Teaching Statement

Ben Greenman

2021-11-29

Teaching is very important to me. The main reason that I am applying to faculty jobs rather than industry positions is to have the opportunity to teach a variety of students. Why? For one, I enjoy the interactions and challenges that come with teaching. Students always end up teaching me something new, at least by sharing their perspectives, and some make intellectual contributions to research projects. On a deeper level, teaching is meaningful. Computing skills are useful for any person to know and essential for many jobs in todays' workplace.

Experience

Teaching. At Brown, I organized and managed a graduate-level seminar focusing on type systems for data science—specifically, for tabular data. This seminar inspired a student-led research project on type system design. The work is ongoing, but has already produced one publication. At Northeastern, I had four TA opportunities. I was the managing TA in an upper-level course on software development for two semesters. I was a TA for the introductory computing course for one semester, during which I ran weekly lab sessions for approximately 30 students. And I was a TA for a sophomore-level course on object-oriented design for one semester. These experiences shaped my teaching philosophy: programming is best taught by rewarding systematic designs over functional-but-unprincipled solutions. At Cornell, I spent five semesters as TA for a sophomore-level course on functional programming. I gave weekly lectures to ~20 students and helped manage the course in four of these semesters. The department recognized my efforts with a **teaching award** two years in a row. My proudest course-management moment came on the eve of a first exam, when I led a few TAs in rewriting the extremely difficult test that our overloaded professor had drafted. (The rewritten test was still too hard.)

Mentoring. I am currently supervising one Ph.D student and one research undergraduate at Brown. The Ph.D student is seeking a dissertation topic at the intersection of gradual type systems and data science; toward that goal, we have designed a benchmark for table types and are formalizing an industry partner's gradual language. The undergraduate is developing a staged modeling language to teach formal methods. At Northeastern, I supervised two research undergraduates on topics in gradual language implementation. One of these students made significant contributions to a research paper and is currently a programming languages Ph.D student at UCLA. The other went to a blockchain startup after graduating.

These talented students have been a pleasure to work with as I learn to give effective feedback and guidance. My general advising plan is to help students set goals and then give them freedom to accomplish the goals. One practice that I have found useful is to keep a shared diary of long-term ideas and concrete next steps. The long-term notes shape priorities, the next steps ease context switches, the diary format is simple to maintain, and the document itself provides a meeting point outside the student's day-to-day workspace. For remote advising over the past year, these diaries have helped us stay connected despite the lack of shared office space.

Volunteering. I have volunteered with the Bootstrap program while at Brown [4]; with E1T1 [1], an after-school program for underprivileged high schools, while at Northeastern; and with both Ithaca Media Arts [2] and the Math Explorers' Club [3] at Cornell. These experiences gave me a chance to reach a wide variety of learners, ranging from high school teachers to elementary school students. Finding ways to connect with the individuals in these groups was a difficult but exciting challenge. Northeastern recognized my tutoring at E1T1 with a **graduate community service award** [5].

Teaching Goals

At the undergraduate level, CS education desperately needs methods to deliver quality education at scale. Lecture, grading, and feedback need to accommodate hundreds of students at a time. My experiences at Northeastern and Brown have fortunately acquainted me with two promising tools to start from:

- *The Design Recipe*: A structured method for writing programs. The recipe gives students milestones to target, provides structure to Q/A sessions, and lets graders focus on the quality of solutions instead of mere functional correctness.
- *Quizius*: A machine-learning platform that uses student input to discover which concepts the class struggles the most with. Student input lets teachers focus on curating questions (rather than inventing them) and may reveal expert blind spots.

Tools like these, which add structure to classroom discussions and provide automated feedback, are essential to scale and diversify CS classrooms. In particular, students who lack confidence as programmers (and might drop out of an unstructured sink-or-swim course) will likely benefit from clear milestones. As a faculty member, I will use these tools and support the development of others.

At the graduate level, my goal is to supervise a small pipeline of Ph.D students in an apprenticeship style. Each student must work individually first of all and carve out a distinct research topic, though a junior student may briefly pair up with a senior student to learn the ropes. I will provide one-on-one feedback, primarily on research and communication skills.

Courses

I would be eager to teach programming, programming languages, and related topics (such as algorithms, discrete math, theory of computation) at any level. I am especially interested in four such courses:

- an upper-level programming languages course,
- an upper-level software course based on code review and face-to-face communication,
- a course on logic as a tool for software engineers, and
- an intro-level programming course that focuses on systematic design.

Teaching at the introductory level is on this list of interests because it presents an opportunity to stimulate non-major and first-year students. For non-majors, an intro course may transform their careers by imparting critical-thinking skills and programming confidence. For the first-year students, the intro course may spark a research interest that they pursue during their remaining years. I experienced both these effects myself as a first-year non-major at Cornell.

At the graduate level, I would like to teach advanced courses on type systems and language design. I believe the most effective way to teach such courses is to ask each student to identify a programming problem close to their own research interests—whether in PL or another area—and create a language-based solution. I would also be interested in organizing graduate seminars on modern topics in programming languages, or even to study a textbook or a codebase in depth.

[1] Each One Teach One (E1T1). URL `https://www.eachoneteachone.is`. Accessed 2021-10-12.

[2] Ithaca Media Arts. URL `http://www.ithacamedia.org/`. Accessed 2021-10-12.

[3] Math Explorers' Club. URL `http://pi.math.cornell.edu/~mec/`. Accessed 2021-10-12.

[4] Bootstrap. Data Science. URL `https://www.bootstrapworld.org/materials/data-science/`. Accessed 2021-05-26.

[5] Shandana Mufti. Community service award highlights student's commitment to tutoring, mentorship. URL `https://www.khoury.northeastern.edu/community-service-award-highlights-students-commitment-to-tutoring-mentorship/`. Accessed 2021-10-12.