NICTA

Half talk: Translation Validation for seL4

Thomas Sewell Magnus Myreen

NICTA & Cambridge

10 May 2013

Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Funding and Supporting Members and Partners

Australian National University

Trade & Investment

THE UNIVERSITY OF SYDNEY

Queensland Government

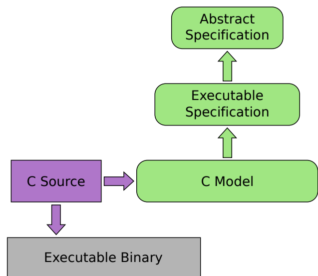
Griffith UNIVERSITY

QUT

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

Copyright NICTA 2013

Proof Composition

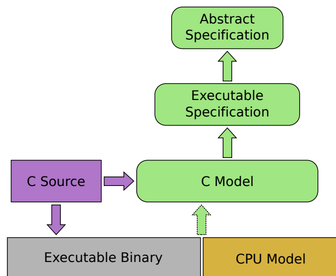


We want to link the seL4 proofs down to the binary, giving the proof deeper foundations.

Copyright NICTA 2013

Proof Composition

NICTA



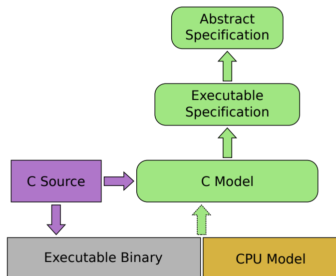
We want to link the seL4 proofs down to the binary, giving the proof deeper foundations.

We link to the Cambridge ARM model, which is extensively validated. This brings our theory about as close to the real world as we can go.

Copyright NICTA 2013

Proof Composition

NICTA



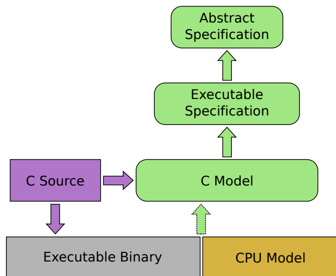
We want to link the seL4 proofs down to the binary, giving the proof deeper foundations.

We link to the Cambridge ARM model, which is extensively validated. This brings our theory about as close to the real world as we can go.

This guards against the compiler being broken, the C semantics being wrong, or the standard being weak.

Copyright NICTA 2013

Comparisons

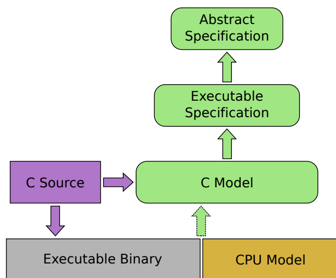


There are similarities to previous really big projects like Verisoft.

Copyright NICTA 2013

Comparisons

NICTA

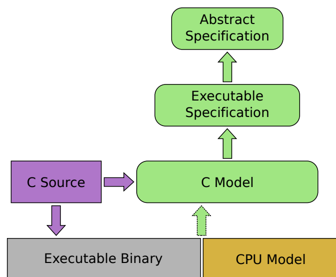


There are similarities to previous really big projects like Verisoft.

There is also a clear difference: this proof is happening incrementally.

Copyright NICTA 2013

Comparisons



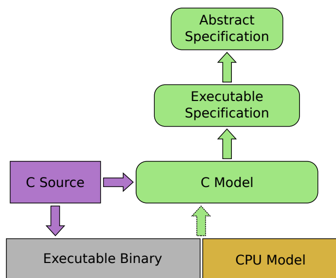
There are similarities to previous really big projects like Verisoft.

There is also a clear difference: this proof is happening incrementally.

Also note this is only half the binary verification issue for seL4.

Copyright NICTA 2013

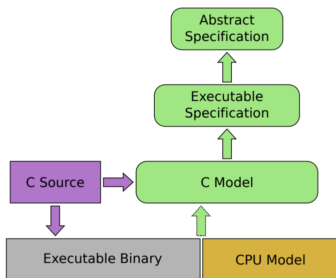
Context



This is **translation validation**, a form of **refinement**.

Copyright NICTA 2013

Context



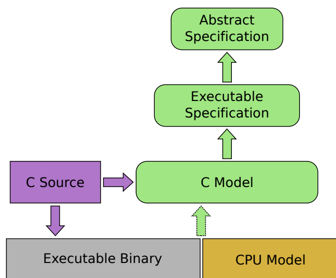
This is **translation validation**, a form of **refinement**.

