HOL Around the World '93

compiled by Sara Kalvala

August 1993

Abstract

This document lists some current projects that use the HOL proof system. The appended bibliography refers to some papers produced in the cited projects. Additions to this list are welcome, and can be sent to sk@cl.cam.ac.uk.

- Phil Weiss (weiss@leopard.cs.uidaho.edu)

  Using a converted Z spec for the **parallel coordination** language Linda, I am proving that deadlock cannot occur in the dining philosophers problem, when only 4 out of 5 (or similar numbers) are allowed to "sit and eat" at once.

- Michael L Coe (coe@leopard.cs.uidaho.edu)

  Hardware verification. I am working for Phil Windley and doing some work HOL. So far, I have reverified the Tamarack processor (who hasn't) using Phil Windley's **Generic Interpreter** theory. Professor Windley has updated the GI theory this year, and now I am updating the previously list proof for the new GI theory.

- Donald Syme (symdchon@cs.anu.edu.au)

  My (honours) project concerns the embedding of the Formal Definition of **Standard ML** into HOL, with the aim of investigating the suitability of HOL for this task. I have concentrated on the Core dynamic semantics of Standard ML rather than embedding the Core static semantics (typing rules etc.). The project has involved intensive use of HOL with lots of definitions, custom tactics etc. It has also involved a good deal of work with type theory, especially **mutually recursive types**.

- Allan Stavely (al@nmt.edu)

  No specific projects yet. I still consider myself a learner.

- Ching-Tsun Chou (chou@cs.ucla.edu)

  Want to use HOL to verify **distributed algorithms**, in particular the distributed minimum spanning tree algorithm by Gallager, Humblet, and Spira (GHS), based on the notion of events. Currently working on the verificaton of a simple distributed flooding and summation algorithm as a preparation for the verification of GHS. By-products are (undirected) Graph Theory (up to important theorems about trees and minimum spanning trees) and a semantic embedding of Lamport's Temporal Logic of Actions. [C.-92a, C.-92b, C.-93]

- Jim Grundy (jg@cl.cam.ac.uk)

  I'm interested in **program refinement**, and in styles of reasoning, in particular 'window inference' that support refinement. [Gru91, Gru92]

- Wishnu Prasetya (wishnu@cs.ruu.nl)

  Verifying **distributed algorithms**. The language chosen is UNITY. I also intend to do the verification of the implementation of the algorithms. The implementation language is OCCAM.

- Tom Melham (`tfm@cl.cam.ac.uk`)

  - Foundations and Tools for Formal Verification (SERC Grant GR/F 36675): This project aims to exploit the theoretical insights of **categorical logic and modern type theory** to further develop the widely-used HOL theorem proving system and its associated approach to formal methods.
  - Dependent Type Theory in HOL: One component of this work has been the development of a translation of **Dependent Type Theory** (DTT) into higher order logic. The expressive power of Dependent Type Theory has several well-known advantages for applications of formal methods. The classical higher order logic supported by the HOL theorem prover is rather simpler to use than DTT but less expressive. The aim of this project is to gain the pragmatic advantages of DTT within the simpler HOL framework by means of a mathematical translation of DTT into higher order logic. This work on DTT in HOL has been done jointly with Bart Jacobs. [JM93]
  - Quantification over Type Variables: I have also been working on an extension of the HOL logic to include a very simple form of **quantification over types**. The aim is to solve certain practical problems with using the definitional mechanisms of HOL, with a view to facilitating the implemention of high-level derived rules of definition for abstract data types. The theoretical work for the proposed extension is described in a paper presented at the 1992 HOL Users Meeting in Leuven. Future development of this work will begin with an implementation based on the HOL90 system, followed by experimental implementation of the derived principles of definition mentioned above. [Mel93]
  - Inductive Definitions: Juanito Camilleri and I have prepared a suite of case studies using my **inductive definitions** package for HOL, which is described in the proceedings of the 1991 HOL workshop at Davis. The case studies are explained in a Cambridge technical report, and the HOL source code has been installed in the contrib directory of version 2.01. It is also available by ftp at ftp.cl.cam.ac.uk (128.232.0.56). [CM92, Mel92b]
  - The Pi-calculus in HOL: Work continues on the development of a mechanized formal theory of the pi-calculus in higher order logic using the HOL. The aim is to provide a practical HOL-based tool to support both reasoning about applications using the pi-calculus and metatheoretic reasoning about the calulus itself. Work completed to date is described in a Cambridge technical report. [Mel92a]
  - Documentation: Mike Gordon and I have (acting as editors) prepared a book on the HOL system based on selected parts of the system documentation, in particular the DESCRIPTION and TUTORIAL volumes. The book it to be published by Cambridge University Press sometime early in the New Year. [GM93] I have also cleaned up the HOL set theory libraries and prepared full Manuals for them.

- Gavan Tredoux (`gavan@elc.mth.uct.ac.za`)

  Formal semantics of programming languages: This is MSc work. Aim is to cover unbounded nondeterminism using a semantics based on execution sequences. Completed mechanization of semantics and derivation of standard programming logics - hoare logic for partial and total correctness, Dijkstra's wp/wlp calculus, a logic of invariant relations, and Pnueli's temporal logic DX. Currently writing up MSc thesis, expected Dec 1992. Making work available in hol/contrib. [Tre92]

- Stephen H. Brackin (`shb@oracorp.com`)

  I've used HOL—both HOL88 and HOL90, but most recently HOL90—as the prover for the Romulus **computer-security** analysis tool. Romulus can be used to determine the extent to which designs are restrictive, a nondisclosure property. Over the last several months, with Shiu-Kai Chin, I've developed a concrete recursive type implementation of a process specification language, an inductively defined operational semantics for this language, inductively defined security properties capable of guaranteeing restrictiveness for processes specified in this language, and theorems effectively defining these security properties by structural induction on process specifications, theorems that can be used as rewrites to prove that processes have these security properties [SS93]. I used HOL88 and an earlier version of Romulus to do a security analysis of a real design, but this work can't be released because of security considerations.

