A review for the Journal of Functional Programming (JFP) of Mike Gordon and Tom Melham, eds., *Introduction to HOL: a theorem proving environment for higher order logic* (Cambridge University Press, 1993), ISBN 0-521-44189-7.

Background

This book is an introduction to the HOL system, an interactive theorem-proving environment for classical higher-order logic. The book is derived from the documentation of the HOL88 system, and aims to serve two purposes: "(i) to provide a coherent and self-contained introduction to the HOL system; (ii) to compress into a single volume most of the material that is needed for day-to-day work with HOL." The book is spring-bound (in a similar way to the Latex book), which re-enforces it's aim as a real day-to-day book for the HOL user. It is also a large book, having some 472 pages, although a large proportion of this is reference material.

Mechanised theorem proving is an important topic in Computer Science, as witnessed both by the increasing number of systems being produced (HOL, Coq, Isabelle, ALF, Nuprl, Lotos, Lego, etc), and the increasing use of these systems for industrial problems. The basic idea of theorem provers is to provide computer support for proving statements within some mathematical logic. Using an automated system can ensure that only logically sound inferences are made, and can help with the management of large proofs. Computer checked or aided verification of systems is increasingly being demanded in industrial applications, and particularly in defence and safety-critical applications.

The HOL system is one of the most widely used theorem provers, both in academia and industry. It is free, comes with extensive documentation, libraries, and an interactive help system. HOL is a direct decendant of the innovative LCF (logic of computable functions) theorem prover developed by Robin Milner in the early 1970s, and is an implementation of a version of Church's simple theory of types, a formalism dating back more than 50 years. Basically, the HOL logic is first-order classical predicate logic, with the following differences: the logic is higher-order (variables can range over functions and predicates); the logic is typed; and there is no separate syntactic category of formulae (terms of boolean type play the role of formulae). Alternatively, the HOL logic can be viewed as a typed lambda-calculus extended with a few logical constants (equality, implication, and a higher-order version of Hilbert's choice operator). Higher-order logic is a powerful system, and much of classical mathematics can be encoded in it.

The basic ideas of HOL are as follows. HOL is built on top of the strict functional language ML, and proofs are constructed interactively using the ML system. The HOL system defines ML types for the various logical entities, including terms, types, theorems, and theories. The ML type of theorems is an abstract data-type, with operators corresponding to the axioms and inference rules of higher-order logic. In this way, the type system ensures that only valid inferences can be made. Types can often, but not always, be omitted in logical terms, and are inferred using a similar inference algorithm (due to

Milner) to that found in ML itself. Perhaps the key use of ML in HOL is as a meta-language to program proof tactics, which are functions that allow one to make top-down (or goal directed) proofs by breaking a theorem into a number of simpler parts, proving these, and then combining the proofs of the parts to give a proof of the whole. Sophisticated tactics are built up from simple tactics using tacticals, which are functions that combine tactics to form other tactics. Without using tactics and tacticals, one would be restricted to making bottom-up proofs beginning with axioms and applying inference rules, which would be impractical for all but simple proofs.

HOL has mainly found application in the verification of hardware systems and communications protocols, as opposed to theorem proving in its own right. One uses the HOL system to construct a proof that a concrete design (with respect to some suitable abstraction of physical components) satisfies an abstract specification. In such proofs one often has an obligation to verify a huge number of conditions, which is a task that can only properly be addressed with computer support.

Summary

The book is divided into five parts. Part one is mostly just a condensed version of the remainder of the book (intended to give a flavour of the HOL system and its use), together with three examples: verification of a parity-checking circuit, building a normalisation tool for propositional terms, and a proof of the binomial theorem. Although the parity-checking problem is quite a simple example, it reveals that considerable experience is needed even to make simple proofs: "Trivial deductions sometimes require elaborate tactics, ... HOL experts can prove arbitrarily complicated theorems if they are willing to use sufficient ingenuity."

The normalisation example begins with the comment that "It is sometimes claimed that LCF-style systems can never be practical, because the efficiency needed to handle real examples can only be obtained with decision procedures coded as primitive rules. It is hoped that this chapter, ..., shows that the truth of such claims is not obvious." A simple (but inefficient) normalisation tool is first built using the built-in rewriting tools. Then a more efficient tool is built by programming a normalisation function directly in ML, and proving after each application of this function that the normalised output term is equivalent to the input term, by using the decision procedure for equality of propositional terms that forms part of the standard library for the HOL system. I was somewhat dissapointed to find the following comment "An even more efficient approach, ..., would be to avoid having to do this proof by verifying the normalisation program by some sort of meta-theoretic reasoning about ML. How to do this in HOL is not clear, ..."

