Teaching Statement

Ningning Xie

One of the most important reasons for my pursuit of an academic career is teaching and mentoring students. I believe that good teaching gives students a firm grasp of *what* they have learned, and more importantly, affects *how* they think. I was fortunate to meet numerous inspiring professors and advisors, and I truly look forward to teaching and advising my future students.

Teaching Experience. I have experience with teaching and tutoring undergraduate students, graduate students, and software engineers.

Teaching assistant for undergraduates. I have served as the teaching assistant for the undergraduate-level course *Functional Programming* on theory and practice of functional programming, and *Principles of Programming Languages* on fundamental concepts of programming languages. Each one had around 100 students. I gave weekly tutorials for both courses, held office hours, answered questions on the course forum and via emails, and designed and graded homework assignments. During teaching, I paid particular attention to motivating students to learn the course in the first place. I designed our homework assignments to be a gradually learning experience of building a language interpreter extended with increasingly challenging language features. This way, students actively engaged in the course to learn the materials needed to accomplish the assignments. At the end of the course, almost all students successfully finished the language interpreter, and many had even added support for extra language features.

Tutorial presenter for graduates. Together with my collaborator, I ran a 3-hour tutorial on *programming with algebraic effects and handlers* at ICFP, a premier conference in functional programming held online during the pandemic. I co-designed the tutorial, presented multiple parts of it, and held Q&A after each part. The tutorial had around 40 attendees, primarily graduate students. The tutorial was recorded and is freely available online, and its video has attracted more than 600 views within two months of release. For this tutorial, my approach is to combine theory and practice. In particular, the tutorial demonstrates practically useful programming patterns with various examples, and in the meantime overviews key research concepts from the underlying theory. This way, the attendees not only learned how to use algebraic effects and handlers, but also gained a solid understanding of their theoretical foundation.

Instructor for software engineers. I have served as the instructor for the course *Haskell 101* at Google, a three hour introduction to functional programming and Haskell taken by more than 80 Google engineers. This is a particularly challenging experience because those engineers are from various backgrounds, and are already used to one or more non-functional programming languages. For this course, I focused on teaching by example. When I introduced basic concepts in functional programming and Haskell, I showed implementations of the same program using different programming languages. This way, the concepts were better conveyed as they built connections between functional programming and other programming paradigms they may already be familiar with.

Teaching Philosophy. My teaching philosophy is to help students develop an understanding of programming from first principles. Broadly speaking, programming is a systematic mechanism for applying mathematics and logic to practical problems. Thus, my teaching focuses on improving students' understanding of the theoretical principles, and helping students develop rigorous and critical thinking.

It is essential to motivate students in the first place, as motivation drives students to learn actively. One practical way is to set up a sequence of specific and measurable goals, such as milestones towards an exciting course project. My experience setting up a language interpreter gradually extended with increasingly challenging language features suggests that students would feel more actively engaged this way. This project-driven learning style makes it clear where to start and how to achieve the results, helping students build confidence to approach new materials and gain a sense of accomplishment after finishing the course.

The interaction between the instructor and students also plays a vital role in guiding students to learn fast and deeply. Instead of showing students directly the solution to a problem in my course tutorials, I always do live programming. Live programming allows students to think about the problem by themselves first, and my instant response to their attempts allows them to realize immediately whether their attempts make sense or not, promoting critical thinking. Another important aspect is the feedback to assignments. While using an auto-grader to mark assignments, I still check the assignment submissions manually to learn what mistakes are made and provide detailed comments and constructive suggestions for improvements.

Similarly, during my conference tutorial, I always answered any questions immediately in the chat or during the tutorial, helping the audience understand concepts quickly.

Last but not least, a good instructor should be consciously aware of the curse of knowledge. That is, years of conducting research in this field makes it difficult for me to recall the inherent difficulties of a subject for someone with little background. Therefore, I always try to understand the audience and tailor the lecture accordingly. My experience with teaching Haskell to Google engineers suggests that teaching by example is an effective way to make abstract concepts concrete. In particular, establishing the connection between what audiences already know and the new subject provides a complete view of how things work from first principles.

Future Courses. My past teaching experience has prepared me well to teach a number of courses, and I am particularly interested in teaching courses related to the broad concept of programming languages. I believe it is immensely valuable for the students to learn and understand the principles of programming languages, which can provide a different perspective of programming (i.e., understanding programming from its foundation rather than simply viewing it as a tool).

At the undergraduate level, I can design and teach a wide variety of introductory courses, including imperative programming, object-oriented programming, functional programming, and principles of programming languages. I can also teach compilers and software engineering, especially for which an exciting course project can be set up. For those introductory courses, I would also like to design lectures to illustrate real-world interests and applications. For example, for the functional programming course, I can briefly overview the industry-leading Haskell Compiler GHC, and demonstrate its use cases in the research community and the industry.

At the graduate level, my research experience qualifies me to teach more advanced courses like programming language theory, type theory, and formal verification. Furthermore, I am excited to hold seminars or design new courses based on my research experience and work from the broader community. That includes specific research areas like dependent types, gradual typing, algebraic effects, program synthesis, etc., and the intersection between programming languages and other broader areas.

Research mentoring. I enjoy my experience in mentoring graduate students. I met with each student weekly and advised on their projects. Some students I have mentored have already published papers in top-tier venues for programming languages, and we also have co-authored papers under submission. I believe these experiences have prepared me to advise students as a faculty in the future. Based on my experience, I would like to emphasize several aspects of effectively advising students.

Students should be self-motivated. Self-motivation allows students to enjoy what they are doing. One way to achieve this is to let students have their *own* projects, instead of simply doing what they are told to do. For example, let students participate in a project in its early stage, even if there is nothing but a research problem. This way, they gain ownership of the project and can feel self-motivated to move forward. I would also encourage students to explore freely in the research area and find their research problems. Such exploration allows students to find what they truly feel excited about, and gives them the opportunity to lead a research project.

Effective communication between the advisor and the student is essential for the student to progress and not get stuck on a problem for too long. Weekly meetings are one way to report progress regularly. But sometimes it can also be inefficient as the students may wait until the weekly meeting to discuss any issue. We address this issue by making use of Slack, an instant workplace communication tool. Students are encouraged to openly ask questions and discuss their ideas on Slack. According to the discussion on Slack, we can set up a follow-up physical meeting if needed.

There should always be a clear and rigorous research plan. A useful research plan covers the research problem with its motivation, the ideas and directions to explore further, and related literature. The research plan can then be used to evaluate students' performance, ensure they are on the right track, and prevent students from spending significant efforts in solving unrelated issues. The research plan can be revised and polished depending on the research progress. A proper evaluation also makes sure that I can give timely advice.

Diversity. Mentoring, including but beyond research mentoring, is also crucial for diversity. Outside the university, I have served as a mentor in PLMW, a programming language mentoring workshop, at two top conferences at programming languages (SPLASH 2020 and POPL 2021), and also as a long-term mentor for SIGPLAN-M (SIGPLAN long-term mentoring). During the mentorship, I discuss with senior undergraduates and masters about graduate school applications, balancing research and life, career planning, etc. Those mentoring experience has prepared me well to build a diverse research group.