TEACHING STATEMENT
Kristopher Micinski (micinski@cs.umd.edu)

Passion for teaching motivates every aspect of my scholarship. My parents were both teachers. From an early age they emphasized the importance of understanding material at a level that allowed me to explain it to others. This desire to get to the bottom of things pervades everything I do from writing research papers and blog articles to explaining concepts in office hours. I have taught both formally and informally throughout my time in grad school. I served as a teaching assistant twice, gave a series of one-off lectures at a variety of levels, and taught a course as the instructor on record. I mentored several undergraduate research projects. I looked for any opportunity I could to get experience working directly with students. To practice explaining things in the best possible way, I wrote blog articles, tutorials, and interactive apps throughout grad school.

Teaching Assistantships During my first year I was a TA for two courses at UMD. The first was the undergraduate programming languages course. I taught discussion sections and led office hours. Interacting with students in office hours, I discovered a wide variation in learning styles. Some learned best with pictures, or being visually guided through explanations. Others learned best by seeing some example code. I immediately discovered a passion for identifying different learning styles and reflecting on how to structure concepts to meet student needs. During this class, I frequently found myself wandering into student study rooms to get more face time with students and work with them directly. To further practice my pedagogy, I led a series of review sessions before exams that gave a frank and honest discussion of how to solve sample questions. These sessions were attended by much of the class.

Also, I served as my advisor’s TA on the first iteration of a revamped compilers course, CMSC 430. Here I got experience refining material and thinking hard about course assignments. I enjoyed pretending to be a student and anticipating stumbling blocks they might encounter. I identified parts that could confuse students and eliminated them from homeworks and projects. Because the course was mostly seniors, this also gave me experience working with advanced students. I realized that for these students, less hand-holding was better. Rather than explaining concepts to them in a rote way, I would identify what key aspects were impeding their understanding. I would clear their confusion and then have them explain the material back to me, so they would discover for themselves what they had been missing.

One-Off Lectures After my time as a TA, I yearned to do more course development. I was vocal with my colleagues about my interest in this and I gave lectures in a variety of undergraduate and graduate level courses. I used these classes to experiment with a mix of teaching styles including live-coding, whiteboard lectures, and using slides. I discovered that a key challenge was to keep students plugged-in throughout class. Teaching these lectures helped me understand how I could mix different presentation devices to keep students engaged. For example, when I used slides I was careful to mix in demonstrations or questions. I also realized that good presentation frequently demanded paying attention to the student tone throughout lecture and modulating presentation based on that feedback. For example, I was careful to note where I could skip coverage of certain concepts or switch over to live-coding and integrate that into lecture to help demonstrate things in a concrete way.

Teaching Undergraduate Programming Languages During the summer of 2015 I served as the instructor for the undergraduate programming languages class for which I had previously been a TA. My course had roughly 40 students. The course content covered programming in a variety of languages (Ruby, OCaml, and Prolog) and also foundational concepts such as programming language semantics. However—inspired by my prior experience working directly with students—I changed the course format to use an active learning style. Instead of using the slides given to me for the course, I changed each lecture to build up material via live-coding and group work. I began most classes by posing a question we could not yet solve. This helped motivate why we needed the technical devices we would learn about during class. Then I gradually introduced new concepts leading students to solve this problem. Each step along the way, I had students work together—frequently relying on interactive tools to check their work. For example, during our coverage of state machines I built a web app to build and run finite automata. At the end of each class we had an artifact that students could use to begin a follow-on project.

As an example, consider teaching regular expressions. I first told students we would be parsing a gradebook file to calculate a class average. I by showing students necessary boilerplate code to manipulate and read files. I find that this allows students to be motivated by real-world applications but avoids their getting stuck on minutiae. I would then ask students to form groups and discuss a high level algorithm for calculating a mean. Engaging students early
with simple questions helped everyone feel confident and motivated throughout class. I would then incrementally guide students toward a solution. For example, I would show components of regular expressions and small examples in an interactive web-based tool. Then I would ask students to cooperatively solve pieces of the problem, building up to the solution.

While I reused some course projects from prior semesters, I also experimented with new projects. For example, to tie together coverage of programming and language semantics, I had students implement an interpreter for the core of the OCaml language. I chose this project because it reiterated topics students learned on multiple levels. First, they were grappling with understanding functional programming. Second, they were having trouble relating the math they learned on the board to how an interpreter for a programming language would work. And last, they were having trouble scaling up their programming to large projects. This project addressed all three. As they implemented the interpreter for our small language, it forced students to reexamine technical concepts they had taken for granted when merely working through them on paper. It also helped them understand the language in which they were programming even better. I believe this strategy of structuring projects to drill down into concepts after teaching them at a high level helps students gradually internalize their understanding of material.

Reflection and Growth  
I strive for transparency with my students. I let them know that I value their input and am willing to change my teaching style or coverage of course material to help them learn. When I cover something poorly, I let students know so they don’t internalize misunderstandings as their fault. To do this, I set up anonymous feedback and let students know they are free to give constructive criticism without damage to their grade. I am honest when I discover mistakes in grading or assignments and work with students to ensure fairness. Students told me this allowed them to identify and correct problems early rather than letting them fester.

After finishing my course I reflected upon what methods worked and which needed modification. Guiding classes by examples and using active learning methods helped focus students and encouraged participation. However, I needed to do more to help place examples in the broader context of the course goals, and carefully relate my each class to a holistic narrative. When I began my course I focused on the first set of modules, rather than thinking about how each incremental topic built upon the last. When I structure courses in the future, I plan to focus more on how each topic fits together before I develop material in each section.

Mentorship and Advising  
Outside of courses, I am also motivated by educating students as part of my research program. I have done so with several undergraduates and high school students at Maryland. Crucially, my experience with these talented students has taught me how to design research projects that offer a mutually rewarding experience. For example, designing research projects for undergraduates balances different goals than for grad students. I focus on starting with something that will bring the student confidence and teach them a practical skill. This is frequently learning a new language or framework, and even this experiences offers the ability to mentor students in a hands-on way. After they learn some concrete skill, I have them use that skill to implement a clear extension to one of my research directions. Last, I gradually teach technical concepts on an as-needed basis, grounding the challenges in terms the students understand.

Course Development and Logistics  
I love taking new concepts and thinking about how to present them in the best way. Because of this, I enjoy teaching a breadth of topics. My research experience offers a strategic advantage in this respect: I have done work in both theoretical areas such as programming languages, but also build large systems to implement my ideas. In developing content, I strive for a connection between theoretical constructs and implementations. I aim for students to be able to take the ideas they learn about, internalize them, and then create something new they are passionate about.

http://cs.umd.edu/~micinski/application