Teaching Statement

Ankush Das

The opportunity to teach and collaborate with undergraduate and graduate students is my primary motivation for seeking an academic career. Teaching provides the unique privilege of making an impact on society by shaping the future generation. It is also a way for me to give back to the academic community.

Teaching Interests. Although I would love to teach many graduate and undergraduate computer science courses, I would be particularly excited to teach **programming languages** and **formal methods**. At the graduate level, I would be interested in teaching courses related to **formal verification**, **resource analysis**, **type systems**, and more broadly **blockchains and smart contracts**, and **analysis of distributed systems**. Also, I am currently applying my research to build secure systems, and could teach courses like **computer security** and **applied cryptography** in the future. At the undergraduate level, I would enjoy teaching courses like **foundations of programming languages**, **mathematical logics**, **program logics**, and **compilers**. Freshman courses are more challenging to teach since first-year students are still adapting to the new university culture, and often transitioning from memorizing in high school to critical thinking in universities. Noting these challenges, once I have gained experience with the above courses, I would like to teach more fundamental courses like **introduction to programming**, and **functional and imperative languages**.

Teaching Philosophy

My research has been driven by two core principles that I plan to carry to my teaching methods: *(i)* exploring the beautiful connections between mathematics and computer science, and *(ii)* learning by programming.

Math and Computer Science. All areas of computer science are deeply rooted in mathematics: machine learning is rooted in statistical theory; cryptography is rooted in number theory; programming languages are rooted in proof theory. In fact, my own interest in computer science was sparked by its mathematical foundations. I want to approach my courses by explaining the math behind each core concept. With a mathematical approach, students would cultivate the ability to design and employ algorithms, as well as critical thinking and analytical skills. Relatedly, studying computer science is an iterative process: we constantly improve the computer systems we study. In that spirit, my approach will empower students to build the next generation of computer systems upon strong theoretical foundations.

Learning by Programming. I believe the most effective way to learn programming is by, well, programming! There are core programming lessons associated with each course: data structures, algorithms, compilers, operating systems, etc. But these lessons can often get lost between the technical jargon and idiosyncrasies that come with traditional languages. Therefore, following CMU's long-standing tradition of creating tools for educational purposes, I would like to create my own prototype languages for my courses directed towards each concept. My own research, which was centered around designing new languages, has taught me unique skills to implement such prototypes. With these core lessons, students will learn how the technological devices around them function, how to develop software in a modular fashion, and utilize algorithms to enhance its safety and efficiency. My learning-by-programming approach has been inspired from **learning-by-doing and active learning theories**. Research shows that these theories increase engagement, improve critical thinking, and deepen understanding.

Teaching Experience

I have had the privilege of serving as the **teaching assistant** for 2 courses: one undergraduate course on **Constructive Logic** and one graduate course on **Types and Programming Languages**.

For the undergraduate course, I particularly enjoyed teaching recitations where I would often use **existing tools to further develop important concepts**. For instance, to illustrate the power of logic programming, I used Prolog to solve well-known logic puzzles like 'The Zebra Problem'. The students not only enjoyed the experience, but also learned how real-world puzzles can be encoded as logic programs, and more broadly, how computers can provide solutions to real-life problems.

Teaching courses at both levels, I learned that the knowledge and expectation of undergraduates is very different from that of graduate students. Consequently, the teaching style needs to be different. I observed that undergraduates would get distracted easily, so I would introduce interactive exercises or recollect some interesting stories related to the topic to retain their attention. On the other hand, graduate students are already committed to the course, and I would encourage group discussion to hear their ideas and would act as a sounding board for affirmation and guidance. Even my evaluation criteria was different. I assessed undergraduate students using assignments, tests, quizzes, and group projects that focused on **programming** and **applications of concepts**. For graduate students, I used more theoretical and often **open-ended questions** demonstrating that some questions cannot be answered with a simple 'yes' or 'no'. The students' performance also helped me **identify the concepts that were not communicated well** and **recognize the students that are lagging behind**. This helped shape my future lectures focusing on concepts that were not taught well. I also routinely held extra office hours and organized extra recitation sessions to support struggling students. Finally, I prepared course-specific feedback forms to evaluate to assess whether my teaching methods are working.

Since graduate courses are advanced, they also usually assume some prerequisite knowledge. However, students usually come from such diverse communities that everyone may not have the same background. In fact, my own undergraduate training was back in India which significantly differs from the training in the US. So, when I joined CMU, I had trouble following the initial courses where instructors naturally assumed a US-based education. Since then, I have realized the importance of **recognizing the diversity in students' background and connecting with them at their level**. To this end, I plan to take an initial test in my courses to survey whether students are familiar with the prerequisite knowledge I presume. If not, I plan to take additional classes to ensure a level-playing field for all students.

Mentoring Experience

I have mentored two excellent undergraduate students for different research projects within my graduate research. Since the students were unfamiliar with my research area, these projects also became teaching ventures. First, I learned that different students have different needs and desires: the first student wanted to work on a core theoretical project while the second student wanted to work on an implementation-based project. Instead of just assigning a project, I worked with the students to design their projects to ensure they were excited and motivated by it. Second, I found it was important to understand the student's background and design a project based on that. The former student was a sophomore with little programming experience, so we started with a smaller project on designing a toy programming language. She then quickly learned the relevant skills which set the stage for the project. With my students, my goal was to develop **research independence, communication skills, and resilience so that they remain calm in the event of failures**. To build independence, I gradually transitioned from a hands-on to a hands-off teacher encouraging their own ideas in the later half of the project. I always treated their ideas with respect and learned that that effective two-way respectful communication was the bedrock of a successful collaboration. To build written skills, I solicited regular project reports and returned them with constructive criticism. To ensure that they were not discouraged from failures, I would recount my own experiences and discuss lessons gathered from each failure. The first project led to a **publication at a top-tier security conference** and we are currently writing a paper about the second project.

Inclusive Teaching during COVID. A UN study reports that 'The COVID-19 pandemic has disrupted the education of nearly 1.6 billion students in more than 190 countries'. The same study also reports that this disruption disproportionately impacts students from underrepresented minority communities. At this time, it is crucial to support these students. The first step here is to raise awareness regarding the different educational, cultural, and socio-economic situations students belong to. It is also important to prepare lecture notes, videos, take home assignments, and examinations taking the class diversity into account. I would also like to post them on a public repository and publicize them so as to reach students beyond my university. It is equally important to assess students that are struggling and reach out to them early with special lectures, recitations, and office hours. Research shows that inclusive teaching builds better thinkers, improves academic outcomes and boosts student involvement.