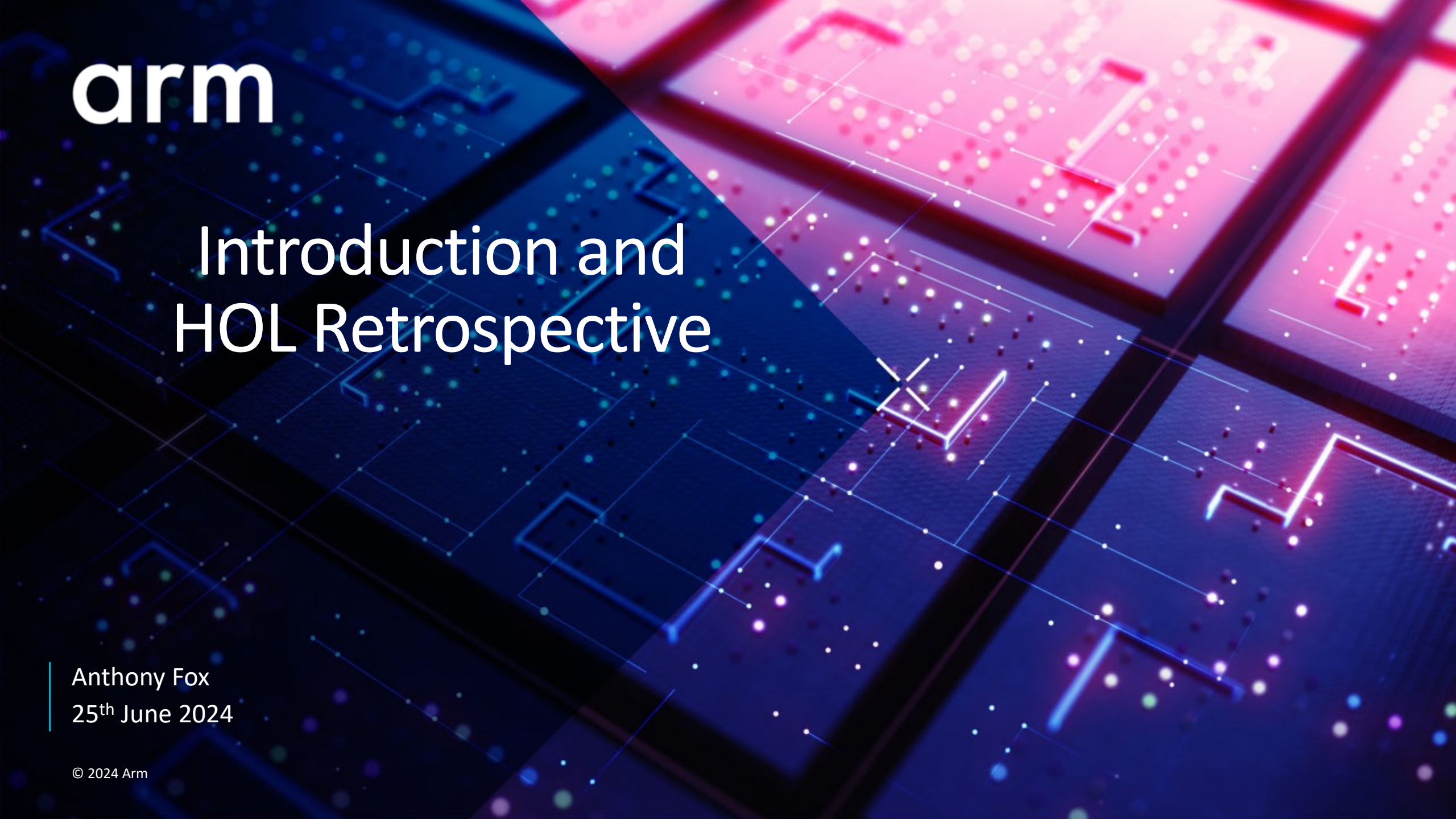


Introduction and HOL Retrospective

Anthony Fox

25th June 2024

Overview

1. Introduction to Arm + Formal

2. Origins of the ITP Conference

3. Origins of HOL4

   Through the lens of Computer Lab technical reports

4. Athabasca (HOL98) to Trindemossen-1

   A nostalgic trawl through the release notes

arm

Part 1. Introduction to ARM + Formal

Arm + Formal

- I will give a quick overview of some formal methods work at Arm
  - With a historical perspective
- Very incomplete, as it just relates to activities of those presenting at this workshop
  - For example, does not cover work of David Russinoff and others on floating-point verification with ACL2
  - Arm presenters at this workshop are all members of ATG (Architecture & Technology Group) at Arm

