Statement of Purpose
Jeremy Lacomis

As a returning adult transfer student, I have tried to make the most of my limited time at the University of Virginia. I have leveraged my motivation to do well in upper-level courses and gain experience with the many aspects of the research process. I have enrolled in a graduate-level programming languages seminar that requires the ability to read and understand several conference and journal papers per week. I am a coauthor of a paper that is in submission to Transactions on Software Engineering. I was an essential contributor to this research and wrote a large portion of the initial drafts of the text. I believe that my motivation combined with my classroom and research experience have well prepared me for a graduate career at Carnegie Mellon.

My research focuses on using automated techniques for software improvement, particularly finding novel optimizations in software with minimal human intervention. I developed a technique that reduces the power consumption of software by automatically modifying the assembly code output of a compiler using techniques from search-based software engineering. To minimize power usage, the technique checks the power consumption of programs through either a model of power or data from a real-world power meter. By modifying assembly instructions and measuring or modeling the power consumed by the mutated program, the technique is able to find a series of modifications that lead to a more efficient but still correct program.

I was impressed by the success of the technique in preliminary experiments with small test programs. Motivated by its effectiveness and the growing expense of power in data centers (projected to cost American businesses $13 billion per year by 2020), I began applying it to indicative examples of software found in server farms. Although there was progress on these larger benchmarks, I observed that it was impractically slow at scale. I decided to take the time to investigate if there were opportunities to speed up the search.

While analyzing the results of searches on larger programs, I was able to identify three key factors that caused slowdowns. First, candidate programs created during the search often did not assemble, wasting time during generation. Second, candidate programs were often semantically identical to others generated earlier in the search, causing redundant energy measurements. Finally, many mutations change code in locations that are rarely or never executed, which have little or no impact on power consumption and cause the search to spend extra time in the generation and evaluation stage.

Encouraged by these observations, I designed and implemented algorithmic modifications to improve the effectiveness of the technique and allow it to scale to larger programs. By carefully implementing code to recognize and reject classes of mutations that do not assemble or are semantically identical to earlier programs I was able to completely prevent their generation, reducing the search space of some programs by over 90%. To confine the search to only parts of the program that are actually executed, I created an execution profile of each program, counting the number of times that each instruction was run. I then designed software to use this profile to map the execution counts to assembly source, weighting the probability that each line in the program would be modified. By combining these two algorithmic improvements, I was able to meaningfully direct the search to perform mutations on a small fraction of the original source lines. For example, in one of the larger benchmark programs, 2.2 million lines of assembly instructions were reduced to only 17,454 valid modification locations.

Testing this technique in the real world required both hardware and software experience. Measuring real-world power consumption required cost-effective low-noise hardware with enough temporal resolution to rapidly test tens of thousands of programs. Since I was unable to find an affordable device meeting these requirements, I helped design and construct hardware to rapidly measure power in a server environment. To test software indicative of what is commonly used in data centers, I identified interesting examples and adapted them for use with our technique. I was responsible for designing and running experiments on benchmarks that I chose, and helped collect and analyze data from over six months of combined experimental runtime. By investigating assembly code modifications by hand, I was able to identify and explain the mechanism of many difficult-to-understand optimizations, some of which involved changes as complex as constant data being interpreted as x86 instructions or an overflow changing a loop guard in an optimal way. I used the understanding gained during these investigations to further refine search space reductions. Although data are still being evaluated, results are promising.

My career goal is to continue to explore the edges of human knowledge as a research scientist. My real-world experience and the advice of friends and colleagues have shown me that I will need a PhD to work on the types of problems that interest me. I believe that Carnegie Mellon is a good fit for me. I am interested in much of the current research in the Software Engineering group, and would particularly like to work with Professor Le Goues on automated program repair and Professor Garlan on self-adaptive systems. I also believe that building on my graduate class and research experience will allow me to get the most out of a graduate career at CMU. The resources and breadth of computer science research performed at CMU will provide me with the experience that I need to pursue my passion in computer science.