Introduction to the HOL System

Mike Gordon
Computer Laboratory
New Museums Site
Pembroke Street
Cambridge CB2 3QG

1 What is HOL?

The HOL system is an interactive theorem-proving environment for higher order logic. It is a development of Cambridge LCF [11] which, in turn, is directly based Robin Milner's original LCF system developed at Edinburgh [3].

HOL provides tools to support the proof of theorems in higher order logic. These tools are functions in the programming language ML, which forms the interface to HOL. It is intended that users of HOL will build their own application-specific theorem-proving infrastructure. ML is designed for this purpose.

The original HOL system (called HOL88) is fully implemented and available now. It is in the public domain and can be obtained by FTP from sites in the UK, USA and Canada. HOL88 is implemented in Lisp and runs on any platform that supports Common Lisp (e.g. Suns, MIPS, HP workstations, Apple Macintosh). Two new versions of HOL, both implemented in Standard ML, are available: HOL90 from the University of Calgary is a public domain system; ICL HOL is a commercial system intended to support applications in the security critical area.

HOL is completely documented. There is a detailed description of the system [5], which includes a formal semantics of the version of higher order logic used, a manual for the ML programming language and a description of the theorem proving infrastructure. There is also a reference manual [6] that documents every ML function in HOL. The text of this manual can be accessed by the help system and an X-windows browsing tool. Finally, there is a tutorial introduction to the system [7] and a training course (including exercises and solutions). All the documentation is public domain and the LaTeX sources are distributed with the system.

2 What is higher-order logic?

Higher-order logic is a version of predicate calculus that allows variables to range over funtions and predicates. The expressive power of this logic is roughly similar to set theory. It is sufficient for expressing most ordinary mathematical theories. The particular formulation of higher-order logic supported by the HOL system is an extension of Church's Simple Type Theory [1]. The fine details have been influenced by the develoment of HOL from LCF. In particular, the HOL logic allows LCF-style type variables. This provides, within the logic, some of the meta-theoretic notations used by Church. A second influence of LCF is the explicit management of logical theories. These form directed acyclic graphs and support the splitting of complicated specifications into a coherent structure. A feature of HOL not found in LCF is the separation of consistency-preserving definitional principles, from arbitrary axioms. Most developments using HOL are purely definitional and are thus guaranteed to be consistent.

3 What is ML?

ML (an acronym for "Meta Language") is an interactive functional programming language. It was originally developed for the LCF system by Robin Milner and his research assistants [2, 3].

ML has a polymorphic type discipline invented by Milner [9]. In LCF-style systems (e.g. HOL) there is a separate type of theorems. The only primitive operations on this type are the inference rules of a logic. The ML typechecker ensures that any value of the theorem type must have been obtained by a sequence of primitive inferences.

0-8186-2460-4/92 $03.00 © 1992 IEEE

ML supports functions as first-class values. It provides an eager evaluation regime. Higher order programming is used in LCF and HOL for combining proof generating tools.

The original version of ML ("Classic ML") was implemented in Lisp. A direct descendent of this is used by the original HOL system (HOL88). Major improvements to the compiler were implemented in the early 1980s by Larry Paulson and Gerard Huet. A new version of ML, called Standard ML (SML), has recently been defined by a committee lead by Robin Milner [8]. There are several implementations of this already available. The two new implementations of HOL (Calgary's HOL90 and ICL HOL) both use Standard ML rather than Lisp.

4 How is the HOL system used?

The HOL system is used in two main ways:

1. Directly for proving theorems.
2. As embedded theorem proving support for an application-specific verification system.

Approach 1 is used when higher-order logic is a suitable specification language (e.g. for hardware verification and classical mathematics). Approach 2 is taken when specifications in specific formalisms need to be supported. For example, ICL have build a theorem proving tool for the Z specification language on top of ICL HOL. At Cambridge, projects are in progress to build theorem proving support for several hardware description languages (ELLA, VHDL, Silage) and simple programming languages (Vista, Safe).

HOL provides many built-in theorem-proving tools for both forward and goal-directed proof, including a powerful rewriting subsystem based on Paulson's higher-order rewriting combinators [10]. There is a library facility that contains useful theories and tools that have been packaged for general use. Libraries have been contributed by users from both universities and industry. Parsing and pretty-printing libraries are provided; these can be used to support different notations and languages by semantic embedding in higher order logic.

References

[1] A. Church, "A Formulation of the Simple Theory of Types', *Journal of Symbolic Logic* **5**, 1940.

[2] M. J. C. Gordon, A. J. R. G. Milner, L. Morris, M. Newey and C. P. Wadsworth, "A Metalanguage for Interactive Proof in LCF", *Proceedings of the ACM Conference on the Principles of Programming Languages*, Tuscon, 1978.

[3] M. J. C. Gordon, A. J. R. G. Milner. and C. P. Wadsworth, *Edinburgh LCF: a mechanized logic of computation*, Springer Lecture Notes in Computer Science **78**, Springer-Verlag, 1979.

[4] M. J. C. Gordon, "HOL: A Proof Generating System for Higher-Order Logic", in *VLSI Specification, Verification and Synthesis*, edited by G. Birtwistle and P.A. Subrahmanyam, Kluwer, 1988.

[5] HVG Cambridge, *The HOL System: DESCRIPTION.* Produced by the Cambridge Research Centre of SRI International with the support of DSTO Australia. Distributed with the HOL88 System (Version 2.0 or later).

[6] HVG Cambridge, *The HOL System: REFERENCE.* Produced by the Cambridge Research Centre of SRI International with the support of DSTO Australia. Distributed with the HOL88 System (Version 2.0 or later).

[7] HVG Cambridge, *The HOL System: TUTORIAL.* Produced by the Cambridge Research Centre of SRI International with the support of DSTO Australia. Distributed with the HOL88 System (Version 2.0 or later).

[8] A. J. R. G. Milner, M. Tofte and R. Harper, *The Definition of Standard ML*, The MIT Press, 1990.

[9] A. J. R. G. Milner, "A Theory of Type Polymorphism in Programming", *Journal of Computer and System Sciences*, **17**, 1978.

[10] L. C. Paulson, "A higher-order implementation of rewriting", *Science of Computer Programming*, 3, pp 143-170, 1985.

[11] L. C. Paulson, *Logic and Computation: Interactive Proof with Cambridge LCF*, Cambridge University Press, 1987.