Teaching Statement

Shan Shan Huang
ssh@cc.gatech.edu

My teaching philosophy centers upon two tenets: Students should be taught to recognize assumptions and encouraged to challenge them; Students should be taught in ways that suit their particular styles of learning.

One of the most difficult skills to acquire is the ability to think critically about the assumptions made in the development of an idea. Students should be reminded every step of the way that there are no absolute truths in computer science (with the exception of mathematical theorems), only solutions built upon assumptions that can be invalidated or shown irrelevant.

In my own experience as a student and as a lecturer for CIS410/510 (Object-Oriented Languages and Systems), I found the best way to encourage this form of critical thinking is to ask questions before giving the answers, and after giving the answers, ask even more questions about the answers. For example, when teaching Object-Oriented design patterns, we can first present a design problem, and ask the students to develop their own solutions. This gives the students a chance to develop their thoughts, before being boxed in by the thinking of those who came before them. When a particular design pattern has been introduced as a solution, we should ask again, "When would this solution *not* work? Is this solution just a band-aid for a deeper problem? Would this problem even exist if we used a different language? What other problems would arise if we did use a different language?" This type of questioning sends students the message that everything in computer science is relative to a set of assumptions; it encourages them to not see the teacher as an authority, but someone whose knowledge can serve as a stepping-stone to discovering new ideas. This type of teaching is equally beneficial for the teacher: what better place than fresh, pre-indoctrinated and brave minds for new ideas to originate?

I also believe that each student has a different learning style. It is encumbered upon the teacher to appeal to these different styles. One of my favorite teaching quotes is accredited to Tony Hoare: *"You cannot teach beginners top-down programming, because they don't know which end is up."* This analogy is broadly applicable to the teaching of any subject, where "top-down programming" may be replaced by "from theory to its applications", or "from the big picture down to the details". Furthermore, not only do beginners not know which end is "up", different beginners see different ends as "up". Thus, it is crucial to interleave the two "ends", the theory and the applications, the general and the specific, when introducing new ideas. For example, when I gave a survey on modern type systems, I started with a broad overview of the different type systems that is accompanied by small example programs written using these different systems. These examples helped illuminate the differences in the expressiveness of these type systems and their trade-offs. When I introduced the soundness of type systems, the progress and preservation theorems were accompanied by examples of exactly how programs may go wrong in unexpected ways if these theorems cannot be proved. This approach helped both the students who learn best from the abstract and those who learn best from the specifics to benefit from the process.

Teaching is an evolving process. I believe in the principles of encouraging students to challenge assumptions, and appealing to different learning styles. But I am sure I will discover and evolve the specific techniques to carry out these principles throughout my teaching career.