Arm + Formal

Early 2000s

- In 2000 Arm partnered with Cambridge and Leeds for two EPSRC projects
  - Grant GR/N13159/01: "**Specification and Verification of ARM6**"
    - PI was AG Cohn but it was led by **Graham Birtwistle** at Leeds
  - Grant GR/N13135/01: "**Formal Specification and Verification of ARM6**"
    - PI was **Mike Gordon** at Cambridge
- The first grant had an emphasis on *specification*
- The second grant had an emphasis on *verification* (using HOL)

Arm + Formal

- Graham Birtwistle had two PhD students working on the first grant

  Dominic Pajak: ***Specification of microprocessor instruction set architectures : ARM case study***, 2005

  Daniel Schostak: ***Methodology for the formal specification of RTL RISC processor designs (with particular reference to the ARM6)***, 2003

- Dominic worked for Arm 2006 to 2011
- Daniel joined Arm around 2003 and is still here

Arm + Formal

- I was a post-doc on the second grant under the supervision of Mike Gordon
  - Produced HOL4 models of the Armv3 architecture and ARM6 micro-architecture
  - An initial verification was completed in 2002
  - The processor model was based on specifications written by Daniel Schostak

- There were follow-on projects at Cambridge…
  - Monadic Armv7 specification was presented at first ITP (Edinburgh) in 2010
  - Work culminated with L3 (DSL) models of Armv7 in 2011 and Armv8 in 2014 (used in CakeML proofs)

Arm + Formal

ISA-Formal

- The following two papers were published in 2016:

  Alastair Reid: ***Trustworthy specifications of ARM® v8-A and v8-M system level architecture***

  Alastair Reid, Rick Chen, Anastasios Deligiannis, David Gilday, David Hoyes, Will Keen, Ashan Pathirane, Owen Shepherd, Peter Vrabel, Ali Zaidi: ***End-to-End Verification of Processors with ISA-Formal***

- Result of five years of work on migrating Arm specifications to the domain specific language ASL
- The Architecture Explorer (ArchEx) tooling processes ASL and generates Verilog
  - Enables model checking with EDA tools such as JasperGold
- The ISA-Formal approach is actively used at Arm, and there are links to the Sail ecosystem (some architecture proofs have been done in Isabelle/HOL)

ARM + Formal

Consistency Models

- An initial formal consistency model for the Arm architecture was released in 2017
  - Will Deacon used Jade Alglave's *cat* language, which integrates with the *herd+diy* tool suite
- This model has been maintained and *enhanced*
- Alternative formulations have been constructed
- This line of work has been summarized in the following 2021 paper

  Jade Alglave, Will Deacon, Richard Grisenthwaite, Antoine Hacquard, Luc Maranget:
  Armed Cats: Formal Concurrency Modelling at Arm

Arm + Formal

Confidential Compute

- The following paper was published in 2023:

  Anthony Fox, Gareth Stockwell, Shale Xiong, Hanno Becker, Dominic Mulligan, Gustavo Petri, Nathan Chong: ***A Verification Methodology for the Arm Confidential Computing Architecture: From a Secure Specification to Safe Implementations***

- Focusses on verification of Arm's Realm Management Monitor (RMM), which is a security critical *firmware* component of Arm CCA
- HOL4 has been used to validate/verify the (Realm Management Interface) ABI specification

