Teaching Statement

Kihong Heo

1 Philosophy

I believe that university-level software education should train students to deeply understand programming language theories and to develop strong implementation skills. On the theoretical side, it should focus on the permanent principles of computer science rather than simply teach coding techniques. On the practical side, it should provide skills on how to write correct programs and interact with other developers. Unlike other fields in science and technology, students in computer science can experience and realize the cutting edge technologies in class. Therefore, all of my lectures will be accompanied by assignments or laboratory classes that enable students to enjoy state-of-the-art programming systems from the programming language and software engineering areas.

2 Program Analysis: Logical and Probabilistic Reasoning (Grad)

Lecture   At the graduate level, I believe lectures should introduce core techniques to analyze the behavior of software. First, it covers the theoretical background of program analysis, and its various applications such as verification, bug-finding, and code optimization. Second, I want to introduce the fundamental limitations of conventional program analysis and a way to proceed to the next level related to my recent research. In particular, I will discuss integrating program analysis with probabilistic reasoning with big data and human interaction. Finally, I will introduce other relevant techniques (e.g., fuzzing, concolic testing, or model checking) developed in the programming language and software engineering areas and discuss possibilities of combining them with probabilistic reasoning. Through the lectures, students will understand the characteristics of each approach and be able to compare their pros and cons.

Assignment   Students will be given assignments for implementing simple program analyzers, with various logical and probabilistic techniques. In particular, students from different research areas will find program analysis problems in their own areas and solve them by themselves. Moreover, assignments will provide a chance for students to experience to detect real security vulnerabilities (e.g., CVEs) using program analysis techniques.

3 Principles of Programming (Junior Undergrad)

Lecture   The goal of this lecture is to introduce principles of programming. The central idea to deliver is the concept and history of software: it is not that software was created to *run* hardware, but that hardware was developed to *realize* software. In addition, lectures will provide training for computational thinking, not merely time-varying trends for coding skills. Finally, I want to let students enjoy the beauty of software—*abstraction*—that simplifies all the non-essential details and enables us to focus on the essence. By understanding the concept of abstraction, students will learn how to build large programs and how to collaborate with other developers.

Laboratory   I plan to let my students experience the key concepts introduced in the class without any low-level distractions. The assignments and practices will be designed with high-level languages in which most of the cutting-edge technologies are embedded. Moreover, I want to emphasize that programming is not only for the interaction between human and computer anymore, but also for the communication between humans (e.g., open source community). This will be accompanied by "citizenship education" for software community including coding convention, basic tools (e.g., editor, compiler, and build tools), and libraries.

4 Programming Languages (Senior Undergrad)

Lecture   The goal is to introduce the common underlying principles behind various programming languages and their theoretical backgrounds. Furthermore, the lecture lets students enjoy high-level technologies in the programming language community such as type systems and program analysis. By doing so, students will think about the limitations of state-of-the-art programming languages and discuss ideas to solve the problems.

Assignment   The assignments will be designed for students to realize the essence of programming languages by implementing an interpreter. They will understand the principles of common concepts in various programming languages such as function, module, and exception. Also, the assignments will help students experience advanced topics such as compilation and type systems that are closely related to language design.

5 Research Advising (Grad)

For graduate students, I want to first emphasize the importance of vision and attitude rather than skills. I believe that world-leading research can only be started from one's own strong motivation, not blindly following other researcher's results. Therefore, I want to help my students cultivate problem finding, critical thinking, as well as reasonable practical experience. During a post-doc study at the University of Pennsylvania, I have advised a Ph.D. student and applied two separated coaching strategies: weekly research meetings for high-level discussion, and casual code reviews

via GitHub. In my experience, this twofold advising process is helpful for students to learn how to distinguish the essence of research problems from the low-level implementation, while also teaching them how to design and build a large software system. I plan to continue this style of advising with the help of post-docs and senior Ph.D. students in my own group in the future.

I also want to emphasize the importance of communication in research. This will help students understand that most of our research is carried out in collaboration with multiple people and evaluated by others in the community. To improve students' interaction skills, I will provide communication and writing opportunities as much as possible including weekly individual and group meetings.