Teaching Statement

Matthew Fluet

It has been a great privilege to have been taught by several outstanding teachers. In addition to enjoying their courses, I observed the great pleasure my teachers took in sharing their knowledge. I have long envisioned myself in their shoes. I believe the road to successful teaching involves hard work, self evaluation, and subsequent revision. I am excited about taking on the role of teacher and consider such a role as a significant component of my academic career. Below, I outline some of the principles that guide my philosophy of teaching.

My time in the classroom as a student has exposed me to several different styles of teaching. As an undergraduate student at Harvey Mudd College, I experienced first-hand the positive impact of small class sizes and accessible instructors. As a graduate student at Cornell University, I was exposed to the very different teachinge style adopted by a large research university. These experiences have left me with clear ideas about what does and doesn't work in a classroom, both for the students and for the teacher. From these ideas, I have identified several distinct challenges that face an educator in computer science.

One challenge is to balance the *creative* and the *observational* aspects of a computer science curriculum. Many students appear to approach computer science as though it were simply "learning how to write programs." For many students, this is the appeal – to *create* something. Indeed, every artifact of the field is the result of some creative effort, whether it is physical hardware or concrete programs or abstract algorithms. Unfortunately, I have observed that many courses in computer science place so much emphasis on creating new artifacts (e.g., programs), that they exclude the development of a student's ability to observe and critically analyze existing artifacts. For example, I cannot recall a single homework assignment before graduate school in which I was asked to look at an existing program and draw some relevant conclusions, whereas I can recall countless homework assignments in which I was asked to write new programs. Certainly, at the graduate level, I have critically evaluated both my own research and that of others, but I believe that it will become increasingly important for students to develop this ability early in their education.

The reality is that no complex scientific or engineering project is the product of a single individual. Higher-level courses must cultivate a student's ability to work in collaboration with others and to organize and design their work with this collaboration in mind. A necessary precursor to the development of these skills is the opportunity to observe the successful application of these skills. Too often, students are charged with creating "good" artifacts (e.g., programs, algorithms), having had little to no exposure to exemplars. Hence, I believe that introductory-level courses should stress both "learning how to write programs" and "learning how to read programs," as one concrete realization of an emphasis on teaching students how to approach the subject of computer science, an approach that balances the creative and observational aspects necessary for success in the field.

As another example, computer science educators are challenged with determining the best use of the technology available. As a field, we should be providing innovative usage scenarios that push

the boundaries of technology as an educational tool. To simply replicate the classroom experience on the Web, with copies of lecture notes, handouts, and homework assignments, seems a limited use of resources. While this availability of course materials begins to communicate that learning can continue to take place at home, not just in the classroom or lab, it often remains a static form of learning. I am interested in pursuing methods to extend this learning experience to include more dynamic components; for example, by providing opportunities for students to give feedback about readings and lectures.

Throughout my education, I have participated seminar courses conducted in varying styles: 1) an open discussion involving the the entire class, occasionally involving a written response by students prepared ahead of time; 2) the presentation of a paper by a single student followed, time permitting, by an open discussion; 3) small-group discussions with directed questions followed by an open discussion. Though each style has its benefits, I consider the third style to be by far the most effective means of running a seminar course. I prefer this style mainly because I have observed significantly more critical thinking and student participation in this style than in either of the others. Upon reflection, the distinct characteristic of this third style is the expression of the instructor's goals for the classroom episode in the form of directed questions. Simply articulating goals more specific than "understand the paper" leads to a more dynamic exchange. I believe that this lesson carries over to courses at all levels: the refinement and sign-posting of specific goals enhances the learning experience in the classroom.

All computer science teachers assign homework, but the effectiveness of homework as a teaching tool can vary considerably. As a student, I valued homework as a means of practicing what was presented during class. As a grader and teaching assistant, I began to value homework as a means of evaluating student understanding of material. However, I have observed that many teachers assign homework without taking advantage of its role as a diagnostic tool. There is no better indication that a class hasn't grasped a given curricular issue than all of the students failing to solve a related problem. When faced with this situation, a teacher often provides a solution to the problem; but, this approach does not always address the root causes of a student's misunderstanding. In order to correct these misunderstandings, there are a number of actions available: examine the student work to diagnose the misconceptions; revisit the material (not just the solution) during the next lecture; assign a similar problem on a future homework to validate the corrected misunderstanding. Although these actions make creating and evaluating homework assignments a challenging task, it remains an important one. Good assignments must not only ensure that correct solutions imply understanding, but that incorrect solutions also become opportunities for teaching and learning.

Lastly, one of my formative experiences involved my early exposure to research at the undergraduate level. I attribute these early experiences with Dr. Arthur Benjamin at Harvey Mudd College with my eventual decision to pursue a graduate degree. I look forward to providing opportunities for undergraduates to be exposed to research in the field of programming languages.

I would be interested in teaching not only the standard courses in programming languages and theoretical computer science, including compilers, programming languages, data structures and algorithms, complexity, and logic, but also an introductory programming course or a software engineering course. At the masters or graduate level, I would be interested in teaching courses on advanced compilation techniques and semantics of programming languages. I hope to run graduate seminars in these areas; possible topics include advanced functional programming and language-based security.