DEYUAN (MIKE) HE

https://ad1024.github.io/

mikehe@princeton.edu

Education

Princeton University, Princeton, NJ *Sept. 2022—Est 2027*

Ph.D. in Computer Science

- Advisor(s): TBD
- Fields of Study: Programming Languages & Formal Verification & Compilers & MLSys

University of Washington, Seattle, WA *Sept. 2018—Jun. 2022*

B.S. in Computer Science

- GPA: 3.89 (ranking not applicable)
- Fields of Study: Programming Languages & Formal Verification & Compilers & MLSys
- Honors: CRA Outstanding Undergraduate Researcher Award 2022 (Honorable Mention)

Selected Publications

1. Marisa Kirisame*, Steven Lyubomirsky*, Altan Haan*, Jennifer Brennan, **Mike He**, Jared Roesch, Tianqi Chen, and Zachary Tatlock. Dynamic tensor rematerialization. In *International Conference on Learning Representations (ICLR'21)*, 2021 (*: Equal Contribution)
2. Bo-Yuan Huang, Steven Lyubomirsky, Yi Li, **Mike He**, Thierry Tambe, Gus Henry Smith, Akash Gaonkar, Vishal Canumalla, Gu-Yeon Wei, Aarti Gupta, Sharad Malik, and Zachary Tatlock. Speicalized accelerators and compiler flows: Replacing accelerator apis with a formal software/hardware interface, 2021
3. Bo-Yuan Huang*, Steven Lyubomirsky*, Thierry Tambe*, Yi Li, **Mike He**, Gus Smith, Gu-Yeon Wei, Aarti Gupta, Sharad Malik, and Zachary Tatlock. From dsls to accelerator-rich platform implementations: Addressing the mapping gap. In *Workshop on Languages, Tools, and Techniques for Accelerator Design (LATTE'21)*, 2021 (*: Equal Contribution)

Experience

Taichi Graphics Jun. 2022—Sept. 2022

Compiler R&D Intern *Remote*

· Implement matrix representations for low-level IR to optimize the performance by vectorization.

· Enable CHI IR to represent matrices and vectors as whole and implement code generation for LLVM.

· Benchmark use cases under different scenarios (e.g. sparse v.s. dense matrices).

Intel Labs Mar. 2022—Jun. 2022

Formal Verification Research Intern *Remote & Hillsboro, OR*

· Implemented Pyrope, a domain-specific language embedded in Python that aims to enable **correct-by-construction hardware modeling**. Pyrope captures a major subset of Python syntax and, in addition, provides interfaces for proof-carrying programming in Python. Pyrope compiles to Dafny for automated verification.

· Encoded the proof of (multi-)montgomery reduction algorithm and successfully verified by compiling to Dafny.

3LA, LATTE '21 Jun. 2020—Now

Research Assistant @ PLSE *Seattle, WA*

· 3LA proposes an end-to-end compilation flow that provides **flexible** and **verifiable** compiler support for custom Deep Learning (**DL**) accelerators.

· Implemented Flexible Matching: using equality saturation to search for optimal operator offloading to accelerators. Flexible Matching is able to efficiently explore the space of equivalent model implementations modulo a set of rewrite rules and extract the optimal model that has been found.

Dynamic Tensor Rematerialization, ICLR '21 Oct. 2019—Aug. 2021
Research Assistant @ PLSE *Seattle, WA*

· Dynamic Tensor Rematerialization (**DTR**) is a greedy gradient checkpointing algorithm. DTR **enables** training Deep Learning models on memory-constrained devices.
· Implemented evaluations and nightly CI for DTR prototype in PyTorch.
· Prepared submission artifact of DTR to ICLR '21.

Teaching

Paul G. Allen School, University of Washington Mar. 2021—Jun. 2021
Teaching Assistant *Seattle, WA*

· Worked as a TA for **Principles of Programming Languages** (CSE 505)
· Helped re-designing CSE 505 and developing course materials for various topics about PL and formal verification (**Hoare Logic**, **Lambda Calculus** and **System F**, etc.) in **Coq**.

Talks & Presentations

1. *Pyrope: Towards Provably Correct Hardware Modeling in Python/HeteroCL*, Jun. 2nd at Intel Labs.
2. *From DSLs to Accelerator-rich Platform: Addressing the Mapping Gap*, Sept. 2021 at Intel (presented jointly with Dr. Steven Lyubomirsky)
3. *Correct & Flexible Compiler Support for Custom Accelerators*, Sept. 2021 at SRC ADA Center

Conference Service

→ **MICRO '21**, Artifact Evaluation

Projects

Sager

- A demonic data structure synthesizer written in ROSETTE, a **solver-aided** language based on **Racket**. **Sager** is able to exploit algorithm bottleneck by performing **symbolic exeuction** over the whole algorithm and using **SMT solver** to synthesize a sample data structure the algorithm works on that pushes the algorithm to its worst-case performance.
- A demonic data structure synthesizer that aims to explore worst-cases performance of graph algorithms.
- **Language** & TOOLS: **Racket**, **Rosette**, Z3
- Keywords: SMT Solver, Incremental Solving, Program Synthesis, Symbolic Execution

veripy

- An easy-to-use auto-active program verification library for Python programs written in **Python**.
- The library is shallowly embedded in Python and the interface is implemented as **decorators**. It compiles annotated Python functions to **SMT** formulae and calls **SMT solver** to check whether it matches the given specification and gives a counter-example input when it violates any constraint.
- **Language** & TOOLS: **Python 3**, **SMT-LIB**, Z3, PYPARSING
- Keywords: SMT Solver, Static Analysis, Hoare Logic, Program Verification

dtlc

- Implemented **dependently-typed** lambda calculus in Martin-Löf style intuitionistic type theory.
- Written in **OCaml**, **dtlc** has a language frontend **Lexer & Parser** implemented using **Menhir**. Core language supports **type unification** with **metavariables** which makes the type inference stronger.
- Implemented eliminators for **naturals**, **identity type**, **union type**, etc.
- **Language** & TOOLS: **OCaml**, MENHIR, DUNE
- Keywords: Type Theory - Dependent Type, Proof Assistant, Functional Programming