- David Fura (`dfura@faygo.boeing.com`)

  Verification of a Processor Interface Unit for a **Fault-Tolerant** Embedded Processor. This is work being performed with Phil Windley at the University of Idaho. We are specifying and verifying a Processor Interface Unit (PIU) for a Boeing-designed fault-tolerant computer called FTEP (Fault- Tolerant Embedded Processor). A significant result from this work is a new modeling approach for transactions that we will describe at HOL93.

- Paul Curzon (`pc@cl.cam.ac.uk`)

  - Compiler Verification: I am verifying a **compiling algorithm** for a subset of Vista: a structured assembly language for the Viper microprocessor. I have combined the compiler correctness theorem with a derived programming logic for the subset. Given a programming logic correctness statement about a Vista program, a derived inference rule can be used to derive a corresponding correctness statement about the compiled code. [Cur92d, Cur92c, Cur92a, Cur92b]
  - I am using HOL to hierarchically verify an existing Asynchronous Transfer Mode (ATM) **communication network -** the Fairisle network. This involves hardware and protocol verification. Initially, I am verifying the Fairisle switch.

- Garrel Pottinger (`garrel@msiadmin.cit.cornell.edu`)

  Worked on building a verification system for **Ada** at ORA Corporation that included machinery for verifying asymptotic correctness of floating point programs. Recommended using HOL as the theorem prover to be used in the system. This was done, though, by that time, I was no longer with ORA. [Pot92b, Pot92a]

- Rachel E. Onate-Roxas (`rachel@cs.anu.edu.au`)

  A simple programming language is used to demonstrate the feasibility of mechanical proof of the validity of **program transformation** rules. The language is formally defined using natural semantics and the system relies on a knowledge base that captures the properties of environments, memories and predicates that are suitable for expressing applicability conditions. The verification system used is the Higher Order Logic system. The proofs that were developed in this study were aided by high level tactics described in the paper. This project is a pilot study for proving the rules which underpin the implementation of the executable specification language REFINE. [RN92]

- Richard J. Boulton (`rjb@cl.cam.ac.uk`)

  My main interest is in providing frameworks and tools for writing **efficient proof procedures** in HOL, with particular emphasis on decision procedures. I am also interested in automation through the use of heuristics, and in using HOL to support formal reasoning about programming and hardware description languages.

  [BGHT90, Bou92d, Bou92b, BGG+92, Bou92c, Bou92a]

- Catia M. Angelo (`catia@imec.be`)

  - The verification of generic and parameterized **hardware modules** in the CATHEDRAL silicon compiler environment: Such modules are often flexible and complex due to optimizations in the hardware image. The project consisted in developing a methodology for structuring the proof of such complex hardware. The correctness proof is divided in two parts: structural and behavioral reasoning. The structural reasoning consists in proving that an irregular implementation behaves like a regular one. That is made using the knowledge of the design process. The methodology is based on the observation and decomposition of the individual design decisions as they have been made by a designer starting from a naive description (the specification) towards the actual optimized hardware. The behavioral reasoning consists in proving that the regular implementation meets the high level specification. [ACD91]

  - The **comparison** of HOL and Boyer-Moore for formal hardware verification. Faced with the challenge of designing correct circuits, the research community has been applying alternative verification methodologies instead of only traditional methods like ad-hoc simulation. The best choice among alternatives like tautology checking, symbolic simulation and theorem proving depends very much on the kind of circuit to be verified and on the level of abstraction considered. When theorem proving is best applicable, one is faced with the problem of choosing a formalism. We compared the proof assistant HOL and the theorem prover Boyer-Moore based on a practical experience with both systems in order to verify a combinatorial and parameterized hardware module from the CATHEDRAL II Silicon Compiler library. Although the comparison is based on a specific application, general features, advantages and drawbacks of both systems were investigated, bearing in mind the verification of other kinds of circuits. [AVCD91a, AVCD91b, AVCD93]
  - The formal semantics definition of a **multi-rate DSP** specification language and the development of proof procedures for a formal transformational design system. Silage is an applicative language for describing multi-rate DSP algorithms. It has been used as the high level input language to the CATHEDRAL Silicon Compilers. Hardware implementations of specifications described in Silage are synthesized and inconsistencies in the synthesis trajectory must be avoided. To build the foundations of a verification environment, the formal semantics of Silage had to be defined. Our main motivation to formally define the semantics of Silage was to build formal tools, in such a way that the interactivity between the user and the silicon compiler could be increased in a safe way. The designer could then interactively transform the specification, fully exploiting his expertise, but without the risks of introducing inconsistencies. Formally verified transformational design techniques could then be applied at the specification level of the Silicon Compilers. I am currently developing proof procedures for the correctness verification of multi-rate transformations of Silage specifications. [ACD92, ACD93]

- Glynn Winskel (`gwinskel@daimi.aau.dk`)

  I supervise MSc and PhD students in the use of HOL such as:

  - Automating recursive types definitions in HOL (2 student projects 1990)
  - Low-level circuit verification in HOL (student project using Winskel's compositional version of Bryant's MOSSIM model to verifying small MOS circuits where signal strength plays a role)
  - Program verification in HOL (MSc project of Sten Agerholm, an extended abstract for which is in HOL Proceedings 1991)
  - Domain theory in HOL (beginning PhD student)
  - Simple resolution in HOL (student project 1989)
  - Model checking in HOL (student project making a "local model checking" of the kind found in the Edinburgh Concurrency Workbench into a tactic returning proof or disproof of a finite labelled transition system satisfying an assertion in the modal mu calculus—based on Winskel's algorithm)
  - Programming with proofs (MSc project using HOL as an infra structure around Krivine's second order intuitionistic logic in which programs were extracted from proofs, 1991)

- Victor Carreño (`vac@air12.larc.nasa.gov`)
  - Verification of **mutual exclusion** algorithm using Transition Assertions. Transition Assertions is a Specification notation with semantics based on higher order logic [V. 93a].
  - Top level description and electronic block model of **data routing** circuit in higher order logic. Proof of correspondence by implication of electronic block model to top level description using HOL.
  - Temporal behavioural generic model for describing **dynamic systems**. Generic model is based on higher order functions and is called Transition Assertions. Eight specific models in higher order logic with graphical counterparts. The eight models include single transitions, multiple transitions, inertia and preemtion. Specification of electronic digital cruise control using Transition Assertions. Proof by implication, using HOL, that specification complies with three requirements [V. 93b].

