Verification of Functional Programs
`Isabelle/HOL`

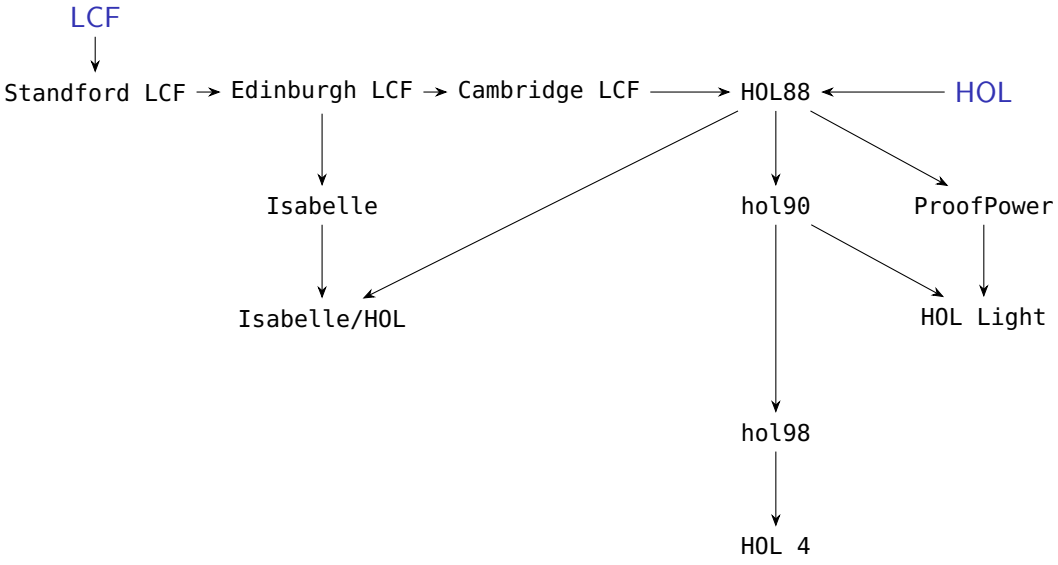
Andrés Sicard-Ramírez

Universidad EAFIT

Semester 2026-1

Introduction

LCF and HOL Proof Checkers and Proof Assistants



Logics

Generalisation of first, second, third, . . . -order logics

"It seems very natural to extend the system F of first-order logic by permitting quantification on predicate, propositional, and function variables as well as individual variables. One thus obtains the wffs of a system of second-order logic. One might then introduce predicate and function variables of higher type to denote relations and functions whose arguments may be relations and functions of individuals as well as individuals. Thus one would be led to a system of third-order logic, and if one permitted quantification with respect to these new variables, one would obtain a system of fourth-order logic. This process can be continued indefinitely to obtain logics of arbitrarily high order. Of course, after a while one runs out of different types of letters to use for different types of variables, but this problem can be solved by introducing type symbols to indicate the types of variables, and using a letter with type symbol α as subscript for a variable of type α." (Andrews [1986] 2002, p. 201)

HOL (Higher-Order Logic)

Paper: Alonzo Church (1940). "A Formulation of the Simple Theory of Types".

Also called "simple type theory" or "Church's type theory".

Types: Base types (ι of individuals and o of truth values), variables have fixed types, function types.

Terms: Variables, constants, λ-abstractions and function application.

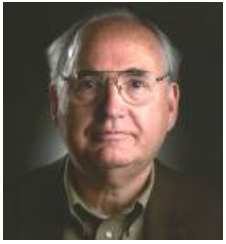
First-order logic with equality: Formulae are of type o.

Sets: Sets of individuals with type $\iota \to o$, sets of sets of individuals with type $(\iota \to o) \to o$, etc.

"Church's type theory has been extensively studied by two of Church's students, L. Henkin and P. Andrews... Henkin also showed in [30] that Church's type theory could be reformulated using only four primitive notions: function application, function abstraction, equality, and definite description... Andrews formulated a version of Church's type theory called Q_0 *that employs the ideas developed by Church, Henkin, and himself." (Farmer 2008, p. 269)*

LCF (Logic for Computable Functions)

Paper: D. S. Scott ([1969] 1993). "A Type-Theoretical Alternative to ISWIM, CUCH, OWHY".



