TEACHING STATEMENT

Ronghui Gu (ronghui.gu@yale.edu)

I am excited about teaching because it is not only a procedure to transfer knowledge to students but also a procedure to examine my research and make real impacts by training the next generation of computer scientists.

Teaching Fellow Experience

My teaching experience spans various activities, such as being a teaching assistant, organizing class projects, and participating in the course development. Among these, I value the teaching assistant role most. During my Ph.D. study at Yale University, I served as teaching fellows for one introductory course (Fall 2012 $\sim$ Spring 2013) and four advanced courses (Fall 2013 $\sim$ Fall 2015). When looking back, it is hard to believe that I have taught more than 600 undergraduate and graduate students directly. These experiences also equip me with the skills to adjust teaching strategies for students with different background and motivations.

Take an introductory course CPSC 112 (Introduction to Programming Languages) as an example. Since most students enrolled in this course were freshmen, and many of them did not have any previous programming experiences, my goal for the class was to inspire their programming interests, instead of emphasizing the coding skills too much. Before the midterm exam, I found that lots of the students were struggling with an assignment related to the regular expression. They were asked to implement some string operations using the regular expression. However, during my office hours, they told me that this concept was too abstract and seemed magical to them. I discussed with the instructor and then gave a tutorial lecture named "Regular Expression in Automata Theory" on the class. I tried to explain the regular expression from the view of automata and utilized slide animations to show how expressions are accepted by the automata. Although such slides took me several days to prepare, it paid when students told me that they finally understood the secrets of regular expression after my lecture.

As for advanced courses like CPSC 424(Parallel Programming Techniques), students are always eager to learn more about debugging and programming skills, which cannot be taught through the lectures alone. To satisfy these demands, I offered a monthly one-to-one training opportunity for all the 15 students in the class. During this training, I worked with the students together on debugging their own parallel code, which is widely considered to be challenging. Sometimes, it took us several hours to locate a single bug. Although it is quite time consuming, I believe this training is very helpful for students to dig out issues in their programs and establish a thorough understanding of how to solve them.

Course Development Experience

Besides the adaptive teaching approach, I believe that advanced courses are best taught with a combination of theoretical depths and application-oriented practices. With this belief, as the only teaching fellow, I worked with Prof. Ruzica Piskac on developing a new course CPSC 439 (Software Engineering) at Yale University from scratch in Spring 2014. Different from a regular software engineering course that focuses on the engineering side, we insisted on teaching more formal methods. I gave several lectures on how to write specifications, how to reason about code using program logics, and how to use Dafny to assist the development of large projects. All these formal techniques were intensively applied in most of the course projects. At the end of the semester, Ruzica and I held a project fair in the department and invited all the CS faculties, as well as some industry experts from IBM and Microsoft. During the project fair, students presented and demoed their products as if they were running start-ups. Most of the attendants were amazed by the code quality and the reliability-focused design of these projects. Some industry experts said it was unbelievable that these projects were managed by the students under an undergraduate-level course. Furthermore, one project that implemented a secure storing system for medical data has resulted in a start-up.

Another example that demonstrates the power of this combination is our re-design of the course CPSC 422 (Design and Implementation of Operating Systems) at Yale. This re-design was developed by my advisor Prof. Zhong Shao in Fall 2015 based on our CertiKOS research project (i.e., an extensible framework to formally verify OS kernels). As the lead developer of CertiKOS, I actively participated in this course development and gained a new view of the relation between teaching and research. In the past, the OS course at Yale was one of the most challenging courses for undergraduate students. Some students were even scared away by the first code lab when they had a hard time understanding the functionality of the missing code, as well as the role these code play in the whole OS kernel. To help students build a high-level view of the kernel's design, we established a series of code labs using the layered specifications and implementations of a verified OS kernel called mCertiKOS. We found that this combination of the latest verification results and the OS course not only provided a cleaner view of the kernel concepts and kernel structures, but also made every programming task more precise and concrete. To further carry out this belief, when mentoring students with their course projects, I tried to help them get used to first writing specifications (maybe informal) before starting the actual programming. Some students who

chose to implement device drivers or network stacks told me that these specifications helped them untangle the hardware details and sped up their development. Through this process, two students expressed strong interests in these layered specifications and kept working on their projects after the course.

Mentoring Experience

Hengchu Zhang is one of the students who continued to work on the OS course project. During his senior project, he attempted to verify an implementation of the synchronous IPC under my guidance. Compared with the original IPC implementation of mCertiKOS, Hengchu's version performs better but heavily depends on the scheduling primitives, which makes it difficult to reason about. During our weekly one-on-one meetings, I helped him reduce this problem into two separate parts: (1) verifying that the implementation satisfies the specification defined in terms of scheduling primitives; and (2) proving the synchronous property over the specification alone. Hengchu solved these problems under my mentorship, and his implementations, along with the proofs, were finally integrated into the CertiKOS code base. After this research experience, Hengchu decided to pursue a Ph.D. degree in the field of formal verification and joined the University of Pennsylvania in Fall 2016.

I enjoyed this mentoring experience a lot. For the past year, I also have been working closely with Mengqi Liu, a second year a Ph.D. student at Yale, on verifying cyber-physical systems with real-time guarantees. Mengqi, our advisor Prof. Zhong Shao, and I divided this project into two phases: (1) porting the existing CertiKOS framework to the ARM platform and verifying the functional correctness of an embedded system; and (2) proving the timing property of the system. I helped Mengqi design the ARM machine model and guided him to finish the first phase. As for the second phase, modeling time in a practical way was a big challenge. Mengqi and I spent several hours a week doing a long run of literature review together and working out a solution to model the flow of time by "counting" the transition steps on the ARM abstract machine. We developed a systematical way to encapsulate these counts into the specification and utilized this information to reason about the timing properties of the whole system. This line of work has been turned into a conference paper submission.

Courses to Teach

Based on my background, I would like to teach courses in compilers, programming languages, operating systems, formal methods, program verification, and software engineering. I believe that formal methods are best taught in association with sophisticated low-level systems software; and software engineering principles are best conveyed by teaching students programming language theory and foundation of logics. I am looking forward to the opportunity to expanding my teaching and mentoring experience in a world-class university.