- Monica Nesi (`mn@cl.cam.ac.uk`)

  The purpose is to build an interactive environment to support verification and reasoning about **process algebra** specifications. The formal theory for a specific behavioural semantics, namely observational congruence, and for a modal logic, a slight extension of Hennessy-Milner logic, over a subset of Milner's CCS, called pure CCS (no value passing), is embedded in the HOL logic, and the resulting formalization supports verification strategies based on mechanized formal proof. These include strategies that exhibit different degrees of user interaction depending on the subsets of CCS under consideration [CIN91], proofs of correctness and verification of modal properties by mathematical induction for parameterized (finite-state or infinite-state) specifications [Nes92b, Nes92a, Nes92c].

- Ray Toal

  Compiler Correctness: Equivalence of Natural and Operational **Semantics**.

- Konrad Slind (`slind@informatik.tu-muenchen.de`)
  - **Implementation of HOL** in SML: I have re-designed and re-implemented the HOL theorem prover in Standard ML. This has been in release for over a year. Current investigation is in fine-tuning the performance, which already exceeds that of hol88 by approximately a factor of 4. [Sli91a]
  - Investigation of **meta-theoretic extension** of the HOL system: Formalization of the syntax and semantics of SML in HOL lends one the ability to reason about the HOL implementation. This self-reference, if used appropriately, can be used to bootstrap the implementation, in effect introducing new primitive rules of inference in a sound way. This can be used to iteratively improve the performance of the theorem prover. What has been solved is how to ensure "appropriate usage"; the remaining problem is to find an appropriate notion of equivalence between programs and build up a stock of theorems and inference rules for reasoning about this equivalence. [Sli93, Sli92, Sli90, Sli91b]
  - [BGS+90, SBHS89, BHSS89, SBGS89]

- Michael J. Healy (`mjhealy@atc.boeing.com`)

  Investigating **formal verification** using HOL. Worked on NASA contract in formal hardware verification 1988-1990. Also used HOL in preliminary investigation of software verification, including knowledge-based systems. Briefly, it is difficult to get a viable effort going in this area. All the chips are not yet in.

- Flemming Andersen (`fa@tfl.dk`)

  The goal is to develop tools in a safe way to be used for verifying (at least parts of) software mainly within the area of telecommunication.

- UNITY: A part of my work with HOL has resulted in the implementation of what I call **HOL-UNITY** [And92]. It is a definitional theory of the UNITY theory as described in Chandy and Misra's book. Due to a progress relation in the UNITY logic, a way of defining non-primitive recursive recursive functions was needed. This lead to the definition of a theory for **recursive boolean functions**. The HOL-UNITY theory is now being used for proving properties of parallel programs. The user interface is however still for experts-only. Hence, it is planned to extend HOL-UNITY to support automated proof techniques and a programming language.
- Tarski: As a refinement of the work with fixed points this theory has become a separate package called Tarski [AP92]. This package allows for defining minimal and maximal **fixed points** for monotonic boolean predicate transformers. The package also supports functions for deriving structural induction principles supported by the fixed points. The Tarski package has been made together with my colleague Kim Dam Petersen.
- Other Projects: Another small project resulted in the implementation of a theory for linear temporal logic, a theory defining a Duration Calculus, the branching time temporal logic CTL, and a version of Roger Hales interval temporal logic. None of these theories have been used, since their only purpose was to present their semantic definitions as derived theories within the HOL logic.

• Kim Dam Petersen (`kimdam@tfl.dk`)

Constructing a theory for constructing Dana Scott's P-omega model in HOL, to be used for constructing **recursive domains and continuous functions**. The project includes defining theories of partial functions, (partial, complete, product, function, algebraic, reflexive) order relations, and of other some concepts (e.g. topology and continuity).

• Joakim von Wright (`Jwright@finabo.abo.fi`)

- Program refinement in HOL: We have formalised predicate transformer semantics and the corresponding bf refinement notion in HOL. This includes high-level refinement concepts such as correctness proofs for loops, data refinement and superposition. [BvW90, vWS91, vWHLL92, vW92, BvW92]
- Temporal Logic of Actions in HOL: We have embedded the semantics of Lamport's **TLA** in HOL and proved the soundness of the TLA proof rules. We are trying to find ways of doing TLA proofs using the formalisation. TLA is used to prove properties and implementation of concurrent algorithms. [vWL92]
- Verifying proofs: I am currently building a **theory of HOL-proofs** in HOL. The idea is to represent the types and terms of HOL inside HOL, as well as inference steps and proofs. Thus a proof generated by HOL (or some other system) could be fed into HOL and checked.

• Phillip J. Windley (`windley@cs.uidaho.edu`)

- Abstract Models for Hardware. This project is developing **abstract descriptions** of hardware devices that can be instantiated to provide the specification and verification of a particular device.
- Linking HOL to VLSI CAD Tools. This project is devloping connections between HOL and a suite of **VLSI CAD tools** in use at the University of Idaho's Microelectronic Research Center. The CAD tools range from in–house tools such as the NOVA simulator to public domain tools such as the Berkeley OCT Tools.
- Implementation of a Verified **Microprocessor with Security Features**. This project is designing and fabricating a verified microprocessor with a 5 stage instruction pipeline, supervisory mode, and interrupts.

