Verifying Multiple Compare and Swap

Joe Hurd
joe.hurd@cl.cam.ac.uk

University of Cambridge
Introduction

This work shows how a parallel architecture can be formalized in HOL, with many processors acting on a shared memory.

Relevant to possible future work of the ARM project (individual processors could be retargeted to ARM instruction set).
The Formalization Task

Create a logical model of a parallel architecture, supporting:

- independent execution of many processors communicating only through a shared memory; ✓
- arbitrary interleaving of instructions at the granularity of the primitive atomic operations; ✓
- plugging-in different memory models, with different rules for access reordering and barrier instructions; ✓
- automatic tools and techniques for specification and verification of concurrent programs. ~current work
Parallel Architecture: Overview

- Identify a processor with its register file:

  \[ \text{processor} = \text{register} \rightarrow \text{value} \]

- The whole parallel architecture can then be modelled by a list of processors and a shared memory:

  \[ \text{architecture} = \text{memory} \times \text{processor list} \]

- The global state advances by any non-halted processor executing a primitive atomic instruction.
Parallel Architecture: Instruction Set

An instruction corresponds to a primitive atomic operation:

\[
\text{instruction} = \begin{cases} 
\text{NOP} \\ 
\text{UP of } \text{value} \times \text{register} \\ 
\text{RD of } \text{register} \times \text{register} \\ 
\text{WR of } \text{register} \times \text{register} \\ 
\text{CAS of } \text{register} \times \text{register} \times \text{register} \times \text{register} \\ 
\text{ALLOC of } \mathbb{N} \times \text{register} \\ 
\text{MB} 
\end{cases}
\]

Some memory models (such as the Alpha) use memory barrier (MB) instructions to impose an ordering on accesses.
The CAS(n) Algorithm

- A case study to investigate verification in our model.
- CAS(n) stands for Multiple Compare And Swap:
  
  \[
  \text{if the } n \text{ memory addresses } a_1, \ldots, a_n \\
  \text{contain the expected values } x_1, \ldots, x_n \\
  \text{then replace them with the values } y_1, \ldots, y_n \\
  \text{(and this must all happen atomically!)}
  \]

- A fast CAS(n) algorithm was recently developed by Harris, Fraser and Pratt in Cambridge.

- So far has led to:
  - instruction macros to simplify programming;
  - and a simulation engine to test arbitrary interleaving.
Conclusion

- We have laid out a formalization of a parallel architecture in HOL.
- We’re working on case studies to develop techniques for specification and verification for a simple instantiation.
- There’s scope for making the memory model more realistic (c.f. Gordon’s model of the Alpha architecture), and also the processor instruction set (c.f. Fox’s model of the ARM).
- The intention is that proofs completed for the simple instantiation lift up to more realistic models (we’ll see).