Typed combinatory logic for reasoning about computable (continuos) functions including partial and non-terminating ones.

Types: Interpreted via domain theory.

Recursive functions: Interpreted via least-fixed points.

Systems

Stanford LCF

Paper: Robin Milner (1972). "Logic for Computable Functions. Description of a Machine Implementation".

Proof checker for LCF.

Edinburgh LCF

Book: Michael J. Gordon, Robin Milner and Christopher P. Wadsworth (1979). "Edinburgh LCF. A Mechanised Logic of Computation".

Edinburgh LCF

Book: Michael J. Gordon, Robin Milner and Christopher P. Wadsworth (1979). "Edinburgh LCF. A Mechanised Logic of Computation".

A problem with `Stanford LCF`: The size of proofs was limited by available memory.

Edinburgh LCF

Book: Michael J. Gordon, Robin Milner and Christopher P. Wadsworth (1979). "Edinburgh LCF. A Mechanised Logic of Computation".

A problem with `Stanford LCF`: The size of proofs was limited by available memory.

A solution:

> *"He [Milner] had the idea that instead of saving whole proofs, the system should just remember the results of proofs, namely theorems... To ensure that theorems could only be created by proof, Milner had the brilliant idea of using an abstract data type whose predefined values were instances of axioms and whose operations were inference rules. Strict typechecking then ensured that the only values that could be created were those that could be obtained from axioms by applying a sequence of inference rules—namely theorems." (M. J. C. Gordon 2000, p. 170)*

Edinburgh LCF

Book: Michael J. Gordon, Robin Milner and Christopher P. Wadsworth (1979). "Edinburgh LCF. A Mechanised Logic of Computation".

A problem with `Stanford LCF`: The size of proofs was limited by available memory.

A solution:

> *"He [Milner] had the idea that instead of saving whole proofs, the system should just remember the results of proofs, namely theorems... To ensure that theorems could only be created by proof, Milner had the brilliant idea of using an abstract data type whose predefined values were instances of axioms and whose operations were inference rules. Strict typechecking then ensured that the only values that could be created were those that could be obtained from axioms by applying a sequence of inference rules—namely theorems." (M. J. C. Gordon 2000, p. 170)*

Introduction: The functional programming language `ML` (Meta Language) and the concepts of "tactics" and "tacticals".

Cambridge LCF

Book: Lawrence C. Paulson ([1987] 2003). "Logic and Computation. Interactive Proof with Cambridge LCF".

Cambridge LCF

Book: Lawrence C. Paulson ([1987] 2003). "Logic and Computation. Interactive Proof with Cambridge LCF".

Many theoretical and technical improvements in relation to `Cambridge LCF`.

HOL

Paper: M. J. C. Gordon (1985). “HOL. A Machine Oriented Formulation of Higher Order Logic”.

HOL

Paper: M. J. C. Gordon (1985). "HOL. A Machine Oriented Formulation of Higher Order Logic".

"The design of HOL was largely taken 'off the shelf', the theory being classical higher order logic and the implementation being LCF." (M. J. C. Gordon 2000, p. 174)

HOL

Paper: M. J. C. Gordon (1985). "HOL. A Machine Oriented Formulation of Higher Order Logic".

"The design of HOL was largely taken 'off the shelf', the theory being classical higher order logic and the implementation being LCF." (M. J. C. Gordon 2000, p. 174)

"In this paper we describe a formal language intended as a basis for hardware specification and verification. This language is not new; the only originality in what follows lies in the presentation of details." (M. J. C. Gordon 1985, p. 2)

HOL

Paper: M. J. C. Gordon (1985). "HOL. A Machine Oriented Formulation of Higher Order Logic".

