JFP **32**, e3, 13 pages, 2022.

doi:10.1017/S0956796821000289

Editorial
On being a PhD student of Robert Harper

Abstract

The Robert Harper Festschrift includes articles by three of Bob's students and colleagues—Karl Crary, Andrzej Filinski, and Jonathan Sterling. Each of these articles touches on themes that are central to Bob's research: module system design, proof-directed program development, and (to use Bob's term) "computational trinitarianism".

In this foreword to the Festschrift, we have additionally compiled reminiscences of Bob Harper from his PhD students. We invited them to reflect on their experiences working with and learning from Bob. We believe these reminiscences, presented in chronological order of dissertation date, deliver a most fitting tribute to Bob in honor of his 64th birthday.

1. Benjamin C. Pierce, University of Pennsylvania (Thesis: Programming with Intersection Types and Bounded Polymorphism, December 1991, co-advised by John Reynolds)

Bob showed up at CMU white hot from a PhD with Bob Constable at Cornell and a long postdoc at Edinburgh with Robin Milner, spewing ideas in every direction like a one-man ticker-tape parade. I was his first student.

The quality I remember best about Bob in that period—well, I guess it hasn't changed much since—was his intensity about pretty much everything. There was the time that he and Peter Lee discovered the first version of SimCity and basically disappeared for a week. And the time he decided he needed to understand—in 100% complete detail—the path taken by a keypress on his office computer's keyboard to transform it into an arrangement of pixels on his screen; he spent the better part of another week tracking it down by reading through the source code of the operating system.

Besides Bob, I was co-advised by another towering figure in the programming languages world, John Reynolds. Looking back, I am astonished both at my incredible good fortune and at the way two such different people managed to convey so many of the same lessons. John was very much a loner, intellectually: he had his own notations, his own terminology, and his own ways of thinking about things; other people had to enter his world not just to benefit from his insights but even to make him understand what they were talking about. Bob was and is very much a collaborator, working intensely (natch) with students and colleagues to establish and deepen common understanding and figure out how best to communicate it to others. In their different ways, though, both taught me innumerable lessons about clear thinking, clear speaking, clear writing, and the drive for excellence.

I used to say to my friends, "The great thing about having two advisors was that we can all play 'good cop / bad cop' together... only I can never figure out who is bad cop and who is good cop." :-)

Seriously, though: Both were good cop!

2. Greg Morrisett, Cornell Tech
(Thesis: Compiling with Types, December 1995, co-advised by Jeannette Wing)

Bob Harper took me under his wing after I gave a disastrous talk as a graduate student in the Principles of Programming (PoP) group seminar. I am eternally grateful.

I had been working on the Fox project, which was an ambitious undertaking to implement an operating system on top of a high-level language—Standard ML, of course. At this point, we were just implementing the TCP stack and what I hated about the code was that it was lower level than the C code it was meant to replace. This was because to represent a packet, all we had were byte arrays, when in C, you could use a record (struct) to directly represent the packet and access the fields. You couldn't directly use SML records to do the same thing because SML implementations, such as SML/NJ, represented the fields uniformly (as a machine word), when we needed fields that had different sizes, from bits to bytes to words and even nested structures.

In my talk, I proposed adding new kinds of types to SML called "flat types". The idea was that a flat record would look like a C-struct: fields would take up only the space needed to represent them and nested records would not be represented by a pointer, but rather flattened into the rest of the record. The problem with flat types is that they didn't play well with polymorphism or abstract types. How much space should we allocate for an abstract type? What alignment constraints should it take on? I proposed some crazy scheme for solving this (the details of which have been conveniently forgotten) and I tried to give a talk on this. John Reynolds was present at the talk and did nothing but scowl and eventually exploded, basically telling me that I was an idiot, that he didn't understand what I was trying to do, and that what I proposed would never work.

