Teaching Statement

Heather Miller

As scientists, it's our job to make more of the unknown known. Beyond that though, I believe we have a duty to the society that supports and surrounds us to take what we do, the "newly known" if you will, and to make it accessible to a broader audience than our peers alone. Education and outreach are how we move forward together.

Teaching has played a large and rewarding role in my academic career so far, via experiences ranging from co-designing and lecturing a new EPFL BSc. course, to leading the development of a popular MOOC on functional programming in Scala. These efforts have gone on to a publication at ICSE'14 about new ways to engage students and professionals alike, as well earning EPFL's highest award for excellence in teaching in 2012. I provide a sketch of these experiences below.

In brief, my teaching style emphasizes learning by doing, and engagement through inspiring examples and interactive exercises that engage students beyond the classroom. In particular, I take a special pride in jolting students out of the juvenile sense of powerlessness they often come to university with by demonstrating to them first-hand that their time, energy, and actions, *right now*, matter.

Graduate Student Strangers → Co-Authors on a Book

In the Fall of 2016, during my first semester as faculty at Northeastern, I was given the opportunity to teach a research seminar covering my research area; *Programming Models for Distributed Computation*.

I wanted my course to do more for my students than to only survey the landscape of programming models for distributed systems. Across my studies both in the US and Europe, I noticed a common allergy amongst graduate students in Computer Science whether English was their native language or not; oral and written communication. Computer Science coursework traditionally does little to help here, and I wanted to change this.

So I flipped the classroom. Each week, we covered a fundamental topic such as message-passing, RPC, or asynchronous programming through a series of ~5 curated research papers on each topic, with some selected papers taking adversarial positions to others. Students would choose one of the selected papers each week. At the start of each class session, students who read the same paper would huddle up for 15 minutes to figure out how to together explain what they understood of their research papers to their peers. Each group would then get 20 minutes to give whiteboard presentations to the class (no slides allowed) covering the important concepts in their papers. For each paper, I stood at the ready to augment their explanations with important details they may have missed, corrections, or to assist in answering questions about the research to the rest of the class. Each session also tended to result in much interaction between students of different paper groups; students of opposing research papers would chomp at the bit to ask insightful and often revealing questions that they believed their papers resolved but which the presenters' had not.

Why were students so engaged each session? On the first day of class, a rather ambitious course project was announced; each week would correspond to a chapter in a book on distributed programming that we would together write and open source to the developer

community at the end of the semester. Students knew that their efforts would be public, and that book would also be reused and potentially depended on by industry developers. It gave them purpose.

Shortly after the end of the semester, our innocuous book repository was discovered and shared on social media, going viral on a few occasions in the tech community. To date, the project has over 2,200 stars on GitHub [2].

The interest in our book exceeded our expectations, and has resulted in the start of a deal with the MIT Press to publish our work as a full-fledged textbook.

The book project demonstrated to an entire class that their efforts, even as students, could have impact. Students took a responsibility and pride in the work, and it paid off. And the best part–I had the opportunity to help a crop of BSc., MSc., and PhD students go from timidly struggling to explain technical concepts, to being able to ingest research ideas without stress and to clearly and concisely explain those ideas orally or in written form to a broad audience.

Instructor, Co-Designer of Reactive Programming & Parallelism Course

In the Spring of 2015 and 2016, I co-designed and lectured a new course for EPFL undergraduates on data-parallel programming and analytics and event-based reactive programming. My portion of the course focuses on distributed systems, bridging the data-parallel paradigm from multicore to clusters, and doing some basic large-scale analytics with those foundations.

A key to my approach is getting real systems quickly up and running in students' hands and challenging them with problems that also pique their curiosity and motivation. For example, one project is a cluster-based service to interactively explore trending tweets on Twitter seeded by students' interests.

In my role as a lecturer, I designed and administered lectures, weekly programming projects, and exams.

Functional Programming in Scala, a Popular MOOC

In 2012, I led the development of EPFL's first and most successful MOOC to date. Our course, *Functional Programming Principles in Scala* [3], taught by Martin Odersky, has attracted over 500,000 students so far, and has the highest known completion rate for a course its size, at ~19.2%[1] (completion rates of around 7% are standard).

Most notably, I helped develop a method of evaluation based on cloud-hosted automated grading, which provided students with a real-time feedback loop as they solved programming assignments. With 60-70% of our participants hailing from full-time jobs in industry, this approach was considered by many to be the key to our high retention and completion rate. We published these and other results related to our MOOC at ICSE'14 [5].

In addition to laying the foundation and establishing the infrastructure for our MOOCs, as lead and teaching assistant, my tasks included: designing programming assignments, developing automated feedback, recording and editing lectures, providing support on course forums, organizing and directing a hierarchy of EPFL and Community TAs, and more.

