Teaching Statement

TALIA RINGER

My goal as an educator is to help students build the knowledge, enthusiasm, curiosity, and independence to succeed in their careers and to continue to teach themselves. It can take just one lecture or conversation to spark a lifetime of curiosity. I experienced this as an undergraduate: I had not taken any programming courses before college, and had been interested primarily in mathematics. Late in college, my compilers professor briefly explained the relation between programs and proofs. This unified everything I'd known about mathematics, philosophy, and computer science, and I have not stopped thinking about it since. I will know I have succeeded as an educator if I can spark the same curiosity in my students, and give them the resources they need to follow that curiosity.

I am excited to bring this philosophy to both my teaching and my advising. When it comes to teaching, I am most excited to teach undergraduate and graduate courses about programming languages and program verification. I would particularly enjoy teaching an advanced undergraduate programming languages course centered around the design and implementation of a programming language or productivity tool. I would also enjoy leading a broad computer science research seminar for undergraduate students who are interested in research but do not know where to begin. I could also teach a traditional compilers course. With preparation, I could teach undergraduate courses on computer security, software engineering, or other core computer science topics.

Teaching Interests. Programming languages is at a crossroads between theory and practice: it draws on elegant theoretical foundations to build practical tools that help programmers. I am well prepared to design programming languages courses that highlight both theory and practice. This is thanks in part to my background: I was drawn to programming languages by its connections to mathematics, then during my time as a software engineer at Amazon saw that many of the beautiful theoretical concepts that I had learned were applicable to my daily work.

I will bring this perspective to any class that I teach. For example, I hope to teach a project-based advanced undergraduate programming language class. I will frame such a class around building a tool with concrete applications, like a domain-specific language or framework for the productivity of software engineers, or a tool to test programs or prove them correct. Along the way, I will highlight the mathematical underpinnings that make building such tools possible, and give students who are inclined the freedom to explore and expand on those concepts.

I also enjoy introducing undergraduate students to computer science research. During my first year of graduate school, I helped an undergraduate student that I was mentoring start an undergraduate research seminar. I helped the student organize the seminar, choose accessible reading material, and lead discussions. The papers spanned domains across all of computer science, and each lecture brought in a graduate student or faculty guest from the domain of discussion.

Teaching & Diversity. Reaching students with different backgrounds is not just about engagement—it is also about diversity. Catering to just one background will reinforce the demographic most likely to have that background. This shows up, for example, when considering to what degree to emphasize the mathematical underpinnings of programming languages: in 2016, among bachelor's degree earners in the US, women were twice as represented in mathematics as in computer science, while Black students were half as represented in mathematics as in computer science [1]. It is prudent to teach with an emphasis on *both* practical applications for building software systems and on mathematical underpinnings, and to build bridges between computer science and other fields. I myself came from a different field, and I was almost scared out of computer science during my first year as an undergraduate, when a lecturer encouraged me not to pursue computer science after I struggled to get a B in his course. I will carry this experience with me as an educator and be sure to always encourage students who are new to the field.

Teaching Experience. I have put my philosophy into practice as a teaching assistant three times, each time with increasing autonomy. During my senior year as an undergraduate, I was a teaching assistant for the undergraduate course in computer and network security. I held office hours and

review sessions before exams. I paid close attention to how each individual student learned and catered to different student backgrounds. Feedback from students was overwhelmingly positive, and most students succeeded in the course.

During my second year of graduate school, I was a teaching assistant for an undergraduate compilers and programming languages design course. I taught sections, and I was given autonomy for how I taught those sections. I chose to give short 10-15 minute lectures and follow them with interactive labs that I designed. Students completed these labs in groups that matched less experienced students with more experienced students. In the meantime, I helped students and made sure that all students in each group were included and heard.

Later in graduate school, I was the sole teaching assistant for a graduate programming languages course. I helped design the course material and wrote the homework assignments myself. I adapted most of the homeworks from a book by a well-known professor at MIT that had been too difficult for his students. I successfully adapted and tutorialized homeworks to make them easier for students. I also made sure to give students elaborate constructive feedback when grading assignments, and to point interested students toward exciting relevant research.

Remote Teaching Experience. Due to unforeseen personal circumstances, I completed my third teaching assistant role while remote from the students—in 2018, long before the pandemic. I learned a lot from this experience that will help me make courses accessible online in the future, which may help not just during a pandemic, but also in reaching students with disabilities or with responsibilities at home. I attended all lectures virtually, while students attended in person. Students and I used an online text chat service to communicate, and set up video calls when needed. I had dedicated office hours during which I was guaranteed to be reachable on the text service and by video chat, but I found that students often preferred to contact me when most convenient for them. When students were stuck, I set up times for interactive pair debugging: I ran students' code locally, then responded with hints that would help them arrive at an answer. I found that this style of interactive debugging worked better than interactive debugging in person. Feedback was positive, and I plan to apply this style of debugging to future online, hybrid, and even in-person classes.

Advising. I advised two undergraduates on their own projects during my time as a graduate student. I coadvised these students with my own advisor: I met with each student weekly and my advisor joined every other week. The first project led to a publication at a top type theory conference; the student is now a graduate student at Carnegie Mellon University. The second project culminated in an honors thesis; the student is now a software engineer at Github.

My advising philosophy is to communicate well with my students, learn from them, and give them the tools that empower them to succeed as independent researchers. Rather than choose a project for a student, I prefer to work with them to design a project they are passionate about, since I believe students thrive most when they are passionate. I believe that advising is most productive when the advisor is not just teaching the student, but also learning from the student and adapting.

For example, my second student came to me with an interest in verifying a class of distributed data structures. I helped the student refine that interest into a project, while also teaching him how I refine broad interests into research projects. I used my expertise in my thesis area and in writing to help the student; the student taught me about the distributed data structures in return. Together, we worked through a book on a verification tool that neither of us had used before. Throughout, the student stretched the boundaries of my own knowledge; both of us came out of the relationship as better researchers. This is the kind of relationship that I strive for with my students.

REFERENCES

[1] National Science Foundation, National Center for Science, and Engineering Statistics. 2019. Women, Minorities, and Persons with Disabilities in Science and Engineering. (2019). https://www.nsf.gov/statistics/wmpd