Teaching Statement

Elena L. Glassman

December 23, 2017

My PhD thesis was born out of my frustration with the tools available to me when I was helping several hundred students learn how to design RISC processors each semester. As a result, my research agenda and teaching philosophy have influenced each other. As I discuss my teaching goals and experiences, I will cite key pieces of literature and a few of my own publications.

I have accumulated both a breadth and depth of traditional lab and classroom teaching experiences. As an undergraduate, I served as a lab assistant for the required core signal processing course in the MIT EECS department, and provided on-demand one-on-one tutoring in the subject for five years through the MIT EECS honor society. As a graduate student, I completed the MIT Teaching & Learning Lab's graduate student teaching certification course. I also served on the teaching staff for the MIT EECS required core computer architecture course for several years, leading a 20-to-30-person classroom several times a week to supplement lectures. In recognition and support of my development of novel tools for enhancing teacher and student experiences in that class, I was awarded the **MIT Amar Bose Teaching Fellowship**.

During my final semester at MIT, my PhD advisor was unable to teach what was, at that time, the only course explicitly devoted to human-computer interaction (HCI) offered by the EECS department. It was uncertain whether or not the course would be taught at all. I felt that it was important that MIT engineers gain an appreciation for the difficulty of designing for humans, so I volunteered to co-lecture the class with an expert in HCI from industry.

The HCI course I co-lectured was a combined undergraduate/graduate course with approximately 175 enrolled students and taught in a flipped-classroom style. Students read lecture notes in advance and spent much of their time in the lecture hall working in pairs on active learning activities. This teaching experience gave me a new appreciation for the challenges and rewards of moving beyond the traditional lecture format.

I supplemented this formal teaching experience with more co-curricular teaching experiences. I was invited by the MIT Teaching & Learning Lab to write and film a short instructional video explaining radio receiver technology to students at the Singapore University of Technology and Design. I was also invited by MIT MEET to teach programming in Jerusalem to a gender-balanced classroom composed of an equal mix of gifted Israeli and Palestinian high school sophomores.

Philosophy

I believe that learning how to consider alternative designs is a core part of becoming a good engineer. Students should implement their own solutions but also benefit from learning about the variety of more or less optimal designs generated by their peers. I also believe that we as teachers should, whenever possible, illustrate concepts with sets of concrete examples and demonstrations. These illustrative sets should be designed with the recommendations of educational psychology and the learning sciences in mind. Specifically, sets of examples should be diverse in particular, strategic ways [1], [2] and, when possible, presented in parallel, [3] to help students construct mental models that generalize well to new situations.

[1] F. Marton and S. A. Booth. *Learning and awareness*. Psychology Press, 1997

[2] M. Ling Lo. *Variation theory and the improvement of teaching and learning*. Göteborg: Acta Universitatis Gothoburgensis, 2012

[3] K. J. Kurtz, C.-H. Miao, and D. Gentner. Learning by analogical bootstrapping. *The Journal of the Learning Sciences*, 10(4):417–446, 2001

Novel Tool Development

In massive programming classrooms where hundreds or thousands of students are all writing their own solutions to solve the same programming assignment, there are a unique set of challenges and opportunities. To help me engage with students in accordance with my teaching philosophy, I designed abstractions necessary to programmatically reveal the space of student-generated solutions in a way I could understand. In other words, I began developing the practice of code demography.

My first code demography project was analyzing and visualizing over two hundred student-written finite state machine (FSM) controllers for a Turing Machine. [4] This visualization revealed that, for the assigned problem, there were two distinct solution types and a few that were incorrect but still passed the teacher-written test cases. At least one fellow teacher realized, in retrospect, that they may have *unnecessarily encouraged students to abandon their work and start over* because the teacher did not know any types of solutions except for the one they had implemented themselves. Teachers can better respond to individual students when they know the space of solutions collectively discovered by the class so far.

[4] E. L. Glassman, N. Gulley, and R. C. Miller. Toward facilitating assistance to students attempting engineering design problems. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ICER '13, pages 41–46. ACM, 2013. URL http://doi.acm.org/10.1145/2493394.2493400

DEAR GAMMA [5] mapped out the space of student-designed 4-bit adders. Many implementations were straightforward, but a small fraction were quite clever. A surprisingly large set of solutions revealed that students were missing a fundamental concept: how De Morgan's Law simplifies digital circuits. Based on the DEAR GAMMA solution map, I designed, deployed, and published an enhancement to that lab, where students were invited to reflect on their own designs, write hints for students with less computationally efficient designs, and learn the answer to the common student question, "Was there a much better solution that I missed?"

