Minghai Lu

(+1) 765-421-0893 | lu1074@purdue.edu | lachinygair.github.io

Education

Purdue University — West Lafayette, IN
Ph.D. *in Computer Science* — *2023 – 2028 (Expected)*

Southern University of Science and Technology — Shenzhen, China
B.E. *in Computer Science and Engineering* — *2019 – 2023*

Publications

LLM-Assisted Synthesis of High-Assurance C Programs
Prasita Mukherjee, ***Minghai Lu****, Benjamin Delaware*
ASE '25: Proceedings of the 40th IEEE/ACM International Conference on Automated Software Engineering

Proof Automation with Large Language Models
Minghai Lu*, Benjamin Delaware, Tianyi Zhang*
ASE '24: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering

Automated Deep Learning Optimization via DSL-based Source Code Transformation
Ruixin Wang, ***Minghai Lu****, Cody Hao Yu, Yi-Hsiang Lai, Tianyi Zhang*
ISSTA '24: Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis

JITfuzz: Coverage-guided Fuzzing for JVM Just-in-Time Compilers
Mingyuan Wu, ***Minghai Lu****, Heming Cui, Junjie Chen, Yuqun Zhang, Lingming Zhang*
ICSE '23: Proceedings of the 45th International Conference on Software Engineering

Research Experience

Distributed System Verification with LLM Agents — Ongoing
Purdue University, West Lafayette, IN — *2025 – Now*

- Built a benchmark of distributed system verification tasks in Dafny, covering inductive invariant discovery, inductiveness proofs, and refinement proofs.
- Performed an empirical study of state-of-the-art coding agents (e.g., Claude Code, Codex) on these tasks, learning their performance ceilings and systematic failure modes.
- Building a collaborative multi-agent LLM system in which specialized agents coordinate end-to-end verification.

Fine-tuning and RL on LLMs for Automated Unit Test Generation — Ongoing
Purdue University, West Lafayette, IN — *2025 – Now*

- Synthesized training data from the OpenCoder dataset: first generated chain-of-thought (CoT) from the code under test, then derived test cases from it, explicitly modeling boundary-case analysis, input selection, and output derivation.
- Supervised fine-tuned Qwen-Coder series models on top of the vLLM framework; built a multi-GPU parallel training and evaluation pipeline.
- Further optimized compile pass rate and code coverage with GRPO reinforcement learning; explored multiple reward function designs.
- Benchmarked on EvalPlus, LiveCodeBench, and others; compared the impact of different base models and training-data construction strategies.

LLM-based Automation for Interactive Theorem Proving — ASE 2024
Purdue University, West Lafayette, IN — *2023 – 2024*

- Built PALM, an automated proof framework for interactive theorem proving that integrates LLMs with symbolic methods through a "RAG – automatic error repair – backtracking search" proof generation pipeline.
- Explored multiple RAG approaches, including dense retrieval, BM25, and hybrid schemes, to improve retrieval quality.
- Conducted an empirical analysis of 520 LLM-generated proof errors, identified 7 common failure patterns, and implemented targeted automatic repair mechanisms.
- Evaluated on 10k+ theorems; improved proof success rate by 76.6% over SOTA tools.

LLM-guided Adaptive Proof Repair via Strategy Selection Under Submission

Purdue University, West Lafayette, IN *2024 – 2025*

- Leveraged LLMs for dynamic strategy selection in formal reasoning, combining error messages, local context, and historical trajectories to adaptively choose the next proof-repair path.
- Designed multiple LLM-driven repair strategies, including proactive context retrieval, auxiliary lemma discovery, and proof regeneration.
- Explored a range of decision-making methods (LLM reasoning, ML models, and rule-based heuristics) and systematically evaluated their performance and cost trade-offs.
- Built CoqDev, a new benchmark mined from the GitHub commit history of real Coq projects (1,720 theorems), masking context introduced by future commits to simulate realistic incremental development.
- Evaluated on 2 benchmarks and 5 LLMs; 18.58% improvement over SOTA tools.

Coverage-guided Fuzzing for JVM Just-in-Time Compilers ICSE 2023

Southern University of Science and Technology, Shenzhen *2021 – 2023*

- Designed multiple mutation strategies targeting common compiler optimizations to systematically trigger their code paths.
- Analyzed Java bytecode (data flow, control flow, method calls, class inheritance, etc.) with the Soot framework to ensure source programs remain valid and semantically equivalent after complex mutations.
- Used a Multi-armed Bandit algorithm to dynamically schedule mutation strategies based on coverage feedback.
- Discovered 36 new bugs on mature JVMs including OpenJDK; 27 confirmed and 16 fixed.

Technical Skills

Programming: Python, Java, Rocq, Lean, Dafny, Rust
Tools: Verus (Rust Verification), Soot (Java Program Analysis), PyTorch
Languages: English, Chinese