Formal models of ARM processors in HOL

More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

Formal models of ARM processors in HOL

- Today's talk
  - very quick overview of ARM and HOL
  - some technical highlights and research challenges
  - making the ARM HOL models accessible (demo)

- General goals of the ARM-in-HOL project
  - accurately model a modern COTS processor
  - use processor model as basis for formal code verification
  - study difficult modelling challenges

- Acknowledgements
  - Graham Birtwistle, Joe Hurd, Guodong Li, Magnus Myreen, Scott Owens, Dominic Pajak, John Regehr, James Reynolds, Susmit Sarkar, Daniel Schostak, Peter Sewell, Konrad Slind, ...

Formal models of ARM processors in HOL

- Today's talk
  - very quick overview of ARM and HOL
  - some technical highlights and research challenges
  - making the ARM HOL models accessible (demo)

- General goals of the ARM-in-HOL project
  - accurately model a modern COTS processor
  - use processor model as basis for formal code verification
  - study difficult modelling challenges

- Acknowledgements
  - Graham Birtwistle, Joe Hurd, Guodong Li, Magnus Myreen, Scott Owens, Dominic Pajak, John Regehr, James Reynolds, Susmit Sarkar, Daniel Schostak, Peter Sewell, Konrad Slind, ...

Formal models of ARM processors in HOL

- Today's talk
  - very quick overview of ARM and HOL
  - some technical highlights and research challenges
  - making the ARM HOL models accessible (demo)

- General goals of the ARM-in-HOL project
  - accurately model a modern COTS processor
  - use processor model as basis for formal code verification
  - study difficult modelling challenges

- Acknowledgements
  - Graham Birtwistle, Joe Hurd, Guodong Li, Magnus Myreen, Scott Owens, Dominic Pajak, John Regehr, James Reynolds, Susmit Sarkar, Daniel Schostak, Peter Sewell, Konrad Slind, ...

More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

ARM processors and ARM-powered products

- Qualcomm Snapdragon
  - 1GHz design based on ARM Cortex-A8
  - Dell Mini 5 slate, Lenovo Skylight, Google Nexus One
- Texas Instruments OMAP3430
  - 550MHz design based on ARM Cortex-A8
  - Motorola Droid, Nokia N900, Palm Pre
- Nvidia Tegra 2
  - 1GHz design based on dual-core ARM Cortex-A9
  - Asus Eee Pad, Notion Ink, Viewsonic, T-Mobile phone
- Marvell Armada 610
  - 1.2GHz based on ARM Cortex-A8, but shorter pipeline
  - Sampling to customers

ARM processors and ARM-powered products

- Qualcomm Snapdragon
  - 1GHz design based on ARM Cortex-A8
  - Dell Mini 5 slate, Lenovo Skylight, Google Nexus One
- Texas Instruments OMAP3430
  - 550MHz design based on ARM Cortex-A8
  - Motorola Droid, Nokia N900, Palm Pre
- Nvidia Tegra 2
  - 1GHz design based on dual-core ARM Cortex-A9
  - Asus Eee Pad, Notion Ink, Viewsonic, T-Mobile phone
- Marvell Armada 610
  - 1.2GHz based on ARM Cortex-A8, but shorter pipeline
  - Sampling to customers

15 Billion processors created; 10 million shipped every day

[From: `http://www.arm.com/markets/showcase/`]

Higher order logic and the HOL4 system

- Higher order logic intended to formalise mathematics
  - very expressive – can represent all mathematics
  - undecidable – decision procedures for fragments
- HOL4 used in the ARM formalisation project
  - accident of history - could have used other systems
  - Isabelle/HOL, HOL Light, ProofPower use similar logics
- ARM model in first order logic?
  - a challenging case study for the ACL2-HOL link
  - ARM VFP modeled by Reynolds in HOL and ACL2

More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

Higher order logic and the HOL4 system

- Higher order logic intended to formalise mathematics
  - very expressive – can represent all mathematics
  - undecidable – decision procedures for fragments
- HOL4 used in the ARM formalisation project
  - accident of history - could have used other systems
  - Isabelle/HOL, HOL Light, ProofPower use similar logics
- ARM model in first order logic?
  - a challenging case study for the ACL2-HOL link
  - ARM VFP modeled by Reynolds in HOL and ACL2

