systems, such as natural deduction and sequent calculi, type systems, operational semantics, compilers, abstract machines, etc. (see [Pfe96] for a survey). These representations often lead directly to implementations, either via the constraint logic programming paradigm [Pfe94] or via general search using tactics and tacticals.

The logical framework derives its expressive power from the use of dependent types together with "higher-order" representation techniques which directly support common concepts in deductive systems, such as variable binding and capture-avoiding substitution, parametric and hypothetical judgments, and substitution properties. The fact that these notions are an integral part of the logical framework would seem to make it an ideal candidate not only for reasoning within various inference systems, but for reasoning about properties of such systems.

Unfortunately, higher-order representation techniques are fundamentally incompatible with the induction principles needed to reason about such encodings (see [DPS97] for a detailed analysis). In the literature three approaches have been...
studied in order to overcome these problems, while retaining the advantages a logical framework can offer. The first called schema-checking [Roh94,RP96] implements meta-theoretic proofs as relations whose operational reading as logic programs realizes the informal proofs. This has been applied successfully in many case studies (see [Pfe96]), but lacks automation. The second is based on reflection via a modal provability operator. At present it is unclear how this idea, developed for simple types in [DPS97], interacts with dependent types, and if it is flexible enough for many of the theorems that can be treated with schema-checking. The third is to devise an explicit (meta-)meta-logic for reasoning about logical framework encodings. For the simpler logical framework of hereditary Harrop formulas this approach has been followed by McDowell and Miller [MM97,McD97] (see Section 5 for a detailed comparison).

In this paper we follow the third approach and develop a simple meta-logic $M_2$ for LF and sketch its implementation in the Twelf system. $M_2$ was designed explicitly to support automated inductive theorem proving and has been applied successfully to prove, for example, value soundness and type preservation for Mini-ML, completeness of a continuation stack machine with respect to a natural semantics for Mini-ML, soundness and completeness of uniform derivations with respect to resolution (which is a critical step in the correctness of compilers for logic programming languages), the deduction theorem for intuitionistic propositional logic using Hilbert's axiomatization, and the existence of an embedding of Cartesian closed categories into the simply-typed $\lambda$-calculus. In each case we specified only the theorem and the induction variable, the proof was completely automatic in every other respect.

We view Twelf as a special-purpose automated theorem prover for the theory of programming languages and logics. It owes its success to the expressive power of the logical framework combined with the simplicity of the meta-logic which nonetheless allows direct expression of informal mathematical arguments. Its main current limitations are the lack of facilities for incorporating lemmas and for proving properties which require reasoning about open LF objects, i.e., objects which may contain free variables. We plan to address the former by adapting standard techniques from inductive and resolution theorem proving and the latter by borrowing successful ideas from schema-checking.

This paper is organized as follows: In Section 2 we briefly describe the logical framework LF and introduce a programming language Mini-ML and a type preservation result as running example. The meta-logic $M_2$ is introduced in Section 3 which is implemented in the Twelf system which we discuss in Section 4. Section 5 compares the most closely related work before we assess the results and discuss future work.

2 The Logical Framework LF

The type theory underlying the logical framework LF is an extension of the simply-typed $\lambda$-calculus by dependent types. It is defined by three syntactic categories of objects, type families, and kinds [HHP93]. We use $\alpha$ for type family