Teaching Statement – Anvesh Komuravelli

My passion for teaching stems from the personal satisfaction I obtain by helping someone learn something new and the numerous benefits it has on my own understanding. Moreover, given the practical nature of Computer Science, I am motivated to take the responsibility for shaping tomorrow's computer scientists who will impact the whole world. With this in mind, I have the following main teaching objectives: students should be able to (a) summarize and explain concepts in *abstract* and *intuitive* ways, (b) develop the right *questioning and problem-solving* skills, and (c) propose *new problems* and invent *new approaches* for solving them and think beyond what exists today.

Abstract and intuitive characterization of concepts is essential for a better understanding which helps in remembering and applying the concepts where and when necessary. To facilitate this, a teacher should be *motivating* and *engaging* with the students to make sure they really understand their significance. This is especially true when the concepts form the core of the subject and the teacher should strive to be as *creative* as possible. I still remember my high school days when I used to enjoy learning the basic concepts of Physics because our teacher had a very illustrative style of teaching with beautiful pictorial representations on the blackboard. Perhaps surprisingly, this also led me to appreciate the mathematics that actually helped solve Physics problems and Physics used to be my favorite subject. Two years ago, I was reading a research paper on logical analysis of hardware systems where the mathematics was so detailed and seemed too technical. But, when I later attended a presentation of the paper at a conference with wonderful pictorial representations using Venn diagrams and such, it all made sense and seemed so intuitive! I wish the paper made use of those representations. A good representation helps students grasp the concept easily, explain it intuitively, and apply it in non-trivial ways in a different setting.

To help students develop questioning and problem-solving skills, a teacher should first create an *interactive* environment, maintaining a natural chain of thought that evolves from simple to complex reasoning. A teacher should always try to put herself in the shoes of the students to figure out how the presentation will be perceived and whether some important detail has been overlooked. Then, the teacher should herself ask relevant questions and take the students through the process of analyzing and answering them by means of active learning. Two years ago, I was co-teaching (with another PhD student) two 8-hour sessions on algorithms for *Sorting and Searching* to 30 bright, high-school students as part of the IdeaMath summer program. To get them thinking about algorithms and their efficiency, we distributed small pieces of paper with numbers written on them with the goal of sorting the numbers using a comparison-based sorting algorithm. The key operation, comparison, was simulated by talking to a neighboring student about his/her number. This simple exercise made them later appreciate questions regarding the computational complexity of various sorting algorithms and why several algorithms exist in the first place. Given that this material is usually taught in introductory undergraduate courses, we were pleasantly surprised from the final course feedback that the students found the course to be very useful.

To be able to propose new problems and invent new approaches, especially at the graduate level, a teacher should engage in one-on-one discussions with the students, listening to their ideas and giving constructive and critical feedback. Such advising style has helped me tremendously to grow in research. In the beginning, I used to pick the simplest and not so important problems as interesting research questions to pursue. But, the discussions with my mentors made me realize their level of importance and dare to look beyond what's right in front of my eyes. Along with such advise, students should also be taught the importance of the values such as *persistence*, *patience*, and *self-confidence*. Unfortunately, there is little emphasis today on teaching the students *how* to acquire such values. I believe that a historical perspective of evolution of the science is one good way of acquiring such qualities. I am currently reading the book *Mechanizing Proof: Computing, Risk, and Trust* where the author talks about the history and the sociology of a subject matter close to my research interests. This book made me realize, among other things, how motivated and *persistent* the forefathers of Computer Science were

to realize their objectives, despite the negative criticism that surrounded them. I am very interested in *creating and teaching a course* that describes how the various concepts came into being and what motivated the great minds of the past for doing what they did. Going back to the grass roots of the problem and critically analyzing the various bits and pieces has helped me discover beautiful insights for my own research. This process is time-consuming and students should be taught to be *patient* and that it is okay to take time. Last year, I met an Italian professor in his late 60's at a conference and when I was trying to understand his research by asking many basic questions, he tells me within the first few minutes of our interaction: "*I think you will be a great teacher!*" That was the greatest compliment I have ever received. When a teacher advises graduate students, she should also be careful to give them enough freedom to let them explore on their own, find out their capabilities and limitations, and more importantly, gain the *self-confidence* necessary for conducting research. I found this especially useful for myself and I am grateful to my advisor for that.

I am very self-motivated at teaching and give my best to improve student understanding. When I was a teaching assistant for a course on automated theorem proving, the introductory lectures assumed a basic knowledge of logical inference and deductive proofs which some of the students lacked. So, I and the other teaching assistant volunteered to teach optional lectures on logical inference where we showed the theory but also some concrete examples explaining the step-by-step process. In one of those lectures, I was quite happy to receive the compliment "*this one hour was more useful than the 90-minute lecture I attended!*" from a student. On another occasion, when I was a teaching assistant for an introductory algorithms course, I found the students confused by the nested minimum and maximum operators involved in the definitions of worst-case lower and upper bounds on computational complexity. This only got more complicated with the probabilities in randomized algorithms. I sensed the problem and realizing that the lecture notes weren't detailed enough, I volunteered to create a supplementary handout explaining the concepts with small examples on a few pages and the students found that to be very useful. Being a student for all my life, I have witnessed the good and the bad in teaching and to be on the better side, I have enrolled myself in the Future Faculty program at my university (CMU) and already finished most of the coursework.

I look forward to teaching courses on introductory algorithms and complexity, automata theory, theory of computation, logic for CS, and imperative programming (with the notions of invariants/proofs) at the undergraduate level and courses on theorem-proving/constraint-solving, automatic verification techniques including model checking, SAT solvers and decision procedures, abstract interpretation, and program analysis at the graduate level. As I mentioned earlier, I am also very interested in designing a course on the history and the sociology of the field and, in particular, of the areas close to my research interest, especially at the graduate level. On the one hand, this will give a cultural perspective of CS to the students. On the other hand, this will raise the social awareness and convey the broader importance associated with the research problems which helps motivate the graduate students. Having said that, I am also happy to take up other courses, including introductory courses for non-majors. As my research expands in the future, I expect to be able to also lead graduate level courses on programming aids in general, such as program synthesis and topics from software engineering like model-driven development.

In conclusion, I believe that teaching is a process which helps both the students and the teacher. Good teaching often involves a divide-and-conquer approach and explaining how the individual pieces fit together often brings out unforeseen connections and improves the teacher's own understanding. Given that I am a finite being with finite resources, I am always ready to acknowledge my limitations while being passionate about learning new concepts and expanding my breadth.