Teaching Statement—Federico Mora (fmora@berkeley.edu)

My teaching and mentoring philosophy is primarily influenced by universal design for learning (UDL) principles [Rose et al. 2006]. The core idea behind UDL is that there is no "normal" student and so focusing on teaching the "normal" student will ensure that no student has an optimal learning experience. Instead, developing a flexible learning environment ensures that all students can take the best path for their success. The UDL perspective draws heavily from work in disability studies and has obvious implications for learners who do not see themselves represented in curriculum designed for "normal" students. In the subsequent sections, I describe how I have put UDL principles into practice while mentoring and teaching. I conclude with a description of the courses I am excited to teach.

1 MENTORING EXPERIENCE AND PHILOSOPHY

I have mentored nine students across four universities in two countries. My students have won ACM student research competitions at POPL and PLDI, placed second at the ACM student research competition grand finals, won a SACNAS National Diversity in STEM presentation award, and gone on to rewarding careers in industry and academic programs at CMU, ETH, Princeton, and Stanford. In 2024, I was awarded a Demetri Angelakos Memorial Achievement Award (UC Berkeley EECS) and an Outstanding Graduate Student Peer Mentor Award (UC Berkeley) for "mentoring which went above and beyond the norm." In line with UDL principles, my mentoring philosophy focuses on strategy development, building knowledge, and fostering interests and collaboration.

Strategy Development. I have had the privilege of working with incredibly ambitious and hard working students. One of the most effective mentoring strategies therefore has been to help them set realistic goals, adjust these goals over time, and, when appropriate, intervene to put them back on track to achieve their goals. For example, I never tell students to participate in a student research competition. Instead, I like showing my students that these exist and, if they show interest, help them make realistic plans to participate. Part of this experience for students inevitably involves dealing with missed deadlines and other "failures." In these cases, I attempt to strike a balance between helping them update their goals and supporting them through extra meetings, proof walkthroughs, or pair-programming, as appropriate. Over time, my mentees become better at setting goals, planning for challenges, and monitoring their own progress.

Building Knowledge. Junior researchers often struggle to engage with existing research. Academic papers use unfamiliar language and symbols, and the ideas within and relationships between papers can be hard to decode. I have been able to mentor students effectively by helping them make explicit connections between research and material they have covered in courses. For example, I often ask students to read and present papers to me—often the same paper multiple times. During these presentations, I ask them to explain the relevant background material and related work. When students are missing background, I give them miniature lessons and, when necessary, encourage them to enroll in relevant classes. Over time, my mentees improve at drawing these connections themselves and become better at reading, understanding, and even communicating new research.

Fostering Interests. Students perform at their best when they are genuinely interested in their work and feel like a respected member of a team. I have been able to mentor students effectively by giving them a meaningful say in our projects. For example, I often ask students what they liked and disliked about the work since the last meeting we had, share what I am excited about, and then discuss how to optimize our plan moving forward. Sometimes students are excited by a new concept they learned and we can read more about it; sometimes they are excited by a programming task and we can budget more time for engineering. Even for projects with fixed goals, I have found that there is always a way to optimize the student's learning experience. Over time, my mentees find topics they are excited about and develop a sense of ownership and pride in their work.

2   TEACHING EXPERIENCE AND PHILOSOPHY

I have taught, either as a teaching assistant or guest lecturer, at the undergraduate and graduate levels for seven unique courses at three universities in two countries. These courses cover topics including programming languages, formal methods, software engineering, and artificial intelligence. My excellence in teaching has been recognized by both my department and university. In 2022, I was awarded an Outstanding Graduate Student Instructor Award (UC Berkeley); in 2023, I was awarded the highly selective Outstanding Teaching Assistant Award (UC Berkeley EECS). I try to apply the same mentoring principles described above while teaching, but these are difficult to scale to large classes. Therefore, my teaching philosophy focuses on complementing my mentoring philosophy with autograders and inclusive course design.

Autograders. Autograders are an important part of my teaching philosophy because they help increase instructor contact time by reducing the grading burden. I have helped develop three autograders, including the autograder used for all assignments in one version of UC Berkeley's introduction to programming languages and compilers course (CS 164). Autograders are also interesting because they can enable flexible deadlines and immediate feedback to students. There is (mixed) evidence to suggest that these features can reduce stress, increase equity, and improve student learning outcomes [Cai et al. 2023; Hills and Peacock 2022]. I am actively searching for the best way to integrate these features into my courses, and I plan to periodically revisit the integration and relevant research throughout my teaching career.

Inclusive Course Design. While teaching programming languages and compilers for the second time, I built an audio interface to the course compiler and designed a section for students to implement their own. The implementation followed Schanzer et al. [2019] but the idea came from Kleege and Wallin [2015], who use audio descriptions as a pedagogical tool. Despite their being no theoretical reason for programming language interfaces to be text-based, all undergraduate compilers courses that I have encountered ignore alternate interfaces, like audio and graphical. Our audio interface highlights this fact; makes students think about the accessibility implications of their work; and serves as a useful motivating example for technical concepts, like tree traversal and editing algorithms. The following semester, I participated in UC Berkeley's "Preparing Future Faculty: Designing Courses through the Lens of Universal Design for Learning" and expanded the material to a full semester syllabus for a new introductory programming languages course. This new course uses my current understanding of research in education to make technical programming languages content accessible to as many students as possible. I am excited to put this new course into practice and refine my approach as I learn more about the relevant research and gain experience in the classroom.

3   COURSES

I am ready to teach a broad range of formal methods, programming language, and software engineering courses. At the undergraduate level, I am excited to teach Discrete Math; Programming Languages; Compilers and Interpreters; Software Testing and Verification; or related topics. I am also open to teaching introductory courses. At the graduate level, I am excited to teach Automated Reasoning; Formal Methods; Program Verification; Program Analysis; or related topics. I would also be extremely interested in developing a graduate course on domain-specific automated reasoning. This course would combine ideas from programming language design and implementation with topics in applied logic, like automated theorem proving and satisfiability modulo theories.

REFERENCES

Zhihui Cai, Yang Gui, Peipei Mao, Zhikeng Wang, Xin Hao, Xitao Fan, and Robert H. Tai. 2023. The effect of feedback on academic achievement in technology-rich learning environments (TREs): A meta-analytic review. *Educational Research Review* (2023). https://doi.org/10.1016/j.edurev.2023.100521

Melissa Hills and Kim Peacock. 2022. Replacing Power with Flexible Structure: Implementing Flexible Deadlines to Improve Student Learning Experiences. *Teaching & Learning Inquiry* (2022). https://journalhosting.ucalgary.ca/index.php/TLI/article/download/73960/56002/230810

Georgina Kleege and Scott Wallin. 2015. Audio description as a pedagogical tool. *Disability Studies Quarterly* (2015). https://dsq-sds.org/index.php/dsq/article/view/4622/3945

David H Rose, Wendy S Harbour, Catherine Sam Johnston, Samantha G Daley, and Linda Abarbanell. 2006. Universal design for learning in postsecondary education: Reflections on principles and their application. *Journal of postsecondary education and disability* (2006). https://files.eric.ed.gov/fulltext/EJ844630.pdf

Emmanuel Schanzer, Sina Bahram, and Shriram Krishnamurthi. 2019. Accessible AST-Based Programming for Visually-Impaired Programmers. In *SIGCSE*. https://doi.org/10.1145/3287324.3287499