William Harris — Teaching Statement

Teaching Philosophy I believe that a highly effective class structure is one in which students introduce themselves to material individually, and then actively work on material collectively with supervision (i.e., the *flipped classroom* [2]). I believe that programming is an extremely effective tool for ensuring that students understand a concept comprehensively. As a result, I believe that courses covering topics in computer systems should include extensive project work.

Teaching Experience An an undergraduate student, I served for one semester as a lab consultant, and then served for five semesters as a lab teaching assistant. As a lab consultant, I interacted with students one-on-one to help them gain a better understanding of material that they found difficult. As a lab teaching assistant, each week I was responsible for a lab group of approximately 30 students. In particular, I reviewed the material presented in lecture by the instructor, administered a lab assignment, and provided in-person feedback to students regarding their solutions to the assignment. Each semester, I also designed one to two of the lab assignments used by all of the approximately six lab groups that constituted the course.

As a graduate student, I have served as a project mentor and a guest lecturer. As a project mentor for an undergraduate course in system security, I provided feedback to small groups of students on their design and implementation of secure systems. I served as a guest lecturer for an undergraduate-level course in programming languages and compilers. All of my teaching experiences have informed my current teaching philosophy: across all experiences, I have consistently found that students internalize material best through actively working problems, and then discussing questions that arise over the course of their work with small groups of students and with instructors.

Teaching Plan I look forward to applying my experience to teach undergraduate and graduate courses in compilers and programming languages. While compilers are a classical topic in the undergraduate curriculum, an undergraduate course in compilers may still be one the best forums in which to demonstrate how theory, in particular, formal languages and automata, can be applied to solve practical problems. While relatively few students today will likely need to develop a full compiler, the principles of designing a compiler can be adapted to teach students how to design and implement a *domain-specific language*, a skill which is both highly enlightening and practical.

I am eager to teach a graduate course in programming languages that exposes graduate students across multiple areas to advanced techniques developed in modern research on programming languages. Such a course will provide a rigorous treatment of how desired properties of a program can be specified independently of the program's implementation, and how a program can be checked to satisfy its specification. In the process, students will be introduced to techniques that could be applied in many fields of computer-science research, including *efficient state-space search* and *automatic theorem proving*.

Online Education I am excited to contribute to and learn from recent efforts in online education. When educating students in person, I plan to use online tools in at least two ways. First, I plan to create and share recorded lectures. Providing recordings of lecture material will allow students to absorb and review material at their own pace, while allowing me to devote valuable lecture time to actively work with students on in-class assignments [2]. Second, I plan to moderate online forums that enable students to raise questions and discuss others' questions; such forums are used effectively in online courses provided by, e.g., edX [1].

I also look forward to educating large groups of students in Massive Open Online Courses (MOOCs). MOOCs allow lecturers both to address an enormous body of students and to collect valuable information about the effectiveness of their teaching. However, multiple challenges must be addressed before the potential of MOOCs can be realized fully. One highly-publicized concern for MOOCs is that their retention rate over the duration of the course is low: typically only approximately 10%. However, some analyses of retention rates have observed that rates vary dramatically with the initial intent of a student (e.g., among students who submit the first assignment in a MOOC, the retention rate is approximately 45%) [3]. I thus

believe that retention rates reported for current MOOCs reveal a more subtle issue than simply maximizing retention rate: the core issue may in fact be to design MOOCs that present material in highly modular, abstracted components that enable students with varying skill sets and objectives to benefit from the MOOC proportionately to the amount of time that they are willing to dedicate to learn material.

References

[1] edx.org FAQ. https://www.edx.org/org-faq, Nov 2013.

[2] Will MOOCs change the way professors handle the classroom? http://chronicle.com/article/Will-MOOCs-Change-Campus/142869/, Nov 2013.

[3] S. Kolowich. Coursera takes a nuanced view of MOOC dropout rates. http://chronicle.com/blogs/wiredcampus/coursera-takes-a-nuanced-view-of-mooc-dropout-rates/43341, April 2013.