Axel Sorenson

951-269-9768 | AxelPSorenson@gmail.com | linkedin.com/in/axel-sorenson/ | github.com/axelcool1234

Education

University of Utah — Salt Lake City, UT

PhD in Computer Science (Compilers and Formal Methods) — *August 2025 - May 2030*

- **Advisor**: Professor John Regehr
- **Lab**: Formal Methods Lab
- **Relevant Coursework**: Advanced Compiler Implementation, Software Verification, Programming Language Semantics, Testing and Verification of Digital Circuits, Advanced Operating System Implementation

University of California, Irvine — Irvine, CA

Bachelor of Science in Computer Science — *September 2022 - March 2025*

- **Cumulative GPA**: 4.0
- **Awards**: Dean's Honors List (every quarter), Phi Beta Kappa Annual Book Award

Experience

Compiler Engineer Intern — May 2026 - August 2026

AMD (AIG-SHARKS) — *San Jose, CA*

- Diagnosed a defect in AMD's current and next generation pre-register-allocation schedulers for gfx1250 (MI450) GPUs that bunched LDS loads too far ahead of the WMMAs consuming them, leading to significant register spillage and causing the matrix cores to majorly stall on s_wait_dscnt instructions.
- Implemented a ScheduleDAGMutation in LLVM's AMDGPU backend adding DAG edges confining each LDS load to a window determined from a statically calculated live VGPR histogram. (#203095)
- On AMD's cycle accurate simulator, raised matrix core utilization from 2.5% to 40.1% on the worst affected mxfp8 GEMM, cutting its VGPR spills from 805 to 16 and its average stall on s_wait_dscnt instructions from 594 to 44 cycles per loop iteration; additionally lifted an fp16 GEMM from 77.8% to 86.2% matrix core utilization. Peak register usage fell on all kernels tested.
- Built a containerized benchmark harness (using Docker, Make, and Python) for kernels in various configurations using AMD's simulator, autogenerating comparison tables used to iterate on the DAG mutation and further understand the defect in the AMD pre-RA schedulers.

PhD Research Assistant — August 2025 - Present

Formal Methods Lab with Professor John Regehr — *Salt Lake City, UT*

- Authored the dataflow analysis framework of VeIR, a work in progress verified compiler in Lean 4 mirroring MLIR capabilities and asymptotic complexity with proof assistant level verification, developed with Tobias Grosser's group at the University of Cambridge and my advisor. The framework is a port of MLIR's C++ DataFlowFramework - a worklist fixpoint solver built on top of the monotone framework. Sole author of VeIR's analysis namespace, across 37 merged pull requests. (#264)
- Laid the groundwork for verified transfer functions by defining VeIR's abstract domain interface as a Lean typeclass carrying its own proof obligations: no instance typechecks until it discharges the bounded join semilattice laws and proves its concretization map monotone. Already instantiated for constant propagation and liveness. (#666, #856)
- Built VeIR's dominance and dead code analyses as clients of that framework: an iterative Cooper-Harvey-Kennedy immediate dominator solver, exposed through block and operation level dominance queries over nested regions, and a dataflow based dead code analysis. (#133, #721, #659)
- Built the extraction harness producing the evaluation corpus for the lab's StableHLO semantics formalization: 121 torchvision models across 21 image and 42 video shapes through two export paths (JAX, torch-xla). (repo)
- Prototyped L2, a loop language with an implemented IC3/property directed reachability invariant prover over Z3, evaluated on 35 matched safe/unsafe benchmarks under lit and FileCheck - an experiment for exploring translation validation of unbounded loops, which Alive2 currently handles by bounded unrolling.

LLVM Open Source Contributor November 2024 - Present
llvm-project (Open Source) *Remote*

- Implemented an extension point in the PassBuilder pipeline enabling developers to insert and run custom passes immediately after vectorization; used upstream to register AMDGPU's InferAddressSpacesPass. (#123494)
- Ported a funnel shift combiner from SelectionDAG to GlobalISel as declarative TableGen, cutting a canonical AArch64 GlobalISel case from 9 instructions to 7. (#135132)
- Ported a rotate transformation from SelectionDAG to InstCombine, giving the fold a middle end equivalent so chained rotates collapse earlier in the pipeline; benchmarked across seven real world codebases including LuaJIT, wasmtime-rs, and ffmpeg, with instruction count reductions and no regressions. (#160628)

Undergraduate Research Assistant August 2024 - March 2025
Secure Systems and Software Laboratory at University of California, Irvine *Irvine, CA*

- Helped document a binary lifter translating machine code into LLVM IR with Professor Michael Franz, learning pointer provenance, memory models, and aliasing rules to keep the lifter aligned with LLVM internals.

Academic Intern March 2023 - March 2025
University of California, Irvine *Irvine, CA*

- Developed automated grading tools handling 400+ weekly student submissions for UCI's lower division Intermediate Programming course.
- Refined students' programming skills by providing constructive feedback to hundreds of students on programming assignments and projects.

Projects

Compiler With x86 ELF Binary Generation (repo) | *C++, x86 Assembly* March 2024 - June 2024

- Constructed an optimizing compiler supporting signed integer arithmetic, I/O, structured control flow, and user defined functions, running a benchmark that calculates the Mandelbrot set ~8x faster than the equivalent CPython program.
- Implemented lexer, LL(1) recursive descent parser, SSA IR with several optimizations, register allocator, assembly code generator, and ELF bytecode assembler.
- Designed and integrated a Graphviz based visualizer for the SSA IR to present optimized control flow diagrams.

Lambda Calculus Interpreter (repo) | *Prolog* August 2025

- Implemented a full lambda calculus interpreter: tokenization, parsing, pretty printing, and evaluation to normal form (normal order beta reduction), with capture avoiding substitution via alpha conversion and free variable analysis.
- Added macros and parameterized macros for encoding higher order functions and Church numerals, plus alpha equivalence checking to resugar results back into macro form.

Search Engine and Web Crawler | *Python* October 2023 - December 2023

- Developed a polite web crawler adhering to robots.txt and sitemap protocols with a team of three, using a runtime thread pool for multithreaded crawling.
- Built partial binary indexing for memory efficient concurrent indexing over 40k+ pages, with a frontend ranking results via TF-IDF and cosine similarity.

Technical Skills

Languages: C, C++, Rust, Lean 4, Python, Nix, Bash, Nushell, Prolog, Haskell, SQL, LaTeX, Typst
Frameworks/Libraries: LLVM, MLIR, Triton, Alive2, Z3
Developer Tools: Linux, Git, Jujutsu, Nix, Docker, CMake