[5] E. L. Glassman, A. Lin, C. J. Cai, and R. C. Miller. Learnersourcing personalized hints. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work & Social Computing*, CSCW '16, pages 1626–1636. ACM, 2016. URL http://doi.acm.org/10.1145/2818048.2820011

DEAR BETA, published alongside DEAR GAMMA, mapped the space of common bugs while designing a simulated RISC processor. When students fix a bug in their own code, I asked them to perform a little extra pedagogically valuable work, i.e., to reflect on how they fixed a bug and write a generalized hint for others who get stuck on the same bug after them. By entering that hint into a hint database indexed by buggy behavior, i.e., failed teacher-written test cases, students can crowdsource a set of debugging hints about common and uncommon bugs for themselves and future students.

I also developed a technique that allows me to better administer and monitor in-class exercises. In-class exercises allow instructors to offer real-time help and hints, but it is difficult to track how many students have arrived at the right answer, and which wrong answers are most common. By using lightweight tools like Google Forms during class, I monitor the number of students who complete each exercise and address common misconceptions that emerge from the distribution of right and wrong answers on the fly.

Mentoring

During my time as a PhD student at MIT and Postdoctoral Scholar at UC Berkeley, I have had the privilege of working with many younger students who are at various stages of their maturation as researchers, from sophomores like Aaron Lin, who helped me build DEAR BETA, to PhD students like Andrew Head, Hezheng Lin, and Bala Kumaravel, who seek out my advice on designing projects and studies. I helped Andrew complete and submit a paper on his vision for a mixed-initiative example extractor [6] and I currently help Björn Hartmann advise Bala on a project in human-robot interaction.

I have also had the special privilege of working with a number of brilliant senior undergraduate students. For example, Stacey Terman (MIT EECS '15) and Eric Pai (Berkeley CS '17) each approached me about working on OVERCODE [7] during their respective senior years, and I supervised both of them through their 5th-year Master's degrees. At the invitation of Finale Doshi-Velez, a Harvard CS professor specializing in machine learning, I co-advised one of her senior undergraduate researchers, Sindy Tan (Harvard '17), as she worked on generalizing ideas from OVERCODE to the realm of solutions written in LaTeX by students in massive machine learning courses. To facilitate more mentorship within our research group at Berkeley, I have led pair research sessions [8] and a weekly quiet writing session.

[6] A. Head, E. L. Glassman, B. Hartmann, and M. Hearst. Interactive extraction of examples from existing code. In *Proceedings of the Annual ACM Conference on Human Factors in Computing Systems*, CHI '18. ACM, 2018

[7] E. L. Glassman, J. Scott, R. Singh, P. J. Guo, and R. C. Miller. Overcode: Visualizing variation in student solutions to programming problems at scale. *ACM Transactions on Computer-Human Interaction*, 22(2):7:1–7:35, Mar. 2015. ISSN 1073-0516. URL `http://doi.acm.org/10.1145/2699751`

[8] R. C. Miller, H. Zhang, E. Gilbert, and E. Gerber. Pair research: Matching people for collaboration, learning, and productivity. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '14, pages 1043–1048. ACM, 2014. URL `http://doi.acm.org/10.1145/2531602.2531703`

Teaching Plans

I would be delighted to teach existing courses on data science and human-computer interaction. I can also teach introductory-level programming, signal processing, probability, and computer architecture. The MIT course I developed tools for is much like other introductory lab courses on digital circuit design. I can assist with courses on programming languages, particularly those that cover the design and usability of domain-specific languages.

I am interested in developing new courses at the intersections of human-computer interaction, programming systems, machine learning & teaching, and data science. I would like to develop a graduate-level course on **Interfaces for Data-Driven Decision Making**, as well as graduate seminars on **Information Visualization for Code and Text**, **Systems for Programming by Demonstration**, and **Human-Centered Machine Learning**. At the undergraduate level, I would like to develop a course called **Systems for Data Science** that guides students through the iterative user-centered design process to prototype and build systems that address the challenges facing data scientists within and beyond the campus, including financial analysts, scientists, and spreadsheet programmers.