VIKRANT GAJRIA
User ID: vikrantgajria@gmail.com

Personal Statement
Applying to PhD and MS programs

I firmly believe that technological advancement today is indebted to decades of research in the field of programming languages. The level of sophistication that these languages have reached, to communicate elaborate logic to computers, fascinates me.

However, the reliance on software today begs the question of correctness and performance of our projects. Even the most experienced developers can err on mission-critical software, ending up with huge losses like the Japanese Hitomi satellite that was devastated due to incorrect parameters in the program.

I aspire to spend the subsequent years of my career studying programming languages at Carnegie Mellon University, especially the topics of functional languages, type systems, program analysis and verification, to help improve the reliability of software systems.

This decision to study programming languages is the culmination of my experience in academia and the industry. As a freshman, working on a real-world problem at Checkapp taught me the issues in developing reliable software, such as verbose yet incorrect logic in procedures and runtime errors in dynamically-typed languages like Javascript. Incidentally, this project of generating prescriptions from natural language was my first insight into language processing, which gave me a direction to find potential solutions to these problems.

In my sophomore year, I realized the fundamental problems that I discovered previously could be solved with better programming language semantics and runtime environments. As an intern at Lido Learning, I played a significant role by working closely with the CTO in major engineering decisions; such as designing a distributed system architecture for the start-up. There, I was introduced to Erlang and its use in fault-tolerant applications for higher availability, as well as functional programming for writing better logic. My foray into learning functional languages acquainted me with contemporary topics in programming language research like type theory and formal verification of programs.

The topic of programming languages wasn't the only field of computer science that I studied, but it was the field that stuck to me. In my junior year, I mentored dozens of my underclassmen with the knowledge and experience in software development that I garnered. Moreover, I got invited to mentor the participants of a hackathon organized by the ACM student chapter of my college. Interacting with several peers and helping them in their projects enabled me to explore the breadth of computer science, like data science, machine learning, and robotics. Yet, none of these fields solved the question of software reliability in my projects.

I developed an inclination towards a graduate program at IIT Bombay. The head of my department introduced me to Prof. Surya Durbha of CSRE as a result of my work as a mentor. I joined his lab to work on the remote-sensing apparatus developed by his PhD students. Working in a research environment taught me the stark contrast between industry and academia; prioritizing novelty and solving unique problems over delivering a product on time. Though my internship was put on a pause due to COVID-19, my experience with the PhD candidates in Prof. Durbha's lab motivated me to pursue a graduate degree for an extensive study of my field of interest.

With a definitive ambition to pursue research, I pondered upon the problems in software engineering that I encountered in three years of college to conclude that I wanted to concentrate on the study of programming languages. I immediately started to look for opportunities to work on these problems and learn from experts in this field.

Today, my research at Accelus Robotics aims to make the development of feature-rich yet portable languages easier, targeted towards programmable logic controllers (PLCs) in the manufacturing industry. Much of the implementations of standard PLC languages are proprietary, which makes it difficult to use or extend such languages. The language specification, compiler, and runtime that I am developing are based on the IEC 61131-3 specification and UCSD Pascal. One of my key achievements is the design of an efficient and extensible virtual machine inspired by the architecture of JVM. After my preliminary implementation, I studied Python's interpreter to improve the performance of my p-code virtual machine by 15-20% using the computed gotos extension of GCC. The liberty to make decisions and search for new approaches to solve problems like the performance and semantics of the programming language excites me more than my experiences in the past.

I wanted to conduct independent research to prepare myself for a such career. Hence, I started to look for opportunities by contacting my college professors. Co-authoring a paper on decentralized voting and a book chapter on blockchain in healthcare introduced me to academic writing. Thereafter, I approached two of my college professors with my project ideas that aim to improve the correctness of programs written in popular languages.

With the interest that I developed in studying type systems of languages like OCaml, I wanted to introduce developers to utilitarian type systems without the requirement of a strong mathematical background. My study of "Refinement Types for ML" by Prof. Pfenning gave me the intuition that they can be represented as pure functions in any language. To test this, I developed a library for Python that uses metaprogramming to implement refinement types using functions and decorators. The library rewrites the ASTs of functions to add Hoare logic - pre-conditions, post-conditions, and other type checks in the body - by leveraging the syntax of annotations in Python that curiously has no semantics. This approach of utilizing the extant syntax of a language would make the programmer feel comfortable to use the type system.

The drawback of my metaprogramming approach was the increased start-up time and the need to refactor existing codebases to add my library's decorators. I knew from experience that the industry is hesitant to edit large codebases. To overcome this problem, I came up with the idea of a static type checker and compiler to move the metaprogramming from run-time to compile-time. Unlike the previous approach that focused on adding semantics to Python itself, I took ideas from LiquidHaskell to design a language extension to parse and generate most of the Hoare logic at compile-time. I aim to present my results at a conference after implementing language-integrated verification using Z3 theorem prover.

My career goal is to explore and apply various methodologies to help developers build reliable software. I have been working towards this objective in my undergraduate program, and I plan to do so moving forward. My experience of working on real-world projects has shown me that with the mentorship of professors at CMU, the PhD program would help improve my understanding of unique problems in the realm of programming languages and dedicate a substantial amount of time to solve these problems with enthusiastic peers. If selected for the MS program, I would prepare for a PhD later in my career.

I want to continue my study at CMU because of the intriguing projects in programming languages by the faculty and students of Principles of Programming group. I believe my career goal and research interests align well with those of the members of this group, which is why I would like to work with many of them in their projects; such as the works in type systems and metalanguages by Professors Pfenning, Harper, and Hoffmann. I plan on pursuing a research career after earning my graduate degree, with an inclination to solve the issues faced by software developers in the industry. The experiences and knowledge that I would gain in the graduate program, coupled with the resources and opportunities at CMU, would help me achieve my career goals and the goals of like-minded people at the university.