Higher order logic and the HOL4 system

- Higher order logic intended to formalise mathematics
  - very expressive – can represent all mathematics
  - undecidable – decision procedures for fragments
- HOL4 used in the ARM formalisation project
  - accident of history - could have used other systems
  - Isabelle/HOL, HOL Light, ProofPower use similar logics
- ARM model in first order logic?
  - a challenging case study for the ACL2-HOL link
  - ARM VFP modeled by Reynolds in HOL and ACL2

More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

Why create formal models of ARM?

- Ultimate goal:
  - high confidence verification of real ARM code
- Need accurate specification of instruction execution
  - ARM supplies 2000 page document in English
- ARM ISA formalised in higher order logic
  - semantics of software is given by hardware execution
- Acknowledgement: pioneering CLI Stack
  - ARM project has similar goals ... but with COTS hardware

Why create formal models of ARM?

- Ultimate goal:
  - high confidence verification of real ARM code
- Need accurate specification of instruction execution
  - ARM supplies 2000 page document in English
- ARM ISA formalised in higher order logic
  - semantics of software is given by hardware execution
- Acknowledgement: pioneering CLI Stack
  - ARM project has similar goals ... but with COTS hardware

Why create formal models of ARM?

- Ultimate goal:
  - high confidence verification of real ARM code
- Need accurate specification of instruction execution
  - ARM supplies 2000 page document in English
- ARM ISA formalised in higher order logic
  - semantics of software is given by hardware execution
- Acknowledgement: pioneering CLI Stack
  - ARM project has similar goals ... but with COTS hardware

More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

Why create formal models of ARM?

- Ultimate goal:
  - high confidence verification of real ARM code
- Need accurate specification of instruction execution
  - ARM supplies 2000 page document in English
- ARM ISA formalised in higher order logic
  - semantics of software is given by hardware execution
- Acknowledgement: pioneering CLI Stack
  - ARM project has similar goals ... but with COTS hardware

More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

Challenges

- How can one be sure the model is correct
  - ARM documentation is voluminous and informal
- IP problems with ARM
  - microarchitectures, tests not easily available to academics
- Several versions of ARM instruction set architecture
  - ARM7 in old phones, Cortex-A8 in smartphones, tablets
- Many modelling research challenges
  - *e.g.* no standard specification of weak memory behaviour

More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

Challenges

- How can one be sure the model is correct
  - ARM documentation is voluminous and informal
- IP problems with ARM
  - microarchitectures, tests not easily available to academics
- Several versions of ARM instruction set architecture
  - ARM7 in old phones, Cortex-A8 in smartphones, tablets
- Many modelling research challenges
  - *e.g.* no standard specification of weak memory behaviour

Challenges

- How can one be sure the model is correct
  - ARM documentation is voluminous and informal
- IP problems with ARM
  - microarchitectures, tests not easily available to academics
- Several versions of ARM instruction set architecture
  - ARM7 in old phones, Cortex-A8 in smartphones, tablets
- Many modelling research challenges
  - *e.g.* no standard specification of weak memory behaviour

Challenges

- How can one be sure the model is correct
  - ARM documentation is voluminous and informal
- IP problems with ARM
  - microarchitectures, tests not easily available to academics
- Several versions of ARM instruction set architecture
  - ARM7 in old phones, Cortex-A8 in smartphones, tablets
- Many modelling research challenges
  - *e.g.* no standard specification of weak memory behaviour

How can we be sure the model is accurate?