- Juergen Froessl (`froessl@ira.uka.de`), Thomas Kropf (`kropf@ira.uka.de`), Ramayya Kumar (`kumar@ira.uka.de`), Ralf Reetz (`reetz@ira.uka.de`), Sofiene Tahar (`tahar@ira.uka.de`), Klaus Schneider (`schneide@ira.uka.de`)

  We are a group of 6 people working in the field of hardware verification at the University of Karlsruhe, Germany. Although we use different approaches and theorem provers for that task, most of our work is based on the HOL90 system. Most of the work aims at automating proofs for hardware verification, modelling hardware behaviour at different abstraction levels, and intregrating hardware verification with commercial design tools. Parts of our work is sponsored the national ministry of science (DFG SFB 358).

  – To embellish MEPHISTO, a tool for automating **hardware proofs** at the RT-level and gate-level. This comprises tactics for automatically decomposing high-level proof goals into simpler subgoals, which are then processed by FAUST. (Kropf, Kumar, Schneider)
  – To embellish FAUST, an automated, **tableau-based theorem prover** for first-order logic. FAUST is also capable of solving a limited set of higher-order formulae. FAUST is fully integrated in HOL and allows two modes of operation — the faster unvalidated mode where the proven subgoal uses `mk_thm` and the slower mode where a forward proof of the subgoal is generated. (Kropf, Kumar, Schneider)
  – High-level specification and verification of hardware: In order to simplify the use of MEPHISTO for normal circuit designers, we are developing methods to specify the intended hardware behaviour in common **hardware description languages**. (Kropf, Reetz)
  – Automating hardware proofs at higher abstraction levels requires the use of **domain specific knowledge**. In this project, we try to automate harware proofs for pipelined processors by concentrating on RISC architectures. (Kumar, Tahar)
  – An extensive theory for describing regular, **parametrizable structures** using ML like descriptions is being developed. This project comprises the development of a structural specification language and basic building blocks for cascadable, tree-like, hybrid and other **interconnections**. A set of induction tactics for automatically performing the equivalence proofs of two such structural descriptions is being developed. (Kumar, Schneider)
  – Most approaches use only fully synchronous models for gate level descriptions, i.e. an AND-gate represents the conjuction of two variables WITHOUT any delay. In this project, we try to extend our current behavioural models down to transistor-(switch)level, which incorporate **complex delay models**. (Froessl, Kropf)
  – Verification approches may also be used to generate test patterns to find fabrication faults. In this project, approaches which are currently based on model checking are being extended to include hierarchical **functional test pattern generation**. (Kropf)
  – The importance of verification tools can be established when they are exercised on realistic applications and furthermore integrated into realistic design environments. In this project, we are integrating our MEPHISTO verification system into a commercial **ASIC design system** (CADENCE). Additionally, complete chip designs are currently being verified with MEPHISTO. A set of benchmark circuits for evaluating different aspects of harware verification are being created. (Kropf, Kumar, Reetz, Schneider)

  [DKR93, TRK92, TRK93, RTK91b, RTK91a, RKT93, KRT91b, KRT91a, KRT92b, KRT92a, KRT92c, KRT93d, KRT93a, KRT93b, KRT93c, SR93b, SR93c, SR93a]

- Sten Agerholm (`sagerholm@daimi.aau.dk`)

  – Formalization of a recursive **modal assertion language** for maximum and minimum fixed points in HOL. An ML program called the model checker has been implemented to prove assertions of finite labelled transition systems are true or false, automatically.

In order to ensure the recursive assertions defined as maximum and minimum fixed points are sensible it is necessary to have theorems in HOL stating certain functions are monotonic. Such theorems are provided automatically by another ML program called the monotonicity prover. [AS90]

– A study of **program verification** in HOL. It treats all aspects from formalizing program semantics to building tools to support actual proofs of program correctness. In particular it introduces a way of formalizing programming language theory in the HOL logic which has obvious advantages over approaches seen elsewhere. Thus more general constructs are allowed and no restrictions are imposed on program variables and their types. The study highlights many important aspects of the HOL system, e.g. the flexibility and tailorability of the system is fundamental to this application. [Age92]

– Domain theory is the mathematical foundation of denotational semantics, providing an understanding of both nontermination and recursive definitions. It has been used to give semantics to a variety of functional (and imperative) programming languages and for instance the semantics of Scott's Logic of Computable Functions was expressed in domain theory. This work concerns a formalization of **domain theory** in HOL. We introduce the notions of cpo's, continuous functions and fixed points, and define a number of constants which give ways of constructing cpo's and continuous functions from trivial ones. An ML program, called the cpo prover, has been implemented to prove automatically that a term constructed using the cpo constructions only, is indeed a cpo. The informal syntax defined by the cpo prover constitutes a metalanguage for cpo's. In a similar way, we have implemented a metalanguage for continuous functions as an ML program, called the continuity prover. We focus on both the actual formalization and the unsuccessful attempts. Our first attempt was based on a representation of the underlying sets of cpo's as HOL types. However, we need dependent subtypes (i.e. term parametrised types) and these are not provided by the HOL logic. But dependent subtypes can be simulated in HOL by predicates denoting subsets of types. Unfortunately, this introduces other complications. For instance, it affects the definition of continuity and a certain (dependent) lambda abstraction must be used in definitions of function constructions. [Age]

– Future work: Reasoning about functional programs in the HOL formalization of domain theory. Denotational semantics of functional programming languages in HOL, program transformation, equivalences of denotational and operational semantics.

