My Teaching Philosophy

Whether teaching freshmen the basics of induction proofs, or supervising seniors on publications, it is a privilege to help students in the classroom and the lab. In both settings, I seek to cultivate an atmosphere of learning, where students feel empowered to ask questions and test their own ideas. This value guides my teaching and mentoring approaches and lays the groundwork for all my instructional experience.

Teaching Approach. To cultivate an atmosphere of learning, I strive to instill passion for the material, to treat students as colleagues instead of subordinates, to inspire independence, and to make myself accessible. For instance, during office hours, when students come to me with questions I first ask "*how do you think we can solve this problem?*" followed by a series of Socratic questions, designed to lead them to the answer. Students need encouragement. I phrase these questions with "we" language to foster inclusivity but also to demonstrate by example my passion for learning in hopes of spreading that passion to them. While this approach often caused my office hours to extend beyond the original time slot, it was worth each additional minute to see students walk away with a sharpened understanding. I found this approach helps students feel like respected colleagues and brings out their independent problem solving. I felt rewarded when students pointed out this detail in my teaching evaluations when many wrote: "Jacob takes time to provide helpful explanations".

Teaching courses that span the full curriculum: from introductory freshman level to advanced senior level, helped me tailor my teaching approach to match the experience of students. Despite the differences in course levels, the fundamentals of my pedagogy remain the same. My explanations always rely on concrete examples. Examples create clarity. I also make sure to only use examples from courses the students have already taken, hence how I tailor my teaching to match students' experience. But examples need relevance. For instance, it is far easier to convince students skeptical about the need for induction proofs when I show how these proofs help them check that their recursive functions do not run forever.

Teaching Objectives. As an instructor, my objectives remain the same across all levels. I want students to cultivate their own independent thinking skills. To accomplish these objectives, I make lectures, code reviews and recitation sections as interactive as possible. Interactivity encourages questions. When I led discussion sections for a Discrete Structures course, I periodically paused throughout my lectures to ask students for their questions. Questions create discussions. Before presenting key concepts, I follow up students' questions by asking "*what do you think comes next?*" to stimulate discussion and hone their ability to arrive at the answer independently. Anecdotally, I noticed student engagement and grades improved after I began to incorporate these pauses and questions into lectures.

My Teaching Experience

Shaped by my positive experiences as an undergraduate course assistant, throughout my PhD I eagerly served as a teaching assistant (TA) for multiple classes over multiple semesters. My TA experience comprises four different undergraduate courses at the University of Illinois Urbana-Champaign: CS126 Software Design Studio, CS173 Discrete Structures, CS374 Algorithms and Models of Computation and CS421 Programming Languages and Compilers. These courses span the full undergraduate curriculum, from freshmen (CS126) to sophomores and juniors (CS374) up to seniors (CS421). These courses also span the entire computing stack: from high level algorithms (CS374) to intermediate level program semantics (CS421) down to low level C++ coding (CS126).

Teaching Responsibilities. As a TA, my responsibilities included running discussion sections (CS173, CS374), helping develop automated grading tools (CS421), leading code reviews (CS126), designing homework and midterm problems and grading them (CS173) and running office hours (CS173, CS374, CS421). I also guest lectured for CS477 Formal Software Development Methods, where I incorporated my research into the lesson by covering the foundations of abstract interpretation and automatic differentiation. This guest lecture gave me my first taste of how to choose material for a lesson plan while ensuring the chosen material remains accessible to undergraduates.

Courses I Can Teach

I envision myself teaching a variety of courses and seminars at both the undergraduate and graduate level, which I describe below:

Undergraduate Courses. At the undergraduate level, I would be delighted to teach upper level Programming Languages (PL) and Compilers courses. However building a robust pipeline of future PL researchers starts even earlier. Hence I also maintain an interest in teaching introductory programming and mathematics courses where I can integrate PL ideas like formal semantics into the curriculum from day one. Also, given my PL research intersects with continuous mathematics involving numerical computations, I also have an interest in teaching a Numerical Methods course.

Graduate Courses. My goals also include teaching graduate level courses in Programming Languages and Formal Methods. These courses would expose students to current PL research trends by reading recent publications and they would also let students pursue their own PL-related course projects.

I also envision designing my own course on Automatic Differentiation and Differentiable Programming. Since AD lies at the intersection of multiple domains including Programming Languages, Machine Learning, Graphics, and Scientific Computing, I plan to incorporate each of these perspectives into the class to make it as interdisciplinary as possible. To forge these interdisciplinary connections, I want students to read AD papers from PL conferences (e.g., POPL, PLDI) and read AD papers from other conferences, like ICLR, SC and SIGGRAPH. My goal is to bring students from these diverse domains together to find the common ground needed to stimulate interdisciplinary AD research.