- First approach: prove ISA and hardware models consistent
  - proved ARMv3 ISA consistent with ARM6 machine
  - ARM6 used in Apple Newton
  - modern microarchitectures not available to us
  - Cortex-A8 proof estimated 10x ARMv3 effort

- Second approach: validate by testing against hardware
  - compare deduction in model with execution on ARM

Board	Core
Olimex LPC-2129P	ARM7TDMI-S
Atmel SAM3U-EK	ARM Cortex-M3
TI OMAP3530 Beagle Board	ARM Cortex-A8

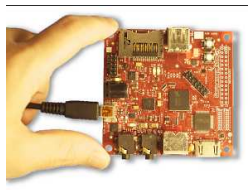
  - can validate different ISA versions
  - less/different assurance than proof
  - ARM's test suites not available to us

How can we be sure the model is accurate?

- First approach: prove ISA and hardware models consistent
  - proved ARMv3 ISA consistent with ARM6 machine
  - ARM6 used in Apple Newton
  - modern microarchitectures not available to us
  - Cortex-A8 proof estimated 10x ARMv3 effort

- Second approach: validate by testing against hardware
  - compare deduction in model with execution on ARM

Board	Core
Olimex LPC-2129P	ARM7TDMI-S
Atmel SAM3U-EK	ARM Cortex-M3
TI OMAP3530 Beagle Board	ARM Cortex-A8



  - can validate different ISA versions
  - less/different assurance than proof
  - ARM's test suites not available to us

Single-step theorems

- Single-step theorems derived by instantiating ARM model
  - describe execution of a single instruction
  - derived from ISA-specific configuration
- Executing models for testing
  - single-step theorems generated on-the-fly in HOL4
  - execute by deduction or translation to ML
- Deriving Hoare triples for code verification
  - Hoare triples derived from single-step theorems
  - derived triples basis for formal code verification

Single-step theorems

- Single-step theorems derived by instantiating ARM model
  - describe execution of a single instruction
  - derived from ISA-specific configuration
- Executing models for testing
  - single-step theorems generated on-the-fly in HOL4
  - execute by deduction or translation to ML
- Deriving Hoare triples for code verification
  - Hoare triples derived from single-step theorems
  - derived triples basis for formal code verification

Single-step theorems

- Single-step theorems derived by instantiating ARM model
  - describe execution of a single instruction
  - derived from ISA-specific configuration
- Executing models for testing
  - single-step theorems generated on-the-fly in HOL4
  - execute by deduction or translation to ML
- Deriving Hoare triples for code verification
  - Hoare triples derived from single-step theorems
  - derived triples basis for formal code verification

Formal verification of ARM code (Myreen)

- Semantic basis is derived Hoare logic for machine code
  - *e.g.* `E0834004` adds `r3` to `r4` – derived Hoare triple is:

$$\{\ \text{r3}\ a * \text{r4}\ b * \text{pc}\ p\ \}$$
$$p : \texttt{E0834004}$$
$$\{\ \text{r3}\ a * \text{r4}\ (a + b) * \text{pc}\ (p + 4)\ \}$$

- A decompiler extracts functional descriptions from code
  - code below calculates length of a list

```
 0: E3A00000      mov r0, #0          ; set reg 0 to 0
 4: E3510000   L: cmp r1, #0          ; compare reg 1 with 0
 8: 12800001      addne r0, r0, #1    ; if not equal: add 1 to reg 1
12: 15911000      ldrne r1, [r1]      ; load mem[reg 1] into reg 1
16: 1AFFFFFB      bne L               ; jump to compare
```

  - automatically decompiles to function definitions in HOL

$$\text{f}(r_0, r_1, m) = \text{let}\ r_0 = 0\ \text{in}\ \text{g}(r_0, r_1, m)$$

$$\text{g}(r_0, r_1, m) = \text{if}\ r_1 = 0\ \text{then}\ (r_0, r_1, m)\ \text{else let}\ r_0 = r_0{+}1\ \text{in let}\ r_1 = m(r_1)\ \text{in}\ \text{g}(r_0, r_1, m)$$

- Automatically derive a Hoare triple from decompiled code:

$$\{\ (\text{r0}, \text{r1}, \text{m})\ \text{is}\ (r_0, r_1, m) * \text{s} * \text{pc}\ p * \langle \text{f}_{\text{pre}}(r_0, r_1, m) \rangle\ \}$$
$$p : \texttt{E3A00000}, \texttt{E3510000}, \texttt{12800001}, \texttt{15911000}, \texttt{1AFFFFFB}$$
$$\{\ (\text{r0}, \text{r1}, \text{m})\ \text{is}\ (\text{f}\ (r_0, r_1, m)) * \text{s} * \text{pc}\ (p + 20)\ \}$$

