MIKE HE

https://ad1024.space

dh63@cs.washington.edu
(206) · 887 · 8588

EDUCATION

University of Washington, Seattle *Sept. 2018—Est. Jun. 2022*
B.S. in Computer Science

- Cumulative GPA: 3.88 / 4
- Field of Studies: Programming Languages & Formal Verification & Compilers & MLSys
- Selected Coursework: Data Structures & Parallelism; Intro to Algorithms; Advanced Programming Languages & Verification; Intro to Artificial Intelligence; System Programming; Computational Complexity
- PLs & Skills: **Java**, **Python**, **C++**, **Coq**, **OCaml**, **Rust**; Algorithm Design, Formal Verification

EXPERIENCE

PLSE Lab, University of Washington Oct. 2019—Now
Research Assistant *Seattle, WA*

· Implemented a Just-in-time compiler (**JIT**) in TVM that can offload deep learning operators, perform simulation-based verification during compile time and address granularity mismatches between high-level intermediate representations and target accelerators by compiling to Instruction-level Abstraction (**ILA**).

· Worked on Dynamic Tensor Rematerialization (**DTR**), an greedy gradient checkpointing algorithm that **enables** training Deep Learning models on memory-constrained devices (e.g. GPUs, FPGA-based accelerators) by evicting and recomputing tensors **on fly** without prior knowledge about the model architecture.

· Implemented continuous integration and evaluation for the DTR implementation in **PyTorch**; the infrastructure was able to detect multiple performance regression and expose bugs in operator dispatcher.

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

· Worked as TA for **Principle 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**.

· Held office hours and shared tricks used in Coq tactics and Coq programming.

· Coordinate grading of all homework assignments.

PROJECTS

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.

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.

PUBLICATIONS

1. Marisa Kirisame, Steven Lyubomirsky, Altan Haan, Jennifer Brennan, Mike He, Jared Roesch, Tianqi Chen, and Zachary Tatlock. Dynamic tensor rematerialization, 2021
2. 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