

MIKE (DEYUAN) HE

✉ mikehe@princeton.edu · in @Mike He

EDUCATION

Princeton University, Princeton, NJ

2022 – Est. 2027

Ph.D. in Computer Science

Advisors: Prof. Aarti Gupta

Fields of study: Compilers; Formal Verification; Distributed Systems; Equality Saturation

University of Washington, Seattle, WA

2018 – 2022

B.S. in Computer Science, GPA: 3.89/4.0 (CUM LAUDE)

Advisors: Prof. Zachary Tatlock & Dr. Steven Lyubomirsky

Selected Honor: CRA Outstanding Undergraduate Researcher Award, Honorable Mention (2022)

RESEARCH

CATS_{TAIL}: Synthesizing Packet Programs via Equality Saturation

June. 2023 – Now

TL;DR CATS_{TAIL} is an equality saturation-based P4 program synthesizer. Previous works use SKETCH to synthesize the program, which takes too long to make debugging on actual hardware possible. Compared with SKETCH, CATS_{TAIL} is up to 30x/2000x faster in finding the optimal stage allocation for Intel Tofino/Domino (Banzai ALU). I lead the design and implementation of CATS_{TAIL}.

Verifying correctness of SW/HW mappings

Dec. 2022 – Now

TL;DR hex is a language for accelerator operation explication and a tool for verifying the software-hardware mapping correctness. My contribution and work in progress are

- Implemented a case study for FlexASR pooling instructions and verified its correctness against the software implementations
- Designing memory layout mapping invariant inference/generation algorithm

Improving Term Extraction with Acyclic Constraints

Sep. 2022 – Feb. 2023

TL;DR To have a better term extraction algorithm for egg, an equality saturation framework, we devise the encoding using Weighted partial MaxSAT and include a set of *Acyclic constraints* that ensures the acyclicity of the extracted optimal term. Our encoding demonstrates better solver time (~3x speed up) for the case study of extracting tensor programs. My contributions include

- Devised the constraint
- Optimized the constraints with Tesitin encoding, which exponentially reduces the search space.
- Developed the application-agnostic term extractor
- Implemented the case study using Glenside examples
- Authored the workshop paper at **PLDI EGRAPHS'23**

3LA: Application-level Validation of Accelerator Designs

June. 2021 – June. 2022

TL;DR 3LA is a software/hardware co-verification methodology for DL accelerators that aids hardware developers in performing early-stage application-level debugging. My contributions are

- Lead the development of Flexible matching
- Extended Glenside to support a more diverse set of input models
- Implemented the compilation pipeline for VTA
- Implemented handwritten digit recognition (on CIFAR) and image classification (on ImageNet) for VTA. Passed the mapping validation using 3LA.
- Co-authored the paper under review at **ACM TODAES**.

Dynamic Tensor Rematerialization

Jan. 2020 – Oct. 2020

TL;DR Dynamic Tensor Materialization (DTR) is an online, heuristic-based checkpointing algorithm that enables DL inference under constrained memory budgets. My contributions are

- Identified problems in the PyTorch DTR implementation
- Designed the evaluation framework for DTR and extended the case studies to multiple new DL applications (e.g. Unrolled GAN, UNet)
- Co-authored the paper published at **ICLR'21**

PUBLICATIONS

-
- [submitted to ACM TODAES] 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. *Application-Level Validation of Accelerator Designs Using a Formal Software/Hardware Interface*, 2022
- 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*, 2021
- Marisa Kirisame*, Steven Lyubomirsky*, Altan Haan*, Jennifer Brennan, **Mike He**, Jared Roesch, Tianqi Chen, and Zachary Tatlock. *Dynamic Tensor Rematerialization*, 2021

SERVICE

- AEC member of POPL'24, MLSys'23, MICRO'21
- Mentor of the Ph.D. application mentoring program (Princeton, 2023)

INTERNSHIPS

Taichi Graphics, Remote and Beijing, China

June. 2022 – Sep. 2022

Compiler R&D Intern (C++/Python)

- Refactored the intermediate representation (IR) of Taichi Language
- Implemented standalone **Tensor type** for better compilation speed
- Adapted **compiler passes** (e.g. Load/Store forwarding, Dead code elimination, reaching definition, etc.) to optimize for tensor type expressions
- Implemented **LLVM**-based code generation for tensor type for Superword-level vectorization

Intel Labs, Hillsboro, OR

Mar. 2022 – June. 2022

Formal Verification Research Intern (Formal Methods/Python/Dafny)

Developed the **Pyrope** framework for **correct-by-construction** hardware modeling.

- Facilitated **correct-by-construction** hardware modeling purely in Python
- Encoded the correctness proof of (multi-)montgomery reduction algorithm in Python and **verified successfully by compiling to Dafny**
- Unified “sources of truth” for correctness proofs and programming model implementations

UWPLSE, Seattle, WA

Oct. 2019 – Sep. 2021

Research Assistant (PL/Compiler)

Responsible for conducting research with Prof. Zachary Tatlock, specifically,

- Implemented evaluations in the Dynamic Tensor Rematerialization project
- Designed a flexible matching algorithm for domain-specific language compilers.
- Led research projects with other undergraduate students
- Attended and presented at reading groups

SELECTED PROJECTS & CONTRIBUTIONS

CATS**TAIL**: Synthesizing Packet Programs via Equality Saturation (Rust) [GitHub](#)
Music Scores: Reverse engineering of some arrangements (Lilypond) [GitHub](#)
flexmatch: Flexible offload pattern matching for DNNs (Python, Rust) [GitHub](#)
egg-taichi: Towards automated super-optimization for Taichi programs (Rust) [GitHub](#)
Taichi*: High-performance parallel computing in Python (C++, Python) [GitHub](#)
Glenside*: Term rewriting for tensor programs (Rust) [GitHub](#)
veripy: auto-active verification for Python programs (Python) [GitHub](#)
dtlc: Dependently-typed lambda calculus (OCaml) [GitHub](#)
Sager: A demonic graph synthesizer for worst-case performance (ROSETTE, Racket) [GitHub](#)
More on my [GitHub](#)
★ : Contributor

TEACHING

- [COS 516: Automated Reasoning about Software](#) (TA, Princeton University)
- [CSE 505: Principles of Programming Languages](#) (TA, University of Washington)

SKILLS

- **Languages**: C/C++, Python, Rust, OCaml, Coq, Dafny, etc. (Open to other languages)
- **Compiler & Applied PL**: Equality Saturation, Static Analysis, Computer-aided Reasoning, SMT
- **PL Theory**: Formal Verification, Type Theory, Mathematical Logic
- **Systems**: Distributed Systems, Machine Learning Systems, Data Center Systems
- **Others**: Algorithms and Data Structures
- **Fun Fact**: I am more experienced in playing the violin than coding 🎻; I have:
 1. The Former Lv.9, the new highest level equivalent, certified by [Central Conservative of Music](#);
 2. > **20-year** violin solo experience;
 3. Multiple 1st Prizes (various local competitions in Beijing) and a Silver medal (Beijing regional);
 4. 6-year experience with symphony orchestras; 3-year experience as *the 2nd Principal Violinist*;
 5. 3-year experience with a piano quartet/quintet and multiple string quartets (with 1 CD made);
 6. 4 public performances with a philharmonic orchestra at the [National Centre for the Performing Arts](#), Beijing, China