Most of the section on the binomial theorem is concerned with modelling monoids, groups, and rings in HOL, which I found quite fascinating. In fact, only an outline of the HOL proof of the binomial theorem is given.

Part two is a substantial introduction (100 pages) to the ML programming language used as a meta-language for HOL system, and is based on *The ML Handbook* (an unpulished

report from INRIA), in turn based on the Edinburgh LCF book. The introduction is paced well, and there are many examples. Anyone familiar with the basic ideas of functional programming should have not trouble learning ML by reading this part of the book.

Part three is a short review (30 pages) of the logical theory on which HOL is based. First a set-theoretic semantics for the HOL logic is given, based upon a universe of sets with assumptions that are slightly weaker than those of Zermelo-Fraenkel set theory with the Axiom of Choice. Then the proof system of HOL is described, and is shown to be sound for the set-theoretic semantics. Finally, theories within HOL are treated in a formal way. Part three is the most technical part of the book, but written in a non-threatening style. But still I suspect that much of this material will be inaccessible to many readers coming from a Computing Science background.

Part four (100 pages) describes the HOL system in detail. Topics covered include the syntax of the logic in ML, the facilities for constructing and managing theorems and theories, the types definition package, an overview of the various built-in theories, and the system features for managing the ML interface to HOL. Part five (60 pages) gives a detailed account of the various techniques for proving theorems using HOL, including bottom-up proofs (using axioms and inference rules), conversions and conversion combining operators (used to implement rewriting tools in HOL), and goal directed proofs (tactics and tacticals).

There are two appendices, which document a small selection of ML and HOL library functions. Appendix A documents some standard ML functions, namely those that are used in the parity checker example in Chapter five. The book admits that this choice is somewhat ad hoc, but HOL comes with an interactive help system which in practice is what users will most likely use in day-to-day work with the system. Appendix B provides an example of the prescribed standard for documenting HOL libraries, in the shape of the library for proving propositional formulae.

Comments

The HOL system is of interest to functional programmers for a number of reasons, not the least of which is that it provides one of the largest and longest developed functional programs. Being brought up on languages like Miranda and Haskell, it was fascinating to see how many of the key ideas in modern functional languages were discovered by Milner in his design of an appropriate meta-language for the LCF system. Polymorphic type inference relieves the user from most of the burden of supplying type information in terms. The use of abstract data types allows the type system to guarantee that only sound inferences are made. The exception handling mechanism takes care of the fact that proof tactics may fail. Last but by no means least, the tacticals library is perhaps one of the first examples of the design of a special purpose library of higher-order functions, now an important style of functional programming; for example, special libraries exist for parsing, pretty-printing, handling I/O, building user interfaces, etc.

Being based on ideas about theorem proving from 20 years ago, there are now a num-

ber of modern theorem provers than are more sophisticated than HOL. For example, the ALF system (under development at Chalmers University, Sweden) implements a variant of Martin-Löf's theory of types, provides an X-windows interface for the development of proofs, and can be used to prove properties of functional programs (something which HOL has not proved well suited to). Nonetheless HOL still remains an important system because of the substantial effort that has been made in producing re-usable theory libraries and proof tools.

I came to this book with only a little knowledge of mechanised theorem proving. My feeling after reading the book (and following up most of the citations pertaining to HOL) is that I have a reasonable understanding of the HOL system, and what its limitations are, but would require a considerable amount of effort to actually learn how to use the system to any degree of competence. My main comment on the book is that it reads as if it has been produced over a long period of time, as indeed the material has been, much having been adapted from the original LCF documentation. A more unified book would have resulted if some of the material had been reworked, although with a book of this size this would have been a substantial undertaking.

If you are seeking an in-depth treatement of the theoretical work on which HOL is based (e.g. higher-order logic, polymorphic type systems, semantics), this is not the book for you. Rather this is a book for someone that wants to learn the basics of the HOL system and the 'tactics and tacticals' approach to theorem proving. It collects the basic material from the vast documentation provided with HOL into a single volume that can comfortably be placed into a briefcase and read on the train.

Graham Hutton
Department of Computer Sciences
Chalmers University of Technology
and University of Göteborg
S-412 96, Göteborg, Sweden