Translating from Higher-Order to Higher-Order*

Chad E. Brown, Thibault Gauthier, and Josef Urban

Czech Technical University, Prague

Introduction

A hammer [5] for an interactive theorem prover (ITP) [13] typically translates an ITP goal into a formalism used by an automated theorem prover (ATP) [16]. Since the most successful ATPs have so far been first-order, the focus has been on first-order translations. There is however interest in producing ATPs working in richer formalisms, such as THF0 [3], THF1 [10], and TFF1 [6]. An interesting related task is to create a (grand) unified large-theory benchmark that would allow fair comparison of such systems and their integration with premise-selectors [1] across the different formalisms. As a step towards creating such benchmarks we would like to use translations that are TD-abstractions [11] and behave similarly on first-order formulas.

HOL4 [17], like many other ITPs, is based on an extension of Church's simple type theory [8] that includes prefix polymorphism and type definitions [12]. Without these extensions it would be possible to directly translate HOL4 terms and propositions into the THF0 format for higher-order ATPs such as Satallax [7] and LEO [2, 18]. It is nevertheless possible to give a translation from a goal in HOL4 into a THF0 problem that takes some advantage of the HOL4 higher-order constructs. The essential idea is to give axioms for a higher-order set theory and translate the HOL4 goal into the higher-order set theory. We propose such a translation here and briefly compare the result to a first-order translation. We have implemented these translations for HOL4 and plan to use them for the first (grand) unified benchmarks, generalizing the existing ones (CakeML [15], HOL4 standard library) used in the CASC LTB competition [19, 20].

Translation to THF0

In order to translate HOL4 into THF0 we begin by including a few basic constants and axioms. Note that base types o (for propositions) and ι (for individuals) are built into THF0. We will think of elements of type ι as being sets. The basic constants we include are as follows:

- $\mathsf{mem} : \iota \to \iota \to o$ corresponds to the membership relation on sets.
- $\mathsf{ne} : \iota \to o$ is for nonemptiness. HOL4 types will be mapped to nonempty sets.
- $\mathsf{ap} : \iota \to \iota \to \iota$ corresponds to set theory level application.
- $\mathsf{lam} : \iota \to (\iota \to \iota) \to \iota$ is used to construct set bounded λ-abstractions as sets.
- $\mathsf{bool} : \iota$ is used for a fixed two element set.
- $\mathsf{arr} : \iota \to \iota \to \iota$ is used to construct the function space of two sets.
- $\mathsf{p} : \iota \to o$ is a predicate which indicates whether or not an element of bool is true or not.

We then include a number of basic axioms summarized as follows: $\mathsf{arr}\ A\ B$ is nonempty when A and B are nonempty, lam and ap satisfy typing properties relative to arr and mem, boolean extensionality, functional extensionality and a beta axiom. If ι is interpreted using a model of ZFC, then the constants above can be interpreted in an obvious way as to make the basic axioms true.

Given this theory, our translation from HOL4 to THF0 can be informally described as follows. We map each HOL4 type α (including type variables) to a term $\hat{\alpha}$ of type ι for which

*Supported by the ERC Consolidator grant no. 649043 AI4REASON, and by the Czech project AI&Reasoning CZ.02.1.01/0.0/0.0/15_003/0000466 and the European Regional Development Fund.

we should always know $\mathsf{ne}\ \hat{\alpha}$ in the context in which α is used. The invariant can be maintained by always having a hypothesis $\mathsf{ne}\ \hat{\alpha}$ when α is a type variable or constant. HOL4 type variables (constants) are mapped to THF0 variables (constants) of type ι. For the remaining cases we use bool and arr. We map each HOL4 term $s : \alpha$ to a THF0 term $\hat{s}$ of type ι for which we should always know $\hat{s} \in \hat{\alpha}$ in the context in which s is used. Again, the invariant can be maintained by including the hypothesis $\hat{x} \in \hat{\alpha}$ whenever x is a variable or a constant. The ap and lam constants are used to handle HOL4 application and λ-abstraction. The axioms corresponding to typing rules maintain the invariant. Finally HOL4 propositions (which may quantify over type variables) are translated to THF0 propositions in an obvious way, using p to go from ι to o when necessary. As an added heuristic, the translation makes use of THF0 connectives and quantifiers as deeply as possible, only using p and $\hat{s}$ when necessary.