- Cui Zhang (`zhang@sakura.cs.ucdavis.edu`)

  I am specifying the semantics of a langauge called microSR with **concurrency and nonderterminism** and deriving a Hoare-like logic for this langauge in HOL, mechanizing the programming logic for microSR in HOL.

- Andrew Gordon (`gordon@cs.chalmers.se`)

  Silage is a simple dataflow language used at several sites to specify **DSP algorithms** for synthesis to custom IC designs. The HOL-Silage project (part of the CHEOPS Esprit project between IMEC, Philips Eindhoven, and Cambridge) investigated how performance-improving transformation of Silage circuits could be rendered as theorem-proving in HOL [Gor92, Gor93b].

  I am also interested in using HOL to reason about the **operational behaviour of programs** written in ML or another functional language. As a preliminary experiment, I am embedding Abramsky's lazy lambda-calculus in HOL, and investigating a theory of program equivalence based on applicative bisimulation. Preliminary experience with embedding lambda-calculus syntax was reported at HUG93 [Gor93a].

- Peter Vincent Homeier (`homeier@cs.ucla.edu`)

  Verification of software written in a **general-purpose programming language**, including expressions with side effects, nondeterminism, and concurrency. The approach includes a tool to automatically create a Verification Condition Generator (VCG) function along with a proof of its correctness.

- Tim Leonard (`Leonard@ricks.enet.dec.com`)

  Our group is exploring formal verification of **Digital's chips**. We're concentrating on Alpha AXP processors and support chips, on the difficult (to verify) functions within those chips, and on abstraction levels from the register-transfer level to the Alpha AXP architecture.

- John Harrison (`jrh@cl.cam.ac.uk`)

  Exploring applications of the **real numbers** in theorem proving, e.g. verification of floating-point hardware, analysis of numerical algorithms and correctness of Computer Algebra Systems.

- Damian Anderson (`de9da34@uk.ac.mmu.cs.sa`)

  Development of a verified **behavioural synthesis** system.

- Mark van der Voort (`vdvoort@cs.utwente.nl`)

  Trying to extend HOL for use in verification of **functional languages**, without using any deep embedding.

- Wai Wong (`ww@cl.cam.ac.uk`)

  A **proof checker** for HOL: A file format for storing HOL proof is defined which will become the interface between HOL and a proof checker. HOL88 is modified to record and output proofs. A prototype of proof checker is being developed.

- Matthew J. Morley (`mjm@lfcs.ed.ac.uk`)

  Safety analysis of geographic data for a data-driven railway signalling control system. The idea is to automate the verification of the data's correctness w.r.t. certain safety criteria.

References

[ACD91] C.M. Angelo, L. Claesen, and H. De Man. "A Methodology for Proving Correctness of Parameterized Hardware Modules in HOL". In D. Borrione and R. Waxman, editors, *Proceedings 10th International Symposium on Computer Hardware Description Languages and their Applications*, pages 63–82. Marseille, France, Elsevier Science Publishers B. V. (North-Holland), Amsterdam, April 1991.

[ACD92] C.M. Angelo, L. Claesen, and H. De Man. "The Formal Semantics Definition of a Multi-Rate DSP Specification Language in HOL". In L. Claesen and M. Gordon, editors, *Proceedings of the IFIP International Workshop on Higher Order Logic Theorem Proving and its Applications - HOL-92*, pages 375–394. IMEC, Leuven, Belgium, Elsevier Science Publishers B. V. (North-Holland), Amsterdam, September 1992.

[ACD93] C.M. Angelo, L. Claesen, and H. De Man. "Degrees of Formality in Shallow Embedding Hardware Description Languages in HOL". In *Participants Proceedings of the 1993 International Workshop on Higher Order Logic Theorem Proving and its Applications*, pages 89–100. Vancouver, Canada, August 1993. To appear in *Lecture Notes in Computer Science*, Springer-Verlag.

[Age] Sten Agerholm. Formalizing domain theory in hol. Draft version, contact author.

[Age92] Sten Agerholm. Mechanizing program verification in hol. In Myla Archer, Jeffrey J. Joyce, Karl N. Levitt, and Phillip J. Windley, editors, *Proceedings of the 1991 International Workshop on the HOL Theorem Proving System and its Applications*. IEEE Computer Society Press, 1992. More details in: DAIMI IR-111, Aarhus University, April 1992.

[And92] Flemming Andersen. *A Theorem Prover for UNITY in Higher Order Logic*. PhD thesis, Technical University of Denmark, March 1992.

[AP92] Flemming Andersen and Kim Dam Petersen. Recursive boolean functions in hol. In Myla Archer, Jeffrey J. Joyce, Karl N. Levitt, and Phillip J. Windley, editors, *Proceedings of the 1991 International Workshop on the HOL Theorem Proving System and its Applications*. IEEE Computer Society Press, 1992.

[AS90] Sten Agerholm and Henrik Skjødt. Automating a model checker for recursive modal assertions in hol. Technical Report DAIMI IR-92, Aarhus University, January 1990.

[AVCD91a] C.M. Angelo, D. Verkest, L. Claesen, and H. De Man. “A Synopsis on the Comparison of HOL and Boyer-Moore for Formal Hardware Verification”. In P. Prinetto and P. Camurati, editors, *Proceedings of the Advanced Research Workshop on Correct Hardware Design Methodologies*, pages 421–426. Turin, Italy, Elsevier Science Publishers B. V. (North-Holland), Amsterdam, June 1991.

[AVCD91b] C.M. Angelo, D. Verkest, L. Claesen, and H. De Man. “Formal Hardware Verification in HOL and in Boyer-Moore: A Comparative Analysis”. In M. Archer, J. Joyce, K. Levitt, and P. Windley, editors, *Proceedings of the 1991 International Workshop on the HOL Theorem Proving System and its Applications*, pages 340–347. Davis, California, U.S.A., IEEE Computer Society Press, August 1991.