Apparently, after my horrible talk, John went up to Bob Harper and told him that he needed to "fix" me. To his credit, Bob invited me up to his office (which was incredibly hard to find, hidden somewhere in Doherty Hall) and listened patiently while I tried to explain my ideas. Bob came up with a wonderfully elegant solution to the problem I was facing based on what we later termed "intensional polymorphism". The basic idea was to re-think language implementations that supported type abstraction and polymorphism, and instead of erasing the types at run-time, you would instead pass around representations and use them to support type-specific representations and optimizations. The trick was doing this in a principled way, one that retained the ability to type-check the intermediate code where you were manipulating the types as values.

I can't take much if any credit for these ideas. They were all Bob's. My modest contribution was that I immediately saw a bunch of applications, from flat records and arrays, to unboxed arguments and results for functions, to tag-free garbage collection. I fondly recall going up to Bob's office over and over again, working out the details for all of these applications. I would write things on the white board, Bob would critique, erase, and then show me the right way to formulate things. Those one-on-one sessions were some of the most

enlightening (and frustrating) times in my life. I could tell that I was often interrupting Bob from doing something else, but he always let me come in and would spend hours working, ever so patiently, with me.

Towards the end of my graduate career, I had figured something out and was excited to show Bob. I went up to his office but couldn't find him. In fact, he disappeared for a whole week. (I later found out he was trying to solve all of the puzzles in the "Myst" CD-ROM game that was all the rage back then.) When I finally tracked him down, I started writing on the white board and got about half-way through explaining whatever it was that so excited me. Bob kept interrupting me with questions. I got so frustrated that I finally just threw the white board marker at him and yelled at him to "Shut up!" At that point, he leaned back in his chair, smiled, and said, "Now you're ready to graduate."

3. Christopher A. Stone, Harvey Mudd College
(Thesis: Singleton Kinds and Singleton Types, August 2000)

I enrolled in graduate school at CMU planning to study Programming Languages with Peter Lee and to focus on the implementation side. Shortly before the first semester started, Peter suggested I might be interested in an elective (15-814: Type Theory for Programming Languages) being offered by his Fox Project collaborator, Bob Harper. I knew nothing about type theory—or Bob—but the class sounded interesting so I signed up. That turned out to be one of my favorite courses ever; I was blown away by the clarity and elegance of Bob's presentation of small-step semantics, preservation and progress theorems, logical relations, and more, and the way small extensions could handle recursion, exceptions, polymorphism, continuations, and other language features. Bob's course notes evolved significantly thereafter, eventually resulting in the book *Practical Foundations for Programming Languages*, but occasionally I still find myself consulting the original handout from that first course.

Although Peter remained my advisor for several years, it became increasingly clear that I had been seduced by type theory; Bob eventually took over the advisor role. Bob is a man of very strong opinions, so he and I had a fair number of heated (technical) arguments during the years I was working on my dissertation. To my great irritation, it almost always turned out that Bob was right. But even at his most skeptical, he still took the time to listen and give fair consideration to my ideas, and to agree with the parts that actually did make sense. While some memories of my time at CMU have begun to fade over the decades, my appreciation for Bob's support and for everything I learned from him remains strong.

4. Perry Cheng, Nimbella Inc.
(Thesis: Scalable Real-Time Parallel Garbage Collection for Symmetric Multiprocessors, September 2001, co-advised by Guy Blelloch)

I cannot recall how I first met Bob, but my lasting impression of him is that of an amiable and supportive mentor who is as enthusiastic about doing his own research as imparting the why of that work. When the elaborate and indirect reasoning in PL proofs would start to overwhelm, he would motivate me by revisiting the goalposts. This vital lesson crystallized

for me the point of dissertation work and future serious endeavors. Although my thesis, co-advised by Guy Blelloch, is in a somewhat different area from his work, the goals and methods are consonant with his approach of balancing soundness with real-world concerns.

If I had to come up with just a few nouns I associate with Bob, they would be types, progressive politics, and cycling. The latter did not interest me but the former two remain memorable and I can still hear the debates on the pragmatics of strongly-typed vs. . . . in daily programming. He was funny, merciless, and often right on the nose by remaining on point. He taught me intellectual self-confidence by example, an enduringly useful skill in both research and commerce. Sadly, I never did pick up his knack and love for writing.

Bob, thanks for getting me through!

