Ravi Chugh

Teaching Statement

On the first day of a Mathematical Foundations for Computer Science course as a freshman, my remaining doubts about pursuing a computer science degree were eliminated by a professor who passionately described elegant ideas in mathematics and their applications to computing. I hope to engage and excite my students in a similar way. A great feature of computer science as a discipline is that an undergraduate curriculum serves as a platform for a wide variety of careers within computer science as well as an increasingly large number of other fields. In my teaching and advising on research projects, I hope to introduce my students to as many of these avenues as I can.

Prior Experience

Over the course of my undergraduate and graduate career, I have been a teaching assistant, instructor, and guest lecturer for high school, undergraduate, and graduate courses.

Working with High School Students. When I took introductory programming in high school, the instructor began the first class by passing out blank index cards and asking us to write down a recipe for making a peanut butter and jelly sandwich. After we finished, he brought out PB&J ingredients and some utensils, chose an arbitrary recipe, and chose a student to read out and follow the instructions, making sure that the student did only as instructed. In short order, the poor student was stuck, in need of extra arms and several leaps of faith to complete the recipe, leaving an unmistakable lesson: unlike a recipe for human consumption, an algorithm for computer consumption must not leave out any details.

I appreciated the PB&J exercise so much (not to mention the tasty snack that followed!) that I used it on the first day of a summer high school course I taught at the University of Pennsylvania. The exercise provided a fun and non-threatening atmosphere for students to participate and meet each other, setting the tone for the 3-week course that followed. To complement the programming material usually taught in the summer high school program, I designed a breadth-first introduction to hardware and software algorithms to give students a flavor of an undergraduate computer science program. Throughout the course, I was thrilled to see the students’ enthusiasm for a subject that — I am happy to be reminded — seems like magic at first sight.

Working with Undergraduate Students. As a teaching assistant for several undergraduate programming language courses, I have helped design homeworks and in-class assignments. In my future courses, I plan to incorporate more in-class assignments, including group assignments and oral presentations, into the testing and grading structure. Collaborative and interactive assignments will help prepare students for more advanced courses and most real-world settings, where software design and development are often collaborative and iterative processes.

Working with Graduate Students. I have given several guest lectures in graduate-level compilers, software verification, and software engineering courses at UCSD, where I presented program analysis techniques that I learned from the research literature as well as those I developed in my own work. In general, I find it invaluable to consider my research ideas in such a way that I can meaningfully explain them to others who are not as familiar with the area, which often leads to fresh insights into the core challenges of the problem.

At UPenn, I served as a teaching assistant for a discrete mathematics course in a Masters program targeted at students with varying backgrounds outside of computer science, some of whom were decades past their formal training. I was inspired by the many students who showed enthusiasm and determination to understand difficult material even when it pushed them past their comfort zones.
Looking Ahead

I plan to pursue several opportunities early in my teaching career.

**Teaching Static Verification with Dynamic Languages.** I think the Racket (or PLT Scheme) approach to teaching introductory programming is very compelling. Using a core untyped, or “dynamic,” programming language with sufficiently expressive features (in particular, first-class functions and mutable variables), Racket provides a progression of *language levels*, which expose only those concepts that are currently under study. This design facilitates a smooth transition between increasingly difficult topics within a single programming environment, so that students can focus on the core challenges without fussing with new languages and environments at the same time. Teaching new languages and tools is important, too, but should not be an inherent side effect of teaching the fundamental programming concepts.

In my research, I have developed static type checking and verification techniques for dynamic languages, and I believe that these ideas can be integrated into a system like Racket to further extend the spectrum of topics that can be taught within a single platform. Building high-quality support for these advanced topics so that they are pedagogically sound poses several engineering and research challenges, which I believe will lead to many fruitful projects for both graduate and undergraduate students.

**Undergraduate Research.** Introducing undergraduates to research early on can, in addition to help set them up for successful careers in graduate school, provide invaluable experience formulating and presenting independent work no matter what career paths they choose. But even the best students often find it difficult to transition from assignment-oriented courses to open-ended projects. To ease this transition, I plan to set concrete, short-term goals that are realistic and to meet regularly to monitor progress and provide guidance along the way. This will be especially important early on, because it takes practice to identify which parts of a problem are crucial and which parts can lead to wasted effort and frustration. Helping a student focus on the “essence” of a problem can lead to more rapid progress, giving the student the confidence to continue and discover his or her own particular interests and strengths. As a guiding principle, I believe this is the right strategy for training and working with students at all levels.

**Flipping the Classroom.** Recently, a popular notion among instructors has been to pre-record lectures for students to watch independently at their own pace (as provided by Khan Academy and Udacity, for example) and then use in-class time for discussion to support, augment, and enrich students’ understanding. I am excited to experiment with this approach so that more face-to-face time is focused on more difficult material, rather than on “first exposure” to the material. Furthermore, if I pre-record my lectures and reuse them as appropriate, then I can devote more preparation time and in-class time to facilitate discussion, organize group assignments, and allow more flexibility to dive into topics based on the interests and needs of students. In most lectures, students ask questions that I have not heard before, which lead the class to interesting discussions and which lead me to new insights. I would like to organize my courses to allow more time for these kinds of fruitful moments. I look forward to joining a generation of teachers that will experiment with technology to cater to the changing needs of and opportunities for students.

**Conclusion**

I believe that striking a balance between teaching and research is mutually beneficial for both endeavors and is a well-rounded approach to training new students who will pursue careers either in engineering or other fields. I have had the great fortune to learn from inspiring teachers at all levels, as well as the opportunity to begin to repay this debt by working with students in a variety of classroom settings. Teaching has been a fulfilling complement to my research efforts, and I look forward to developing my teaching experience further. Beyond my own interests, I look forward to exploring topics and projects that excite my students, and I hope they will take me along for the ride!