[AVCD93] C.M. Angelo, D. Verkest, L. Claesen, and H. De Man. “On the Comparison of HOL and Boyer-Moore for Formal Hardware Verification”. *Formal Methods in System Design: An International Journal*, 2(1):45–72, February 1993. Kluwer Academic Publishers, ISSN: 0925-9856.

[BGG+92] R. Boulton, A. Gordon, M. Gordon, J. Harrison, J. Herbert, and J. Van Tassel. Experience with embedding hardware description languages in HOL. In *Proceedings of TPCD Conference 1992 (Theorem Provers in Circuit Design: Theory, Practice and Experience)*, Nijmegen, The Netherlands, June 1992. North-Holland/Elsevier.

[BGHT90] R. Boulton, M. Gordon, J. Herbert, and J. Van Tassel. The HOL verification of ELLA designs. Technical Report 199, University of Cambridge Computer Laboratory, August 1990. Revised version in *Proceedings of the International Workshop on Formal Methods in VLSI Design*, Miami, January 1991.

[BGS+90] Graham Birtwistle, Brian Graham, Todd Simpson, Konrad Slind, Mark Williams, and Simon Williams. Verifying an secd chip in hol. In *Proceedings of the IFIP WG 10.2/WG 10.5 International Workshop on Applied Formal Methods for Correct VLSI Design*, pages 369 – 378. North Holland, 1990.

[BHSS89] Graham Birtwistle, Mike Hermann, Todd Simpson, and Konrad Slind. From formal proof to netlist. In *Proceedings of the Canadian Conference VLSI*, pages 217 – 224, Vancouver, British Columbia, Canada, 1989.

[Bou92a] R. J. Boulton. Boyer-Moore automation for the HOL system. In L. J. M. Claesen and M. J. C. Gordon, editors, *Higher Order Logic Theorem Proving and its Applications: Proceedings of the IFIP TC10/WG10.2 Workshop*, pages 133–142, Leuven, Belgium, September 1992. North-Holland/Elsevier.

[Bou92b] R. J. Boulton. A HOL semantics for a subset of ELLA. Technical Report 254, University of Cambridge Computer Laboratory, April 1992.

[Bou92c] R. J. Boulton. A lazy approach to fully-expansive theorem proving. In L. J. M. Claesen and M. J. C. Gordon, editors, *Higher Order Logic Theorem Proving and its Applications: Proceedings of the IFIP TC10/WG10.2 Workshop*, pages 19–38, Leuven, Belgium, September 1992. North-Holland/Elsevier.

[Bou92d] R. J. Boulton. On efficiency in theorem provers which fully expand proofs into primitive inferences. Technical Report 248, University of Cambridge Computer Laboratory, February 1992.

[BvW90] R.J.R. Back and J. von Wright. Refinement concepts formalised in higher-order logic. *Formal Aspects of Computing*, 2:247–272, 1990.

[BvW92] R.J.R. Back and J. von Wright. Predicate transformers and higher order logic. In *REX Workshop for Refinement of Distributed Systems*, The Netherlands, 1992.

[C.-92a] C.-T. Chou. A note on interactive theorem proving with theorem continuation functions. In L.J.M. Claesen and M.J.C. Gordon, editors, *Higher Order Logic Theorem Proving and its Applications*, volume A-20 of *IFIP Transactions*, pages 59–70, Leuven, Belgium, 1992. North-Holland.

[C.-92b] C.-T. Chou. A sequent formulation of a logic of predicates in HOL. In L.J.M. Claesen and M.J.C. Gordon, editors, *Higher Order Logic Theorem Proving and its Applications*, volume A-20 of *IFIP Transactions*, pages 71–80, Leuven, Belgium, 1992. North-Holland.

[C.-93] C.-T. Chou. Predicates, temporal logic, and simulations. In *HOL User's Group Workshop*, Vancouver, Canada, August 1993. Springer Verlag.

[CIN91] A.J. Camilleri, P. Inverardi, and M. Nesi. Combining interaction and automation in process algebra verification. In S. Abramsky and T.S.E. Maibaum, editors, *Proceedings of the 4th International Joint Conference on the Theory and Practice of Software Development*, volume 494 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[CM92] J. Camilleri and T. Melham. Reasoning with inductively defined relations in the hol theorem prover. Technical Report 265, University of Cambridge Computer Laboratory, August 1992.

[Cur92a] Paul Curzon. Compiler correctness and input/output. In *Proceedings of the 3rd IFIP Working Conference on Dependable Computing for Critical Applications*, Dependable Computing and Fault-Tolerant Computing. Springer-Verlag, 1992.

[Cur92b] Paul Curzon. Deriving correctness properties of compiled code. In L. Claesen and M. Gordon, editors, *International Workshop on Higher Order Logic Theorem Proving and its Applications*, 1992.

[Cur92c] Paul Curzon. A programming logic for a verified structured assembly language. In A. Voronkov, editor, *Logic Programming and Automated Reasoning*, volume 624 of *Lecture Notes in Artificial Intelligence*, pages 403–408. Springer-Verlag, 1992.

[Cur92d] Paul Curzon. A verified compiler for a structured assembly language. In Myla Archer, Jeffrey J. Joyce, Karl N. Levitt, and Phillip J. Windley, editors, *Proceedings of the 1991 International Workshop on the HOL Theorem Proving System and its Applications.* IEEE Computer Society Press, 1992.

[DKR93] D. Eisenbiegler, K. Schneider, and R. Kumar. A functional approach for formalizing regular hardware structures. In C. Seeger and J. Joyce, editors, *International Workshop on Higher Order Logic Theorem Proving and its Applications*, Vancouver, Canada, August 1993.

[GM93] M. J. C. Gordon and T. F. Melham, editors. *Introduction to HOL: A theorem proving environment for higher order logic.* Cambridge University Press, 1993.