5. Umut A. Acar, Carnegie Mellon University
(Thesis: Self-Adjusting Computation, May 2005, co-advised by Guy Blelloch)

I started my PhD with Guy Blelloch. I was familiar with Guy's work and came to Carnegie Mellon to work with him. After we started working together, Guy mentioned a few potential projects. Among them was the idea of using persistence and functional programming to derive dynamic algorithms automatically. It was a high-risk project, with uncertain outcomes. Perfect fit, I thought, and got to work. After about a year, the basic ideas started to take shape but it was all quite complicated. So, Guy suggested that we collaborate with Bob, who could help us reduce the matter to its essence.

So we did. It is hard for me to hide the grin on my face when I think of those days. I would wake up early, work for a few hours at home, go to campus, socialize with other graduate students, stop to talk with Bob and Guy, do some more work, and head off to rock climbing later in the afternoon. The three of us would meet regularly to review progress. During our meetings, Bob and Guy would often express opposing opinions of whatever I proposed. This would typically lead to a lively, forthright debate. I relished these debates. Sometimes, things would heat up, and we would all find ourselves yelling at each other, presumably to make our points heard, or perhaps just for fun. I have learned in these debates that Bob likes to be right. This can be a challenge, especially when he is not right. I once invited him to settle an hours-long disagreement by arm wrestling. I digress. After some more work, I was able to reduce the big beast of a system that we started with to a relatively simple one revolving around the notion of a "modifiable". I recall the meeting when I first presented this idea. There was a sense of disbelief that this could all be so "simple". It was probably the shortest, most eventless meeting that we have had—how exciting.

Those days are long gone, but they have produced knowledge that will long be remembered. They have also seeded a fruitful collaboration between Bob, Guy, and me, and our students, that have sprung off new research directions, many revolving around the idea of unifying Church and Turing's models of computation (a.k.a., "Theory A" and "Theory B"). I hope that this idea will endure into the distant future and even perhaps reach other galaxies ("thank you Amazon shoppers").

There are a scant few that have the courage to seek the truth. I view Bob Harper as one such rare individual. One may agree or disagree with him on any one specific topic, but how could one not agree with the intent?

6. Derek Dreyer, Max Planck Institute for Software Systems
(Thesis: Understanding and Evolving the ML Module System, May 2005, co-advised by Karl Crary)

I don't remember what the context was exactly, but one day in grad school I revealed to Bob that I didn't know anything about logical relations. In typical Bob fashion, he was very surprised and gave me a hard time for not having already educated myself about this fundamental thing that he thought every student should know. To rectify this lapse in my education, he spent the afternoon giving me a private lesson in his office about logical relations, starting with Tait's method for proving strong normalization of the simply-typed λ-calculus, and then Girard's method for proving the same of System F. If I recall correctly, he described these as "book proofs", a reference to Paul Erdös's idea of proofs so beautiful that they belonged in God's book of perfect mathematical proofs.

Apparently, Bob's impromptu lecture made a big impression on me, because over 20 years later, I'm still grinding out a living trying to come up with my own book proofs based on logical relations.

More generally, Bob instilled in me a deeply ingrained sense of the importance of modularity. My thesis work, for starters, was on the ML module system, a topic near and dear to Bob's heart. But even when I moved on to other problems, including some that were perhaps less directly up Bob's alley (e.g. weak memory models, verification of low-level concurrent programs), modularity has remained for me the central concern. How can we reason about concurrency and higher-order state *modularly*? How can we verify compilers *modularly*? How can we factor the soundness proof of a concurrent separation logic *modularly*? How can we establish safety of a language like Rust *modularly*? I am indebted to Bob for planting the seed of modularity in my mind and giving me the confidence that these were the right questions to ask, even if (or maybe precisely because) they often seem so devilishly hard to answer.

This brings me to my last, and perhaps weirdest, thought. In the past decade, particularly since his focus turned to computational higher type theory, Bob has embraced the motto, "Dare to be irrelevant." I think that, while catchy and provocative, this slogan is but one instance of a broader lesson that I took away from Bob's research and mentorship, namely: "Dare to think for yourself. Question the conventional wisdom. If everyone else is going one direction, go the other way. *It's OK to be weird.*" Bob is the living embodiment of these principles: his research interests span wildly disparate topics (type-directed compilation, low-level memory management, avant-garde type theory, module system design, parallel algorithms, mechanized proof, and more), he pursues his vision regardless of prevailing trends in the field, and he approaches everything he does from an original and often idiosyncratic perspective. Bob's work is weird, and gloriously so!

