Object-level reasoning with logics encoded in HOL Light

Petros Papapanagiotou
School of Informatics, University of Edinburgh
ppapapan@inf.ed.ac.uk

Jacques Fleuriot
School of Informatics, University of Edinburgh
jdf@inf.ed.ac.uk

Higher order logic (HOL) proof assistants, such as HOL Light [4], provide the means to encode other logics and prove conjectures specified within them (*object-level reasoning*). This can help understand how the objects of an encoded logic behave or even develop practical applications using that logic.

In practice, performing proofs within an encoding of a custom logic in HOL typically requires the development of specialised tools and tactics for fine-grained structural manipulation of the encoded formulae, customised management of the inference rules for backward and forward chaining, etc. Users often need to develop such tools in an ad-hoc way in order to reason specifically about the logic they are interested in, and this drastically increases the effort and time required to obtain a useful encoding. Moreover, such tools may not scale across other logics, leading to a replication of effort.

We present a generic framework and toolset for object-level reasoning with custom logics encoded in a sequent calculus style in HOL Light. It aims to facilitate the exploration of different logical theories by minimizing the effort required between encoding the logic and obtaining an object-level proof.

There are strong similarities between our system and Isabelle [10, 11], which exposes an intuitionistic meta-logic (*Isabelle/Pure*) that enables the embedding of different theories, and the development of proof tactics and procedures for both interactive and automated theorem proving within those theories.

A key feature which sets our framework apart is the handling of type theoretical correspondences between embedded logics and processes. These are inspired by the *propositions-as-types* paradigm (or *Curry-Howard correspondence*) [6] to enable the construction of executable terms in some calculus via proof, which we will refer to as *computational construction*. Terms are attached to each logical formula, so that the formula represents the type of the term. The application of inference rules on such annotated terms results in the construction of a process with some guaranteed correctness properties.

Our work is motivated by the variety of correspondences between linear logic [3] and distributed processes, including *proofs-as-processes* [1], session types [2], dependent session types [12], *propositions-as-sessions* [13], and πULL [5]. Our framework can work jointly with the meta-theoretic efforts in these strands of work, providing a formal setting to produce object-level proofs and generate process instances.

We present an example from the well-known propositions-as-types paradigm. Note that a similar example from one of the linear logic correspondences would be just as easily managed by our system (see below), but would require setting up enough context, which is not possible here due to space restrictions. We consider a subset of propositional logic involving only conjunction ($\times$) and implication ($\rightarrow$) and its correspondence to a simply typed λ-calculus (inspired by Wadler's discussion of the same topic [14]). The proof of commutativity of $\times$ in this setting is as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{fst(Var\ x):X \vdash fst(Var\ x):X}\,Ax}{Var\ x:X\times Y \vdash fst(Var\ x):X}\,L1\times \qquad \dfrac{\dfrac{}{snd(Var\ x):Y \vdash snd(Var\ x):Y}\,Ax}{Var\ x:X\times Y \vdash snd(Var\ x):Y}\,L2\times}{Var\ x:X\times Y,\ Var\ x:X\times Y \vdash (snd(Var\ x), fst(Var\ x)):Y\times X}\,R\times}{Var\ x:X\times Y \vdash (snd(Var\ x), fst(Var\ x)):Y\times X}\,C}{\vdash \lambda x.(snd(Var\ x), fst(Var\ x)):X\times Y \rightarrow Y\times X}\,R\rightarrow \tag{1}$$

Submitted to:
LFMTP 2020

This can be viewed as a *type checking* proof that the type of $\lambda x.(snd(Var\ x), fst(Var\ x))$ is $X \times Y \to Y \times X$. Alternatively, assuming an original goal of the form $\exists f.\ \vdash f{:}X \times Y \to Y \times X$, this can be viewed as a *construction* proof of a term f with type $X \times Y \to Y \times X$, resulting in $f = \lambda x.(snd(Var\ x), fst(Var\ x))$. Ignoring the λ-calculus annotations this can also be viewed as a simple *logical* proof.

Assuming an encoding of sequents as a function of logical consequence ($\vdash$) over a multiset of formulae, performing even such a simple object-level proof with an embedded logic in HOL Light requires tedious manipulation of the multisets in the inference rules and proof goals using rewriting or appropriate instantiations. In addition, the construction of the computational component f also requires careful, gradual, delayed instantiation of metavariables that the user would normally keep track of manually. Our framework automates the involved structural and metavariable reasoning, facilitating the mechanization of proof (1) with direct rule applications as shown in the following script:

Step	Tactic	Goal(s)
		$\varnothing \vdash X \times Y \to Y \times X$
1	`ruleseq` $(R\to)$	$\{X \times Y\} \vdash Y \times X$
2	`ruleseq` (C)	$\{X \times Y\} \uplus \{X \times Y\} \vdash Y \times X$
3	`ruleseq` $(R\times)$	$\{X \times Y\} \vdash Y \qquad \{X \times Y\} \vdash X$
4	`ruleseq` $(L2\times)$	$\{Y\} \vdash Y \qquad \{X \times Y\} \vdash X$
5	`ruleseq` (Ax)	$\{X \times Y\} \vdash X$
6	`ruleseq` $(L1\times)$	$\{X\} \vdash X$
7	`ruleseq` (Ax)	

