Teaching Statement
Xujie Si

I believe the most appealing part of an academic career is teaching and mentoring students. The process of conveying knowledge to enthusiastic young talents is exciting and rewarding, and I frequently get inspired during my interaction with students. I want to become a professor because I enjoy this collective improvement process of both students and the instructor.

Teaching Experience. I have been a teaching assistant for six courses at three different universities, namely Vanderbilt University, Georgia Institute of Technology, and University of Pennsylvania. These courses range from introductory courses like data structures, computer networks and databases, to graduate-level courses like software analysis and testing, as well as research courses like graduate study on how to do research. I am very delighted to see students taking these courses join top IT companies like Facebook, Google and Microsoft and publish their research projects in top conferences and journals. I enjoyed my experiences of teaching all these courses, and two of them are particularly unique, which I would like to elaborate further.

The software analysis and testing course (CS6340, instructor: Mayur Naik) is one of the key Massive Open Online Courses (MOOCs) from Georgia Tech offered on the Udacity platform, through which students can earn Online Master of Science in Computer Science (OMS CS), the first accredited online Master's degree. Given that the number of students is large (around 300), all teaching perspectives (e.g. lectures, office hours, assignment design, grading, exams, projects) face unprecedented challenges. Being the lead of five teaching assistants, I re-designed many assignments from the past so that automated grading could be done easily and, at the same time, provide useful feedback to students. My TA experience of this online course suggests that peer question answering and peer project review could serve as an effective mechanism for large-scale teaching, while motivating active engagement of students.

The introduction to graduate study course (CS7001, instructors: Alex Endert and Mayur Naik) aims to get new graduate students better prepared for research by going through various milestones of an academic paper publication — from formalizing the initial idea to paper writing, review, rebuttal and eventually presenting in a conference. My TA duties of this course included helping students figure out a research plan following their own interests at each milestone, review the outcome and provide constructive feedback. This is a both challenging and rewarding experience because it is a mandatory course for all new CS graduate students from all research areas of computer science. From this experience, I learnt how to mentor a large number of students, who have very diverse backgrounds, on research topic selection, idea formalization, evaluation and time management.

Teaching Philosophy. My teaching philosophy is to let students learn and understand computer science principles and theories thoroughly via sufficient practice. I view computer science as an art that incorporates mathematical rigor into the engineering process of utilizing computational resources, especially with the goal of solving practical problems. My teaching will focus on improving students' computational and rigorous thinking, problem solving capability as well as faster learning capability. These are critical for their future successes because computer science is a rapid changing field, and many programming skills that students have learned are useful today but will become obsolete in the near future.

I believe students should be well-motivated to learn the subject in the first place. One good way is to set up an interesting course project for each student at the beginning. My past teaching experiences indicate that students actively learn the course materials, which are necessary to finish the given project, and reinforce their learned skills during implementing the project. Beyond motivating students, this project-driven style learning also encourages students to explore skills and ideas that are not covered in the course but will be helpful for their own chosen project. Also, this fits well to students with diverse background so that they can freely learn new things in a demand-driven way.

I would also like to promote computational and rigorous thinking during my teaching. In particu-
ular, I would encourage students to choose problems from their daily life as programming projects and think about solutions in a systematic manner. Computational resource is becoming a new utility like electricity, and there is no reason not to take advantage of it whenever possible. Many seemingly complicated problems in daily life like planning for meetings, courses and physical exercises, can be easily and systematically solved via programming, rather than manually addressing and struggling with various conflicts. The concepts of abstraction and decomposition enable us to simplify a complicated problem and break it into small pieces. These computational concepts are generally helpful for attacking various problems we face in daily life.

Grand challenges foster rapid progress of the entire field, and similarly, small challenges guide students to learn faster. Setting up extra bonus for a few challenging (but optional) tasks is a very helpful way to encourage students to challenge themselves, who are usually capable of far more than they think they are. One critical thing to make this happen is instant feedback so that students can give an attempt at any time and immediately realize whether their attempt is meaning or not. That is, feedback for these challenging tasks should be automated. I observed two great successes during my past teaching. One challenging task we have in the database course (CIS-550, instructor: Mayur Naik) is SQL query optimization, for which I created a leaderboard for all students’ submissions so that students can submit their solution any time and then immediately observe whether their solution is correct and how well it performs. Another challenging task is from the Software Foundations course (CIS-500, instructor: Benjamin Pierce), which promotes this automation feedback idea to extreme — any homework assignment can be automatically checked by the Coq proof assistant. Many students are so motivated that they successfully solved all course assignments in the first month by learning from the feedback of the proof assistant.

Last but not least, inspired by the computing education work from Kathi Fisler and Shriram Krishnamurthi, I believe an instructor should be consciously aware of the fact that many seemingly natural ideas can be inherently difficult for students who have little computing background. This is crucial and should be handled with great care especially for introductory courses. Therefore, I will provide instructional scaffolding, break down concepts, prepare comprehensible examples, and solicit active, continuous feedback from students to ensure that my classroom style is effective.

**Future courses.** My past teaching and research experience makes me immediately ready to teach a number of introductory courses like data structures, algorithms, compilers, programming language, and software engineering. My recent research qualifies me to teach more advanced courses like program analysis, program synthesis, formal verification, deep learning, reinforcement learning, and artificial intelligence. Also, I am excited to design new courses based on my recent work and work from many others in the programming languages and machine learning communities, which cross-fertilizes ideas between these two communities. The new courses include topics like numerical and probabilistic approaches for program reasoning, software testing techniques on neural networks, deep learning for software verification and testing, and neuro-symbolic reasoning.

**Advising.** I enjoy all my past experiences of mentoring and inspiring both undergraduate and graduate students, and look forward to advising my own students in the future. As the TA of graduate study on how to do research, I helped to advise more than fifty new graduate students on their mini-research projects involving almost all research topics in computer science. At Penn, I have advised one master’s thesis on program synthesis, and mentored three PhD students (two of whom are female) on machine learning and program reasoning. All of them published at least one paper in a top-tier venue from software engineering or machine learning. I am now advising one senior undergraduate student from India and two first-year PhD students from Penn and Georgia Tech.

From my research and mentoring experience, I would like to emphasize three things for effectively advising students. First, let the students explore freely, as long as the exploration still sticks to the general vision the advisor suggests. I appreciate such a freedom I got over the years and feel it extremely helpful to motivate innovations and exciting discoveries. Second, timely feedback and effective communication between the advisor and the students are crucial. Weekly meetings are necessary but sometimes inefficient, as students may get stuck on some unimportant details for a few days, which could be addressed by a quick consultation with advisor. Tools like Slack are quite useful for such cases. Third, set up a rigorous evaluation plan as early as possible, which will guide progress in the right direction. I frequently observed that students may spend significant effort solving imagined issues, which are far from the true difficulties of the actual research problem. I believe these experiences would help my students do great and exciting research in the future.