Arm + Formal

Present Day…

- We have four talks at this workshop
- Jade Alglave (formal lead at Arm ATG)
  - An overview of her team's work on "specifications at Arm":
    - Latest on memory models
    - Newer work on standardizing and formalizing ASL
- Hrutvik Kanabar
  - Talk about enabling HOL4 verifications on top of ASL specifications
- Eleni Vafeiadi Bila
  - Talk about the latest RMM specification and verification work using HOL4
- Magnus Myreen
  - Talk about a "constraints validator" that has been implemented using HOL4

arm

Part 2. Origins of the ITP Conference

Origins of the ITP Conference

- The first *HOL Users Group* meeting was held **36 years ago** (1988) at Sidney Sussex College, **Cambridge**
- There were no published papers and I've not seen a proceedings
- **Aims likely coincided with today's workshop**
  - Find out what people are doing with HOL
  - Learn about how to get the most out of HOL
  - Discuss upcoming/possible changes to HOL

Origins of the ITP Conference

- The HUGs meetings evolved as follows:
  - International HOL Users Meeting (1988-1990)
  - International Workshop on Higher Order Logic Theorem Proving and its Applications (1991-1995)
  - International Conference on Theorem Proving in Higher Order Logics (TPHOLs, 1996-2009)
  - International Conference on Interactive Theorem Proving (ITP, 2010-)

International HOL Users Meeting, Sidney Sussex College, Cambridge, 29–30 September 1988

International HOL Users Meeting, Trinity Hall, Cambridge, 14–15 December 1989

arm

Part 3. Origins of HOL4

Through the lens of Cambridge Computer Laboratory technical reports

Origins of HOL4

- The Cambridge Computer Laboratory technical report series contain quite a few HOL related gems: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-table.html

- I will highlight a few to give a sense of how things evolved
- This is a *very* incomplete picture (just for fun)

1983

- UCAM-CL-TR-39: Larry Paulson, *"Tactics and tacticals in Cambridge LCF"*, July 1983

- UCAM-CL-TR-41: Mike Gordon, *"LCF_LSM, A system for specifying and verifying hardware"*, September 1983

- UCAM-CL-TR-42: Mike Gordon, *"Proving a computer correct with the LCF_LSM hardware verification system"*, September 1983

1983: LCF_LSM, A system for specifying and verifying hardware

- Context: 10+ years on from the birth of Stanford LCF, then Edinburgh LCF (1973)

- LCF: **Logic of Computable Functions** (based on Dana Scott's theoretic foundation)

  "A computer system, designed and implemented by Robin Milner and his colleagues, for generating formal proofs interactively"

- LSM: **Logic of Sequential Machines**

  "A formal system which extends the logical calculus embedded in LCF with terms based on the behaviour expressions of Robin Milner's Calculus of Communicating Systems (CCS)"

1983: LCF_LSM

Note that terms, types and formulae can be directly input using the quotation syntax (details below), but theorems can only be created by the predefined inference rules. A tutorial introduction to ML and the concepts underlying the representation of OL in ML is [Gordon4]. I strongly suggest reading this if the brief remarks above are confusing.

Here is a session to illustrate the manipulation of OL from ML:

```
#let fm = "!t. t==T \/ t==F";;
fm = "!t. t==T \/ t==F" : form

#let th1 = ASSUME fm;;
th1 = . |- "!t. t==T \/ t==F" : thm

# let th2 = SPEC "x:bool" th1;;
th2 = . |- "x==T \/ x==F" : thm
```

- ML = Meta Language and OL = Object Language
- The LCF approach will be very familiar to HOL users