Teaching Assistant for Functional Programming

In 2012, I also assumed the role of head TA of EPFL's Functional Programming course, a required 2nd year bachelor's course of 160 students (which I have TAed from 2011-2014).

That year, we began a flipped classroom approach, running our MOOC in tandem with our EPFL course, which I oversaw. The switch required our students to complete their assignments via our MOOC's automated grading system for the first half of the course. In an expanded course evaluation that semester, we found that a vast majority of students–87%–preferred the online format, with 70% indicating that they would prefer the course be "flipped" for the entire duration of the semester instead of just half [5].

As head TA, in addition to designing projects and exams, I was additionally responsible for all the logistics of the course, including fielding all student inquiries, all grading, and oversight of all other teaching assistants.

For my commitment to the development of our MOOC and the successful new course format, I earned EPFL's highest award for excellence in teaching in December 2012.

---

[1] 19.2% corresponds to the first iterations of the course in 2012, which accounted for ~100,000 learners. Coursera as a company has since transitioned away from measuring completion rates as a marker of success for their MOOCs.

Teaching 120,000 People How To Do Distributed Programming

Fast-forward to 2017. Following the popularity of *Functional Programming Principles in Scala*, we were approached to put together a Scala mini-degree on Coursera which included two new courses. I was given the opportunity to develop and teach a MOOC of my own, *Big Data Analysis with Scala and Spark* [4], focused on distributed programming in Spark, a popular framework for large-scale data-parallel processing written in Scala.

Cognizant that Spark has seen massive interest from the Data Science community–a community not typically familiar with functional programming or Scala–I chose depart from the approaches taken in developer books on the subject. I developed all of my own lecture material, rich with visual examples and concrete data for each example. My goal was to teach both the basics of numerical computing with higher-order functions (needed by those unfamiliar with functional programming and Scala) and the basics of distributed computation at the same time. And of course in an effort to keep exercises hands-on and engaging, assignments were focused on solving analytics problems on real-world data and were evaluated by cloud-based automated graders.

The approach appears to have been a success. Between March-November 2017, my course has attracted over 120,000 learners, many of which had no experience with Scala before my course.

Mentoring

Throughout my career, I have been fortunate to supervise many graduate and undergraduate students as a mentor and supervisor. My involvement ranged from supervising a PhD student here at Northeastern as his primary advisor, to summer interns at EPFL that helped on various projects around the Scala compiler and libraries, to regular semester and Master's thesis projects, most of which surrounded my thesis work on programming models for distributed systems. I also guided an undergraduate student, Tobias Schlatter, in a research project that led to our publication on FlowPools [3]. In my mentorship, I work with students to find projects that excite them, while striving to give them the confidence that they can learn new technologies, tackle hard problems, and identify new challenges on their own.

Beyond the Classroom

As one who began her own career in Computer Science on a non-standard path[2] largely due to a lack of exposure to the field, it has long been important to me to help other women discover and try their hand at Computer Science.

To this aim, I have spearheaded a new intensive academic summer program young women at Northeastern. The goal is simple; bring 100 high school-aged girls to live on the Northeastern University campus, and put them through an abridged version of *Fundamentals I* course– Northeastern's "CS 101" course. To pique girls' interest in what they can *do* with these concepts, afternoons will be devoted to hands-on workshops in robotics, game design, and web/mobile app development.

On weekends, I also organize ScalaBridge [1] workshops worldwide; so far in Basel (CH), Zürich (CH), Copenhagen (DK), and soon in Boston (US). The goal of ScalaBridge is to inclusive Scala community with free introductory programming workshops for women–and so far we have reached women and girls from all walks of life! Stay-at-home moms, women with careers in fields like business, biology, and food service, and curious teenage girls have all spent a day with us, trying their hand at programming in Scala.

What next?

As a new faculty member, I would be qualified and excited to teach courses in distributed programming for big data (touching upon topics from statistics and machine learning), parallel, concurrent, and asynchronous programming, functional programming, type systems, and introductory computer science. As one who hailed from a non-technical background before entering Computer Science myself, I'd also be excited and well-suited to teach introductory computer science for non-majors.

References

[1] http://www.scalabridge.org/.
[2] https://github.com/heathermiller/dist-prog-book.
[3] https://www.coursera.org/learn/progfun1.
[4] https://www.coursera.org/learn/scala-spark-big-data.
[5] H. Miller, P. Haller, L. Rytz and M. Odersky. *Functional Programming For All! Scaling a MOOC for Students and Professionals Alike*. in ICSE 2014.

[2] By way of fine arts and electrical engineering with a touch of neuroscience.