In recent years, deep learning techniques have achieved remarkable advancements across various domains, revolutionizing many fields of computer science. Inspired by these breakthroughs, my research aims to leverage the power of deep learning in the field of program analysis. Productively writing correct and secure code for software projects is a significant challenge. As such, program analysis has been an active research area for many decades. Many fruitful techniques based on rules and formal logic have emerged. Despite their successes, these approaches have some noteworthy limitations. Ultimately, my research endeavors aim to bridge the gap between deep learning and program analysis, harnessing the potential of these cutting-edge techniques to advance the state-of-the-art in software development, ensure code correctness, and bolster the security of software systems.

Neural Program Reasoning

In my PhD research, I produced novel applications of machine learning to the field of program analysis. Program analysis tools aim to check that a program satisfies some correctness property. These properties are typically handwritten universal rules of correctness (i.e. a NullPointerException should never occur). However, developers often want to check for program specific rules of correctness (i.e. “You must be logged in before posting to a blog application”). Program specific rules of correctness must be supplied to the analysis tool typically in the form of formal logic, greatly hindering the ease of use. The innovations in my PhD research address this shortcoming by inferring developer intent with deep learning. My innovations are broadly over three program reasoning domains: static program analysis\(^1\), merge conflict resolution\(^2\), and automated testing\(^3\). My work has achieved improvements in performance over traditional reasoning tools through innovations in novel input representations, creating large-scale program reasoning datasets, and refining neural modelling architectures. The works in each domain have the overarching goal of increasing ease of use of program reasoning tooling by inferring developer intent using neural models.

Future Work

I envision two broad synergies of program reasoning tools and language models that hold immense potential for enhancing software correctness and programmer productivity. Firstly, a general purpose neural specification inference library to address the scalability and ease of use limitations of program analysis tools. Inferring a function summary (pre and post-conditions) of a segment of code, could drastically improve the performance of current correctness checking tools. Secondly, to enhance the user experience of a program analysis tool, developers could interact with it using natural language. In the short term, this means expressing the desired correctness property in everyday language instead of formal logic. Looking further ahead, it could involve a unified application that utilizes program analysis tools to address developers’ questions about correctness.

\(^1\) Hoppity: Learning Graph Transformations to Detect and Fix Bugs in Programs. Dinella et al ICLR 2020
\(^3\) TOGA: A Neural Method for Test Oracle Generation. Dinella et al. ICSE 2022 (Distinguished Paper Award)