1983: LCF_LSM

Summary of LSM's Terms for Defining Devices

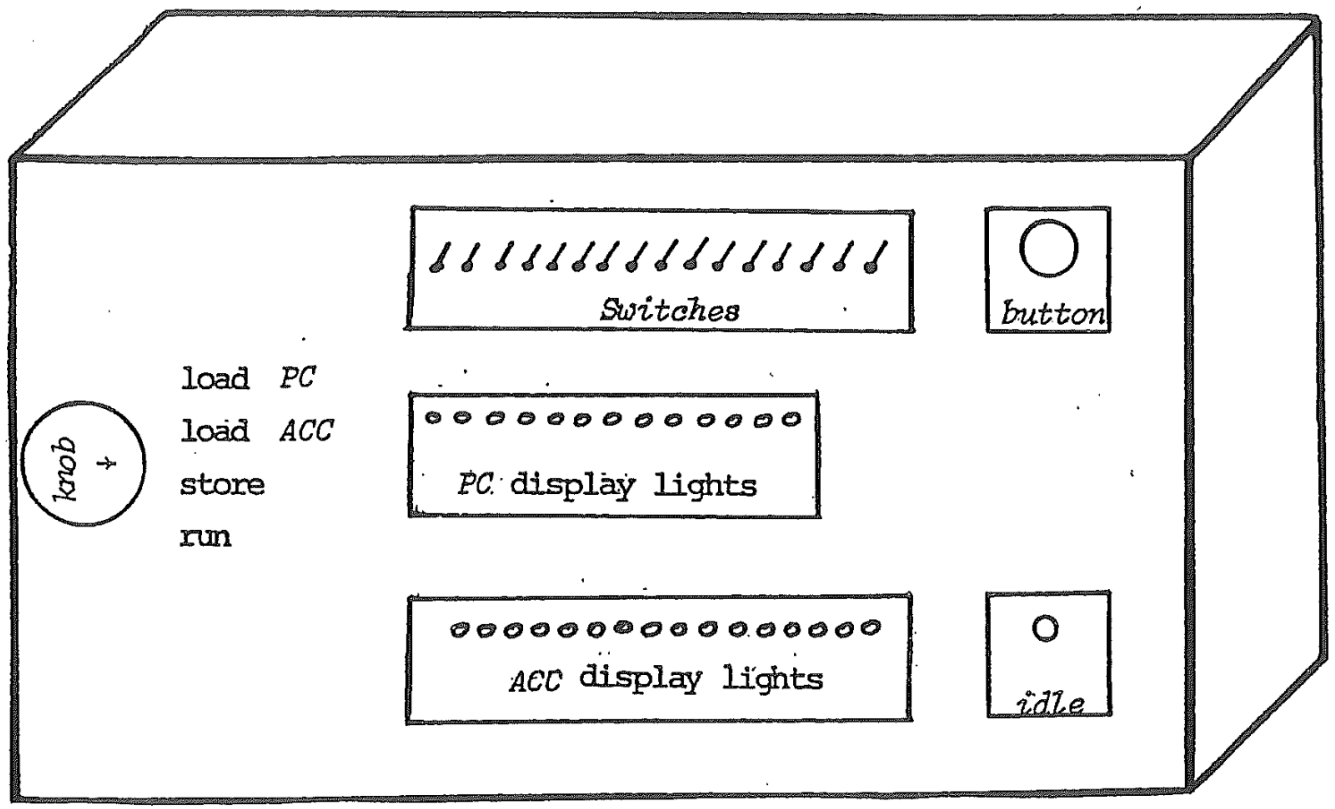
Sequential Machine:	*dev{x1,...,xm}.{l1=t1,...,ln=tn};t*
Renaming:	*t rn[l1=l1';...;ln=ln']*
Joining:	*[| t1 | t2 | ... | tn |]*
Hiding:	*t hide{l1,...,ln}*
Merging Microcycles:	*until l do t*

In Appendix 1 we list various ML functions for manipulating (e.g. constructing and destructing) these terms.

- LSM: Likely unfamiliar to HOL users
- Ben Moskowski persuaded Mike Gordon that Higher-Order-Logic could do the job (instead of LSM)
- LCF_LCM turned into HOL

1983: Proving a computer correct…

- Introduced “Gordon’s Computer”
- Two registers: the program counter PC (31 bits wide), and the accumulator ACC (16 bits wide)
- Random access memory with 13-bit addresses, pointing to 16-bit words
- Eight instructions:
  HALT, JMP, JZR, ADD, SUB, LD, ST, SKIP