More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

Formal verification of ARM code (Myreen)

- Semantic basis is derived Hoare logic for machine code
  - *e.g.* `E0834004` adds `r3` to `r4` – derived Hoare triple is:

$$\{\ \text{r3}\ a * \text{r4}\ b * \text{pc}\ p\ \}$$
$$p : \texttt{E0834004}$$
$$\{\ \text{r3}\ a * \text{r4}\ (a + b) * \text{pc}\ (p + 4)\ \}$$

- A decompiler extracts functional descriptions from code
  - code below calculates length of a list

```
 0: E3A00000      mov r0, #0          ; set reg 0 to 0
 4: E3510000   L: cmp r1, #0          ; compare reg 1 with 0
 8: 12800001      addne r0, r0, #1    ; if not equal: add 1 to reg 1
12: 15911000      ldrne r1, [r1]      ; load mem[reg 1] into reg 1
16: 1AFFFFFB      bne L               ; jump to compare
```

  automatically decompiles to function definitions in HOL

$$\text{f}(r_0, r_1, m) = \text{let}\ r_0 = 0\ \text{in}\ \text{g}(r_0, r_1, m)$$

$$\text{g}(r_0, r_1, m) = \text{if}\ r_1 = 0\ \text{then}\ (r_0, r_1, m)\ \text{else let}\ r_0 = r_0{+}1\ \text{in let}\ r_1 = m(r_1)\ \text{in}\ \text{g}(r_0, r_1, m)$$

- Automatically derive a Hoare triple from decompiled code:

$$\{\ (\text{r0}, \text{r1}, \text{m})\ \text{is}\ (r_0, r_1, m) * \text{s} * \text{pc}\ p * \langle \text{f}_{\text{pre}}(r_0, r_1, m) \rangle\ \}$$
$$p : \texttt{E3A00000}, \texttt{E3510000}, \texttt{12800001}, \texttt{15911000}, \texttt{1AFFFFFB}$$
$$\{\ (\text{r0}, \text{r1}, \text{m})\ \text{is}\ (\text{f}\ (r_0, r_1, m)) * \text{s} * \text{pc}\ (p + 20)\ \}$$

More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

Verifying compilation (Myreen)

- From

  $$\mathsf{f}(r_1) \quad = \quad \text{if } r_1 < 10 \text{ then } r_1 \text{ else let } r_1 = r_1 - 10 \text{ in } \mathsf{f}(r_1)$$

  the compiler generates code and proves that it calculates f

  $\{\mathsf{r1}\ r_1 * \mathsf{pc}\ p * \mathsf{s}\}$
  p : `E351000A, 2241100A, 2AFFFFFC`
  $\{\mathsf{r1}\ \mathsf{f}(r_1) * \mathsf{pc}\ (p{+}12) * \mathsf{s}\}$

- A separate proof establishes $\forall x.\ \mathsf{f}(x) = x \bmod 10$, hence

  $\{\mathsf{r1}\ r_1 * \mathsf{pc}\ p * \mathsf{s}\}$
  p : `E351000A, 2241100A, 2AFFFFFC`
  $\{\mathsf{r1}\ (r_1 \bmod 10) * \mathsf{pc}\ (p{+}12) * \mathsf{s}\}$

  which can be used for subsequent synthesis

- Acknowledgement: collaboration with Slind, Owens & Li

Verifying compilation (Myreen)

- From

  $$\mathsf{f}(r_1) \quad = \quad \text{if } r_1 < 10 \text{ then } r_1 \text{ else let } r_1 = r_1 - 10 \text{ in } \mathsf{f}(r_1)$$

  the compiler generates code and proves that it calculates f

  $\{\mathsf{r1}\ r_1 * \mathsf{pc}\ p * \mathsf{s}\}$
  p : `E351000A, 2241100A, 2AFFFFFC`
  $\{\mathsf{r1}\ \mathsf{f}(r_1) * \mathsf{pc}\ (p{+}12) * \mathsf{s}\}$

