Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 323 (2016) 125–142

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

# Canonical HybridLF: Extending Hybrid with Dependent Types

Roy L. Crole [1] Amy Furniss [2]

*Dept of Computer Science, University of Leicester, University Road, Leicester, LE1 7RH, U.K.*

**Abstract**

We introduce Canonical HybridLF (CHLF), a metalogic for proving properties of deductive systems, implemented in Isabelle HOL. CHLF is closely related to two other metalogics. The first is the Edinburgh Logical Framework (LF) by Harper, Honsell and Plotkin. The second is the Hybrid system developed by Ambler, Crole and Momigliano which provides a Higher-Order Abstract Syntax (HOAS) based on un-typed lambda calculus.
Historically there are two problems with HOAS: its incompatibility with inductive types and the presence of exotic terms. Hybrid provides a partial solution to these problems whereby HOAS functions that include bound variables in the metalogic are automatically converted to a machine-friendly de Bruijn representation hidden from the user.
The key innovation of CHLF is the replacement of the un-typed lambda calculus with a dependently-typed lambda calculus in the style of LF. CHLF allows signatures containing constants representing the judgements and syntax of an object logic, together with proofs of metatheorems about its judgements, to be entered using a HOAS interface. Proofs that metatheorems defined in the signature are valid are created using the M2 metalogic of Schurmann and Pfenning.
We make a number of advances over existing versions of Hybrid: we now have the utility of dependent types; the unitary bound variable capability of Hybrid is now potentially finitary; a type system performs the role of Hybrid well-formedness predicates; and the old method of indicating errors using special elements of core datatypes is replaced with a more streamlined one that uses the Isabelle option type.

*Keywords:* dependent types, HOAS, logical frameworks, metalogical reasoning, variable binding

## 1 Introduction

This paper is about reasoning about deductive systems such as logics, programming languages and so on. In general we refer to a typical deductive system as an *object logic*. One may reason about an object logic by translating it into a *metalogic* and then performing reasoning in the metalogic, provided that properties of the object logic are suitably reflected in the metalogic.

In particular this paper is concerned with Higher Order Abstract Syntax. This is a well-known metalogical technique that can be applied to object logics that have

[1] Email: rlc3@le.ac.uk

[2] Email: mjf29@le.ac.uk

http://dx.doi.org/10.1016/j.entcs.2016.06.009

variable binding: variable binders of the HOAS metalogic are used to implement the variable binders of an object logic.

The first author, Crole, along with Ambler and Momigliano, developed the HYBRID system [1]. This is an implementation of Higher Order Abstract Syntax within an Isabelle HOL package. The key novel feature is that a user may write down *higher order* abstract syntax using "user friendly" named bound variables and then the package *converts* such syntax to a "machine friendly" *first order* de Bruijn notation for its "internal reasoning". The use of HOAS is not without its problems. One issue is that HOAS is typically not compatible with induction. However HYBRID provides a partial solution and provides a user with a form of HOAS which embeds consistently within (Isabelle) HOL and hence with principles of induction [1].

The HYBRID system is underpinned by the untyped $\lambda$-calculus. While HYBRID has been successfully used by the authors and other researchers (see for example [12,9]) we thought it interesting to consider whether one could develop a typed version. This is not simply intellectual curiosity. When using HYBRID one typically has to implement well-formedness predicates which are deployed within inductive definitions. However within a typed system these predicates might be rendered redundant, instead using types to build well-formed expressions. This is both interesting and a potential important simplification and advantage for the user.

At the heart of HYBRID are *conversion* functions mapping untyped higher order syntax to first order syntax. *Our contribution is to show that the conversion technique extends not just to a simply typed setting, but in fact to a new system,* CANONICAL HYBRIDLF, *that is dependently typed. We can indeed dispense with well-formedness predicates. Further, the new system provides for Twelf-style reasoning with the judgements-as-types methodology.*

First we developed an extension of HYBRID based on the simply typed $\lambda$-calculus. Having developed suitable conversion functions which were faithful to the conceptual ideas underpinning HYBRID, we began to consider dependent types. We decided to explore using Edinburgh LF [10] as a basis since it is a system for meta-reasoning and so too is HYBRID. However, whereas HYBRID (and the simply typed version) provide a $\lambda$-calculus HOAS interface for encoding an object system, after which a user reasons directly within (Isabelle) higher order logic, basing a version of HYBRID on LF could mean the possibility of reasoning using judgements-as-types *within a general purpose tactical theorem prover.*

In fact Canonical LF underpins CANONICAL HYBRIDLF (see Section 3.1 for further details). Both LF and Canonical LF have pros and cons. Key factors behind the choice of Canonical LF rather than LF as the basis for our system are the simplicity of equality in Canonical LF and technical issues regarding termination of the unification algorithm that we employ for LF. This, along with type decidability, will be discussed in future work, but here we concentrate on CANONICAL HYBRIDLF.

There is more about the theory of HYBRID in [5]. A version of HYBRID was created by Martin [12] and Amy Felty. Versions of HYBRID developed in Coq appear in [3] and [9] with work by Capretta, Felty and Habli. Our methodology of reduction of higher to first order syntax is exploited in [15].