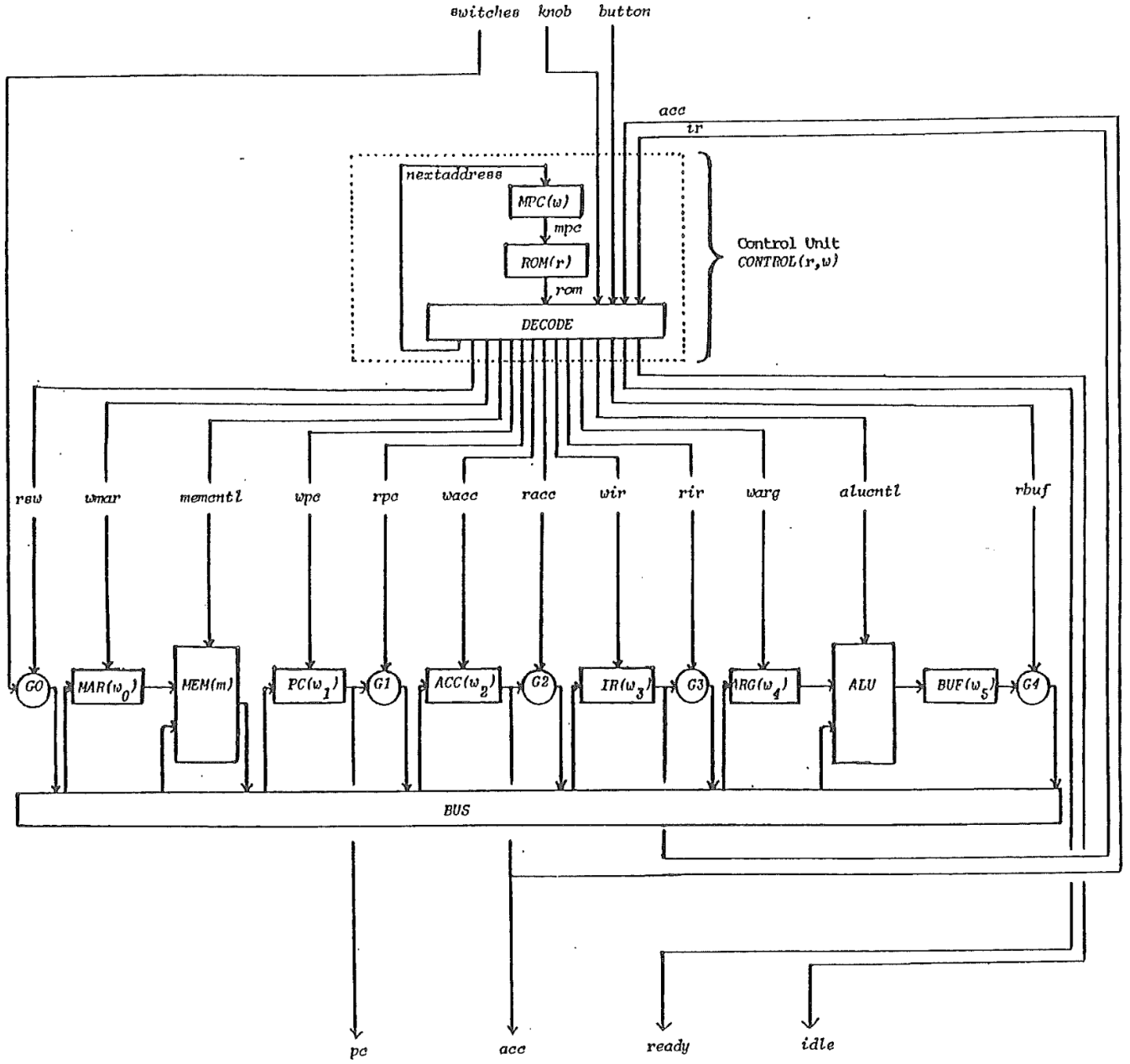
1983: Proving a computer correct…

The eight instuctions are:

