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) — *August 2025 - Present*

- **Advisor**: Professor John Regehr
- **Lab**: Formal Methods Lab
- **Relevant Coursework**: 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
- **Relevant Coursework**: Database Management, Computer Organization, Data Structures and Algorithms, Information Retrieval, System Design, Compiler Construction, Artificial Intelligence, Machine Learning, Data Mining, Operating Systems, Embedded Software

Experience

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, increasing flexibility for downstream compiler projects. (#123494)
- Ported a funnel shift combiner from SelectionDAG to GlobalISel, improving codegen efficiency and reducing instruction counts on backends using GlobalISel. (#135132)
- Ported a rotate transformation from SelectionDAG to InstCombine, enabling the optimization to run earlier in the compiler pipeline. This improves codegen for projects such as LuaJIT and wasmtime-rs. (#160628)

PhD Research Assistant — August 2025 - Present

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

- Formalizing the semantics of StableHLO to enable verified compiler optimizations across Torch, JAX, and TensorFlow. Gathered hundreds of megabytes of StableHLO for evaluation of the defined semantics. (repo)
- Investigating program cut points and loop invariant inference to scale Alive2 verification to larger programs and unbounded loops, extending the reach of formal methods to real-world workloads. (repo)
- Exploring memory model research to expand the breadth of programs Alive2 can employ translation validation on.

Undergraduate Research Assistant — August 2024 - March 2025

Secure Systems and Software Laboratory at University of California, Irvine — *Irvine, CA*

- Collaborated with Professor Michael Franz on maintenance, enhancement, and documentation of a state-of-the-art binary lifter for translating machine code into LLVM IR.
- Studied pointer provenance, memory models, and aliasing rules to guide enhancements to the lifter and ensure alignment with LLVM internals.

Academic Intern — March 2023 - March 2025

University of California, Irvine — *Irvine, CA*

- Improved the efficiency of grading by ~300% by developing automated grading tools handling 400+ weekly student assignments 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.
- Collaborated with the course instructor to ensure consistency in handling the grade rubric and providing feedback to students in order to facilitate a more standardized and fair grading process.

Projects

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

- Constructed an optimizing compiler for a language 721% faster than Python supporting signed integer arithmetic, I/O, structured control flow, and user-defined functions.
- 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).
- Built robust substitution with alpha-conversion and free-variable analysis to prevent variable capture.
- Added macros and parameterized macros enabling concise encoding of higher-order functions and Church numerals.
- Implemented alpha-equivalence checking to resugar results back into macro form, improving readability.

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.
- Increased crawling speed ~4x by implementing a runtime thread pool for multithreaded crawling.
- Achieved ~200x speedup in indexing via optimized partial binary indexing for memory-efficient concurrent indexing (40k+ pages).
- Built a simple front-end with integrated ChatGPT functionality; improved ranking via TF-IDF and cosine similarity.

Multithreaded Assembly Unittesting Library | *Python, Assembly* September 2023 - December 2023

- Developed a MIPS assembly unit-testing framework with automated test generation and execution.
- Implemented parallel test execution to reduce runtime using Python threading.
- Added modular test suite architecture with automated memory/register state tracking and convention checks.

Canvas Autograder | *Python* September 2023 - March 2025

- Built an automated grading system for Canvas submissions, reducing grading time by ~300% via Python autograding and PDF parsing.
- Implemented secure/effective testing: AST filtering, unit tests, and time/memory limits to prevent infinite loops.
- Designed API-based grading pipeline using Canvas API to update grades/comments programmatically, bypassing slow SpeedGrader UI.

Movie Ratings Classification AI Models | *Python* November 2024 - December 2024

- Developed ML models for IMDB sentiment analysis, surpassing 80% accuracy using MLP, SVM, and Random Forest.
- Applied NLP techniques: TF-IDF vectorization, tokenization (lemmatization/stemming), and stop-word removal.
- Performed hyperparameter tuning and anti-overfitting strategies to balance accuracy and generalizability.
- Evaluated ensembles: stacking and gradient boosting to further improve classification.

Technical Skills

Languages: C, C++, Rust, Python, Nix, Bash, Nushell, Prolog, Lean, Haskell, SQL, LaTeX, Typst
Frameworks/Libraries/DB: LLVM, MLIR, Alive2, Z3, SQLite, MySQL, PostgreSQL, Couchbase, Neo4j, Spark, MongoDB, Cassandra
Developer Tools: Linux, Git, Nix