Teaching statement of Konstantine Arkoudas

January 2005

I view teaching as an integral part of an academic career, and as a craft of the highest social importance—we all know that committed teachers can have a decisive positive impact on the lives of their students. As a teacher, I will strive for such impact, doing my best to not only transmit knowledge but to instill an attitude of intellectual curiosity and critical thinking, and to cultivate a lifelong love of scholarship. I particularly welcome the opportunity to help underrepresented students from disadvantaged backgrounds to achieve their full potential.

Teaching can be just as rewarding for the teacher as it is for the student. I have always found it deeply gratifying to help others to understand complex ideas. It can also be a great aid to one’s own research. The process of having to present a topic in a structured and lucid manner forces the teacher to take a bird’s eye view of the subject, to examine the material from multiple perspectives, to discover new connections, and to detect new research possibilities. Feedback from students has often led me to consider new angles, and even to detect problems and weaknesses that had previously escaped my attention.

My teaching tends to be very example-oriented, especially when I am first introducing a new concept. In my experience, it is only after going through several concrete examples of varying degrees of difficulty that most students are capable of making the leap to the general case. Examples are also indispensable for motivation purposes, and coming up with such examples is one of my favorite things about teaching.

Inevitably, aspects of my personal philosophy about Computer Science become reflected in my teaching. One of them is the view that logic and discrete mathematics are to Computer Science what calculus and differential equations are to traditional engineering disciplines; they are the foundations, and need to be emphasized as such. Solid problem-solving skills can only be built on top of deep understanding, not just shallow heuristic skills, and deep understanding invariably requires mastery of the underlying foundations. I also try to inculcate the belief that computer programs are formal mathematical objects amenable to rigorous analysis, not amorphous masses of code to be coaxed into producing more or less acceptable results.

I have acquired extensive teaching experience over the years. In the Spring of 2004 I co-taught the graduate seminar Logic and Artificial Intelligence at RPI, which covered research topics in the intersection of logic and artificial intelligence (such as synthetic character generation, inductive logic programming, etc.) I had to deliver lectures, as well as oversee and evaluate student projects. As a postdoc at MIT in 2001, I co-taught the course Object-oriented Dynamic Languages (6.894), in which we taught advanced implementation techniques for highly dynamic object-oriented programming languages (such as Dylan). My duties were the same as for the aforementioned course.

As a graduate student at MIT I was a TA for Patrick Winston’s famous Artificial Intelligence (6.034). I had to teach weekly recitation sections, grade homework assignments, and hold weekly office hours. At MIT I was also a TA for another AI course, Knowledge-based Systems (6.871), taught by Professor Randall Davis, which covered expert systems and fault diagnosis. In addition,
at certain times during graduate school I was a free-lance teacher and consultant. One of my most rewarding teaching experiences was tutoring the programmers of Zoesis Multimedia Inc. (a company in the Boston area creating synthetic characters for the entertainment industry) advanced functional programming techniques.

In summary, teaching is something I enjoy greatly and take very seriously. I have taught several subjects in my career so far, and am eager to tackle new ones in the future. In view of my background, I would be best qualified to teach courses in programming languages, artificial intelligence, data structures and algorithms, compilers, and logic and/or discrete math, both at the undergraduate and graduate levels. I could also teach graduate seminars on formal semantics and type theory; automated deduction (theorem proving); and formal methods in software engineering.