|0|0|0|----------addr----------|	*HALT*	Stops exececution
|0|0|1|----------addr----------|	*JMP L*	Jump to *addr*
|0|1|0|----------addr----------|	*JZR L*	Jump to *addr* if *ACC=0*
|0|1|1|----------addr----------|	*ADD L*	Add contents of *addr* to *ACC*
|1|0|0|----------addr----------|	*SUB L*	Subtract contents of *addr* from *add*
|1|0|1|----------addr----------|	*LD L*	Load contents of *addr* into *ACC*
|1|1|0|----------addr----------|	*ST L*	Store contents of *ACC* in *addr*
|1|1|1|----------addr----------|	*SKIP*	Skip to next instruction

```
NEXT(m,w1,w2) ==
  let op   = VAL3(OPCODE(FETCH13 m w1)) in
  let addr = CUT16_13(FETCH13 m w1)     in
   (op=0 -> (m, w1, w2, T) |
    op=1 -> (m, addr, w2, F) |
    op=2 -> (VAL16 w2 = 0 -> (m, addr, w2, F) | (m, INC13 w1, w2, F)) |
    op=3 -> (m, INC13 w1, ADD16 w2 (FETCH13 m addr), F) |
    op=4 -> (m, INC13 w1, SUB16 w2 (FETCH13 m addr), F) |
    op=5 -> (m, INC13 w1, FETCH13 m addr, F) |
    op=6 -> (STORE13 addr w2 m, INC13 w1, w2, F) |
            (m, INC13 w1, w2, F))
```

1983: Proving a computer correct…



Simple micro-programmed implementation

1983: Tactics and tacticals…

LCF includes the predefined ML types

```
lettype proof = thm list -> thm;;
lettype goal = form list # form;;
lettype tactic = goal -> ((goal list) # proof);;
```

Note that the function dest_thm has type "thm->goal".

form = formula (term)

Setting the initial goal

```
set_goal: goal -> void
```

Expanding the current goal

```
expand: tactic -> void
```

Applies the tactic, validated using VALID, to the goal. Prints resulting subgoals. If there are none, applies the validation, and those above it whose subgoals have been proved. Prints the resulting theorems.

Fast expand (does not apply VALID)

```
expandf: tactic -> void
```

Printing n levels of the goal stack

```
print_state: int -> void
```

Saving the topmost theorem (onto the theory file)

```
save_top_thm: token -> thm
```

Rotating the current subgoals (by n steps)

```
rotate: int -> void
```


1986

- UCAM-CL-TR-100: Jeff Joyce, Graham Birtwistle, Mike Gordon, *"Proving a computer correct in higher order logic"*, 1986

  “We have subsequently redone this example in higher order logic using the HOL hardware verification system.”

- Led to Tamarack (`examples/hardware`) and later Viper

1988

- UCAM-CL-TR-133: Larry Paulson, *"A Preliminary User's Manual for Isabelle"*, 1988

  "Although the theorem prover Isabelle is still far from finished, there are enough intrepid users to justify the effort of writing a manual"

- UCAM-CL-TR-104: Avra Cohn, *"A proof of correctness of the Viper microprocessor: the first level"*, 1988

- UCAM-CL-TR-134: Avra Cohn, *"Correctness properties of the Viper black model: the second level"*, 1988

Late 1990s

- UCAM-CL-TR-408: John Harrison, *"Theorem proving with the real numbers"*, 1997
- UCAM-CL-TR-428: John Harrison, *"Floating point verification in HOL Light: the exponential function"*, 1997

- UCAM-CL-TR-421: Michael Norrish, *"An abstract dynamic semantics for C"*, 1997
- UCAM-CL-TR-453: Michael Norrish, *"C formalised in HOL"*, 1998

2000s

- UCAM-CL-TR-545: Anthony Fox, *"A HOL specification of the ARM instruction set architecture"*, 2001
- UCAM-CL-TR-548: Anthony Fox, *"Formal verification of the ARM6 micro-architecture"*, 2002
- UCAM-CL-TR-765: Magnus Myreen, *"Formal verification of machine-code programs"*, 2009
- UCAM-CL-TR-879: Ramana Kumar, *"Self-compilation and self-verification"*, 2016

arm

Part 4. Athabasca (HOL98) to Trindemossen-1

A nostalgic trawl through the release notes

HOL Development

The Description provides a summary of the “four”

1. **HOL88**: Lisp substrate, with Classic ML
2. **HOL90**: Port to Standard ML, principally using the SML/NJ (first released in 1988)
3. **HOL98**: Athabasca and Taupo
4. **HOL4**: Kananaskis onwards

Aside: HOL98 development benefited from the PROSPER Toolkit project (Proof and Specification Assisted Design Environments: ESPRIT LTR Project 26241), which formally completed in 2001.