- A separate proof establishes $\forall x.\ \mathsf{f}(x) = x \bmod 10$, hence

  $\{\mathsf{r1}\ r_1 * \mathsf{pc}\ p * \mathsf{s}\}$
  p : `E351000A, 2241100A, 2AFFFFFC`
  $\{\mathsf{r1}\ (r_1 \bmod 10) * \mathsf{pc}\ (p{+}12) * \mathsf{s}\}$

  which can be used for subsequent synthesis

- Acknowledgement: collaboration with Slind, Owens & Li

More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

Verifying compilation (Myreen)

- From

  $$\mathsf{f}(r_1) \quad = \quad \text{if } r_1 < 10 \text{ then } r_1 \text{ else let } r_1 = r_1 - 10 \text{ in } \mathsf{f}(r_1)$$

  the compiler generates code and proves that it calculates f

  $\{\mathsf{r1}\ r_1 * \mathsf{pc}\ p * \mathsf{s}\}$
  p : `E351000A, 2241100A, 2AFFFFFC`
  $\{\mathsf{r1}\ \mathsf{f}(r_1) * \mathsf{pc}\ (p{+}12) * \mathsf{s}\}$

- A separate proof establishes $\forall x.\ \mathsf{f}(x) = x \bmod 10$, hence

  $\{\mathsf{r1}\ r_1 * \mathsf{pc}\ p * \mathsf{s}\}$
  p : `E351000A, 2241100A, 2AFFFFFC`
  $\{\mathsf{r1}\ (r_1 \bmod 10) * \mathsf{pc}\ (p{+}12) * \mathsf{s}\}$

  which can be used for subsequent synthesis

- Acknowledgement: collaboration with Slind, Owens & Li

More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

Code verification: achievements and goals

- Tiny Lisp interpreter (Myreen)
  - runs on bare metal
  - verified against an independent Lisp language semantics
  - verified garbage collector; tail-call optimisation
  - large word arithmetic for crypto applications (in progress)
  - IO handled by host platform (see Nintendo DS demo)
- Current activity
  - better treatment of IO and exceptions (with John Regehr)
  - realistic multi-core memory (with Peter Sewell)
- Long term goal
  - completely verified useful FP system
  - more work than we can do ... collaborators welcomed!
  - want examples of need for verified FP implementations
- Related non-ARM work at Cambridge
  - Lisp interpreter also runs on x86 and PowerPC models
  - x86 and PowerPC models less complete than ARM
  - Broadcom FirePath processor code equivalence via SMT
  - verify compilation Clight $\rightarrow$ x86 + weak memory semantics

More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

Code verification: achievements and goals

- Tiny Lisp interpreter (Myreen)
  - runs on bare metal
  - verified against an independent Lisp language semantics
  - verified garbage collector; tail-call optimisation
  - large word arithmetic for crypto applications (in progress)
  - IO handled by host platform (see Nintendo DS demo)
- Current activity
  - better treatment of IO and exceptions (with John Regehr)
  - realistic multi-core memory (with Peter Sewell)
- Long term goal
  - completely verified useful FP system
  - more work than we can do ... collaborators welcomed!
  - want examples of need for verified FP implementations
- Related non-ARM work at Cambridge
  - Lisp interpreter also runs on x86 and PowerPC models
  - x86 and PowerPC models less complete than ARM
  - Broadcom FirePath processor code equivalence via SMT
  - verify compilation Clight $\rightarrow$ x86 + weak memory semantics

More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

Code verification: achievements and goals

- Tiny Lisp interpreter (Myreen)
  - runs on bare metal
  - verified against an independent Lisp language semantics
  - verified garbage collector; tail-call optimisation
  - large word arithmetic for crypto applications (in progress)
  - IO handled by host platform (see Nintendo DS demo)
- Current activity
  - better treatment of IO and exceptions (with John Regehr)
  - realistic multi-core memory (with Peter Sewell)
- Long term goal
  - completely verified useful FP system
  - more work than we can do ... collaborators welcomed!
  - want examples of need for verified FP implementations
- Related non-ARM work at Cambridge
  - Lisp interpreter also runs on x86 and PowerPC models
  - x86 and PowerPC models less complete than ARM
  - Broadcom FirePath processor code equivalence via SMT
  - verify compilation Clight $\longrightarrow$ x86 + weak memory semantics

