TEACHING STATEMENT

JEAN YANG

My goal in teaching is to provide students with a solid foundation to pursue the projects they want, regardless of discipline. Even though I am a computer scientist, I have diverse interests ranging from the arts to the physical sciences. I have established my teaching interests through designing and teaching undergraduate and graduate courses, as well as through mentoring and cross-disciplinary collaborations.

I have taught introductory courses on programming, formal systems, and programming languages, as well as a graduate-level course in program analysis and software verification. As a professor, I can teach undergraduate courses on programming, compilers, and programming languages and graduate courses on program analysis and software verification.

1. Teaching Interests

As programming becomes more ubiquitous, computer science departments are increasingly responsible for training broad-minded software engineers, as well as people from other disciplines. As writer and activist Astra Taylor says, "...programmers and the corporate officers who employ them are the new urban planners, shaping the virtual frontier into the spaces we occupy, building the boxes into which we fit our lives, and carving out the routes we travel" [1]. Taking this breadth and responsibility into account is important when designing lectures, projects, and courses, without sacrificing rigor and depth.

I have the interest and experience to incorporate these cross-disciplinary principles into rigorous computer science courses. An example of a topic I would like to explore in teaching is the connection between functional programming and music: I was exposed to this through working with a composer interested in programmatically generating music using OCaml. I would also like to design projects and seminars that allow students to explore computational tools for social science and journalism. One of the courses that most expanded my own point of view was media scholar Ethan Zuckerman's new media course, where we discussed the impact of technology on civic participation, experimented with using existing technologies in novel ways, and built new technologies.

2. Teaching Experience

As a Teaching Fellow at Harvard for introductory courses in programming, formal systems, and programming languages, I enjoyed helping students accumulate an arsenal of computational tools that allows them to apply theory to practice. I like teaching the ins and outs of pointer manipulation as much as showing the elegance of high-order functions. I learned that with the right presentation, it is possible to engage students in even the most seemingly dry and difficult topics. For instance, to demonstrate how memory management works, another Teaching Fellow and I acted out the roles of the Stack Pointer and of Malloc, Guardian of the Heap. We showed how the Stack Pointer moves up and down to track function memory and how the program interacts with Malloc to allocate and deallocate persistent memory. Such teaching strategies led to positive reviews and a nomination for a teaching award.

During my Ph.D., I have sought out opportunities to teach introductory courses because I want to encourage undergraduates to think critically about their computational tools. I proposed, designed, and taught several short courses on different programming languages. During MIT's Independent Activities Period, I co-taught a two-day Haskell introduction and a six-day, for-credit C/C++

course. The C/C++ course had sufficiently high enough enrollment that I was given permission to hire three graders. I also designed a three-day Scala introduction in the undergraduate Elements of Software Construction course (6.005), taught in Java and Python. I especially enjoyed teaching the Scala mini-course because I got to show students how a different language could help them more succinctly express concepts they had just learned. Two of the students became so excited about what I presented on Scala that they started doing languages research with me. One of them worked with me for two years building case studies in my research programming language, Jeeves.

I enjoy teaching graduate courses because they allow me to introduce students to tools on another level. At MIT, I helped Arvind, Martin Rinard, and Armando Solar-Lezama create the graduate course Foundations of Program Analysis (6.820), combining modules about type theory, model checking, and abstract interpretation. I learned to present the ideas and tools in a way so that graduate students across disciplines could see how to apply the material to their own research. A recitation I led on recent advances in Satisfiability Modulo Theories (SMT) solvers led a student to investigate whether constraint-solving could help his research in natural language processing. I look forward to disseminating cutting-edge results from language design, software verification, program analysis, type systems, and constraint-based programming.

3. Mentoring

As someone who has thought much about motivating people–both through individual mentoring and through creating ecosystems–I am looking forward to starting my own research group. I have worked with two high school students, through the MIT PRIMES program, and several undergraduate research assistants on developing case studies for my research language, Jeeves. One of my best collaboration experiences was with a Masters of Engineering student I supervised to build a Python embedding of Jeeves and a web framework based on Jeeves. Through my mentoring experiences I learned the importance of student compatibility with the group and project. Through recruiting and supervising students, I observed that my best students were often the ones who were most excited about the project rather than the ones with the highest grades. I also observed how mentors can improve compatibility through seeking feedback: I learned valuable lessons from listening to students about the kinds of pressure and feedback they wanted–and seeing how modifying my behavior helped them take more ownership of their work.

4. Leadership

In 2010, I started the Programming Languages and Software Engineering Offsite, an annual one-day event that assembles the MIT research groups in programming languages and software engineering to give research updates, give advice to graduate students, and discuss the future of the field. The initial Offsite made such a difference in inter-group communication that the event has continued each year.

I am particularly interested in creating environments that promote diversity in STEM: I want to make women and other minorities feel supported in pursuing STEM careers. In 2009, with two other students, I co-founded Graduate Women at MIT (GWAMIT), an institute-wide group dedicated to the professional and personal development of graduate women. GWAMIT now has over 1,800 mailing list members, over 80 graduate women involved in planning committees, and an annual budget of over $20,000 with which it runs two annual conferences, a year-round mentoring program, and events throughout the year. I plan to continue supporting STEM diversity by raising awareness and creating opportunities for discussion.

References

[1] Astra Taylor. Tomgram: Astra Taylor, misogyny and the cult of internet openness, April 2014. [Online; posted 10-April-2014].