

TEACHING STATEMENT

Alyssa Glass

Teaching and mentoring students have been a highlight of my university experience. I have taught students as a course assistant, teaching assistant, teaching fellow, and instructor for several different courses, giving me a chance to see which teaching techniques really spark student interest, in addition to observing which teaching tools help to teach skills that students can use outside of the particular subject of the course. I believe that deeply engaging students in problems is the key to getting them to think about concepts and ideas. Lecturing alone will only go so far toward drawing students into a topic. To really “grab” them, you have to bring them to the point where they are discovering things for themselves. Getting them to this point of self-discovery involves arming them with the right tools and information so that they can start to ask their own questions and care about the answers. I have also found that, although it is well understood that explaining ideas to others can help to clarify confusing topics, this technique is not often used in the teaching of computer science. Providing opportunities to talk about and explain course topics increases student comprehension and provides valuable practice for written and oral technical communication skills.

My first university teaching experience was two years as a course assistant for Martin Feldstein’s *Principles of Economics* at Harvard, a year-long introductory economics course with the largest undergraduate enrollment of any course taught at the university, averaging 1,000 students. My role as a course assistant involved meeting with small groups of students to discuss and grade short quizzes. In a class as large as this one, these grading sessions were often the only personal contact students had with any course staff, so my job went beyond simple grading to include probing students for understanding of the material, working to resolve misunderstandings, and teaching particularly confusing concepts. My involvement with this course taught me two important lessons. First, the right example or metaphor is often key to helping a student to understand difficult theoretical topics. I repeatedly saw students who were attempting to (incorrectly) regurgitate memorized information, who then suddenly had “aha!” moments when a concept was explained with an enlightening example or with a comparison to a topic that they already understood. Second, I saw first-hand how important group activities are for fostering understanding. I would often meet with groups of students who were all confused about the same topic. By asking the right questions and prompting them to discuss and explain the answers among themselves, I could steer them so that they would much more quickly collectively come to a firm understanding of the material than if they were working alone.

I have been able to apply these lessons in a larger classroom setting as a teaching fellow/assistant for several computer science classes. As a teaching fellow for Charles Elkan’s *Intelligent Machines* at Harvard, and twice as a teaching assistant for Jean-Claude Latombe’s *Introduction to Artificial Intelligence* at Stanford, I took a much larger role in designing a course syllabus, writing course assignments and exams, and teaching. For these courses, I taught weekly sections in addition to the usual office hours and grading. During my weekly sections, I was able to apply the model of small-group learning in a larger setting by designing illustrative example problems for each section, then breaking up the class into smaller groups to work through the solutions, with students reporting back to the whole class on key insights and lessons learned. I was also given the opportunity to give several guest lectures in these courses: lecturing on planning for *Intelligent Machines*, and on game playing for *Introduction to Artificial Intelligence*.

In addition, I have had experience building a course from the ground up, in a less traditional setting, by spending a summer volunteer teaching a new course in computer literacy at a housing development in Cambridge, Massachusetts. I was responsible for designing the course syllabus, creating all course materials, leading each class, grading all assignments and exams, and producing

written materials for future course instructors. For this course, in which I had a classroom full of students who had never touched a computer before and had backgrounds very different from my own, I had to really focus on teaching from fundamentals. Everything that I did in this class was hands-on. The students learned not just practical skills, but also the underlying metaphors for how a computer system works. I was able to build their knowledge so that they understood not just the exact ideas and applications taught in class, but were also able to troubleshoot problems, figure out new programs, and explain what they knew. Even though most of my students had never used a computer before, several of them were able to secure office jobs using computers at the end of the summer.

One area of education that I am particularly passionate about is the teaching of good communication skills. Clear communication, both written and oral, is vitally important in technical fields, whether preparing for academia, graduate education, or industry, yet many computer science students are never taught these skills during their university years. Often, students are unable to write succinctly about technical topics. They struggle to communicate with team members to reach consensus. They do not know how to structure a technical talk that will both engage and educate their audience. They cannot speak before a crowd without burying their faces in slides. I have helped to teach these skills in a wide variety of settings: as a reviewer for papers and talks by fellow graduate students; as a mentor to student interns while working in an industrial research lab; and as a high school speech and debate coach. I have found that, while some communication skills can be explicitly taught, they are better developed through repeated practice with constructive critiques. Opportunities to practice these skills can and should be integrated into computer science courses of all sizes through written assignments and oral presentations.

My time as a teaching fellow for *Intelligent Machines* had a particularly large impact on my belief about teaching technical communication skills. The assignments in this course went far beyond problem sets or even traditional software projects. Students worked in groups on software projects and were required to produce a written report for each project. These group reports included details about the project, in addition to thoughtful assessments of how the students functioned as a group. Students received detailed feedback about each report, and could therefore apply lessons about clear writing to future papers within the same course. I found this assignment format to have more relevance for later work than any other format I have experienced. Students learned how to work in groups; how to design large projects and follow through on their execution; how to clearly communicate about scientific results; how to structure research papers; and how to identify their own strengths and weaknesses in relation to their fellow students. Undergraduates who are planning to continue on to graduate school, graduate students who are learning about the research process, and students who are hoping to build careers in industry all benefit from these lessons.

I hope to continue teaching and mentoring students in and out of the classroom. In particular, I am looking forward to teaching undergraduates and graduate students in my areas of expertise, potentially including courses on artificial intelligence, computational logic, knowledge representation and reasoning, semantic web, and adaptive agents.