More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

Code verification: achievements and goals

- Tiny Lisp interpreter (Myreen)
  - runs on bare metal
  - verified against an independent Lisp language semantics
  - verified garbage collector; tail-call optimisation
  - large word arithmetic for crypto applications (in progress)
  - IO handled by host platform (see Nintendo DS demo)
- Current activity
  - better treatment of IO and exceptions (with John Regehr)
  - realistic multi-core memory (with Peter Sewell)
- Long term goal
  - completely verified useful FP system
  - more work than we can do ... collaborators welcomed!
  - want examples of need for verified FP implementations
- Related non-ARM work at Cambridge
  - Lisp interpreter also runs on x86 and PowerPC models
  - x86 and PowerPC models less complete than ARM
  - Broadcom FirePath processor code equivalence via SMT
  - verify compilation Clight $\rightarrow$ x86 + weak memory semantics

Code verification: achievements and goals

- Tiny Lisp interpreter (Myreen)
  - runs on bare metal
  - verified against an independent Lisp language semantics
  - verified garbage collector; tail-call optimisation
  - large word arithmetic for crypto applications (in progress)
  - IO handled by host platform (see Nintendo DS demo)
- Current activity
  - better treatment of IO and exceptions (with John Regehr)
  - realistic multi-core memory (with Peter Sewell)
- Long term goal
  - completely verified useful FP system
  - more work than we can do ... collaborators welcomed!
  - want examples of need for verified FP implementations
- Related non-ARM work at Cambridge
  - Lisp interpreter also runs on x86 and PowerPC models
  - x86 and PowerPC models less complete than ARM
  - Broadcom FirePath processor code equivalence via SMT
  - verify compilation Clight $\rightarrow$ x86 + weak memory semantics

More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

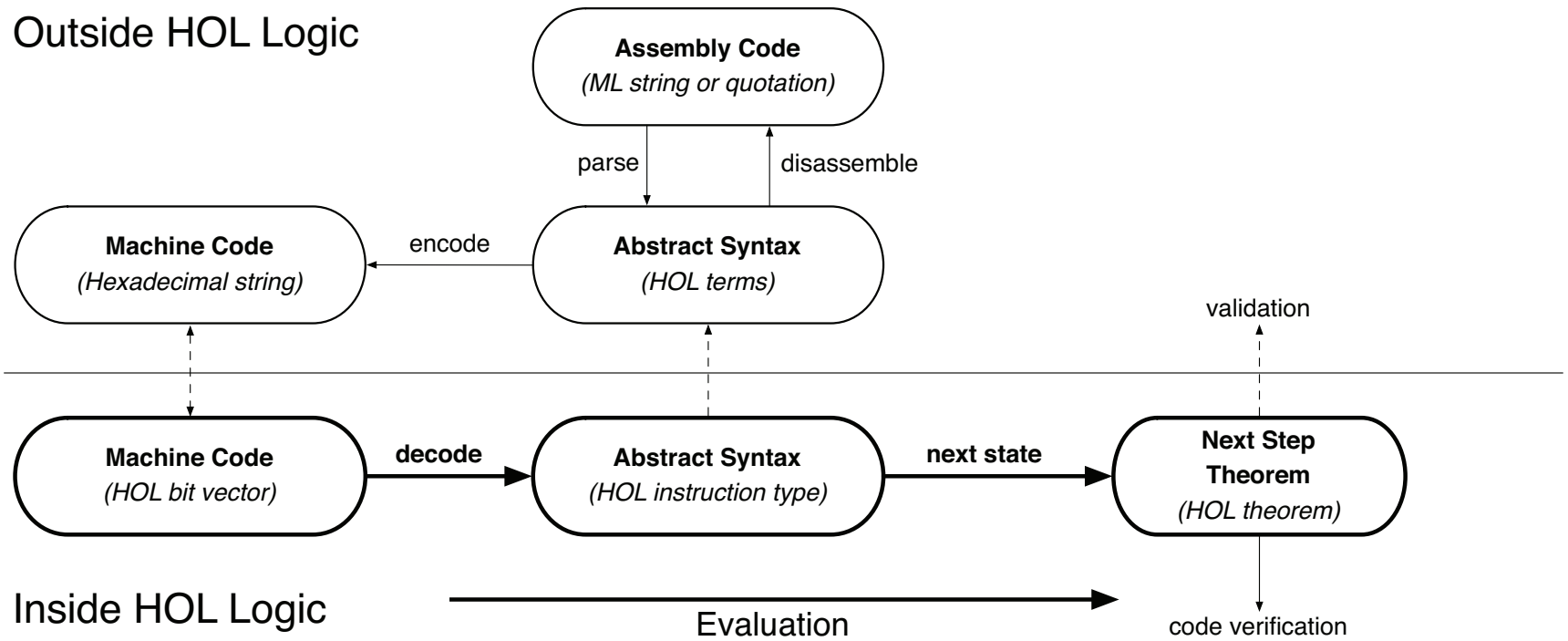
Switch to Anthony ...

