To be honest, I find it surprising that I have to write a teaching statement and am somewhat skeptical that anyone will read it—–if you are, I appreciate the fact that you are actually reading these words.

Teaching Background: Computer Science and Martial Arts

I am a teacher in two capacities. First, I am obviously a professor of computer science and teach in that capacity. Second, I teach at a local martial arts school in Chicago. I am struck by these two fields' very different implicit teaching philosophies. Similar to other technical disciplines I have encountered, within computer science there is a systemic, binary, and perhaps unintentional message communicated to new students: you are either ”good” at CS or you are not. People who are ”good” stay and people who are not, tend to find other majors. In my martial arts school, the opposite is communicated: anyone can become a great martial artist if they simply put in the time, effort, and training.

My teaching philosophy is shaped by these two different disciplines which send such different messages to beginners. Computer science is not diverse. Yet my martial arts school is, in part due to the city and neighborhood I live in, but also because everyone who walks through the school door is told they can be successful with effort—even if they are naturally awkward at first. This sets up a virtuous cycle: new students from all different backgrounds show up, see someone who they identify with (because that person was told they could succeed and stuck with it) and absorb the message that success comes from hard work. In short, in a school where we learn how to hit and choke each other, everyone feels they belong.

Teaching Philosophy

I try to bring this spirit from my martial arts school to my classroom teaching and advising. I tell all my students that being naturally good at computer science will only get you so far: at some point we all hit a problem that we do not intuitively know how to solve, and we can either quit or find a way to break the unsolvable problem into manageable chunks. The beauty and irony of computer science is that we build systems that are so complex that no single person can be an expert from application to physics. And yet, one of the most exciting intellectual contributions of computer science is the idea of modularity: that we can break a problem into subproblems in such a way that we guarantee they will compose to a correct solution. Therefore, I spend a lot of time in the classroom and with my PhD students explaining what I have learned about how to make progress on large problems that initially feel impossible simply by consistently solving small problems. A lot of this work involves dissecting the mistakes I made throughout my career and attempting to explain the processes I have learned to overcome problems that defy my initial intuition or natural abilities.

Statement on Equity, Diversity, and Inclusion[1]

I believe these issues discussed above are fundamentally equity issues. One of my goals is to run an equitable research group where people from different backgrounds all feel a sense of belonging and but are also given the tools they need—based on their individual backgrounds—to succeed.

I pursue such efforts because they are essential to my core mission of advancing computing knowledge. To find the best answers to the hardest problems we must ensure that the broadest possible pool of people are (1) excited to participate, (2) evaluated without historical and societal biases, and (3) feel a sense of belonging and community that supports rigorous work at the cutting edge of the field.

Equity efforts recognize that merit is difficult to quantify and must be evaluated and debated by admissions, hiring, and promotion committees that are composed of humans, and thus subject to human biases. Equity, diversity, and inclusion efforts do not sacrifice merit, they ensure that the most meritorious individuals are recognized despite systemic or societal barriers.

Furthermore, such efforts do not stop at getting people in the door. They also include making sure that people from a wide variety of backgrounds have the support and sense of belonging to achieve their best once they are here.

I have graduated 11 PhD students in my 10 year faculty career, 5 are from backgrounds that the NSF recognizes as historically under-represented in computer science. In addition 5 of my PhD students (4 who graduated and 1 who will graduate this year) came to me after some difficulty with their initial advisor.[2] I also founded and chair our department's Equity, Diversity, and Inclusion committee and I co-authored our department's Broadening Participation in Computing plan.[3]

Advising During Covid

The last three years of world events greatly tested the perspectives and philosophies I stated above. During the last three years I sometimes have felt that I served more as a grief counselor than a research advisor. I know that anyone who reads this statement has experienced some similarities, and I think we should all discuss these experiences more.

Like most people, my students were nervous about how to function during an unknown, worldwide pandemic that threw the economy, politics, the workplace, and the home into disarray. They were anxious about making research progress during quarantine but also so much more. Two members of my group members were hospitalized—–one for months. Several group

[1] Please note that in this section I am rephrasing a statement that I wrote on behalf of the Physical Sciences Division. I originally wrote it for myself and then reformulated it with a committee to be put forth as the division's position on equity, diversity and inclusion. You can find that statement here: https://physicalsciences.uchicago.edu/about/diversity-inclusion/committees/edict/. This version is my preferred version for personal reasons.

[2] Please note there is no intention to assign blame here—the initial relationship just did not work and students were in need of a new advising situation.

[3] The current version of the verified BPC plan can be found here: UChicago Verified BPC Plan.

members lost close family to COVID. One of my students not only lost their close relative but due to family issues had to handle all the arrangements remotely from Chicago.

Then George Floyd was murdered. This was upsetting to the world and hit all my students, from undergraduate to graduate, in different ways. Two of my graduate students of color came to me, distraught, and needed me to listen to what they had experienced in academia. I was and am embarrassed at the treatment they received based solely on their appearance. I wish I could share their stories because there are so few BIPOC students in CS, and the CS community should understand how different their experiences are. However, they are not my stories to share. Suffice to say I felt impotent and could only listen. This did motivate me to found the CS EDI committee, but forming a committee is a small thing in comparison.