The *same* script can perform this proof, whether it is seen as a type checking, construction, or a purely logical proof. This allows users to easily construct computational terms and test and compare different logics and the behaviour of their correspondences (or even multiple correspondences of the same logic at the same time), merely by manipulating the encoding of the inference rules rather than having to rebuild or repurpose the implemented tools and proofs each time. All of this is made possible through the following key features:

1. Adapted procedural tactics for forward and backward reasoning inspired by Isabelle and extended from the *Isabelle Light library* [7].
2. Smart matching of sequents using multiset matching and metavariable unification.
3. Augmented proof system to allow extensions of the proof state with metadata, for instance about metavariables.
4. Systematic management of metavariables in the extended proof state for each subgoal.
5. Special functions that facilitate the extraction of constructed components.

The implemented generic tactics integrate fully within the (extended) tactic system of HOL Light, and can be used interactively, in packaged proofs, or programmatically in proof automation. The full implementation can be found online[1]. It includes the example of this paper, as well as 2 example embeddings of linear logic: Intuitionistic Linear Logic with no correspondence and the *propositions-as-sessions* paradigm, i.e. a correspondence of Classical Linear Logic to CP [13].

We believe our framework can be a powerful tool for research in type theory and process calculi, in addition to its already successful application to process specification and composition within healthcare [8, 9]. We are planning to discuss other aspects of this work, such as key challenges and solutions, technical details, the use of HOL Light, and a comparison with other systems in a forthcoming publication.

[1] `https://github.com/PetrosPapapa/hol-light-embed` and `hol-light-tools`

References

[1] G. Bellin & PJ Scott (1994): *On the π-calculus and linear logic*. Theoretical Computer Science 135(1), pp. 11–65.

[2] Luís Caires & Frank Pfenning (2010): *Session Types as Intuitionistic Linear Propositions*. In Paul Gastin & Francois Laroussinie, editors: CONCUR 2010 - Concurrency Theory, Lecture Notes in Computer Science 6269, Springer Berlin Heidelberg, pp. 222–236, doi:10.1007/978-3-642-15375-4_16.

[3] Jean-Yves Girard (1995): *Linear Logic: its syntax and semantics*. In Jean-Yves Girard, Yves Lafont & Laurent Regnier, editors: Advances in Linear Logic, London Mathematical Society Lecture Notes Series 222, Cambridge University Press.

[4] J. Harrison (1996): *HOL Light: A tutorial introduction*. Lecture Notes in Computer Science, pp. 265–269.

[5] Bas van den Heuvel & Jorge A. Prez (2020): *Session Type Systems based on Linear Logic: Classical versus Intuitionistic*. Electronic Proceedings in Theoretical Computer Science 314, p. 111, doi:10.4204/eptcs.314.1. Available at `http://dx.doi.org/10.4204/EPTCS.314.1`.

[6] William A. Howard (1980): *The formulas-as-types notion of construction*. In J. P. Seldin & J. R. Hindley, editors: To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism, Academic Press, pp. 479–490. Reprint of 1969 article.

[7] Petros Papapanagiotou & Jacques Fleuriot (2010): *An Isabelle-Like Procedural Mode for HOL Light*. In ChristianG. Fermüller & Andrei Voronkov, editors: Logic for Programming, Artificial Intelligence, and Reasoning, Lecture Notes in Computer Science 6397, Springer, pp. 565–580, doi:10.1007/978-3-642-16242-8_40.

[8] Petros Papapanagiotou & Jacques Fleuriot (2015): *Modelling and Implementation of Correct by Construction Healthcare Workflows*, pp. 28–39. Springer International Publishing, Cham, doi:10.1007/978-3-319-15895-2_3.

[9] Petros Papapanagiotou & Jacques D. Fleuriot (2013): *Formal verification of collaboration patterns in healthcare*. Behaviour & Information Technology, pp. 1–16.

[10] Lawrence C. Paulson (1990): *Isabelle: The Next 700 Theorem Provers*. Logic and Computer Science 31, p. 361386.

[11] L.C. Paulson (1994): *Isabelle: A Generic Theorem Prover*. Lecture Notes in Computer Science 828, Springer.

[12] Bernardo Toninho, Luís Caires & Frank Pfenning (2011): *Dependent session types via intuitionistic linear type theory*. In: Proceedings of the 13th international ACM SIGPLAN symposium on Principles and practices of declarative programming, PPDP '11, ACM, New York, NY, USA, pp. 161–172, doi:10.1145/2003476.2003499.

[13] Philip Wadler (2014): *Propositions as sessions*. Journal of Functional Programming 24(2-3), p. 384418, doi:10.1017/S095679681400001X.

[14] Philip Wadler (2015): *Propositions As Types*. Commun. ACM 58(12), pp. 75–84, doi:10.1145/2699407.