Hyperkernel: Push-Button Verification of an OS Kernel

Luke Nelson, Helgi Sigurbjarnarson, Kaiyuan Zhang, Dylan Johnson, James Bornholt, Emina Torlak, Xi Wang

locore.cs.washington.edu/hyperkernel

Goal: Fully automated kernel verification

- Kernels are an essential component of systems
- Manual proofs are costly: 11 p.y. for seL4

Challenges

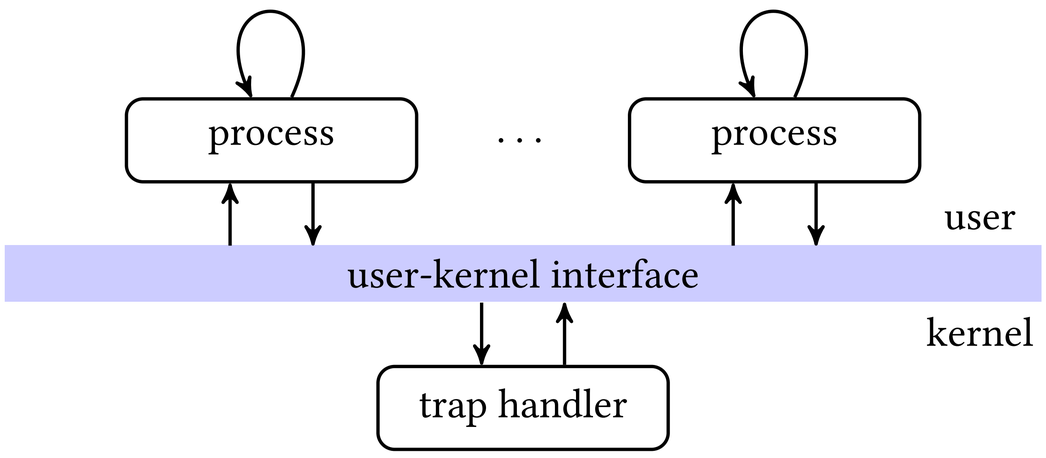
- API must be amenable to automated reasoning
- Virtual to physical mapping can be arbitrarily complex
- C is known to be difficult to reason about

Idea: co-design kernel w/ automated verification

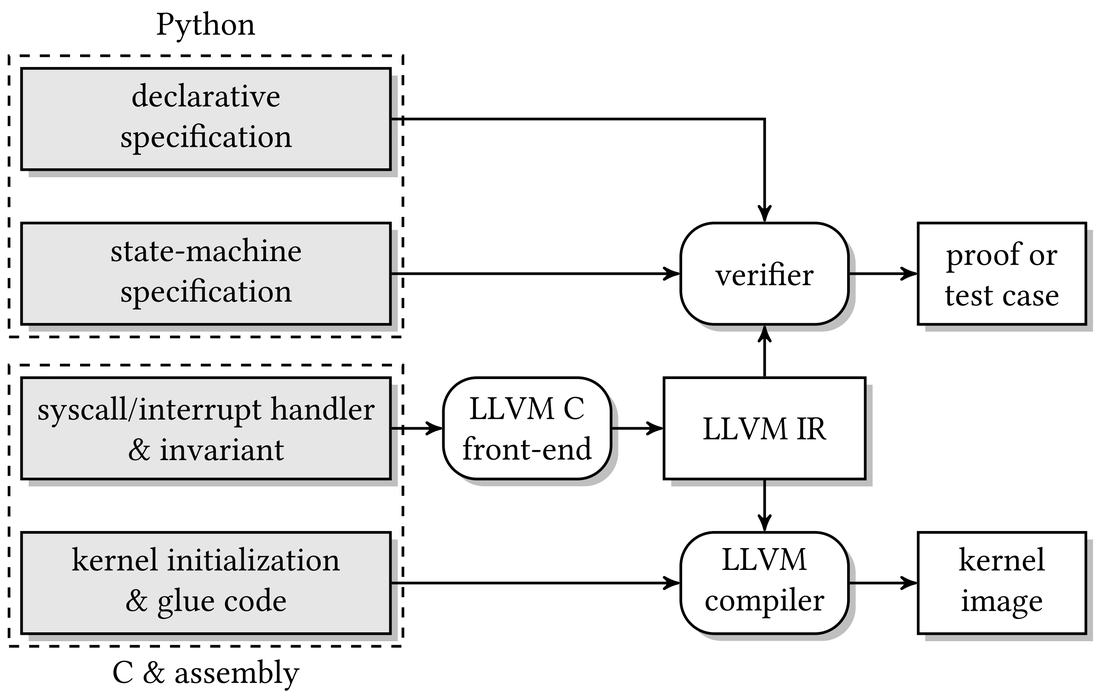
- Make kernel interface finite (free of loops & recursion)
- Limit address space complexity – separate user / kernel
- Verify at the LLVM IR level using the Z3 SMT solver