> *“The project was a collaboration between the Universities of Glasgow, Cambridge, Edinburgh, Tübingen and Karlsruhe, and the industrial partners IFAD and Prover Technology.” – Thomas Melham*

Michael Norrish and Konrand Slind were based at the University of Cambridge and working on HOL

Athabasca (Canadian river) and Taupo (New Zealand lake)

Version	Release Date (approximate)	Selected Highlights (beyond bug fixes)
Athabasca	April 1998	• Runs on MoscowML (better than SML/NJ) • New "oracle" tag • Theories redesigned: `*Theory.{sml,sig}` • Introduces `Holmake`
Taupo-1	December 1999	• Parser and pretty-printer completely re-implemented, e.g., gains records • `if-then-else` syntax • `Hol_datatype` • Simplifier • `computeLib` (with kernel change) • `MESON_TAC` • Presburger arithmetic: `numLib.ARITH_TAC`
Taupo-2	January 2000	Theory of lazy lists
Taupo-3	March 2000	• `intLib.COOPER_TAC` • Tracing
Taupo-4/5/6	July 2000	• Local/selectable term grammars • `Q.SPEC_THEN` • Probability theory

Kananaskis (Canadian river)

Version	Release Date (approximate)	Selected Highlights
Kananaskis-1	June 2002	• Code moved to SourceForge (SVN, previously CVS) • `Holmakefile` • Per-theory name spaces: `bool$/\` • Type abbreviations • `case-of` syntax • `intLib.OMEGA_TAC` (current `intLib.ARITH_TAC`) • `EVAL` (simpler interface to `computeLib`) • Stateful simpset (`srw_ss`) • `HolSatLib` (initially SATO, GRASP and ZCHAFF) • ARM6 verification example (`examples/arm/arm6-verification`)
Kananaskis-2	2004?	• Bug-fix release • Includes experimental “name-carrying” alternative kernel (from late 2002)
Kananaskis-3	July 2005	• Smart configure (`mosml < tools/smart-configure.sml`) • Location information for term/type errors • Improved abbreviations in goals (`Abbrev` marker) • `METIS_TAC`

Kananaskis

Version	Release Date (approximate)	Selected Highlights
Kananaskis-4	October 2006	• Support for literals in case expression patterns • Theory of rational numbers • Improvements to `HolSatLib` (now based on MiniSat) • New bit-vector theory (current `wordsTheory`)
Kananaskis-5	August 2009	• Support for Poly/ML • Overloading via patterns • Strings are now `:char list` • Unicode • `HolSmtLib` (oracle only)
Kananaskis-6	September 2010	• Pretty-printer uses colours • LaTeX munging (`EmitTeX` came a little earlier) • Vim mode • Support `do`-notation monadic syntax • Bit-blasting over bit-vectors

Kananaskis

Version	Release Date (approximate)	Selected Highlights
Kananaskis-7	August 2011	• Formalization of complex numbers • Move to GitHub?
Kananaskis-8	October 2012	Milawa theorem prover example (`examples/theorem-prover`)
Kananaskis-9	December 2013	• Tweaks to `Define` and compilation of pattern-matching in function definitions • New examples: parsing and ARM security properties (KTH)
Kananaskis-10	November 2014	• Improved `Datatype` syntax • New/improved floating-point point theory • `sptreeTheory`
Kananaskis-11	March 2017	• Heaps via Poly/ML's save state feature • Holyhammer
Kananaskis-12	June 2018	Concurrent builds with `Holmake`
Kananaskis-13	August 2019	New syntax: `Theorem <name>: .. Proof .. QED`, `Definition <name>: .. End`, etc.

