Sankha Narayan Guria

Teaching Statement

My teachers have inspired and encouraged me in my academic journey thus far, and I view teaching as my way of giving back to the next generation of young minds that will become engineers and scientists and have a huge impact on society. I discuss my teaching philosophy and my teaching and curriculum design experience and how it provides me with a solid ground for the future courses I want to teach.

Teaching & Curriculum Design Experience

Curriculum Design Experience. I was a part of a Teaching Innovation Grant at UMD to redesign the Organization of Programming Languages class[1] to modernize the course to support the online modes of teaching and update the materials. I specifically helped research Property-based Testing materials as an addition to the course. This involved designing suitable examples that enable students to test basic data structures like binary search trees in languages such as Ruby and OCaml, *i.e.,* the different language paradigms already taught in the class. However, the main takeaways for me were the discussions with lecturers and professors to get a glimpse of the thoughtfulness behind designing materials, assignments, and exams to ensure best learning outcomes for students. I learned about backward design [4], *i.e.,* designing stratified student learning goals followed by development of course materials and exams accordingly. For example, grades students earn should show the depth of knowledge they have about the subject and accordingly points for exam questions are distributed to test the students skills in those topics respectively. As a future faculty, I will apply these learnings to all my future courses.

Mentoring Experience. I mentored a team of students at Technica Tech + Research track, the largest hackathon for women and non-binary students, to introduce them to the idea of Property-based Testing (PBT) in Python using the Hypothesis library [1] and using it to check correctness in large software. I designed some exercises for the students to get familiar with the idea of PBT [2]. I also mentored the students on a one-on-one basis to setup their laptops, discuss and clarify their ideas on Property-based Testing, and helped them write their own properties for testing some numeric algorithms in Python. Mentoring is an important part of academic career and as a faculty I look forward to mentoring my research group and students from my class at a larger scale.

Teaching Philosophy

Ground examples in the real-world. I believe students learn best when examples and exercises are rooted in real-world applications. As a TA for the first iteration of the redesigned undergraduate Compilers class at UMD, it gave me a unique experience in this aspect. The redesign of the class used a Nanopass style [3] compiler where in each class students compiled a gradually increasing subset of the Racket language to a x86 native binary that directly ran on the host operating system. This was a good way to give students an idea of how real compilers work, as the produced code ran natively without any emulators/simulators. Moreover, this removed the disconnect between compiler textbooks, that disproportionately focus on parsing, and modern compilers that need to compile and optimize many sophisticated language features. Additionally, being a new course, the curriculum was being designed as the semester went on, giving me a unique opportunity where I could interact with the students in office hours and pass feedback to the professor. This feedback was incorporated into the next lecture, making the course adapt to the requirements of the students as the semester went on. Certain topics like foreign function calls were included into the curriculum as a result of this feedback.

I continue the spirit of grounding examples in real-world with supporting the Build It, Break It, Fix It (BIBIFI) [2] contest infrastructure at UMD on a volunteer basis. It provides a platform for students to write secure software, and exploit vulnerabilities in other student's submissions for the Computer Security class at UMD. Running the infrastructure involves talking with TA and students to take care of the their changing requirements over each iteration of the class and shipping improvements and security updates to keep the contest server running during the duration of the course. This approach works much better for students who can apply the concepts rather than just hearing about it in lectures.

[1] https://www.cs.umd.edu/article/2020/07/department-receives-three-teaching-innovation-grants

[2] https://github.com/plum-umd/tech-plus-research-PBT

Understand student background and adapt. I have seen the powerful effects of success and discouraging results of failure when instructors do not understand student background. In one of the courses I took as an undergraduate, the instructor assumed some preliminary knowledge for students, that many—including myself did not posses. Despite confusion and lack of engagement with course material among the students, the instructor went forward with his original lesson plans, resulting in little learning and lower grades for many students. Based in part on this experience, I make a concerted effort to understand student background and adapt. This has been my guiding principle for my teaching assistant duties during discussions and office hours. For example, when I served as a teaching assistant for the Introduction to Systems (C Programming) class, I spoke to students in my discussion section to realize they have trouble with pointers and memory allocation. I spent extra time to revise these topics before midterms, and it gave much better engagement from students and my section's scores were also much better than the class average. This has further reinforced my belief that this is an effective way to engage with students and make them connect with the material better.

Synergy of teaching and research. I firmly believe that teaching and research are activities that support each other. This is one of my motivations to design tools in my research that can be used to teach better. One of my main observations through previous teaching and mentoring activities has been that students are primarily taught how to write code and by the time they graduate with a CS degree they are good at it. However, checking if the code does what it was intended to do requires some careful testing at minimum, a skill that is not a focus of present day CS curriculum. In my interaction with students, I have found they have difficulty thinking through their code or designing test cases for checking different behaviors of their program. In fact, test cases are crucial when writing code in the industry, a skill students learn on the job. I believe my research on RbSyn can be adapted to design interactive tools that ask a student to write tests, while the tool synthesizes the corresponding program. This flips the problem, *i.e.,* the student writes the test, while the computer synthesizes the code. The student is successful when the computer generates a program they were expecting. If it does not then their tests are not sufficient to cover all behaviors of the program under test. As a future faculty, it will be a focus area to design tools that assist education.

Future Courses

As a professor, I look forward to teaching graduate course in programming languages, verification, and synthesis. I will teach classes for undergradutes on programming languages, compilers, and other introductory computer science classes.

References

[1] David R. MacIver, Zac Hatfield-Dodds, and many other contributors. "Hypothesis: A new approach to property-based testing". In: (Nov. 2019). DOI: 10.21105/joss.01891.

[2] James Parker, Michael Hicks, Andrew Ruef, Michelle L. Mazurek, Dave Levin, Daniel Votipka, Piotr Mardziel, and Kelsey R. Fulton. "Build It, Break It, Fix It: Contesting Secure Development". In: *ACM Trans. Priv. Secur.* 23.2 (2020), 10:1–10:36. DOI: 10.1145/3383773.

[3] Dipanwita Sarkar, Oscar Waddell, and R. Kent Dybvig. "Educational Pearl: A Nanopass framework for compiler education". In: *J. Funct. Program.* 15.5 (2005), pp. 653–667. DOI: 10.1017/S0956796805005605.

[4] Grant Wiggins, Grant P Wiggins, and Jay McTighe. *Understanding by design.* Association for Supervision and Curriculum Development, 2005.