"The design of HOL was largely taken 'off the shelf', the theory being classical higher order logic and the implementation being LCF." (M. J. C. Gordon 2000, p. 174)

"In this paper we describe a formal language intended as a basis for hardware specification and verification. This language is not new; the only originality in what follows lies in the presentation of details." (M. J. C. Gordon 1985, p. 2)

Logic: Church's HOL with polymorphism and the Axiom of Choice is built-in via Hilbert's ϵ-operator.

(continued on next slide)

Primitive definitional principles

> *"The HOL system, unlike LCF, emphasises definition rather than axiom postulation as the primary method of developing theories. Higher order logic makes possible a purely definitional development of many mathematical objects (numbers, lists, trees, etc.) and this is supported and encouraged." (M. J. C. Gordon 2000, p. 174)*

(continued on next slide)

Derived definitional principles (high-level principles programmed in ML)

Derived definitional principles (high-level principles programmed in ML)

- Melham (1989) *"implemented a derived type-definition principle that converts descriptions of recursive datatypes into primitive definitions and then automatically derives the natural induction and primitive recursion principles for the datatype." (M. J. C. Gordon 2000, p. 174)*

HOL

Derived definitional principles (high-level principles programmed in ML)

- Melham (1989) *"implemented a derived type-definition principle that converts descriptions of recursive datatypes into primitive definitions and then automatically derives the natural induction and primitive recursion principles for the datatype." (M. J. C. Gordon 2000, p. 174)*
- Melham (1991) also implemented a derived definitional principle that *"allows inductively defined relations to be specified by a transition system, and then a rule-induction tactic to be automatically generated" (M. J. C. Gordon 2000, p. 174)*

Derived definitional principles (high-level principles programmed in ML)

- Melham (1989) *"implemented a derived type-definition principle that converts descriptions of recursive datatypes into primitive definitions and then automatically derives the natural induction and primitive recursion principles for the datatype." (M. J. C. Gordon 2000, p. 174)*
- Melham (1991) also implemented a derived definitional principle that *"allows inductively defined relations to be specified by a transition system, and then a rule-induction tactic to be automatically generated" (M. J. C. Gordon 2000, p. 174)*
- Slind (1996) implemented a derived definitional principle for handling general recursive definitions of functions.

Isabelle

Paper: Lawrence C. Paulson (1986). “Natural Deduction as Higher-Order Resolution”.
Paper: Lawrence C. Paulson (1989). “The Foundation of a Generic Theorem Prover”.

Isabelle

Paper: Lawrence C. Paulson (1986). "Natural Deduction as Higher-Order Resolution".
Paper: Lawrence C. Paulson (1989). "The Foundation of a Generic Theorem Prover".

Isabelle is a generic interactive proof assistant for a broad class of logics.

Isabelle

Paper: Lawrence C. Paulson (1986). "Natural Deduction as Higher-Order Resolution".
Paper: Lawrence C. Paulson (1989). "The Foundation of a Generic Theorem Prover".

Isabelle is a generic interactive proof assistant for a broad class of logics.

Meta-logic (logic framework): Intuitionistic higher-order logic.

Isabelle

Paper: Lawrence C. Paulson (1986). "Natural Deduction as Higher-Order Resolution".
Paper: Lawrence C. Paulson (1989). "The Foundation of a Generic Theorem Prover".

Isabelle is a generic interactive proof assistant for a broad class of logics.

Meta-logic (logic framework): Intuitionistic higher-order logic.

Isabelle
- Isabelle/CTT (extensional constructive type theory)
- Isabelle/FOL (first-order logic)
- Isabelle/ZF (Zermelo-Fraenkel set theory)
- Isabelle/HOL (higher-order logic)

Isabelle/HOL

Book: Tobias Nipkow, Lawrence C. Paulson and Markus Wenzel (2002). "Isabelle/HOL. A Proof Assistant for Higher-Order Logic".

Features: HOL + polymorphism + inductive data types + inductive predicates + recursion

References

References

Peter B. Andrews [1986] (2002). An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof. 2nd ed. Vol. 27. Applied Logic Series. Kluwer (cit. on p. 5).