Additionally, I aim to teach non-technical seminars that cover important PhD lessons. These seminars would cover how to pick research problems, how to mentor students, and how to navigate the job market.

My Mentoring Philosophy

Many of the most important lessons come from outside the classroom. Hence, I actively sought the opportunity to mentor students in research. During my time at UIUC, I had the privilege of mentoring five different undergraduate students in research. I have published research papers with four of these students. Furthermore, three are now enrolled in graduate school (MIT, UT Austin, UW Madison) and a fourth is now applying for PhD programs. These students have each expressed that their decision to pursue graduate studies was in part due to their positive experience working with me.

Mentoring Approach. My mentoring approach strikes a balance between providing attention and nurturing independence. To provide attention, I make myself accessible to my mentees. For example, I will meet with undergraduates multiple times a week to discuss their results and ideas. To nurture independence, I give mentees the freedom to tackle their assigned tasks however they think is best instead of micromanaging their solutions. For instance, when discussing programming tasks, I often ask undergraduates "*how do you think we can solve this problem?*". Much like my teaching, this question also serves to foster inclusivity and to show that I can learn from them just as they learn from me.

Mentoring Objectives. My end goal is to put the students I mentor on the project's critical path. This goal stems from my observations that students stay motivated when they feel included and take ownership of a contribution instead of working orthogonally on tasks separate from a publication.

My Mentoring Experience

My first mentee, Rem Yang, began working with me as a freshman. Over the next three years, I guided Rem to successful co-authorship of three papers [DAC21, POPL22, OOPSLA22]. In each paper, I made sure he got experience with all parts of the project: from writing code to running experiments to writing sections of the paper. By his senior year, these lessons prepared him to lead his *own* project [ICLR23] as the first author, which I supervised. That project, titled "Provable Defense Against Geometric Transformations" earned a **notable designation** at ICLR 2023, and, in combination with the other three papers he published alongside me, helped him earn an Honorable Mention for the CRA Outstanding Undergraduate Researcher Award and admission into MIT's CS PhD program where he is now enrolled.

I helped another undergraduate student Robert Nagel craft his own year-long independent study. In the plan we crafted, I set the scope of both experimentation and writing tasks to give him. This work culminated with Robert earning co-authorship on our paper [OOPSLA22].

My teaching experiences also shape my research experiences. I first met Atharva Sehgal and learned of his interest in research, while I was his TA. When the pandemic hit and his internship was cancelled, I reached out to see if he wanted to try research for a summer. A summer turned into a full year of collaboration. Observing his talent for experimentation led me to steer the project down a more empirical path to capitalize on his strengths. This success culminated with him co-authoring [DAC21] and helped inspire him to pursue a PhD in programming languages at the University of Texas. This experience taught me how inquisitive students in the classroom can become impactful collaborators in the lab.

In addition, I mentored and helped onboard two junior PhD students. One of the students, Shubham Ugare, co-authored a paper [DAC21] with me during the COVID-19 pandemic while he worked remotely from India due to travel restrictions. This experience taught me how to adapt my mentoring style to the remote setting by checking in regularly on both academic progress *and* personal well-being.

The experiences of guiding students through different parts of research, steering projects and their scope to match student strengths, and adapting to unforeseen circumstances all lay the groundwork of my advising approach for future PhD students.

References

[DAC21] Jacob Laurel, Rem Yang, Atharva Sehgal, Shubham Ugare, and Sasa Misailovic. "Statheros: Compiler for efficient low-precision probabilistic programming". In: *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE. 2021, pp. 787–792.

[POPL22] Jacob Laurel, Rem Yang, Gagandeep Singh, and Sasa Misailovic. "A dual number abstraction for static analysis of Clarke Jacobians". In: *Proceedings of the ACM on Programming Languages* 6.POPL (2022), pp. 1–30.

[OOPSLA22] Jacob Laurel, Rem Yang, Shubham Ugare, Robert Nagel, Gagandeep Singh, and Sasa Misailovic. "A general construction for abstract interpretation of higher-order automatic differentiation". In: *Proceedings of the ACM on Programming Languages* 6.OOPSLA2 (2022).

[ICLR23] Rem Yang, Jacob Laurel, Sasa Misailovic, and Gagandeep Singh. "Provable Defense Against Geometric Transformations". In: *The Eleventh International Conference on Learning Representations*. **Designated Notable - Top 25% of Accepted Papers**. 2023.