Arguably the apotheosis of this weirdness was the 2006 ICFP Programming Contest, which was organized by a team of Bob, Karl, and their students (not including me—I had graduated the year before). The contest, titled *Cult of the Bound Variable*, was completely unlike any other programming contest I've seen before or since. It comprised a number of problems that were all elaborate jokes about terribly impoverished or cumbersome programming languages, such as the Qvickbasic language (where line numbers were written in Roman numerals) or the 2D language (where circuits had to be written out laboriously

in ASCII graphics). These were all tied together by a ridiculous framing story about "computational archaeolinguistics": a thinly veiled reference to Bob's perennial complaint that so much PL research focuses on remedying the flaws of poorly designed mainstream languages, treating them as archaeological artifacts, rather than building better languages on solid formal foundations.

I'm sure that to a large extent the 2006 ICFP Programming Contest was the brainchild of Bob's amazing students, especially Tom Murphy VII, whose one-of-a-kind Dadaist sensibility was on full display. (See Tom's reminiscence below for his own perspective on the contest.) Nevertheless, it is a testament to the offbeat culture of Bob's research group that something this weird and wonderful could emerge from it: a thing of surreal beauty whose irreverence—and irrelevance!—I find inspirational.

I like to think that with my borderline obsessive focus on modularity and logical relations, inherited as it clearly was from Bob, I am carrying on the CMU tradition of weird and independent thinking. I will try to keep it weird, Bob, and I know you will, too!

7. Leaf Petersen, Google
(Thesis: Certifying Compilation for Standard ML in a Type Analysis Framework, May 2005, co-advised by Karl Crary)

Some oddly vivid and specific memories of Bob. Walking with him to the Oakland O concession in the student center for a hot dog after a long session in his office. Eating out at a Chinese restaurant to celebrate a Fox project milestone. His enthusiasm for a homework assignment I proposed as his TA based on Strassen's algorithm. Visiting him during renovation work at his house.

I arrived at CMU in 1996 with a fairly ad hoc understanding of programming languages picked up "on the job" as it were, and graduated in 2005 with a broader and deeper understanding, somewhat less ad hoc perhaps, but also largely picked up "on the job". Much of what I learned there was still just shared knowledge by a small community, only starting to be more broadly written down and disseminated. The early lecture notes that I believe eventually evolved into "Practical Foundations for Programming Languages" were a startling revelation to me in their clarity and concision. Deep concepts made simple, and almost obvious (almost!). I still have my copy of the notes: "Type Systems for Programming Languages (DRAFT)", copyright 1996. The text and notation is re-assuring and familiar after many years.

It is certainly the case that I learned a great deal about programming languages during my time as one of Bob's Sorcerer's Apprentices—much that was interesting, true, and useful. Far more valuable though, I think, was a rigorous education in the value of a kind of clarity of thought, and co-equally of communication. Bob was never satisfied with something that just worked. It had to *obviously* work, and it had to explain by its very expression *how* it worked. The occasional marathon sessions in Bob's office working through a problem were as much about arriving at an explanation for a solution as at the solution itself. This appreciation for thinking and communicating clearly was almost certainly by far the most valuable thing I took away from my time with Bob, and something that I find myself valuing, respecting, and aspiring to more and more every year.

I'm very grateful to Bob for all that I learned from him, and I'm very pleased to congratulate him on all that he has accomplished and the tremendous influence he has had on those of us represented here, as well as on the broader Computer Science community.

8. Ashish Agarwal, Solvuu Inc.
(Thesis: Logical Modeling Frameworks for the Optimization of Discrete-Continuous Systems, May 2006, co-advised by Ignacio Grossmann)

I was a PhD student in the Department of Chemical Engineering at Carnegie Mellon, which I had chosen for its well known computational focus. However, my interest in Computer Science was too deep to be satiated there. By the second year of my PhD, I was emailing faculty in the CS department. One of them was Bob. I asked him several questions, which all clearly communicated that I knew nothing about his whole field. Nonetheless, he replied immediately saying "you could just stop by my office". So that's the first thing I'd like to say about Bob. He is exceedingly generous.

