Teaching Statement

Jonathan Laurent

January 2021

Inspiring and passionate teachers have had a transformational impact on my life. I am committed to emulating their example and giving back to the community through my own teaching. In this statement, I illustrate my teaching philosophy using examples from my experiences as a teaching assistant at CMU.

Creating Learning Material

When I design new learning material, I aim to leverage the curiosity of my students along with their thirst to build in order to push them to excel. Because I am demanding with them, I must be especially mindful of their time and make sure it is invested as effectively as possible.

For example, while I was a teaching assistant for 15-414 (*Bug Catching*), I designed a new set of lab assignments[1] for teaching students to write formally verified software using the Why3 platform. For such an introductory class[2], it is unusual to have students verify anything more complex than a sorting algorithm. As a challenge, I decided I would have them write a fully verified SAT solver with Boolean Constraint Propagation (BCP). This drew some fair skepticism initially.

I designed all four assignments as a progression towards this goal. To make students as productive as possible without having them constantly spend time browsing documentation, I wrote detailed handouts with many examples. I gave live coding sessions in class to provide additional methodological advice. Finally, I iterated through several coding templates, with a focus on nudging students away from the idiosyncrasies of Why3 while giving them complete freedom on core verification challenges. These assignments were particularly appreciated by students and I was thrilled to see them so motivated as they welcomed the challenge and outdid it.

Teaching Recitations

I see recitations as discussions rather than lectures, whose course is not predetermined but shaped by student participation. In my experience, this not only enables keeping students maximally engaged but more importantly, this makes recitations an ideal forum for meta-learning.

The task of understanding a concept, modelling a situation or solving a technical problem is fundamentally a search process. My primary goal in recitation is to emphasize this process rather

[1] These assignments were designed in 2017 and are still in use today. They can be found at the following address: `https://www.cs.cmu.edu/~15414/s20/assignments.html`.

[2] The class was worth 8 credits and it was mostly taken by people with little to no background in formal verification or programming language theory.

than taking the shortest path to an answer. To do so, it is essential to involve students at every step and engage with their ideas. When a student proposes an approach to a problem, I will make my best to run with it until everyone can understand why it works or where it breaks. I am especially encouraging of students making mistakes as these present valuable pedagogical opportunities in terms of learning to perform sanity checks and looking for counterexamples. A more subtle benefit of letting students drive me off-road is that it forces me into a form of honesty and spontaneity that is otherwise hard to fake. Indeed, when I am looking for a fresh example or following a student's lead, there is no way for me to cheat and I have no choice but to demonstrate the problem-solving skills I am trying to transmit.

For example, as a teaching assistant for 15-424 (*Logical Foundations of Cyberphysical Systems*), my most popular recitations were recitations in which I worked with students on larger, open-ended problems that admit a variety of approaches and foster creativity. In particular, to illustrate the concept of a differential game, I imagined a scenario where a thief had to go through a guarded corridor without being seen. The goal was for us to go through the whole process of modelling the situation using logic, designing a controller for the robot and proving it formally correct. The students collectively managed to find a much better and simpler controller than the one I had in mind[3], which was wonderful to observe.

One-on-one Interactions with Students

One of my favorite aspect of teaching – and for which I often get the most positive feedback – lies in the one-on-one interactions that happen with students during Office Hours or on online forums. In my experience, these interactions can be very powerful in terms of helping students improve their problem-solving skills, deepen their understanding of the class material and gain confidence.

The best way to answer a question is often with another question. When a student is stuck on a problem, I like to offer a smaller problem that best captures the idea they are missing or shines the most light on a faulty reasoning pattern. For this reason, I am usually helping several students in parallel during my Office Hours, giving each of them frequent time to think on their own. This typically translates into me circling between breakout rooms on Zoom, or into a cramped office where students sit everywhere on the floor, a computer or a sketchpad on their knees.[4]

Finally, it frequently amazes me how *little* intervention is often necessary for a student to get unstuck. For example, while I was a TA for 15-424, a student submitted an almost blank proof for a computer-assisted verification assignment. I invited this student to come to an Office Hour[5] so that we could work on their proof together. Then, this student wrote the full proof before my eyes, without a hint and with me mostly acting as a human version of a *rubber duck debugger*[6]. It turns out that all this student needed were a few encouragements to augment their confidence, along with frequent reminders never to give up on an attempt before they could explain clearly the reason why it did not work.

[3]My controller involved some nested loops, which made it hard to verify. Interestingly, it took the concerted effort of many students to come up with the better solution.

[4]When I was a TA for 15-414, the success of this model led me to book larger rooms to host optional *work sessions*.

[5]Some students can be shy to ask for help, especially when they need it most, so it pays off to be proactive.

[6]See `https://en.wikipedia.org/wiki/Rubber_duck_debugging`.