Translation of HOL4 to FOF for Large-theory Benchmarks

Our translation to first-order follows approximately [14] and keeps the same type encoding [4]. However, there are two major differences: We create new constant symbols and give independent definitions for lambda-abstraction and nested predicates as in [9, 21]. The same constant c used with two different arities i, j is translated to two different constants c_i and c_j. Arity equations relating c_i and c_j to c_0 are added and can be used to recover the dependency between c_i and c_j. Thanks to these modifications, the translation of a formula to first-order does not depend anymore on formulas co-occuring in the same problem.[1] This is essential to export large HOL4 theories in a consistent manner for the first-order LTB competition.

Example and Discussion

As a small example, suppose a HOL4 constant $\mathsf{B} : (\alpha \to \alpha) \to (\alpha \to \alpha) \to \alpha \to \alpha$ were defined so that $\forall \alpha.\ \forall f, g : \alpha \to \alpha.\ \forall x : \alpha.\ \mathsf{B}\ f\ g\ x = f\ (g\ x)$ were a HOL4 theorem (we call this Bdef). From this theorem we could prove $\forall \alpha.\ \forall x : \alpha.\ \mathsf{B}\ (\lambda x.x)\ (\lambda x.x)\ x = x$ (we call this Bid) To translate this to a THF0 problem, we would translate Bdef as the axiom

$$\forall A.\ \mathsf{ne}\ A \Rightarrow \forall f, g, x : \iota.\ \mathsf{mem}\ f\ (\mathsf{arr}\ A\ A) \Rightarrow \mathsf{mem}\ g\ (\mathsf{arr}\ A\ A) \Rightarrow \mathsf{mem}\ x\ A \Rightarrow$$
$$\mathsf{ap}\ (\mathsf{ap}\ (\mathsf{ap}\ \hat{\mathsf{B}}\ f)\ g)\ x = \mathsf{ap}\ f\ (\mathsf{ap}\ g\ x)$$

and Bid as the conjecture

$$\forall A.\ \mathsf{ne}\ A \Rightarrow \forall x : \iota.\ \mathsf{mem}\ x\ A \Rightarrow \mathsf{ap}\ (\mathsf{ap}\ (\mathsf{ap}\ \hat{\mathsf{B}}\ (\mathsf{lam}\ A\ (\lambda x.x)))\ (\mathsf{lam}\ A\ (\lambda x.x)))\ x = x.$$

Disregarding the type encoding the translation of Bdef to FOF is essentially

$$\forall f, g, x.\ \mathsf{B}_3(f, g, x) = \mathsf{ap}(f, \mathsf{ap}(g, x))$$

where B_3 is an arity 3 function. To translate Bid to FOF, we create two new constants c_0 and c_1, give a first-order definition of $c_1 = \lambda x.x$ as $\forall x.\ c_1(x) = x$ and an arity equation $\forall x.\ c_1(x) = \mathsf{ap}(c_0, x)$. After this the conjecture can be expressed as $\forall x.\ \mathsf{B}_3(c_0, c_0, x) = x$.

One advantage of the THF0 translation over first-order translations is that there is no need to deanonymize λ-abstractions inside terms. This means no new names need to be created simply to represent the problem. Since all names used will be common across a collection of problems, this may help techniques which learn to do premise selection.

The talk will include results comparing first-order and higher-order provers on HOL4 problems and discuss examples of possible benefits of using the proposed THF0 translation.

[1] With the exception of the counter used for generating new fresh constants

References

[1] Jesse Alama, Tom Heskes, Daniel Kühlwein, Evgeni Tsivtsivadze, and Josef Urban. Premise selection for mathematics by corpus analysis and kernel methods. *J. Autom. Reasoning*, 52(2):191–213, 2014.

[2] Christoph Benzmüller, Lawrence C. Paulson, Frank Theiss, and Arnaud Fietzke. LEO-II - a cooperative automatic theorem prover for classical higher-order logic (system description). In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *IJCAR*, volume 5195 of *LNCS*, pages 162–170. Springer, 2008.