Bob referred me to reading material like Frank Pfenning's course notes and Milner's Pi Calculus. I would ask him questions about my research, which regarded automating algebraic transformations in the field of linear programming. Bob had no experience in this area, but he answered every question I asked so thoroughly that each meeting kept me busy for a couple of months. This went on for a year, at which time Bob suggested we meet weekly. And that was how I became his PhD student.

Until I met Bob, I thought I just needed help with my C++ code. Now Bob had introduced me to a new world, and I knew there was a theory behind my efforts. For the next two years, I had intense meetings with Bob. He personally taught me more mathematics than I had ever learned before. He would solve every problem I posed on the spot and not stop until the solution was complete (one particularly difficult problem led to a 10 hour session). He filled up the many whiteboards in his office multiple times, and I would frantically take notes. Somehow I absorbed most of what he was explaining, which I credit to his exceptional communication skills.

Bob was also demanding. I sent him a version of my dissertation that I thought was quite polished. He called me in the evening, rather unhappy. I headed directly to his office and he spent several hours tearing it apart. In the end, I asked "so it appears that you are happy with all of my technical work". He replied that yes he was. The problem was all in the introduction. It took me several months to rewrite the introduction because I had to first familiarize myself with a century of mathematics. Only then could I truly understand where my dissertation stood in the universe of all knowledge. I feel privileged. His high standards elevated me.

Thinking about my next step, I realized I miss science. I had learned how to build complex software systems, and I wanted to use that knowledge to help scientists. I decided to pursue a postdoc in the field of Bioinformatics, which is the field I've been working in ever since. Bob's advice was that it takes 10 years to gain expertise in a new field, and that I should be ready for that. I think it depends. If you're lucky enough to have an advisor like Bob, that time frame can be drastically reduced. Bob continues to support me, e.g. writing reference letters and serving as an advisor to my startup. To Bob, thank you for accepting me as your student. Your influence is present in the work I do every day.

9. Tom Murphy VII, Google
(Thesis: Modal Types for Mobile Code, May 2008, co-advised by Karl Crary)

I first encountered Bob Harper as an undergraduate in about 1998. I was a cowboy C programmer and probably pretty annoying. The CS fundamentals course 15-212 was taught in ML (Bob's doing, I later understood) and it was the first computer-related thing I ever found difficult, which was puzzling to me. I didn't like it. I wanted to get back to whatever I thought programming was.

Bob didn't teach this course, but he gave a guest lecture at the very end, which dazzled me. This brilliant fellow was animated with passion for what had seemed to me a dry, uptight subject. He had the most comprehensive vision for the discipline of computer programming I had ever seen. It was almost holy for him. And he *made sense*. This was not just a boring version of programming where rules prevented you from having fun and being creative. This was a *better* version of programming that I didn't know about yet.

Bob inspired me with that lecture to try his programming languages course. He is of course a fantastic teacher as well, and this (with help from the rest of the incredible POP Group) led to me staying at CMU for graduate school, which had not been the plan at all.

I believe Bob is the smartest person I've met, although to be polite I should exclude anyone currently reading this (can't exclude Bob, though, or the statement is ill-formed). As an advisor his high standards can be hard to meet. He does not pull punches, but for the same reason his approval or praise is truly meaningful. I definitely benefited from him giving me a hard time, and count some of the successes among my proudest moments.

Despite the intellectual toughness, Bob was not a taskmaster, and he gave us plenty of latitude to explore. I was privileged to spend a full semester working on the design of the 2006 ICFP Programming Contest, which was dangerously ambitious and probably hard to characterize as "part of the PhD program." But it was also very formative for me and others who worked on it, and a treasured accomplishment. I'm sure part of this was a calculated risk for our benefit, but it was also because Bob believes in doing good work and having a good time doing it. I vividly remember the late night "launch" of the contest, which the organizing students celebrated in the CS grad lounge with some illicit beers. When we found out Bob was going to join us for the celebration we hastily put our "responsible student" faces on and hid the beer—Bob showed up, beaming, with a six-pack of his own!

