Teaching Statement

William T. Hallahan

As an educator, my aim is to increase the knowledge and curiosity of the students, and give them the information and skills they need to succeed not only in the class, but independently in pursuits outside the classroom. During my time as a graduate student, I have acted as a teaching assistant for seven semester (five more than required) because I find working with and supporting students is exciting and fulfilling. I have also advised multiple undergraduate research projects, and have enjoyed the opportunity to introduce undergraduates to the research process, resulting in publications at top tier conferences such as PLDI, and AAAI [1] [3]. I am also proud that one of these undergraduate student won the National Science Foundation Graduate Research Fellowship. During the fall of 2020, I designed and advised a directed reading course, which included reading and discussing academic papers and programming with advanced features of the Haskell language.

I've been interested in tutoring and teaching since college, where I frequently tutored high school students, both in one-on-one settings, and in small groups. During the second semester of my senior year, I led a tutorial session for an Algebraic Structures class, where I led a group of students in solving additional example problems and writing additional proofs, beyond those covered in the normal class period. I know from personal experience that a single class or teacher can have a strong positive influence. My interest in formal methods began after I talked to one of my undergraduate professors and he briefly mentioned program verification. This conversation led me to begin reading more about formal methods and ultimately led to me choosing my current direction in graduate school. Of course, one cannot have this large an impact on every student, but I still hope to spark some curiosity in every student that I work with.

Advising Experience As a PhD student, I coadvised multiple students with my advisor, four example of which I will briefly highlight. The first such student, Anton Xue, worked with me on the development of a symbolic execution engine for Haskell programs, and eventually co-authored two papers (one at a top tier conference) with us on that topic [1] [2]. Anton is currently a PhD student at the University of Pennsyvania, where he continues to work in formal methods. I also worked closely with an undergraduate student named Max Levatich who spent a summer exploring synthesis and automated repair of code, and a student named Rushyendra Maganty who worked on adding additional features to the symbolic execution engine. Max has also gone on to pursue a PhD in formal methods, at Columbia University. Finally, I coadvised two high school students, Kairo Morton and Elven Shum, who were interested in the combination of formal methods and machine learning. This work led to a publication [3], with Kairo as the first author.

I believe successful advising is about guiding and encouraging. In order to be successful in research, I believe it is important to be excited and passionate about the topic, so I aim not to convince students to work on specific projects that I choose, but rather find and develop projects that fit both within my and the students interests. Along these lines, I have learned from every student I have worked with- it is important to remember that advising is not at all a one way street. A particularly striking example of this is with Kairo and Elven. Despite being high school students, both had significantly more experience than I did with machine learning frameworks and concepts. While I taught them about formal methods, they taught me about machine learning. The project we worked on and published would not have happened without their input.

Teaching Interests As someone who conducts researching in verification and synthesis, I would be most excited to teach courses that explore one or both of these topics (for graduate students, undergraduate students, or both.) I could also teach courses on software engineering, or, of course, core undergraduate computer science courses. In addition to serving as the teaching assistant for verification courses, I am experienced in working with large courses and diverse subject areas. I previously served as the sole graduate teaching assistant for a systems programming course with over 100 undergraduate students in it.

I would be interested in designing and teaching a course introducing students to verification and synthesis techniques. In this course, I would introduce students to the theoretical insights that make such work possible, and show how those insights can be practically used to build real world tools. By covering both verification and synthesis in the same course, I aim to highlight the connection between the two areas. Verification is essential to certain synthesis techniques (such as CEGIS) while synthesis can aid verification (as in loop invariant synthesis.) To give students concrete experience, as part of the course they would work (individually or in small groups) on tools applying verification and synthesis techniques. To make the course as engaging as possible, I'd aim to give students a certain amount of latitude and the ability to design their own plan. For example, some students might be most interested in designing a verifier for a small, domain-specific imperative language, while others might be more excited to applying synthesis techniques to a small fragment of SQL.

During the time I was the teaching assistant for Software Analysis and Verification, three students who have taken the class have gone on to work on research with me (and other students from the class have gone on to work on research with other members of my advisors research group.) My aim is to design courses that are well suited both to students interested in pursuing research and industry. I would encourage any students in a software verification course I was teaching to come and talk to me if they had an interest in diving deeper into software verification research.

Teaching Experience At Yale, I have served as a teaching assistant for four courses, some multiple times. In total, I have been a teaching assistant for seven semesters. For each class, I held regular office hours and graded homework assignments and tests. In many of the classes, I assisted in the development and writing of tests, homeworks, and projects.

The course I have most frequently been the teaching assistant for is Software Analysis and Verification. During Fall of 2020 this course, like almost all courses at Yale, shifted to online only. With the courses instructor, I worked to adjust the course to this new format. The largest change was the replacement of the midterm and final with two new projects. For both projects, I helped design and write the material outlining the requirements and guiding the students through necessary steps.

Yale allows students who have an academic interest not covered by the normal curriculum to, with the assistance and guidance of a faculty member, design their own individual Directed Reading course. In Fall 2020, I advised a Directed Reading course for an undergraduate student, Laurence Lu, who was interested in a deep dive into functional programming, and, in particular, Haskell. Initially, Laurence created an outline heavily focused on exploring various monads an important, but far from all encompassing, topic in understanding Haskell. I helped Laurence put together a curriculum for the course that covered not just monads, but also a wider variety of relevant topics, such as GADTs, linear types, property based testing, and LiquidHaskell. Throughout, the goal was to ensure the course would be interesting to Laurence, while also exposing him to new ideas and covering a variety of topics related to programming in Haskell.

My Work

[1] William T Hallahan, Anton Xue, Maxwell Troy Bland, Ranjit Jhala, and Ruzica Piskac. Lazy counterfactual symbolic execution. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 411–424, 2019.

[2] William T Hallahan, Anton Xue, and Ruzica Piskac. G2q: Haskell constraint solving. In *Proceedings of the 12th ACM SIGPLAN International Symposium on Haskell*, pages 44–57, 2019.

[3] Kairo Morton, William Hallahan, Elven Shum, Ruzica Piskac, and Mark Santolucito. Grammar filtering for syntax-guided synthesis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1611–1618, 2020.