Then the US government threatened to cancel student visas for students attending schools offering remote instruction. The anxiety compounded and my foreign students needed me to listen to their stories. I learned in detail how much they gave up to come to the US and create a better situation for themselves and their family. They felt betrayed by the system and again, all I could do was listen and assure them that I would do everything in my power to ensure they finished their PhD, even as I had no idea if there was anything I could actually do.

Now——in our ‘post-COVID’ world(?)——as I prepare my promotion materials I am challenged to integrate these experiences into a statement about my teaching philosophy. It is both extremely simple to state, and very hard to execute: Listen to students; be empathetic and realize that two people can experience the same event completely differently based on their backgrounds. These seem like the bare minimum, yet this is what I could do to keep my group together during an incredibly tumultuous time.

Curriculum Development

This statement is already longer than most teaching statements and I believe I am supposed to discuss curriculum. I teach three types of courses: intro (for undergrads starting CS), advanced (for either later year undergrads or early graduate students), and graduate seminars (for advanced PhD students). Specifically, I have taught: (1) Introduction to Computing Systems, (2) Honors Introduction to Computer Science 2, (3) Parallel Computing, (4) Graduate Computer Architecture, and (5) Topics in Computer Systems (covering things from Power and Energy Aware Computing, to Adaptive Computing, and Self-aware Computing).

As stated earlier, my approach to is to emphasize that success is less about some natural inclination towards a problem (although that is certainly helpful) and more about finding ways to break a large problem into manageable chunks whose solutions compose. I emphasize to them that this **the core of computer science**. In intro classes, I facilitate this by starting with a problem statement and leading discussions, rather than lectures, about how to use the **manageable chunk approach** to solve the problem statement. Over the years I have become skilled at prompting partial solutions to the problem from students. I emphasize that it is a good thing to start with a partial solution, recognize its flaws, and build from there. I repeatedly point out it is a more sustainable approach than attempting a flawless solution immediately.

In my more advanced classes I emphasize this core skill by assigning projects and then conducting in class **design reviews** where we discuss the structure of a solution and think through potential risks before students begin coding. I warn students that they should spend much more time than they realize thinking through the design before they ever start to code. In the graduate classes we also emphasize not just design, but also articulating why the proposed design represents a novel advancement over the state of the art.

Some Cherry-picked Quotes from my Course Evaluations

Unlike my research statement, I cannot back up my claims with citations to my papers. So here are some quotes taken from my course evaluations.[4] I, of course, am going to present only the quotes that make me look like a titan of CS education and avoid the ones that make me look bad by saying things like: ”The way the prof explains things is just so confusing,” ”Hoffmann really cares for his students though his lecture was not quite good,” or ”Hank Hoffman was pretty arrogant.”

So here are some quotes that make me look great. In seriousness, I hope that these show some evidence that the approach described above is yielding results:

- ”Attending lectures is like watching Rocky Balboa teach you computer science. It’s riveting, it’s insightful, and it’s so inspiring. Hoffmann’s an absolute legend. He talks through things in such a down-to-earth manner. He’s a well-meaning and experienced computer scientist who just wants to share all his wisdom with you. So go listen to him. I promise you won’t regret it.”
- ”Hoffmann is fantastic. He somehow managed to make extremely complex topics easy to understand in lectures, and even the somewhat drier topics (like linearization) were presented in an interesting way. His lectures were always engaging, and he’s very funny. He put a huge emphasis on the importance of thorough program design, which was not something I was expecting in the class, but loved learning more about and wound up seriously helping with the assignments.”
- ”A problem I often have with systems classes is I get behind on the implementation work so the lectures seem useless and

[4]Unfortunately, I cannot link them as they can only be viewed by people with UChicago IDs.

disjointed and I'd rather just spend time programming instead of listening. The 'design review' approach dealt with this very well by connecting lectures to projects. I would like to see it in other classes but they may not have projects with as much to talk about."

- "I got a lot out of Hoffmann's teaching style of posing computer science problems for the students to solve, and then showing us how one might come to the solutions, which were the data structures that we had to learn for the class. I really enjoyed this, and it heightened my interest in computer science."
- "Hoffman is very good at instilling in his students the skills necessary to be a good software engineer. The setup of the class wherein we discussed design documents before actually starting to implement our projects was unlike any other class I have taken at UChicago but it helped me grow immensely as a CS major. He also clearly really cares about his students and goes above and beyond to help out as long as you reach out."
- "He always brought a positive attitude to lecture and design reviews. He seemed excited for us to learn about parallelism. His lectures were very useful for understanding the content of the course. His coaching in how to approach the assignments was the most useful. The design reviews were helpful because I often missed important aspects of the project in my own brainstorming."
- "The lecture about code development and Hank's philosophy on both CS education and software development shaped the way I approached the rest of the course."