

# Teaching Statement

Stephen Checkoway

During my time as an undergraduate at the University of Washington and a graduate student at UC San Diego, I've had the privilege of being a teaching assistant for 17 classes—including theory, architecture, and programming languages—and helped advise three masters and junior Ph.D. students. Most of those classes involved teaching sections of between 10 and 40 students and all involved holding office hours for one-on-one interactions with students. That experience shaped my teaching philosophy. My goal as a teacher and an advisor is to engage my students in the learning process. My methods for doing so are two-fold: (1) providing the right level of difficulty while teaching the core material and (2) demonstrating the relevance of the material to computer science more generally. Engaging students is critical to ensuring that the students really learn the material and not merely memorize enough to pass tests without a lasting understanding. One important aspect of advising students is a form of one-on-one instruction which requires a different approach than teaching a class. Rather than teach a fixed curriculum, the goal is to teach the student to be an effective researcher.

Like many others who enjoy teaching, the most rewarding part of instruction for me is the “Ah ha!” moment where the previously confused student suddenly realizes the key insight for understanding a concept or solution to a problem. When done well, computer science education is filled with many such moments and this is vitally important for capturing and keeping a student's interest in the material. Material that is too easy and immediately understood by students quickly becomes boring; material that is too hard is frustrating. However, if the students have to struggle and work hard to solve problems, they simultaneously learn the material better and find it more interesting. By engaging with the material, the moment they have the epiphany—that “Ah ha!” moment where all becomes clear—is rewarding for the student who is now more likely to retain the knowledge.

*What* material to teach is only part of the equation; *how* it is taught is just as important. I have found that engaging with students by relating the skills being taught to real-world applications works well. For example, when teaching about finite automata and regular languages, I cover their practical use in day to day applications such as performing regular expression searches or by relating them to other areas of computer science such as compilers. The use of real-world applications can not only motivate the study of a topic, but can often be used to teach a new skill by relating it to something with which they are already familiar. That is, *transferring* knowledge and skills from a familiar context to the problem at hand.

Since I was a TA for multiple courses several times, I was able to observe how some teaching tactics worked well whereas others did not. In one instance of the latter, the instructor and I noticed that students were having a harder than usual time solving the homework problems. We decided that the sections would be devoted to solving homework problems—not similar problems, but the actual problems. The thinking that was that this would encourage more students to attend section (since they would get homework answers for free) and therefore be exposed to correct methods of solving problems. Not only did section attendance rates not increase, the students were obviously bored. They were not engaged at all since I would just walk them, step by step, through the process of solving their homework. Consequently, the students were not pressured to learn the material and this was reflected in the final exam scores. As an instructor, this was a failure, but one which has informed my philosophy of teaching. In particular, it is vital to structure the course work such that students are naturally encouraged to do the work and thus learn the material. In future courses, I concentrated on teaching the methods the students needed to understand rather than on solutions to specific problems. Students were more engaged and performed better overall.

Fortunately, that experience was an exception rather than the rule. As part of a guest lecture on security for Prof. George Varghese's course *Perspectives in Computer Science*, I asked the students to identify a hole in airport security. To that end, I showed a table listing the three locations (the ticket counter, the security checkpoint, and the gate) where various checks of the boarding pass and ID are made (is the ID

valid? Is the boarding pass valid? Does the name on the boarding pass match the ID?). After several minutes discussion amongst themselves, one student visibly lit up (“Ah ha!”) and I could tell that that he had just realized the solution. In fact, not only did he have a correct solution, he had realized that there was a hole that I had never noticed! This is an example of the previously mentioned technique of using something students are familiar with—in this case, airport security—to teach a new skill: thinking like an adversary in order to discover holes in security.

In addition to being a teaching assistant, I have had the privilege of working with several masters and junior Ph.D. students in an advisory capacity. This goes beyond the informal mentoring that is a normal task of senior students in a research group. A major part of advising students is one-on-one teaching where the goal is not only to give advice about solving particular problems but also to teach the students to be good researchers. Advising the students involved having regular meetings, discussing results, and suggesting new approaches. One-on-one instruction is very different from the classroom instruction and has its own set of rewards and challenges. The close interaction enables focused feedback to the advisee and the results of that feedback are more directly apparent than when teaching a class with 30 students. As an advisor, I can suggest that the advisee follow a particular course of action or pursue a line of investigation in response to an encountered problem. At the next meeting, if the student followed my advice, I can see the results of that advice. More importantly, the next time a similar situation arises, I can see how the student tackles the problem. In this way, I can see how the student grows as a researcher.

Like teaching a class, picking a topic with the right level of difficulty for the student is essential. For a new student, a topic that is too hard is discouraging whereas one that is too easy is unlikely to hold the student’s attention. However, the problem must be interesting to the student since ultimately, the project must be completed, a paper written, and a talk given. Fortunately, this is easier than teaching core material in a class since the student has the final choice in what to study and can choose not to spend time on problems which hold no interest for them.

I look forward to the chance to teach and develop classes at the undergraduate and graduate level. In particular, I am interested in teaching courses in undergraduate and graduate security as well as undergraduate courses in areas such as theory of computation, algorithms, discrete mathematics, and programming languages. Teaching is immensely rewarding and I learn more every time I do it. In the same vein, I look forward to working with great undergraduate and graduate students on interesting research.