Designing finite kernel interfaces

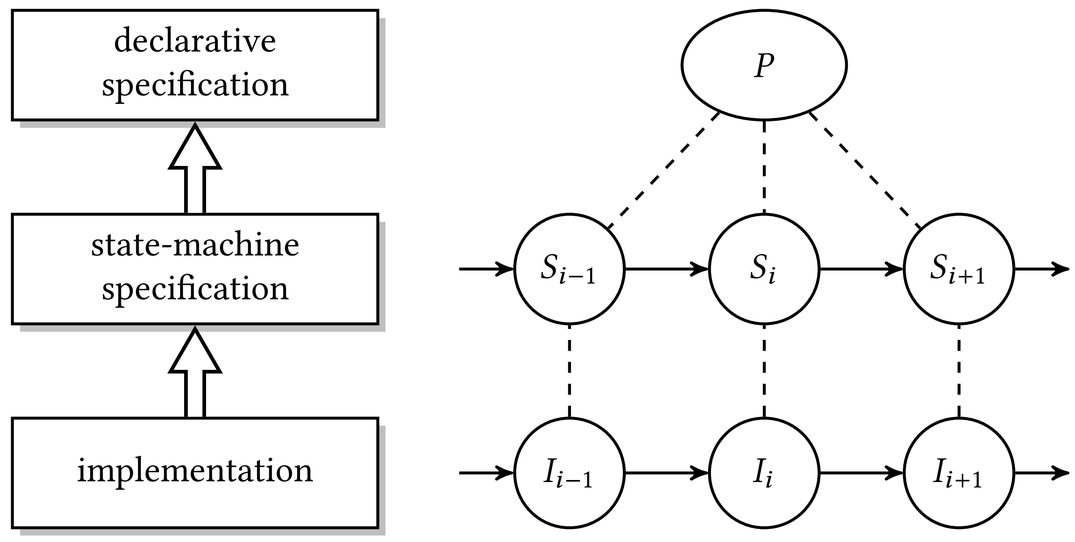
- Make resource management explicit
- Enforce resource lifetime with reference counters
- Validate complex data structures when possible
- Model interface after Unix-like xv6



Development Flow



Main theorems



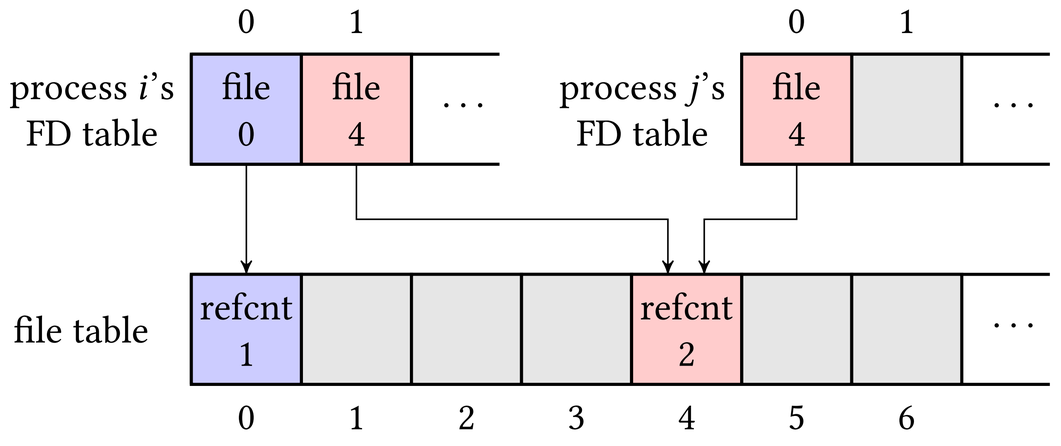
Theorem 1: The implementation is a refinement of the state-machine specification.

Theorem 2: The state-machine specification satisfies the declarative specification.

Example Declarative Specification

file_nr_fds(f) = | {(pid, fd) | proc_fd_table(pid, fd) = f } |

- The reference count for each file equals the number of file descriptors referring to it



Evaluation

- 50 trap handlers verified using Z3 SMT solver
- 15 minutes to verify on an 8-core Intel i7 CPU

Lines of code

Component	Lines	Languages
kernel implementation	7,419	C, assembly
data-structure invariants	197	C
state-machine specification	804	Python
declarative specification	263	Python
user-space implementation	10,025	C, assembly
verifier	2,878	C++, Python

Performance Benchmarks (cycles)

Benchmark	Linux	Hyperkernel	Hyp-Linux
syscall	125	490	136
fault	2917	615	722
appel1	637,562	459,522	519,235
appel2	623,062	452,611	482,596

UNIVERSITY of WASHINGTON

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING