Yueyang Tang

https://github.com/Emanon42 | yueyang@purdue.edu | +1(765)694-5902

Education

Purdue University, West Lafayette, IN, USA — Jan 2023 - Present

PhD in Computer Science

- Supervisor: Professor Tiark Rompf.
- Research topics: programming languages and software engineering, including functional programming, type systems, and compiler constructions.
- Currently working in Reachability Types: a lifetime/ownership mechanism for high-order functional programs.

The University of Edinburgh, Edinburgh, Scotland, UK — Sep 2017 – Jun 2022

BSc Hons Artificial Intelligence and Computer Science, 1st Class Honours

- Had a gap year in the 2020-2021 academic year for two 6-month intern chances in the industry.
- Awards: GitHub Sponsor Prize in System Development Project, Third Prize in Functional Programming Competition (sponsored by Galois).
- Relevant courses: Security (100%), Compiler (99%), Haskell (95%), Network (88%), Architecture (82%), Machine Learning (80%).

Research Experience

Developing Holbert *(Haskell, Compiler Frontend)* — Jan 2022 – Mar 2022

Part-time Research Intern at Laboratory for Foundations of Computer Science, Edinburgh

- Worked in Holbert project, a graphic higher order logic proof assistant running in the browser written with Haskell (GHCJS), supervised by Dr. Liam O'Connor.
- Implemented the user-defined infix operator by using the Earley parsing algorithm.
- Fixed bugs in elaborating of elimination rules.

Developing Aya-prover *(Java 16, Type System, Dependent Type)* — Feb 2021 – Jun 2021

Full-time Research Intern at PLCT Lab, Intelligence Software Research Center, ISCAS, Nanjing

- Worked in Aya-prover project, an interactive theorem prover based on cubical type theory written with Java.
- Implemented `Interval` and `Path` type by applying CuTT in Aya's core type system for inferring the equality of dependent type.
- Fixed bugs in elaborating of `Condition` (a special dependent type constructor) by applying matrix-based pattern traversing.
- Added 100+ coverage tests and 200+ incremental tests.

Improving Links-lang *(OCaml, Compiler Midend)* — Jun 2020 – Sep 2020

Summer Research Intern at Laboratory for Foundations of Computer Science, Edinburgh

- Worked in Links-lang project, a functional language for fullstack web dev written with OCaml, supervised by Dr. James Cheney and Dr. Simon Fowler.
- Improved the serialization of PostgreSQL queries (especially nested ones) by applying modular functor and Format Module, which achieved a 10x increase in speed.
- Extended the syntax to allow SML-style function definition with pattern matching.

Work Experience

Heterogeneous compiler Intern *(C++, LLVM, Compiler Backend)* Jul 2021 – Nov 2021

Full-time Software engineering Intern at The Wake Systems, Beijing

- Worked in the Compiling Optimization Team on the automatic vectorization and parallelization targeted to aarch64 and rv64 ISA.
- Implemented an LLVM static analysis pass based on domination tree and loop invariant to detect and evaluate the legality and cost of vectorization.
- Helped to migrate and adapt DawnCC to the latest LLVM 13.

Project Experience

Lightweight JIT compiler as Undergraduate Thesis

- Designed and implemented a JIT compiler for a non-trivial C subset, targeting RISC-V ISA, supervised by Professor Bjoern Franke.
- It consists of 4 main components: an interpreter as entry point, a compiler for code generating on the fly, a heat function as JIT trigger, and a page manager for managing generated binaries in memory.
- The compiler is in a two-pass style: AST to 3AC IR to RISC-V assembly. Applied type and name analysis at AST level, applied peephole optimization at 3AC IR level.
- Implemented the dynamic linking to glibc functions such as printf.

Register Machine Interpreter as Theoretical Computer Science (TCS) coursework

- Implemented a register machine simulator with two primitive instructions: “inc(i)” (add 1 to i^{th} register) and “decjz(i, j)” (if i^{th} register = 0 then goto j^{th} instruction, else subtract 1 from i^{th} register).
- It supports user defined macro instructions like jump and loop.
- It is Turing complete, the proof can be given by writing a Brainfuck interpreter using it.

Skills

- **Programming Languages: multilingual** (not limited to any specific language or paradigm), especially experienced in **OCaml** (in Links-lang dev), **Modern Java** (16+, in Aya-prover dev), **Rust** (in undergrad thesis), and **Coq** (in Prof. Rompf’s research group). Comfortable with C/C++, Haskell, Agda, Kotlin, Python, Bash, F#, Idris, Koka, Elm, Scala.
- **Compiler:** understand various program representations like static single assignment (SSA), continuation passing style (CPS), and sea-of-nodes.
- **Semantics:** understand small/big-step operational semantics, studying denotational semantics and axiomatic semantics.
- **Utilities:** LaTeX, Docker, Git, LLVM, Linux sysadmin (NixOS and Arch).