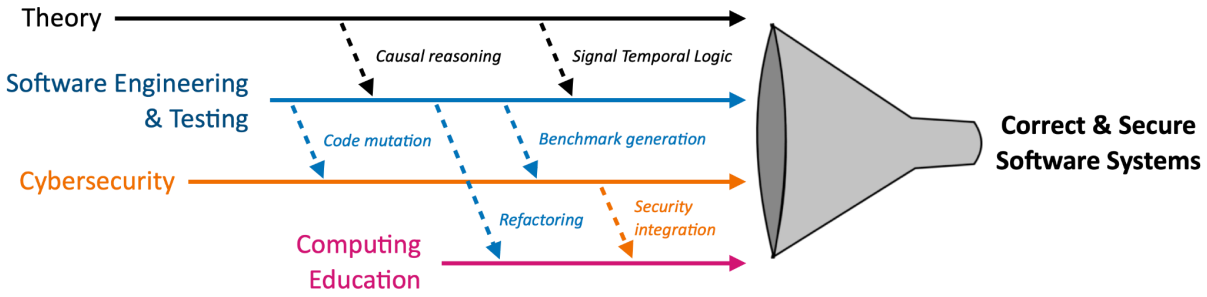
Research Statement

Chris Poskitt
School of Computing and Information Systems
Singapore Management University
Tel: (65) 6828-1376; Email: cposkitt@smu.edu.sg
17th December 2024

Research Overview

My research focuses on **engineering correct and secure software systems**, aiming to make them reliable and trustworthy, especially in contexts where dependability is crucial. I am particularly interested in adapting traditional software quality techniques to non-traditional and complex systems such as autonomous vehicles, critical infrastructure, and (more recently) AI agents. I work at the **intersection of theory and practice**, applying rigorous software theory to develop practical solutions for emerging domains. For instance, using signal temporal logic to formally define traffic laws, then fuzzing to find ways an autonomous vehicle could inadvertently violate them. Additionally, I explore how insights from my research can be translated to the classroom and help prepare the next generation of software engineers.



Some of the key overarching questions my research aims to address include:

- How can traditional approaches for software quality be generalised to complex cyber-physical systems?
- How do we build defence mechanisms for critical infrastructure that effectively detect/prevent attacks?
- How do we intelligently fuzz cyber-physical systems to uncover rare (but feasible) scenarios that violate safety properties?
- How can undergraduate curricula better prepare students to engineer correct, secure, and maintainable code?

Recent Research Highlights

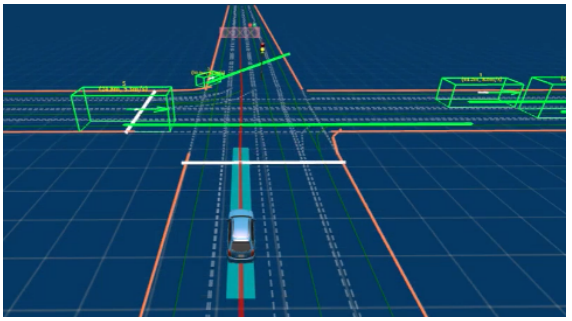
In the following, I summarise some recent results from projects I have collaborated on.

Testing Critical Infrastructure

In [1], we tackle the problem of generating diverse and meaningful test cases for industrial control systems (e.g. water treatment plants), as existing methods often produce tests that are too similar and fail to comprehensively explore different failure scenarios. We solve this problem through a guided fuzzing approach that identifies causal events in tests, generalises them into equivalence classes, and updates the fuzzing strategy as it goes to find new, causally distinct tests. In another paper [2], we tackle the problem of the lack of a formal and complete reference for the Structured Text (ST) programming language, which makes it challenging to verify the correctness of PLCs in critical infrastructure. We solve this problem by introducing K-ST, a formal executable semantics for ST, enabling the evaluation and comparison of different ST implementations to ensure consistency and correctness.