[Gor92] Andrew D. Gordon. The formal definition of a synchronous hardware-description language in higher order logic. In *International Conference on Computer Design, Cambridge, Massachusetts, October 11–14, 1992*, pages 531–534. IEEE Computer Society Press, 1992.

[Gor93a] Andrew D. Gordon. A mechanisation of name-carrying syntax up to alpha-conversion. In *HUG'93 Higher Order Logic Theorem Proving and its Applications, Vancouver*, Lecture Notes in Computer Sciences. Springer Verlag, 1993.

[Gor93b] Andrew D. Gordon. A mechanised definition of Silage in HOL. Technical Report 287, University of Cambridge Computer Laboratory, February 1993.

[Gru91] Jim Grundy. Window inference in the HOL system. In Phillip J. Windley, Myla Archer, Karl N. Levitt, and Jeffrey J. Joyce, editors, *Proceedings of the International Tutorial and Workshop on the HOL Theorem Proving System and its Applications*, University of California at Davis, August 1991. ACM-SIGDA and IEEE Computer Society, IEEE Computer Society Press.

[Gru92] Jim Grundy. A window inference tool for refinement. In Cliff B. Jones, B. Tim Denvir, and Roger C. F. Shaw, editors, *Proceedings of the 5th Refinement Workshop*, Workshops in Computer Science, Lloyd's Register, London, England, January 1992. BCS FACS, Springer-Verlag.

[JM93] B. Jacobs and T. Melham. Translating dependent type theory into higher order logic. In M. Bezem and J. F. Groote, editors, *Typed Lambda Calculi and Applications: Proceedings of the International Conference, Utrecht, March 1993*, volume 664 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[KRT91a] K. Schneider, R. Kumar, and Th. Kropf. Automating most parts of hardware proofs in HOL. In *Lecture Notes in Computer Science 575 (International Workshop on Computer Aided Verification)*, pages 365–375. Springer, Aalborg, July 1991.

[KRT91b] K. Schneider, R. Kumar, and Th. Kropf. Structuring hardware proofs: First steps towards automation in a higher-order environment. In A. Halaas and P.B. Denyer, editors, *International Conference on Very Large Scale Integration*, pages 81–90. North Holland, 1991.

[KRT92a] K. Schneider, R. Kumar, and Th. Kropf. Efficient representation and computation of tableau proofs. In L.J.M. Claesen and M.J.C. Gordon, editors, *Higher Order Logic Theorem Proving and its Applications*, volume A-20 of *IFIP Transactions*, pages 39–58, Leuven, Belgium, 1992. North-Holland.

[KRT92b] K. Schneider, R. Kumar, and Th. Kropf. The FAUST prover. In *Lecture Notes in Computer Science 607 (11th Conference on Automated Deduction)*, pages 766–770. Springer, Saratoga Springs, New York, 1992.

[KRT92c] K. Schneider, R. Kumar, and Th. Kropf. Modelling generic hardware structures by abstract datatypes. In L.J.M. Claesen and M.J.C. Gordon, editors, *Higher Order Logic Theorem Proving and its Applications*, volume A-20 of *IFIP Transactions*, pages 165–176, Leuven, Belgium, 1992. North-Holland.

[KRT93a] K. Schneider, R. Kumar, and Th. Kropf. Accelerating tableaux proofs using compact representations. In R. Brayton, M. Clarke, and P.A. Subrahmanyam, editors, *International Journal of Formal Methods in System Design*. Kluwer Academic Publishers, 1993.

[KRT93b] K. Schneider, R. Kumar, and Th. Kropf. Alternative proof procedures for finite-state machines in higher-order logic. In C. Seeger and J. Joyce, editors, *International Workshop on Higher Order Logic Theorem Proving and its Applications*, Vancouver, Canada, August 1993.

[KRT93c] K. Schneider, R. Kumar, and Th. Kropf. Eliminating higher-order quantifiers to obtain decision procedures for hardware verification. In C. Seeger and J. Joyce, editors, *International Workshop on Higher Order Logic Theorem Proving and its Applications*, Vancouver, Canada, August 1993.

[KRT93d] K. Schneider, R. Kumar, and Th. Kropf. Hardware verification with first-order BDD's. In *Conference on Computer Hardware Description Languages*, Ottawa, Canada, 1993.

[Mel92a] T. Melham. A mechanized theory of the π-calculus in hol. Technical report, Computer Laboratory, University of Cambridge, 1992.

[Mel92b] T. Melham. A package for inductive relation definitions in hol. In Myla Archer, Jeffrey J. Joyce, Karl N. Levitt, and Phillip J. Windley, editors, *Proceedings of the 1991 International Workshop on the HOL Theorem Proving System and its Applications.* IEEE Computer Society Press, 1992.

[Mel93] T. F. Melham. The hol logic extended with quantification over type variables. In L. J. M. Claesen and M. J. C. Gordon, editors, *Higher Order Logic Theorem Proving and its Applications: Proceedings of the IFIP TC10/WG10.2 International Workshop, Leuven, September 1992*, number A-20 in IFIP Transactions. North-Holland, 1993.

[Nes92a] M. Nesi. Formalizing a modal logic for ccs in the hol theorem prover. In L. Claesen and M. Gordon, editors, *International Workshop on Higher Order Logic Theorem Proving and its Applications*, 1992.

[Nes92b] M. Nesi. Mechanizing a proof by induction of process algebra specifications in higher order logic. In K.G. Larsen and A. Skou, editors, *Proceedings of the 3rd Workshop on Computer Aided Verification*, volume 575 of *Lecture Notes in Computer Science.* Springer-Verlag, 1992.

[Nes92c] M. Nesi. Verifying the correctness of an infinite counter in the hol theorem prover. In *Proceedings of the ERCIM Workshop on Theory and Practice in Verification*, 1992.