Bob taught me that beauty matters, that an idea can always be improved, and that the *Fest* can be as important as the *Schrift*. I will be forever grateful for his investment in me and our fulfilling collaborations. Cheers!

10. Daniel Spoonhower, LightStep
(Thesis: Scheduling Deterministic Parallel Programs, May 2009, co-advised by Guy Blelloch)

I learned many things from Bob during my time working with him, including, of course, many things about programming language semantics as well as how to approach programming language research in general. But there are many important aspects of this work that are not just about doing the research itself—at least, according to Bob, and I found that I came to agree. Communicating about research, both in written and spoken form, is

a critical part of being a researcher. As a student, Bob was always direct with feedback about my writing, and I am proud to say that, over the years, my drafts came back with fewer and fewer redlines. And not just proud, as these skills have played an important role in my career, both in academia and industry.

Bob also felt it was important to be able to communicate about our work in more casual settings, say, during a coffee break at a conference, and to engage with others in our research community. In fact, as a first year graduate student, I remember being admonished by Bob for gathering with other students from our group during one such coffee break. We could see each other any day: conferences were for meeting new people! And despite how stressful I found it, I'm glad Bob pushed us to do so. Later in my tenure as one of Bob's students, and as students from our group graduated and moved on to new positions, I really looked forward to seeing those former students at conferences: now that they were no longer part of our institution, I was free to socialize with them!

I am very thankful to Bob for the kinds of students and collaborators he attracted and the community he helped to build. Of course these were also often my collaborators, but in many cases they also became good friends.

11. Dan Licata, Wesleyan University
(Thesis: Dependently Typed Programming with Domain-Specific Logics, February 2011)

The first time I met Bob, I was—like so many other students before and after me—a student in the audience for a lecture. My undergraduate research group had driven up to see him give a talk on type theory for module systems. I didn't understand much of the details at the time, but I was instantly captivated by the material. The way that Bob can give a talk that is chock full of technical details, where you don't understand all of the details, but you nonetheless feel the shape of the ideas, is a rare magic trick. Despite seeing it performed many times, and getting advice on innumerable talks of my own from the magician, I still don't quite understand how he does it.

During my first three or so years, Bob was very helpful and patient and supportive as I learned the ropes. In retrospect, what Bob taught me then—about dependent types, metatheory, representations of variable binding, and hacking in proof assistants, but also about really digging into every detail of a very technical subject, and about expository writing—has been the foundation for everything I've done since.

In the second half of grad school, when I started to have some new technical ideas of my own, Bob was invaluable for refining and presenting them. I remember many an advisor meeting where I would come in excited about something, it wouldn't quite resonate with him, and I'd leave feeling a little deflated—until the next round of revision made the work so much better. As we approached paper deadlines, these meetings would spill into evenings and weekends too, often at a coffee shop. Bob would come in, often after a bike ride, and we'd spend the evening talking about the details of the work or about how to present it. One time, Bob and I were working on a paper with Noam Zeilberger, relating representations of variable binding to the idea of polarity that Noam was studying. The main idea of the paper was that variable binding was a "positive" connective, but we realized, quite soon before the deadline, that it could also be represented by a related

"negative" connective. While this didn't change any of the technical details of what we had done so far, it threw a wrench in the story we were telling about the work (there's always a crisis just before the deadline, I remember Bob telling me). We stayed at 61C until it closed and then went back to Bob's living room until after midnight to iron out the story. Until this phase of grad school, I didn't appreciate how much work it was to fit some technical work and the story presenting it together into a compelling paper.