1. *Finding Causally Different Tests for an Industrial Control System*, C.M. Poskitt, Y. Chen, J. Sun, and Y. Jiang. **ICSE'23**
2. *K-ST: A Formal Executable Semantics of the Structured Text Language for PLCs*, K. Wang, J. Wang, C.M. Poskitt, X. Chen, J. Sun, and P. Cheng. **TSE'23**

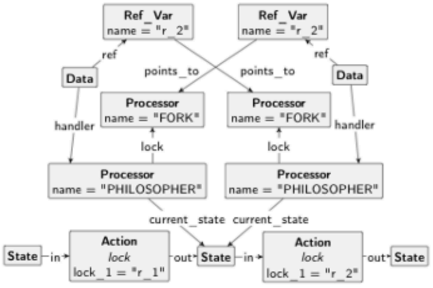
Fuzzing Autonomous Vehicles



We extended our earlier work on testing for traffic law violations [3] to *enforcing* compliance with those laws. In particular, our work [4] solves this problem by introducing REDriver, a runtime enforcement framework that monitors the planned trajectory of an autonomous driving system (ADS) and adjusts it using a gradient-driven algorithm when a potential violation is predicted, thereby enhancing adherence to the specified safety constraints with minimal disruption to the vehicle's journey. Nonetheless, accidents and safety violations still happen, and we need to understand the causes of them when they do occur. Thus, in [5], we propose ACAV, an automated framework that conducts causality analyses on AV accident records by extracting relevant features, simplifying the records, and applying a novel causal analysis tool to identify the causal events that led to the accident or safety violation.

3. *LawBreaker: An Approach for Specifying Traffic Laws and Fuzzing Autonomous Vehicles*, Y. Sun, C.M. Poskitt, J. Sun, Y. Chen, Z. Yang. **ASE'22**
4. *REDriver: Runtime Enforcement for Autonomous Vehicles*, Y. Sun, C.M. Poskitt, Z. Zhang, and J. Sun. **ICSE'24**
5. *ACAV: A Framework for Automatic Causality Analysis in Autonomous Vehicle Accident Recordings*, H. Sun, C.M. Poskitt, Y. Sun, J. Sun, and Y. Chen. **ICSE'24**

Reasoning about Graph-like Structures



As part of my PhD, I studied techniques for proving the correctness of problems modelled as graphs and algebraic graph transformations (i.e., “graph programs”). While no longer my core focus, I do maintain a thread of research in the community. For example, in recent work [6,7], we generalised Peter O’Hearn’s “incorrectness logic” to this setting. Unlike traditional program logics (which prove facts about *all* possible executions), an incorrectness logic proves the *presence* of certain executions, which could be particularly important for graph programs given the large degree of nondeterminism in most of their execution steps. I anticipate, for example, that my contribution could be used as a basis for sound reasoning in symbolic execution tools.

6. *Monadic Second-Order Incorrectness Logic for GP 2*, C.M. Poskitt and D. Plump. **JLAMP'23**
7. *Incorrectness Logic for Graph Programs*, C.M. Poskitt. **ICGT'21**

Software Engineering Education

In one line of work [8], we addressed the difficulty of teaching code refactoring to undergraduate students, noting that traditional methods using instructor-provided code can hinder the learning of novices. We solve this with a 'mistake-based' approach, where students first create code that intentionally contains 'smells’, allowing them to become familiar with the code they will refactor, thereby enhancing their ability to identify and correct these issues confidently. In a different line of work [9], we tackle the problem of students struggling to find credible and relevant supplementary resources to complement their (typically concise) lecture slides. We introduce Slide++, a web platform that automatically extracts keywords from lecture slides to search for and recommend relevant online resources, which are displayed alongside the lecture slides as students study.

8. *Fixing Your Own Smells: Adding a Mistake-Based Familiarisation Step When Teaching Code Refactoring*, I. Tan and C.M. Poskitt. **SIGCSE’24**
9. *Towards Automated Slide Augmentation to Discover Credible and Relevant Links*, D. Dinushka, C.M. Poskitt, K.C. Koh, H.N. Mok, and H.W. Lauw. **AIED’24 Late Breaking Results**