Teaching Statement

RYAN KAVANAGH, Carnegie Mellon University, United States of America

My overarching goals as an educator are to excite students about computer science, and to teach them to think critically about course material so that they may apply it in new settings. I strongly favour an interactive teaching style, and assignments that help students discover new applications of course material. By helping students discover content in the structured setting of lectures and assignments, this pedagogical approach gives students the skills and confidence required to apply their knowledge to new situations. Interactive discovery is not enough though: I believe that we must also teach students to think critically about what they learn and about their own work. For example, in an algorithms course, we must teach students to consider the trade-offs of various algorithms, and how to evaluate which algorithm to use in a given situation. Regarding their own work, we must teach students to ask: is their code correct and efficient? Is their proof correct, elegant, and clearly communicated?

I have had the opportunity to pursue my teaching goals and to see how they positively influence student learning. In fall 2018, I was the teaching assistant for a graduate course on type systems and programming languages. I received an overall rating of 4.92/5.0 (with 13 students completing the TA evaluation forms). I gave three lectures for this course. In these lectures, I adopted an interactive style in which the students and I collectively "rediscovered" each lecture's material. To help guide the students, I presented motivating examples and reminded them of material we had already seen. When I sensed that we were getting lost in details, I encouraged students to take a step back and to consider what we were trying to accomplish and why we wanted to accomplish it. When faced with several approaches, we considered their respective advantages and disadvantages. This interactive approach not only taught students how to think abstractly and critically, but it also taught them how to approach new problems. By actively engaging with the material and rediscovering it first-hand, the students developed a deeper understanding of it than they would have by passively listening to me lecture at the board.

My experiences as an educator and as a student have taught me that including research results in courses is an effective means of engaging students. For example, when I was a teaching assistant for an undergraduate course on constructive logic in fall 2017, we taught students how to program using computational interpretations of substructural logic. While leading recitations, I saw first-hand how students were excited to learn these techniques from the forefront of programming languages research. As a result, I try to include research results wherever it is appropriate when teaching.

I have also pursued my teaching goals when developing assessments. I enjoy developing homework assignments that carefully guide students through discovering new material not covered in class. For example, I developed an assignment that guided students through a proof of the Church-Rosser theorem. To do so, I partitioned the proof into a sequence of small lemmas, each of which formed one problem. To avoid double jeopardy, I ensured that these problems could be solved independently, and than an incorrect solution to one problem did not affect the rest of the assignment. This style of assignment helps students actively engage with course material by applying it in new settings. These assignments also help students deeply understand the new material by guiding them through its discovery. In my experience, students also find this style of assignment rewarding, and they have reported that they enjoyed the associated sense of discovery.

I firmly believe that computer scientists should be taught how to communicate clearly, both through prose and through code. Indeed, experience has taught me that students do not fully understand a concept until they can explain it clearly. Analogously, we do not fully grasp a computational solution to a problem until we can implement it clearly. This is because code is primarily a means of *communicating*

computational ideas, and clear code follows naturally from a clear understanding of those ideas. However, clear communication is difficult, and students often need to be taught how to communicate effectively. When grading, I frequently provided detailed feedback on how to write clearer proofs. As a result, I saw a dramatic improvement in the quality of written work in the span of a semester.

To give students an opportunity to practice communicating technical content, I would like to assign exploratory term papers in upper-level undergraduate courses. These open-ended assessments involve learning about a result or technique not covered in class and writing a short 6–8 page paper about it. I particularly enjoyed this style of assessment as an undergraduate because it gave me room to explore my own interests. These assessments also featured a review component, where students exchanged papers and provided each other with written feedback. This review process further helps students think critically about how to communicate technical content.

Just as I assess student learning throughout the semester, I believe that it is important to continually assess my teaching and to adjust it to meet the needs of students. As a teaching assistant, I invited my students to leave me feedback and suggestions in an "anonymous feedback" envelope that I kept outside of my office. To ensure that I gave assignments of appropriate length and difficulty, I asked students to report on each assignment how long they had spent completing it. As a result, I learned that students were spending much longer on assignments than I had expected, and I adjusted the length and difficulty of assignments accordingly. While teaching, I also constantly gauge the understanding and needs of students, and I adjust my pace and explanations accordingly. I found this particularly important during office hours, where I adjusted my teaching to address the individual needs of students from diverse backgrounds to help them succeed.

I get immense satisfaction from helping students discover and engage with new material. Particularly satisfying is showing students the hidden beauty of many parts of computer science and mathematics. I am eager to teach existing courses, especially introductory courses or those related to programming languages, logic, concurrency, or theory. I am also eager to develop courses related to these topics. For example, I would like to develop a course on constructive logics and their applications to computer science. I envision this course covering topics such as proof search, the design of logical systems, proofs-as-programs interpretations of linear and substructural logics, etc.

Outside of the classroom, I am eager to mentor students in undergraduate research, e.g., through undergraduate student research awards, and by supervising undergraduate honours theses. My own undergraduate research assistantships taught me invaluable research, technical, and communication skills. For example, my mentors taught me how to read papers, how to approach research problems, and how to communicate results. They also opened my eyes to the exciting world of computer science research. I would like to foster these skills and this excitement in undergraduates and to help mentor the next generation of computer science researchers. I am happy to supervise projects related to my own research, as well as projects broadly related to programming languages.