[Pot92a] Garrel Pottinger. Classical type theory with transfinite types. In L. Claesen and M. Gordon, editors, *International Workshop on Higher Order Logic Theorem Proving and its Applications*, 1992.

[Pot92b] Garrel Pottinger. Completeness for the hol logic, preliminary report. posted to info-hol, January 1992.

[RKT93] R. Kumar, K. Schneider, and T. Kropf. Structuring and automating hardware proofs in a higher-order theorem-proving environment. In R. Brayton, M. Clarke, and P.A. Subrahmanyam, editors, *International Journal of Formal Methods in System Design*, pages 165–230. Kluwer Academic Publishers, 1993.

[RN92] R. Roxas and M. C. Newey. Proofs of program transformations. In Myla Archer, Jeffrey J. Joyce, Karl N. Levitt, and Phillip J. Windley, editors, *Proceedings of the 1991 International Workshop on the HOL Theorem Proving System and its Applications.* IEEE Computer Society Press, 1992.

[RTK91a] R. Kumar, Th. Kropf, and K. Schneider. First steps towards automating hardware proofs in HOL. In *International Workshop on the HOL Theorem Proving System and its Applications*, pages 190–193, Davis, California, August 1991. IEEE Computer Society Press.

[RTK91b] R. Kumar, Th. Kropf, and K. Schneider. Integrating a first-order automatic prover in the HOL environment. In *International Workshop on the HOL Theorem Proving System and its Applications*, pages 170–176, Davis, California, August 1991. IEEE Computer Society Press.

[SBGS89] Sue Stodart, Graham Birtwistle, Brian Graham, and Konrad Slind. Chrestomathy of hol specifications and proofs. prepared under Contract W2213-8-6363/01-SS for the Canadian Department of National Defence, 146 pages., 1989.

[SBHS89] Konrad Slind, Graham Birtwistle, Mike Hermann, and Todd Simpson. From logic to layout: transforming hol specifications into gate array netlists. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, pages 352 – 355, Montral, Quebec, Canada, 1989.

[Sli90] Konrad Slind. Completion as a derived rule of inference. Technical Report 90-409-33, University of Calgary Computer Science Department, 1990.

[Sli91a] Konrad Slind. An implementation of higher order logic. Technical Report 91-419-03, University of Calgary Computer Science Department, 1991.

[Sli91b] Konrad Slind. Object language embedding in standard ml of new jersey. In Robert Harper, editor, *Proceedings of the Second ML Workshop*, Carnegie-Mellon University, Pittsburgh, Pennsylvania, USA, September 1991.

[Sli92] Konrad Slind. Adding new rules to an lcf-style logic implementation: Preliminary report. In L.J.M. Claesen and M.J.C. Gordon, editors, *Higher Order Logic Theorem Proving and its Applications*, volume A-20 of *IFIP Transactions*, Leuven, Belgium, 1992. North-Holland.

[Sli93] Konrad Slind. Ac unification in hol90. In *HOL User's Group Workshop*, Lecture Notes in Computer Sciences, Vancouver, Canada, 1993. Springer Verlag.

[SR93a] S. Tahar and R. Kumar. A formalization of a hierarchical model for risc processors. In P. Spies, editor, *European Informatics Congress Computing Systems Architecture,*, Informatik Aktuell, Munich, Germany, October 1993. Springer.

[SR93b] S. Tahar and R. Kumar. Implementing a methodology for formally verifying risc processors in hol. In C. Seeger and J. Joyce, editors, *International Workshop on Higher Order Logic Theorem Proving and its Applications*, Vancouver, Canada, August 1993.

[SR93c] S. Tahar and R. Kumar. Towards a methodology for the formal hierarchical verification of risc processors. In *International Conference on Computer Design*, Cambridge, Massachusetts, October 1993. IEEE Computer Society Press.

[SS93] S. Brackin and S-K Chin. Server-process restrictiveness in HOL. In *HOL User's Group Workshop*, Vancouver, Canada, August 1993. Springer Verlag.

[Tre92] Gavan Tredoux. Mechanizing execution sequence semantics in HOL. *South African Computer Journal*, 7, July 1992.

[TRK92] Th. Kropf, R. Kumar, and K. Schneider. Bridging the gap between hardware-verification and design. In *IFIP World Conference*, Madrid, September 1992. IFIP.

[TRK93] Th. Kropf, R. Kumar, and K. Schneider. *Embedding Hardware Verification within a Commercial Design Framework*, pages 242–257. Lecture Notes in Computer Science. Springer, Arles, France, May 1993.

[V. 93a] V. Carreño. The transition assertions specification method. Technical Report 279, University of Cambridge Computer Laboratory, January 1993.

[V. 93b] V. Carreño. Verification in higher-order logic of mutual exclusion algorithm. In *HOL User's Group Workshop*, Vancouver, Canada, August 1993. Springer Verlag.

[vW92] J. von Wright. Doing lattice theory in higher order logic. Reports on Mathematics and Computer Science 136, Åbo Akademi University, 1992.

[vWHLL92] J. von Wright, J. Hekanaho, T. Långbacka, and P. Luostarinen. Mechanising some advanced refinement concepts. In *Proceedings of the 1992 International Workshop on Higher Order Logic Theorem Proving and its Applications*, Leuven, Belgium, September 1992. North-Holland.

[vWL92] J. von Wright and T. Långbacka. Using a theorem prover for reasoning about concurent algorithms. In *Proceedings of the 4th Workshop on Computer-Aided Verification*, Montreal, Canada, June 1992. Springer-Verlag.

[vWS91] J. von Wright and K. Sere. Program transformations and refinements in HOL. In *Proceedings of the 1991 International Workshop on Higher Order Logic Theorem Proving and its Applications*, Davis, USA, August 1991. ACM.