The abstraction that a programming language provides influences the structure and algorithmic complexity of the resulting programs: just imagine creating an artificial intelligence engine using assembly or building an operating system in a language disallowing direct access to memory. My goal is to use programming languages to improve the reliability and extensibility of software systems.

I am currently working with Professor Greg Morrisett on polymorphism in Haskell. For my senior thesis, I am interested in using run-time type analysis to provide more expressive language constructs for polymorphism. By exposing me to features such as higher-rank polymorphism, Haskell has taught me what one can expect of a language. Using Haskell has also made me more aware of the influence of research languages in mainstream programming: the most recent is the new C++ concepts inspired by Haskell's type classes. My experience so far confirms that I enjoy programming languages research and that what I want to do is relevant to the rest of the world.

My decision to study programming systems is the culmination of my experiences in a variety of areas. My positive experience with a high school robotics program led to my college participation in RoboCup, an international autonomous robotic soccer competition, with the Harvard-MIT team RFC Cambridge. Soon after joining the newly-formed team I found myself technical director of a computer science subteam frantically constructing a vision system that processes information from two overhead cameras to determine the locations of the robots and ball, an AI system that determines what the robots should do, and a hardware interface for sending commands via radio signal.

RoboCup taught me how to approach complex technical problems. Developing the vision system alone required an immense amount of consideration: the first time we left our tiny, low-ceilinged clubhouse and arrived at the US Open arena with its regulation-sized fields and high camera bar, we discovered that the additional field space and mounting height caused so much peripheral distortion that our system could not see at crucial locations on the field, including the kickoff circle. We watched in wonder as other teams lay down sheets of dotted paper for geometric calibration. After realizing that our problem was even larger than anticipated, we read technical papers, talked with other teams, and consulted with professors to improve our systems. At the international competitions in Bremen, Germany, and Atlanta, Georgia, our robots could see to kick off.

Participating in RoboCup taught me more than how to build soccer-playing robots. Most importantly, I learned about the planning and coordination involved in building a large, multi-part system with a team of people with varied skills. I also learned what a team could accomplish. When my teammates proposed in December to make the February US Open qualification deadline, I was skeptical because we had not even begun mechanical design. The team effort surpassed my expectations. We went from having nothing to having three goal-scoring robots at the US Open, from having a small amount of departmental funding to having tens of thousands in sponsorship, and from having a tiny clubhouse to acquiring a full-fledged lounge. My experience made me much confident about committing myself to ambitious dreams.

Though I enjoyed RoboCup, I wanted to solve problems that I found more relevant. Because of this and because of my interests in biochemistry, I decided to do research in computational biology. I worked at the Center for Biomedical Informatics (CBI) at Harvard Medical School to trace the evolution of a set of presynaptic receptors. We were particularly interested in the evolutionary boundary between hydra and sponge: while sponges are not known to have central nervous systems, hydra have fully functioning ones. Gaining insight to the role of these receptors is useful because they play a role in the development of the central nervous system and many are implicated in autism.

At CBI I was able to complete my analyses ahead of schedule. My initial goal was to examine protein sequences to trace the evolution of the receptors LAR and Liprin-a to their points of evolutionary emergence. By mid-summer I had combined existing methods of statistical sequence alignment and domain identification to develop a method for tracing the evolution. I applied the method to the larger set of presynaptic receptors and presented my procedures and results at UC Santa Barbara at a mini-symposium. After a delay waiting for the release of a clean nematostella genome, we are now preparing our results for publication.

My experience at CBI was formative. Working in a multi-disciplinary area taught me that most problems are not contained in a single field. Observing how others approach problems taught me how to ask questions and how to develop methods for answering them. From having to learn evolutionary biology, I learned to teach myself from textbooks and technical papers. Because I was given independence in my work, I learned to make my own decisions, to communicate my results often and effectively, and to ask for feedback and assistance when it was necessary. I left CBI feeling positive about research but still searching for a problem that really excited me.

I found this in computational tools. After courses in compilers and hardware, I became interested in the tools that make the applications possible. For a course project in biologically-inspired distributed systems, I worked on decentralizing Google's MapReduce paradigm for distributed computing. The motivation for the project is that as chip manufacturers increase number of cores per chip, it becomes necessary to find ways to distribute tasks among the processors. Google's MapReduce algorithm, which abstracts away the division of labor in parallel programs, suggests a solution. However, the algorithm involves a centralized master node which becomes a bottleneck for information flow as the number of computing agents increases. My partner and I proposed an amorphous computing-inspired solution in which communication occurs by the passing of gradients. Considering the possibility of so much processing power left me in awe of the potential of computational tools.

My experiences have continued to confirm the importance of having the proper tools. In my programming languages course, I learned about appropriate language abstractions. When I spent a summer at Google, I saw how much widely-used abstractions could be improved. Though I enjoyed the craftsmanship involved with writing C++ code disciplined enough to be correct, efficient, and readable, I saw a glaring need for language constructs with better abstraction mechanisms, better error prevention mechanisms, and better ways of demonstrating code correctness. However, I also noticed that production pressures and the need for backwards compatibility made tool development a much lower priority than product development. After some investigation of other companies, I concluded it was best to return to academia to work out my ideas for better build systems, better error-checking mechanisms, and better language constructs for data abstraction and correctness guarantees.

My career goal is to remain in academia as a professor so that I can pursue my research interests while training and recruiting others to solve relevant problems in programming languages. I want to continue my studies at Carnegie Mellon because it has a large programming languages group with many professors with whom I would like to work, including Professors Harper and Aldrich. Because I find it important to talk to with different areas of focus than my own, I also like the diversity of CMU's computer science research. Because I look forward to teaching as a graduate student, I like that CMU has a reputation for having motivated undergraduates who can challenge my understanding of the materials. For these reasons, I would like to spend my next six years at Carnegie Mellon.