Similar to certified compilation, certifying compilation, binary verification and proof carrying code.

Copyright NICTA 2013

Context

NICTA



This is **translation validation**, a form of **refinement**.

Similar to certified compilation, certifying compilation, binary verification and proof carrying code.

There's a lot of other work in this space. All that really distinguishes us is our motivation.

Copyright NICTA 2013

Motivation

NICTA

Motivation: We care about getting a result for one system and proof.

Copyright NICTA 2013

Motivation

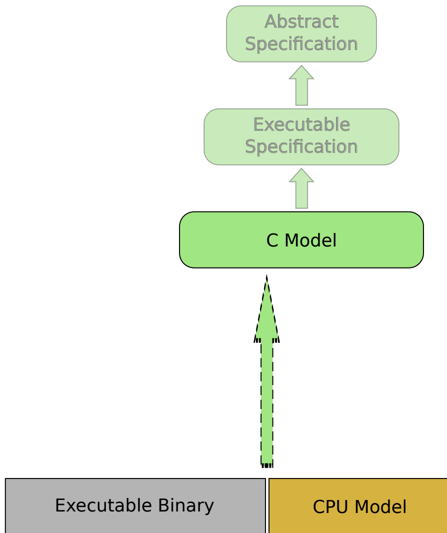
NICTA

Motivation: We care about getting a result for one system and proof. Period.

We don't care about performance, coverage of the C language or of C compiler optimisations. We don't care about gcc.

Copyright NICTA 2013

Approach

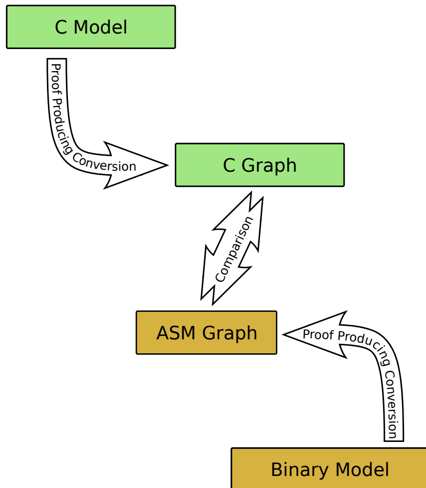


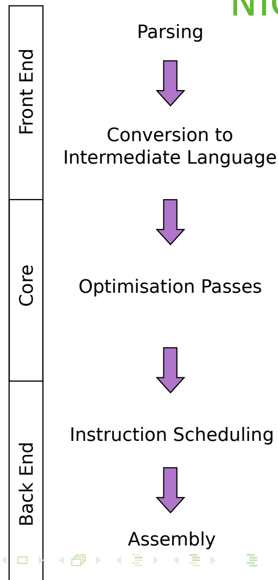
Copyright NICTA 2013

Approach

Copyright NICTA 2013

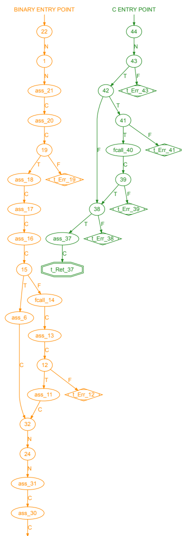
Approach





Copyright NICTA 2013

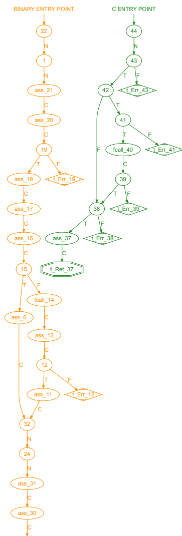
Proving Graph Refinement



The big challenge is the inner graph refinement.
This is proven one function at a time.

Copyright NICTA 2013

Proving Graph Refinement



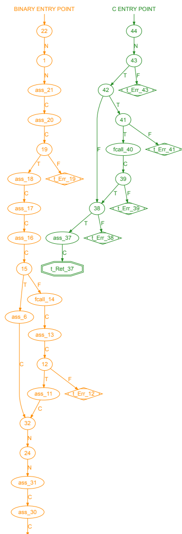
The big challenge is the inner graph refinement.
This is proven one function at a time.

Proven by:

- Implementing compiler-like transforms.