Background to current ARMv7 model

- Work on ARMv7 model started towards the end of 2008.
- **Monadic model** — initially requested by Peter Sewell's group (Cambridge).
- Suited to their work on formalizing weak memory models.
  - Cortex-A9 has **multi-core** configurations with **shared memory**.
  - Can reason about the order of register and memory accesses.
- Now used by Magnus Myreen for his code verification work.

Formalization

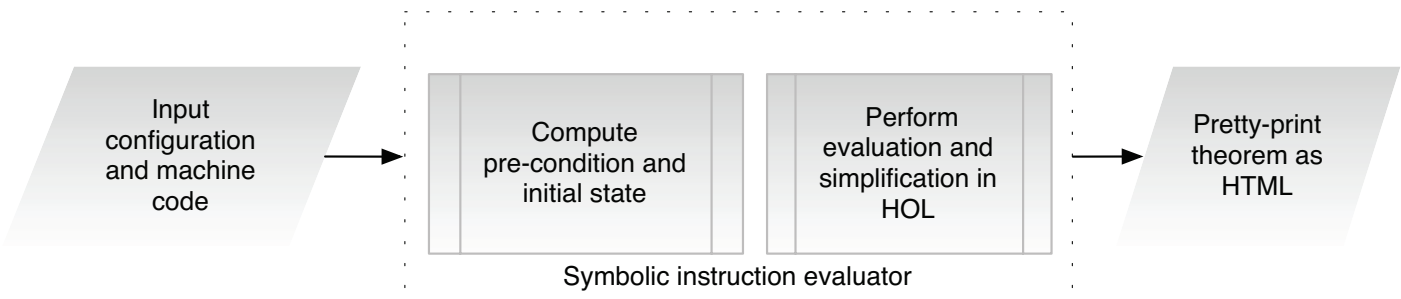
- Around 24 thousand lines of Standard ML and HOL code. Just under half is HOL definitions and proofs. Only critical parts reside in HOL logic.



More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

Web Interface (demo)

- A web interface has been built on top of the instruction evaluator.
- **User friendly** input and output. **Hides complexity** of model.
- Easy way to query the model — avoids trawling through the HOL specification.
- Encourage students and formal methods community to use the model.
- Based on CGI scripting: a HOL4 session generates HTML output.



More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**

Summary

- **Large** HOL specification — all standard **ARMv7** instructions covered.
- **Validated** against ARM development boards.
- Instruction evaluator and web interface provides **easy access to the model**.
- Has many uses:
  - Formalizing **shared memory models**.
  - Hardware, system code and compiler **verification**.
- Further collaborations welcomed.
- **Questions?**

More details at: **http://www.cl.cam.ac.uk/~acjf3/arm/**