Teaching Statement

Charlie Murphy, CS Department, University of Wisconsin–Madison

A core responsibility of a professor is teaching and advising the next generation of computer scientists. I have had the opportunity to do both in my capacity as a grad-student and post-doctoral researcher and am excited to do both in my capacity as a professor.

During my time as a graduate student, I took many opportunities to teach including both instruction and development of course material. I was an assistant in instruction (AII) twice for COS 226 (Algorithms and Data structures) at Princeton University, where I held recitations recapping that week's lectures and leading students through practical exercises. Furthermore, as part of COS 226, I held weekly office hours where I assisted dozens of students on a one-on-one basis answering any question about course material. I was also an AII for COS 448 (Distributed Systems) at Princeton University, where I held a recitation every week. As part of COS 448, I was responsible for creating homework assignments as well as developing slides and course material for each recitation based on that week's topics (e.g., consistency models, snapshot algorithms, and RAFT). Finally, I was an AII for COS 518 (Automated Reasoning About Software) at Princeton University. As an AII for COS 518, my primary duties were to host office hours each week, and give several of the course's lectures. Finally, during my time as a graduate student I was involved with the Prison Teaching Initiative (PTI), a volunteer collaboration between Princeton University and several local community colleges to teach university courses within prisons in New Jersey. As part of my involvement with PTI, I was a lecturer for MAT 015C (Basic College Math) at Edna Maham correctional facility. Furthermore, I and several other graduate students led a multi-year movement to develop PTI's computer science curriculum, which was taught for the first time in Fall of 2023. As part of the course design process, we needed to develop a flipped-classroom design for each course as the students would have limited (in class) access to computers. Finally, my research often includes learning and exploring less common topics in logic, programming languages, and math. I enjoy learning about these topics, creating slides/tutorials and giving a lecture at my research group's seminars.

1 Teaching

My philosophy on teaching arises from my experiences working with students in and out of the classroom. I believe a key component of successfully teaching is having an understanding of where students are coming from both in terms of background knowledge and learning style. As a teacher, it can be easy to recite information at students in the same way I learned the material; however, I found that this rarely connects with students and instead it is important to work with students to explain and go through material in multiple ways to ensure students grok the material.

Student interaction. This need to tailor instruction to students became apparent as I taught COS 226 a second time. I found that during my first time teaching COS 226 the students in my recitation would quickly learn the material and could follow the examples while I worked through the problems on the chalkboard; however, during the second time teaching the course, I found the same technique didn't work well with the students in my recitation. I found instead that students responded much better to being given a chance to first independently work on the example, have a volunteer and/or randomly selected student work out the problem on the board,

and as a class review the solution and propose any fixes with justification. This approach allowed students to see multiple ways to attempt the problem as well as hear justification for why certain changes were made. Through this experience, I strongly believe **lectures should be interactive**: asking students questions and working through examples when possible. Thus, gauging students understanding by sampling both volunteers and students at random.

Backward design. When developing a course, lecture, or talk I start by asking the question "**What should the student/listener take away from the course/lecture/talk?**" My first time developing course material was during my time as an AII for COS 448, where I was responsible for creating lectures for each recitation. Each lecture for COS 448 was centered around a general topic in distributed systems (e.g., consistency models). While the topics could cover multiple lectures worth of material, I needed to distill the topic to a few key points. For example, when covering consistency models, I wanted students to be able to learn the different models (e.g., eventual consistency, causal consistency, linearizability, and serializability), their assumptions, and what effect they have on a user of the distributed system. I then worked backwards from each of the concepts I wanted students to learn and created lecture content and examples to emphasize the key concepts the students should take away.

2 Advising

During my time as a post-doctoral researcher at the University of Wisconsin – Madison, I have had the opportunity to co-mentor two graduate students, Jiangyi Liu and Anvay Grover, while working on the paper "Synthesizing Formal Semantics from Executable Interpreters" (OOPSLA'24). As part of this process, I've learned that being a good mentor/advisor sometimes requires letting a student go off and study a problem by themselves, and at other times requires a stronger hand in steering the direction of research. Additionally, and more importantly being **a good advisor requires kindness and understanding** that mentees come from different backgrounds and educations and as such may need more help in some areas of research/learning than others. In my area of research, both topics in core programming languages and logic are a must. Unfortunately, these topics are not always covered in an undergraduate CS education and are often instead learned through research.

Research Group. During my time as a graduate student and post-doc, I have found that being part of a research group with students/researchers at various levels of experience has had a positive impact on my research and career. As such, my ideal research group would consist of 5-6 graduate students (1-2 joining per year). Ideally, my research group would start small and grow as more students join. I believe this mixture of research levels provides junior students with a greater ability to seek guidance and learn about research from more senior graduate students in a way that professors cannot always provide to their mentees. In terms of research, I would expect each student I mentor to have a project of their own that they are leading, and when appropriate assist and join on projects where their expertise can be used. Finally, I plan to run my research group by meeting with each student weekly (on a per-project basis) and with the full research group every week. While the project specific meetings provide a great way for students to discuss in depth problems and solutions, the group meeting provides a sense of community within the research group and allows everyone to stay informed about what everyone else in the research group is working on.

3 Curriculum

I am excited to incorporate topics form my research including automated reasoning about software and logic into various course curricula. At first, I would incorporate the material as a stand-alone seminar course with a focus on recent research—to introduce students to a breadth of research related to my work. On a longer horizon, I am excited to create an upper-level undergraduate or graduate course that focuses on the fundamentals of logic and automated reasoning principles. This course would include topics such as logic (e.g., propositional logic, first-order logic, satisfiability modulo theories, constrained horn clauses), fixed-points (e.g., lattices, widening, narrowing, and applications to logic and programming languages), decision procedures for logic (e.g., resolution for SAT, conflict driven clause learning, Nelson-Oppen theory combination, and property directed reachability), and verification techniques (e.g., invariant generation, strongest/weakest preconditions, bounded model checking, and reachability analysis). Beyond creating new courses which focuses on advanced topics in logic and automated reasoning, I can also teach introductory CS courses, data structures, algorithms, discrete math and logic, programming languages, and compilers.