[3] Christoph Benzmüller, Florian Rabe, and Geoff Sutcliffe. THF0 – the core of the TPTP language for classical higher-order logic. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, volume 5195 of *LNCS*, pages 491–506. Springer, 2008.

[4] Jasmin Christian Blanchette, Sascha Böhme, Andrei Popescu, and Nicholas Smallbone. Encoding monomorphic and polymorphic types. In Nir Piterman and Scott A. Smolka, editors, *TACAS*, volume 7795 of *LNCS*, pages 493–507. Springer, 2013.

[5] Jasmin Christian Blanchette, Cezary Kaliszyk, Lawrence C. Paulson, and Josef Urban. Hammering towards QED. *J. Formalized Reasoning*, 9(1):101–148, 2016.

[6] Jasmin Christian Blanchette and Andrei Paskevich. TFF1: The TPTP typed first-order form with rank-1 polymorphism. In Maria Paola Bonacina, editor, *CADE*, volume 7898 of *LNCS*, pages 414–420. Springer, 2013.

[7] Chad E. Brown. Satallax: An automatic higher-order prover. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *IJCAR*, volume 7364 of *LNCS*, pages 111–117. Springer, 2012.

[8] Alonzo Church. A formulation of the simple theory of types. *J. Symbolic Logic*, 5:56–68, 1940.

[9] Lukasz Czajka. Improving automation in interactive theorem provers by efficient encoding of lambda-abstractions. In Jeremy Avigad and Adam Chlipala, editors, *Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs, Saint Petersburg, FL, USA, January 20-22, 2016*, pages 49–57. ACM, 2016.

[10] Thibault Gauthier, Cezary Kaliszyk, and Josef Urban. TacticToe: Learning to reason with HOL4 tactics. In Thomas Eiter and David Sands, editors, *LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, May 7-12, 2017*, volume 46 of *EPiC Series in Computing*, pages 125–143. EasyChair, 2017.

[11] Fausto Giunchiglia and Toby Walsh. A theory of abstraction. *Artificial Intelligence*, 57(2-3):323–389, 1992.

[12] M. J. C. Gordon and T. F. Melham, editors. *Introduction to HOL: A theorem proving environment for higher order logic.* Cambridge University Press, 1993.

[13] John Harrison, Josef Urban, and Freek Wiedijk. History of interactive theorem proving. In Jörg H. Siekmann, editor, *Computational Logic*, volume 9 of *Handbook of the History of Logic*, pages 135–214. Elsevier, 2014.

[14] Cezary Kaliszyk and Josef Urban. Learning-assisted automated reasoning with Flyspeck. *Journal of Automated Reasoning*, 53(2):173–213, 2014.

[15] Ramana Kumar, Magnus O. Myreen, Michael Norrish, and Scott Owens. CakeML: a verified implementation of ML. In Suresh Jagannathan and Peter Sewell, editors, *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 179–192. ACM, 2014.

[16] John Alan Robinson and Andrei Voronkov, editors. *Handbook of Automated Reasoning (in 2 volumes).* Elsevier and MIT Press, 2001.

[17] Konrad Slind and Michael Norrish. A brief overview of HOL4. In Otmane Aït Mohamed, César A. Muñoz, and Sofiène Tahar, editors, *Theorem Proving in Higher Order Logics, 21st International Conference, TPHOLs 2008, Montreal, Canada, August 18-21, 2008. Proceedings*, volume 5170 of

LNCS, pages 28–32. Springer, 2008.

[18] Alexander Steen and Christoph Benzmüller. The higher-order prover Leo-III. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Automated Reasoning. IJCAR 2018*, volume 10900 of *LNCS*, pages 108–116. Springer, Cham, 2018.

[19] Geoff Sutcliffe. The 9th IJCAR automated theorem proving system competition - CASC-J9. *AI Commun.*, 31(6):495–507, 2018.

[20] Geoff Sutcliffe and Josef Urban. The CADE-25 automated theorem proving system competition - CASC-25. *AI Commun.*, 29(3):423–433, 2016.

[21] Josef Urban. MPTP 0.2: Design, implementation, and initial experiments. *J. Autom. Reasoning*, 37(1-2):21–43, 2006.