Alonzo Church (1940). A Formulation of the Simple Theory of Types. The Journal of Symbolic Logic 5.2, pp. 55–68. DOI: 10.2307/2266170 (cit. on p. 6).

Willam M. Farmer (2008). The Seven Virtues of Simple Type Theory. Journal of Applied Logic 6.3, pp. 267–286. DOI: 10.1016/j.jal.2007.11.001 (cit. on p. 6).

Michael J. Gordon, Robin Milner and Christopher P. Wadsworth (1979). Edinburgh LCF. A Mechanised Logic of Computation. Vol. 78. Lecture Notes in Computer Science. Springer. DOI: 10.1007/3-540-09724-4 (cit. on pp. 10–13).

Mike Gordon (July 1985). HOL. A Machine Oriented Formulation of Higher Order Logic. Tech. rep. 68. Computer Laboratory. University of Cambridge. URL: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-68.pdf (cit. on pp. 16–19).

Mike Gordon (2000). From LCF to HOL: A Short History. In: Proof, Language, and Interaction: Essays in Honour of Robin Milner. Ed. by Gordon Plotkin, Colin Stirling and Mads Tofte. Foundations of Computing Series. MIT Press. Chap. 6. DOI: 10.7551/mitpress/5641.003.0012 (cit. on pp. 10–13, 16–24).

References

T. F. Melham (1991). A Package For Inductive Relation Definitions In HOL. In: 1991 International Workshop on the HOL Theorem Proving System and Its Applications. Ed. by Myla Archer, Jeffrey J. Joyce, Karl N. Levitt and Phillip J. Windley, pp. 350–357. DOI: 10.1109/HOL.1991.596299 (cit. on pp. 21–24).

Thomas F. Melham (1989). Automating Recursive Type Definitions in Higher Order Logic. In: Current Trends in Hardware Verification and Automated Theorem Proving. Ed. by Graham Birtwistle and P. A. Subrahmanyam. Springer. Chap. 9. DOI: 10.1007/978-1-4612-3658-0_9 (cit. on pp. 21–24).

Robin Milner (1972). Logic for Computable Functions. Description of a Machine Implementation. Tech. rep. Stanford Artificial Intelligence Project, Memo AIM-169. STAN-CS-72-288. Stanford University (cit. on p. 9).

Tobias Nipkow, Lawrence C. Paulson and Markus Wenzel (2002). Isabelle/HOL. A Proof Assistant for Higher-Order Logic. Vol. 2283. Lecture Notes in Computer Science. Springer. DOI: 10.1007/3-540-45949-9 (cit. on p. 29).

Lawrence C. Paulson (1986). Natural Deduction as Higher-Order Resolution. The Journal of Logic Programming 3.3, pp. 237–258. DOI: 10.1016/0743-1066(86)90015-4 (cit. on pp. 25–28).

References

Lawrence C. Paulson (1989). The Foundation of a Generic Theorem Prover. Journal of Automated Reasoning 5.3, pp. 363–397. DOI: 10.1007/BF00248324 (cit. on pp. 25–28).

Lawrence C. Paulson [1987] (2003). Logic and Computation. Interactive Proof with Cambridge LCF. Digitally printed version. Vol. 2. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (cit. on pp. 14, 15).

Dana S. Scott [1969] (1993). A Type-Theoretical Alternative to ISWIM, CUCH, OWHY. Theoretical Computer Science 121.1–2, pp. 441–440. DOI: https://doi.org/10.1016/0304-3975(93)90095-B (cit. on p. 7).

Konrad Slind (1996). Function Definition in Higher-Order. In: Theorem Proving in Higher Order Logics (TPHOLs'96). Ed. by Gerhard Goos, Juris Hartmanis, Jan Leeuwen, Joakim Wright, Jim Grundy and John Harrison. Vol. 1125. Lecture Notes in Computer Science. Springer, pp. 381–397. DOI: 10.1007/BFb0105417 (cit. on pp. 21–24).