Designing Systems for Push-Button Verification

Luke Nelson

Joint work with James Bornholt, Dylan Johnson,

Helgi Sigurbjarnarson, Emina Torlak, Xi Wang, Kaiyuan Zhang

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

OSes are everywhere

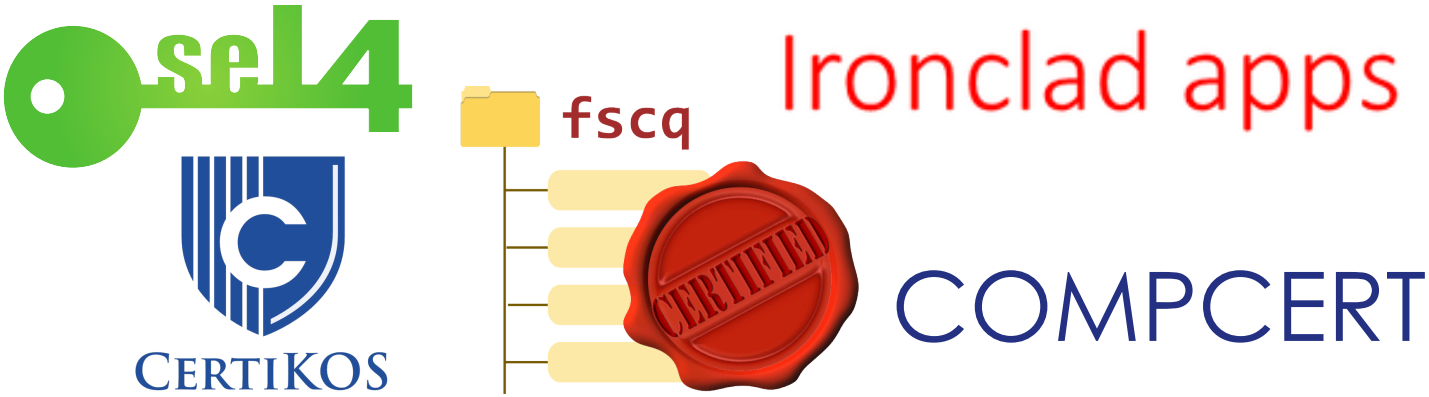
OSes (& bugs) are everywhere

Goals

- Develop “bug-free” systems software
- Correctness, performance, and programmability

Formal verification of systems

- Eliminate entire classes of bugs
- Write a spec & prove impl meets the spec



- Verification projects at UW: Bagpipe [OOPSLA'16], Neutrons [CAV'16], Verdi [PLDI'15], …

Challenges 1/3: non-trivial efforts

- Time-consuming: often person-years
- Require high-level of expertise
- Example: the seL4 kernel
  - 10 KLOC code,
  - 480 KLOC proof
  - 11 person-years

Challenges 2/3: spec

- What *is* a correct system
  - Low-level correctness is well-understood: no overflow
  - Some fields have been using formal specs: TLA+
  - Difficult in general

- Examples
  - The file system must ensure *crash safety*
  - The OS kernel must enforce *process isolation*

Challenges 3/3: integration

- Integrating verification with daily development
  - Learning curve
  - Write good specs
  - Improve upon testing (e.g., Driver Verifier)
  - Catch up with new features: no code base is static
  - Incremental deployment: co-exist with legacy code

Push-button verification

- System design for minimizing proof efforts
- Verifiability as a first-class concern
- Leverage advances in automated SMT solving

Outline

- Yggdrasil [OSDI ‘16]: File system
- Hyperkernel [SOSP ’17]: OS kernel
- Ongoing Work
- Conclusion

Approach

- Model as event-driven systems
  - A set of “atomic” handlers
  - Each handler is bounded
  - Add layers to scale up verification

- Proof automation using SMT solvers
  - Use effectively decidable theories only (if possible)
  - Smart encodings of systems properties
  - Need research on SMT “symbolic” profiling

Yggdrasil [OSDI ‘16]

- Yxv6: journaling file system similar to ext3
- Guarantees: functional correctness & crash safety
- Verified: 1.6 hours w/ 24 cores - no manual proofs

	spec	impl	consistency inv.
Yxv6	250	1,500	5
infrastructure	--	1,500	--
FUSE stub	--	250	--

Hyperkernel [SOSP '17]

- Unix-like teaching operating system based on xv6
- Functional correctness
- High-level properties, ex: Process isolation
- 15 minutes to verify on an 8-core Intel i7 CPU

Component	Lines	Languages
Kernel implementation	7,419	C, assembly
Representation invariant	197	C
State-machine specification	804	Python
Declarative specification	263	Python
User-space implementation	10,025	C, assembly
Verifier	2,878	C++, python

Ongoing work

- Generalize to more systems
  - Push the boundary to more complex systems
  - Example: Hypervisors
- Deployability
  - How to use push-button verified systems in practice

Conclusion

- Push-button verification
  - Examples: file systems, OS kernels
  - Reusable design patterns and toolchains
- Verifiability as a first-class design concern
- How to integrate with daily development