David Van Horn
Teaching statement

Writing programs is the most precise form of thinking, and as such, everybody should—and everybody can—be taught to program as part of a college education. My teaching approach is informed by and supports this claim.

I am well-equipped to teach programming, software engineering, compilers, theory of computation, programming languages, algorithms, and theorem proving at the undergraduate or graduate level.

Inside the Classroom

For the past several years at Northeastern, I have taught programming from first principles, making no assumptions on students' background beyond basic high school level training in algebra. I focus on (1) the *design recipe*, originally developed as part of the *How to Design Programs* curriculum [1], which is an intellectual tool to help go from a blank page to a fully developed solution in a systematic and step-by-step way; (2) the *design-by-contract* methodology, originally developed by Bertrand Meyer, which focuses on a specification driven program development, and (3) *pair programming* and *public code review*, which instills social skills needed to articulate and evolve complex designs in groups.

In the first semester, I teach structured, functional programming where one of the most important lessons is that the structure of a program follows the structure of the data it operates on. Using the simple and uniform means of abstraction and combination of functional programming and basic algebraic reasoning, students develop critical thinking and general problem solving skills. They learn how to structure their ideas and articulate complex concepts—to themselves and to machines. In the second semester, I teach how to design programs in an object-oriented style, building on all of the conceptual basis of the first semester. Both of these course form the basis of the core first-year curriculum for first-year computer science students.

I believe the only way to learn the fundamentals of computation is to *write programs* and I use interactive, distributed, multi-player games to engage students (see [2] for details of the approach). By the end of the first semester, my students have strong analytic problem solving abilities, they can program web servers, chat clients, and sophisticated games—complete with AI players—like Tetris and Risk, and they are prepared for their follow-up course in theorem proving using ACL2, which builds on all of the reasoning skills taught in the first semester.

By the end of their second semester, they are prepared for and consistently receive high reviews in their industrial co-ops positions. (About 98% of students at Northeastern's College of Computer Science participate in the university co-op program that places students in semester long industrial jobs beginning as early as their first summer.)

Outside the Classroom

Beyond the classroom, I have led a group of eight undergraduates, with Matthias Felleisen, in the writing an illustrated book on programming interactive, distributed video games called

Realm of Racket.[1] The students were selected from my class after their first semester. The two year long project involved the students in the complete book production life-cycle; students wrote, illustrated, typeset, edited, marketed, and programmed the book. It will be published by No Starch Press in early 2013.

I have significant experience teaching and managing a large teaching staff. I proposed, designed, and taught an "honors" track of our introductory first year programming curriculum. I have served on a faculty committee tasked with redesigning the undergraduate cirriculum on both algorithms and object-oriented design. I've trained dozens of graduate TAs and undergraduate tutors. I have co-advised and published research papers with multiple PhD students. I have cultivated a relationship with the organizers of "Bootstrap," a program that works with schools, districts, and tech-educational programs across the country to teach students age 12-16 to program their own videogames using purely algebraic and geometric concepts.[2] Many of my undergraduate tutors have gone on to work with Bootstrap, teaching in Boston-area public middle schools serving underprivileged communities. I have also recruited several undergraduates to work on collaborative research projects in their second through senior year.

Graduate and Undergraduate Advising

Currently, I am advising 5 graduate students (4 PhD, 1 MS) and 2 undergraduate students. Phillip Mates (co-advised with Amal Ahmed) is working on counter-example generation for contract verification. Alex Marquez (co-advised with Olin Shivers) is working on security analysis of Dalvik bytecode, which we plan to submit to SAS'13. J. Ian Johnson is working on optimizing program analyzers, which we submitted to PLDI'13, and how to make analysis highly memory efficient, which we plan to submit to ICFP'13. Stephen Chang (co-advised with Matthias Felleisen) is working on tools for mixing lazy and strict programming language features. Our initial work was published in TFP'10, for which Chang received the best student paper award. Phil Nguyen (co-advised with Sam Tobin-Hochstadt) is a MS student on track to be a PhD student next fall who is working on contract verification. Adam Alix and Nicholas Labich are senior undergraduates currently working on analysis of JavaScript. They will both attend the 2013 POPL in Rome with scholarships from the SIGPLAN Programming Languages Mentoring Workshop,[3] and subject to acceptance, will present their research at the Student Short Talk Session.[4]

References

[1] Matthias Felleisen, Robert B. Findler, Matthew Flatt, and Shriram Krishnamurthi. *How to design programs: an introduction to programming and computing*. MIT Press, 2001.

[2] Matthias Felleisen, Robert B. Findler, Matthew Flatt, and Shriram Krishnamurthi. A functional I/O system or, fun for freshman kids. In *ICFP '09 Proceedings of the 14th ACM SIGPLAN International Conference on Functional programming*, pages 47–58. ACM, 2009.

[1] `http://www.realmofracket.com/`

[2] `http://www.bootstrapworld.org/`

[3] `http://www.doc.ic.ac.uk/~gds/PLMW/grants.html`

[4] `http://wrigstad.com/popl13/`