Copyright NICTA 2013

Proving Graph Refinement



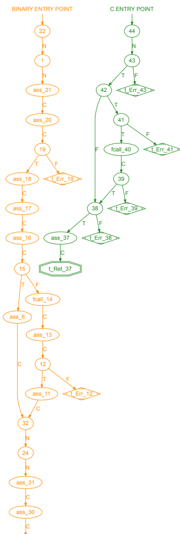
The big challenge is the inner graph refinement.
This is proven one function at a time.

Proven by:

- Implementing compiler-like transforms. ✖

Copyright NICTA 2013

Proving Graph Refinement



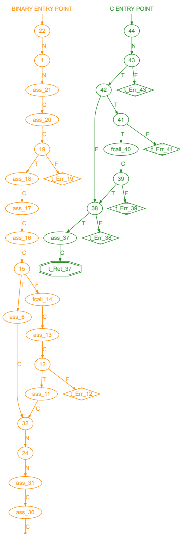
The big challenge is the inner graph refinement.
This is proven one function at a time.

Proven by:

- Implementing compiler-like transforms. ✖
- Showing equivalences one basic block at a time.

Copyright NICTA 2013

Proving Graph Refinement



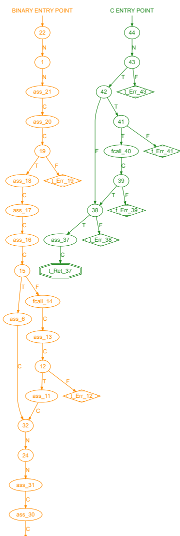
The big challenge is the inner graph refinement. This is proven one function at a time.

Proven by:

- Implementing compiler-like transforms. ✖
- Showing equivalences one basic block at a time. ✖

Copyright NICTA 2013

Proving Graph Refinement



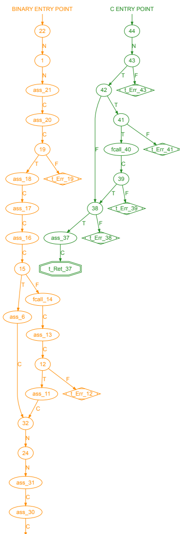
The big challenge is the inner graph refinement. This is proven one function at a time.

Proven by:

- Implementing compiler-like transforms. ✖
- Showing equivalences one basic block at a time. ✖
- Conversion of whole problems to SMT

Copyright NICTA 2013

Proving Graph Refinement



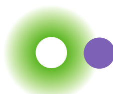
The big challenge is the inner graph refinement. This is proven one function at a time.

Proven by:

- Implementing compiler-like transforms. ✖
- Showing equivalences one basic block at a time. ✖
- Conversion of whole problems to SMT ✔ modulo cycles.

Copyright NICTA 2013

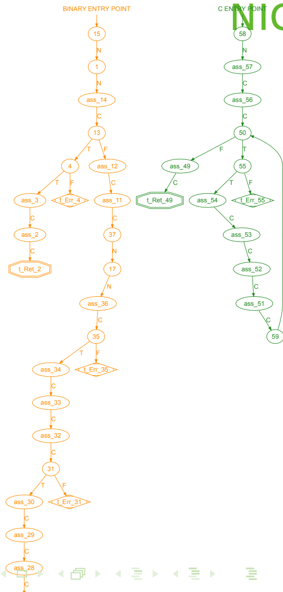
What about cycles?



What about cycles?

We have two approaches:

1. Discover a loop bound.
2. Perform split point induction.



Copyright NICTA 2013

Challenge 1

Copyright NICTA 2013

Challenges

NICTA

Challenges:

- Inlining & problem size.
- Counterexample size.
- Finding split induction parameters.
- Functions marked `const` or `pure`.
- Partiality from C standard, binary semantics, decompiler.
- SMT theory extension for C standard symbols.
- Special memory regions:
  - Pointer memory regions (types matter for strict-aliasing).
  - Global objects.
  - ELF sections `.rodata` `.text` etc.
  - Usable Memory.

Copyright NICTA 2013

Results & Conclusion

NICTA

Results: Works for previously-verified seL4 code with gcc-4.5.1 -O1.

Nested loops and higher optimisation levels not yet handled.

Copyright NICTA 2013

Results & Conclusion

Results: Works for previously-verified seL4 code with gcc-4.5.1 -O1.

Nested loops and higher optimisation levels not yet handled.

Conclusion: It is possible to build a certified compilation environment out of gcc, SMT and tape.

Copyright NICTA 2013