Shiyou(Alan) Huang

427A HRBB, Texas A&M University
College Station, TX, USA 77840
979-422-6478
huangsy@tamu.edu
huangshiyou.github.io/

Research

My areas of interest are model checking, system security, and bug reproduction. The goal of my research is to help develop automated systems combining program analysis, hardware advances, and systematic testing to improve the reliability and security of the software. During my PhD study, I mainly focus on the dynamic analysis of concurrent systems. So far, I have built tools for model checking concurrent softwares, replaying concurrency failures and instrumenting bound and type checks for Java Unsafe APIs.

Education

9/2015–present	**Ph.D. Candidate in Computer Science**, *Texas A&M University*, USA. *Advisor: Jeff Huang*
9/2011–6/2015	**B.E. in Computer Science**, *Huazhong University of Science and Technology (HUST)*, Wuhan, China. GPA: 4.0/4.0

Publication

ICSE 2019	*SafeCheck: Safety Enhancement of Java Unsafe API* **Shiyou Huang**, Jianmei Guo, Sanhong Li, Xiang Li, Yumin Qi, Kingsum Chow and Jeff Huang Proceedings of International Conference on Software Engineering Companion, 2019
USENIX ATC 2017	*Towards Production-Run Heisenbugs Reproduction on Commercial Hardware* **Shiyou Huang**, Bowen Cai and Jeff Huang Proceedings of USENIX Annual Technical Conference, 2017
ECOOP 2017	*Speeding Up Maximal Causality Reduction with Static Dependency Analysis* **Shiyou Huang** and Jeff Huang 31st European Conference on Object-Oriented Programming, 2017
OOPSLA 2016	*Maximal Causality Reduction for TSO and PSO* **Shiyou Huang** and Jeff Huang Proceedings of ACM SIGPLAN conference on Object-Oriented Programming,Systems, Languages, and Applications, 2016

Work Experience

5/2018 - 8/2018	**Research Intern**, *Alibaba Group US.*, 525 Almanor Ave., Sunnyvale, California. Study the crashes caused by the improper use of Java Unsafe API, and present a bytecode-level transformation to enhance the safety of Java Unsafe API.

Research Projects

2015-Present Stateless Model Checking with Maximal Causality Reduction (MCR)

- ***Extend MCR to TSO/PSO*** In this project, I develop novel extensions to MCR by solving two key problems under TSO and PSO: 1) generating interleavings that can reach new states by encoding the operational semantics of TSO and PSO with first-order logical constraints and solving them with SMT solvers, and 2) enforcing TSO and PSO interleavings by developing novel replay algorithms that allow executions out of the program order. On average this technique explores 5X to 10X fewer executions and finds many bugs that the other tools cannot find.

- ***Use static dependency analysis to improve performance*** MCR is limited to the dynamic information of the program and has to construct complicated constraints to ensure the feasibility of the derived execution. In this work, I use static dependency analysis to reduce the size of the constraints, thus speeding up the efficiency of the state space exploration. I use the system dependency graph to capture the dependency between a read and a later event e in the trace and identify those reads that e is not control dependent on.

2016-2017 Concurrency Bug Reproduction Using Intel PT

In this work, I develop H3, a new *record* and *replay* system to reproduce Heisenbugs with commercial hardware features. H3 consists of two phases. First, users run the target program on a COTS (commercial off-the- shelf) hardware with PT enabled. Once a failure occurs, H3 collects the failure PT trace and generates the instructions executed on each core. Second, H3 infers the instructions executed by each thread based on the thread context switch log and generates a symbolic trace for each thread. It then constructs symbolic constraints with the core-based constraint reduction, and computes a global failure reproducing schedule with an SMT solver.

5/2018-9/2018 Memory Bound Check for Java Unsafe APIs

Java runtime provides an Unsafe API as a backdoor for the developers to access the low-level system code. The Unsafe API is powerful, but dangerous, which leads to data corruption, resource leaks and difficult-to-diagnose JVM crash if used improperly. In this work, I study the Unsafe crash patterns and propose a memory checker to enforce memory safety at the bytecode level to avoid the JVM crash caused by the misuse of the Unsafe API. I evaluate the technique on real crash cases from the openJDK bug system and real-world applications from AJDK.

Teaching Experience

Fall 2018 **CSCE111: Introduction to Computer Science Concepts and Programming**, Texas A&M University.

I work as an instructor to teach undergraduate students the basics about Java for this course. My responsibilities include designing the course schedule, deciding the content, homework, labs and exams.

Spring 2018	**CSCE314: Programming Languages**, *Haskell & Java*, Texas A&M University. Teaching Assistant
Spring 2017	**CSCE431: Software Engineering**, Texas A&M University. Teaching Assistant
Fall 2016	**CSCE606: Software Engineering**, Texas A&M University. Teaching Assistant

Honor & Awards

Fall 2018	**Graduate Teaching Fellow**, Department of Engineering, Texas A&M.
June 2017	**Student Scholar for 50^{th} Turing Award Celebration**.
March 2017	**Graduate Research Excellence Award**, Department of CSE, Texas A&M.
OOPSLA 2016	**SIGPLAN PAC Award**, OOPSLA, 2016.
ICSE 2016	**ACM's SRC Travel Award**, ICSE, 2016.
ICSE 2016	**SIGSOFT CAPS Award**, ICSE, 2016.
ICSE 2016	**Third Place, Student Research Competition**, ICSE, 2016.

Service

Sub-reviewer	**ICSE'19, PLDI'19, PLDI'17, PPoPP'17, FSE'16**.
AE Committee	**PPoPP'17**.

Skills

Research	Model Checking, Bug Reproduction, Testing, Static Analysis
Languages	Java, C/C++, Python, Ruby
Misc.	ASM, Z3, LLVM, Intel PT, Java Unsafe, JVM