

I find teaching to be tremendously rewarding, and it is one of the major reasons why I enjoy academia. My experience is a bit unusual—when I joined Caltech in 2000, there were 5 Computer Science faculty and no undergraduate major. I designed and taught new courses, developed an undergraduate major, wrote a textbook, and had a lot of fun. Here are some details of my goals and accomplishments.

CS134: Operating Systems and Compilers The first course I designed and taught was this upper undergraduate course covering operating systems and compilers. The first quarter is a fairly standard OS class, using Silberschatz, Galvin, and Gagne as a textbook. In the lab, student add features to the Linux kernel (using VMware or User-mode Linux), starting with simple concepts like process synchronization and finishing with the implementation of a distributed shared memory system. During the second quarter, students construct a mini-Java to Intel x86 compiler. The textbook is Appel’s “Constructing a Compiler in ML,” but unlike Appel we use an A-normal form intermediate representation. Implementing an entire compiler in a quarter is quite a challenge. The third quarter focuses on student-chosen projects.

CS136: Languages, Semantics, and Type Systems This is the second course I designed and taught, focusing on the foundations of languages. The first quarter studies operational semantics, lambda calculus, runtime architectures, virtual machines, *etc.*, using Mitchell, “Concepts in Programming Languages” as a textbook. This is still a lab class, and students build interpreters, compilers, and a virtual machine. The second quarter focuses on type systems, using Pierce, “Types and Programming Languages” as a textbook, though I focus more on constructive logics than Pierce does. The third quarter is student-chosen projects.

CS24: Computer Architecture I co-designed and co-taught this sophomore-level course with André DeHon when we revamped the sophomore curriculum for the CS undergraduate major.

Stylistically, I like to introduce current research, ideas, and perspective into my classes. The connection with the textbook may be loose at times. My courses tend to be popular, and enrollments tend to be high. I am also a strong supporter of undergraduate research, and I’ll often recruit students to work in my research lab.

CS undergraduate major I was one of the chief architects behind the design and establishment of the Caltech undergraduate major in Computer Science; I worked together with André DeHon. Before the major, computer science was classified as a kind of “independent study,” and students would obtain a generic degree in Engineering & Applied Science. The development of the major required that we construct a new curriculum, but perhaps the most important role that André and I played was in convincing the faculty that an undergraduate major was not only possible, but also in the department’s interest. This required several years of effort, but eventually we succeeded, and Caltech now has a CS major with a small set of required courses and a great deal of flexibility. Student feedback has been quite positive.

OCaml Based on my work in CS134 and CS136, I wrote a textbook on the Objective Caml programming language. A draft I have placed on the web has received

considerable attention, and is used in numerous college courses. The course topics are primarily in the field of theory of programming languages, but range as far as computational linguistics. The final text is to be published by Cambridge University Press in summer 2008.

If I were to summarize, I would say that teaching can be both wonderful and difficult. My approach has been to bring parts of my own research into the classroom, and to confront students with difficult questions. My students have been generally stellar, and it is tremendously rewarding to watch them grow. They bring new ideas and excitement that I then mix back into the workings of my own lab.