

Teaching Statement

Amal Ahmed

Teaching Experience and Philosophy

I developed an early appreciation of some of the challenges, rewards and ingredients of successful teaching thanks to several opportunities to teach as an undergraduate at Brown. My first teaching job was as a recitation instructor for Financial Accounting during my second year at Brown. I taught weekly sections, graded homework problems, and conducted reviews before the midterm and final exams. Never having taught before, I was bit nervous at the outset. But from the beginning, I tried to maintain an atmosphere that was informal and interactive. As a result, students were at ease asking questions, which in turn made me feel at ease. I think the fact that I was myself an undergraduate at the time made me more attuned to those moments when the teacher has gone on uninterrupted for too long and the students seem to have disengaged. To avoid that, I constantly worked to encourage interaction and tried to step back from the details to provide some context. An important benefit of grading homeworks each week was that it gave me a good sense of the gaps in students' understanding of the material. Hence, I was able to tailor the classes to what the students had trouble with, which further helped keep them engaged.

That first teaching experience was an extremely positive one and I decided I wanted to continue teaching. The following semester I again taught weekly sections for Financial Accounting and, in addition, served as a teaching assistant for an introductory programming course in the Computer Science department. During my junior year, I served as a teaching assistant for two more computer science classes. One was a course for non-computer-science majors that sought to give students a basic understanding of computing. The other was a challenging introductory course for computer science concentrators, taught by Andy van Dam.

This last class significantly influenced my perspective on teaching large introductory classes. The class had about 130 students and about a dozen undergraduate teaching assistants, three of whom were designated "Head TAs". There were no exams, only several increasingly challenging programming assignments, a few short homeworks, and a final project. The course had a well-deserved reputation as being both fun and demanding. The course staff worked to maintain students' interest in numerous ways, from engaging lectures where Andy insisted on student input despite the large class size, to fun but tough assignments, to hilarious skits performed by the TAs aimed at reinforcing programming concepts. As a teaching assistant, I held several hours of office hours each week, graded assignments, performed in skits, and created demos for lectures.

The course was incredibly well organized. Teaching assistants went through training at the beginning of the semester where they got valuable guidance on an array of issues, from how to deal with students demanding too much help with their programming assignment, to handling contention about grades, to recognizing when a student needed help with a concept rather than just help debugging a program. Head TAs enforced timely grading and met with TAs to go over graded assignments so as to ensure consistency in grading, and so they could pick up on signs of problems with individual students or common misunderstandings. Procedures, policies, and what to expect from the course were detailed in handouts at the outset. Teaching assistants were encouraged to report back if several students found the wording of assignments unclear so it could be clarified for the next iteration of the course. Weekly TA meetings were used to discuss common questions and confusions that would likely arise as students worked on the next assignment.

Perhaps what the course was most effective at was creating a class community that facilitated learning. The amount of interaction between students and course staff was remarkable. The course demanded a significant time commitment from students, but in return provided substantial support and personal attention from TAs.

This experience convinced me that students learn best when they are engaged by the lectures and challenged by the assignments, and that the more they interact with course staff and fellow students, the more they absorb and understand. I also saw that creating an effective learning environment requires constant effort and creativity, and that enthusiasm and organization are critical.

In graduate school, I chose teaching assistantships that gave me the opportunity to teach weekly recitation sections. Specifically, I served as a teaching assistant for two second-year courses: Introduction to Programming Systems, and Algorithms and Data Structures. I also assisted with Medical Informatics; my responsibilities included helping with the creation of assignments, supervising student projects, and grading.

The most extensive teaching experience I have had was when Umut Acar and I co-taught a graduate-level course, Type Systems for Programming Languages, at the University of Chicago in the winter of 2008. The class had first-year graduate students in programming languages as well as graduate students from other areas who had not had an undergraduate programming languages class before. Thus, in the first half of the course we taught material more typical of an advanced undergraduate programming languages course, while in the second half we covered graduate-level material. Umut and I collaborated on the design and syllabus of the course. Umut taught the first half of the course while I taught the second, with each of us creating assignments and preparing lecture notes for our respective parts of the course.

One thing that I tried to do in my lectures, whenever feasible, was to go down paths that I knew to be dead ends, and only after that show students how we could rectify the problem we encountered to arrive at a solution. When teaching students how to tackle a certain proof or explaining to them the specifics of a type system, I believe it is important to focus on the search for a solution and not just on the final answer. In other words, it is not sufficient to simply provide students with a recipe to apply to a problem. Instead we should help them develop a way of thinking that allows them to come up with solutions when existing recipes do not apply. It is tremendously valuable for students to understand why various naïve solutions fail; it makes them better able to appreciate the rationale behind the actual solution, be it a proof method or the design of a type system or algorithm.

But while an understanding of dead ends and alternative solutions is valuable, it is hard to acquire such understanding without trying out an approach and seeing either where it gets stuck or why it works. In a lecture, however, it is only feasible to explore certain paths effectively, namely those where the problem, or lack of one, becomes obvious very quickly. In the future, when I want students to explore more involved alternatives, I plan to try to do so through carefully crafted homework exercises and problem sets, so they can work through the details themselves and have more time to digest the point of the exercise.

In conclusion, I view both teaching and advising as an integral part of an academic career and am excited about taking on these responsibilities. I have found teaching to always be extremely challenging, yet also fun and rewarding. Teachers are successful when they manage to make students interested in a course. That demands hard work and creativity, as well as ongoing evaluation and revision of one's approach. I look forward to continuing to improve my teaching skills and to the rewards as well as the challenges.

Teaching Interests

At the undergraduate level, I would like to teach courses in programming languages and theoretical computer science, such as Principles of Programming Languages, Compilers, Logic in Computer Science, Discrete Mathematics, Data Structures and Algorithms, and Theory of Computation. I would also like to teach introductory programming courses in computer science. On all of these topics, I would be happy to either adapt an existing course or develop a new course that integrates well into the existing curriculum. I also look forward to working with undergraduate students on independent research projects.

At the graduate level, I would like to teach courses on advanced concepts in programming languages, including Type Systems, Semantics, and Language-Based Security. I also plan to develop graduate seminars focused on more specialized topics.