Kananaskis

Version	Release Date	Selected Highlights
Kananaskis-14	February 2021	• Parallel builds across multiple directories • More top-level syntax: `Type` and `Overload` • `gs` • Induction improvements (e.g., new `using` infix)
Trindemossen-1	April 2024	• Use hidden directory: `.holobjs` • Updates to `REAL_ARITH` (port from HOL-Light) • `cv_compute`

25 years later

```
.
├── COPYRIGHT
├── README
├── bin
├── doc
├── examples
├── help
├── install.txt
├── sigobj
├── src
├── std.prelude
└── tools
```

```
src
├── 0
├── BoyerMoore
├── IndDef
├── arith
├── bag
├── basicHol90
├── bool
├── boss
├── combin
├── datatype
├── decision
├── finite_map
├── goalstack
├── ho_match
├── hol88
├── hol90
├── ind_def
├── integer
├── list
├── lite
├── meson
├── num
├── one
├── option
├── pair
├── parse
├── portableML
├── pred_set
├── q
├── quote-filter
├── real
├── reduce
├── refute
├── relation
├── res_quan
├── robdd
├── set
├── simp
├── string
├── sum
├── taut
├── tfl
├── tree
├── unwind
├── utils
└── word
```

```
examples
├── MLsyntax
├── README
├── autopilot.sml
├── bmark
├── euclid.sml
├── fol.sml
├── ind_def
├── taut.sml
└── tempScript.sml
```

HOL98: 16M

```
.
├── CONTRIBUTORS
├── COPYRIGHT
├── INSTALL
├── Manual
├── README.md
├── bin
├── developers
├── doc
├── examples
├── help
├── sigobj
├── src
├── std.prelude
├── tools
└── tools-poly
```

```
src
├── 0
├── 1
├── AI
├── Boolify
├── HolQbf
├── HolSat
├── HolSmt
├── IndDef
├── TeX
├── algebra
├── bag
├── basicProof
├── bool
├── boss
├── coalgebras
├── combin
├── compute
├── coretypes
├── datatype
├── emit
├── experimental-kernel
├── finite_maps
├── float
├── floating-point
├── hol88
├── holyhammer
├── integer
├── list
├── lite
├── marker
├── meson
├── metis
├── monad
├── n-bit
├── num
├── opentheory
├── parallel_builds
├── parse
├── pattern_matches
├── pfl
├── portableML
├── postkernel
├── pred_set
├── prekernel
├── probability
├── proofman
├── q
├── quantHeuristics
├── quotient
├── rational
├── real
├── refute
├── relation
├── res_quan
├── ring
├── search
├── simp
├── sort
├── string
├── tactictoe
├── taut
├── tfl
├── thm
├── transfer
├── unwind
└── update
```

```
examples
├── AI
├── AKS
├── CCS
├── Crypto
├── Hoare-for-divergence
├── HolBdd
├── HolCheck
├── MLsyntax
├── PSL
├── README
├── RL_Environment
├── STE
├── acl2
├── algebra
├── algorithms
├── arm
├── balanced_bst
├── bnf-datatypes
├── bootstrap
├── category
├── computability
├── countchars
├── cv_compute
├── decidable_separationLogic
├── dependability
├── dev
├── developers
├── diningcryptos
├── dpll.sml
├── elliptic
├── euclid.sml
├── fermat
├── finite-test-set
├── formal-languages
├── fun-op-sem
├── generic_finite_graphs
├── hardware
├── hfs
├── imperative
├── ind_def
├── l3-machine-code
├── lambda
├── logic
├── machine-code
├── miller
├── misc
├── muddy
├── padics
├── parity
├── pgcl
├── probability
├── rings
├── separationLogic
├── set-theory
├── simple_complexity
├── taut.sml
├── tempScript.sml
├── theorem-prover
├── vector
└── zipper
```

Trindemossen: 136M

arm

Closing Remarks

Closing Remarks

- Thanks to everyone!
- Many people have contributed to the development of HOL4 over the years
- I have always found the HOL4 community to be friendly and supportive
- Special thanks to
  - Hrutivk Kanabar for organizing this event
  - Our keynote speaker Michael Norrish
- Hope you all enjoy this event and your visit to Arm in Cambridge!

arm

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה
ధన్యవాదములు