**Research Summary:**  The one-page research summary gives the nominee the opportunity to describe research project(s) in detail.  While the award committee comprises computer science professors, it’s likely that most of them will not have expertise in the specific field in which the nominee worked.  So, while one can use well-known concepts without defining them (e.g., names of data structures and terms like “NP-completeness”, “compiler”, or “k-means clustering”), one should imagine writing to a computer science professor who works in a different subdiscipline.

It’s important to explain the problem, its significance to the field and/or society, what was challenging about the work done, and – most importantly – the nominee’s contributions.  Be specific and precise, as it is critical for the awards committee to understand and evaluate the individual contributions of the nominee; if the project involved other people, it’s important that the nominee’s particular contributions to that project be described explicitly.

**The Problem**

By default, Haskell uses lazy evaluation such that expressions are not evaluated until they are absolutely needed. While lazy evaluation has many advantages, it also can result in serious performance costs through excessive memory allocations due to the accumulation of too many unevaluated expressions. To alleviate this problem, Haskell allows users to force eager evaluation at certain program points by inserting strictness annotations, known as bangs. Unfortunately, manual bang placement is labor intensive and difficult to reason about. The Autobahn 1.0 optimizer uses a genetic algorithm to automatically infer bang annotations that improve runtime performance. However, Autobahn 1.0 often generates large numbers of superfluous bangs, which is problematic because users must inspect each such bang to determine whether it introduces non-termination or other semantic differences. My research aims to minimize bangs while maintaining performance in the optimized Haskell programs.

**Project Details**

Autobahn 2.0 uses GHC (Glasgow Haskell Compiler) profiling information to add a Pre-search and Post-search phase to Autobahn 1.0. We define source code locations that are associated with a large runtime cost in the GHC profile report as a *hot spot*, and locations that are not associated with a large cost as a *cold spot*. The Pre-search phase reduces the search space for Autobahn 1.0’s genetic algorithm by preserving files that contain at least one hot spot, and eliminating files that do not. The search space reduction allows Autobahn 1.0 to produce far fewer and potentially more useful bangs. The Post-search phase tests and removes bangs that have minimal performance impact. Bangs located in cold spots are first permanently eliminated, and bangs located in hot spots are individually tested for whether they cause significant performance slowdown when removed. This phase removes unnecessary bangs and trims down the number of bangs generated by Autobahn 1.0.

When evaluated on the NoFib benchmark suite, Autobahn 2.0 reduced the number of inferred bangs by 90.2% on average, while only degrading program performance by 15.7% compared with the performance produced by Autobahn 1.0. In a case study on a garbage collection simulator, Autobahn 2.0 eliminated 81.8% of the recommended bangs, with the same 15.7% optimization degradation when compared with Autobahn 1.0.

Autobahn 2.0 assists Haskell programmers that wish to use strict evaluation to improve their program performance but do not have the knowledge or time to do it from scratch. The improvement on Autobahn 1.0 both provides a less time consuming user experience and potentially improves code quality and program safety by eliminating bangs that are distractingly useless or cause non-termination.

**Nominee Contributions**

I was responsible for implementing and debugging Autobahn 2.0, coding scripts for evaluating the tool on different benchmarks, and running experiments and collecting results. Professor Kathleen Fisher provided guidance for generating solution ideas, oversaw the progress of the project and helped write the research paper.

**Personal Statement:**   The personal statement (not to exceed 300 words) should address interests and activities outside of research (e.g., club leadership, teaching, peer tutoring) and/or special circumstances or adversity that the student has confronted.

As a women studying computer science, I am interested in activities that support and empower female-identifying individuals in technology. I am both an active member of Tufts Women in Computer Science and served as the Conference Logistics Coordinator for the second annual Tufts Women in Tech Conference in September. The conference included discussion panels, technical workshops and networking opportunities, and created a platform for female and non-binary high school and university students to connect. For the four months leading up to the conference, I was in charge of arranging conference venues, coordinating the registration process, preparing conference materials, and planning shifts for a team of 30 other conference volunteers. On the day of the conference, I worked alongside my logistics committee members to facilitate operations for ensuring a smooth conference experience for all 250 attendees and sponsors.

I started being a Teaching Assistant (TA) at my school in my sophomore year, and became a Teaching Fellow (TF) for the Programming Languages course in 2018 spring. My TF role is to support all other 20 student TAs and facilitate communication between them and the senior course staff. Every week, I lead a TA staff training meeting to discuss teaching strategy and tips for the upcoming assignment. I also help new TAs better transition to their role at the beginning of every semester by organizing a day of TA training and periodically visiting them at office hours. Outside of my TF duties, I still perform regular TA tasks such as holding office hours and grading assignments. I enjoy directly interacting with students to help them understand course material and sharing my knowledge on and passion for the topic. I have received immense amounts of guidance and help from mentors and upperclassmen when I started in Computer Science, and I strongly believe in doing the same by giving back to the community.