Just after I graduated, Bob entrusted me with giving the lectures in CMU's new introductory functional programming class 15-150, a revamp of his longstanding sophomore-level 15-212. In addition to all of the research mentorship, these three semesters of teaching mentorship and experience are a big part of why I got hired for the job I have today. Bob and I designed the syllabus and mapped out what should be in each lecture together, but I got to put my own spin on the presentation. This mostly went well, except for one lecture just after spring break when I was a little underprepared, and didn't quite nail the explanation of the correspondence between parallel functional programs and cost graphs. Bob sat quietly in the front row, frowning and shaking his head (I'm guessing you all know the look). I've taught this course most years since, and every time I give that lecture (or any lecture, really), I'm reminded to make sure I understand every detail of what I'm about to say and to take the time to get the story just right.

12. Kuen-Bang Hou (Favonia), University of Minnesota
(Thesis: Higher-Dimensional Types in the Mechanization of Homotopy Theory, February 2017)

Perhaps surprisingly to many people, my research area before coming to CMU was in algorithms, and I had never been exposed to programming language theory until I took Bob's course on Types and Programming Languages (CMU 15-814). Bob's vivid explanation demonstrated the depth of his knowledge and how one should approach the field. I was blown away by his charisma and had since been preparing myself for PL research.

Throughout my Ph.D., Bob offered me great freedom and resources to pursue what I wished to accomplish; I was able to participate in various seminars and related activities even before becoming his student. After becoming his student, I was able to work with anyone in the field on any topic that interested me. Moreover, Bob has been selflessly providing guidance and help even when undergoing events as significant as kidney transplantation. Only after becoming a Ph.D. advisor on my own did I realize how much effort Bob must have put in.

I have learned many things from Bob: the formal analysis of programming languages, the pursuit of fundamental principles, and the respect for mathematical truth over personal preference. If I have become a successful researcher in any sense, it is all thanks to these lessons.

13. Joseph Tassarotti, Boston College
(Thesis: Verifying Concurrent Randomized Algorithms, January 2019)

My earliest memory of Bob is from the CMU visit days for admitted students, where he gave a talk on the propositions-as-types principle. Bob's fervor was inspirational, and when

I arrived at CMU in the fall, I had sworn off low-level systems verification work and was committed to studying type theory with Bob. As it turned out, a little less than a year later I was verifying concurrent code on weak-memory architectures, a complete reversal of my plans.

Fortunately, Bob was very patient as I found my way, and I will always be grateful for his support and guidance as I explored my research interests. When I would tell people what I was working on and who my advisor was, they would sometimes express confusion because they did not see the connection to Bob's own interests. But I could always trace a thread between what I was working on and some remark or question posed by Bob in one of our meetings. In particular, many of the research topics I have pursued were motivated by two guiding principles that Bob imparted to me.

The first is the idea that research can and should be motivated by teaching. If something important is too difficult or confusing to explain to students, that's a sign that the underlying ideas can be improved. When Bob was teaching the analysis of randomized parallel quicksort's running time, he asked me to see if there was a way to make the analysis simpler or more uniform. Answering that question directly led to our ITP 2018 paper together, which introduced a technique to make it easier to produce machine-checked proofs for such algorithms.

The second idea was the use of logical relations to reason about the encapsulation of benign effects. Studying Bob's book and discussing the topic with him many times gave me a deep appreciation for the centrality and beauty of these ideas. Just from looking at the work of Bob's students, one can get a sense of how versatile logical relations can be, with applications ranging from cubical type theory to (for my own part) transactional storage systems.

I'm thankful to Bob for teaching me these and many other lessons during my time as a student. It's a pleasure to be able to celebrate his career and his broad impact on the field.

14. Carlo Angiuli, Carnegie Mellon University
(Thesis: Computational Higher-Dimensional Type Theory, September 2019)

In 2011 I arrived at Carnegie Mellon University knowing only that I wanted to study the theory of programming languages. When I met with Bob, he told me about "homotopy type theory," an up-and-coming area that Vladimir Voevodsky had spoken about at CMU just one year earlier. Neither of us (well, nobody) knew much about it yet, but the topic seemed like a natural fit for me.

The early days of HoTT consisted mostly of mathematicians, computer scientists, and philosophers trying to understand one other. It felt to me like a real age of discovery, and I remember many meetings where Bob would explain some subtlety of dependent type theory to me, and I would explain to him what little algebraic topology I knew at the time. My second year was the Special Year on HoTT at the Institute for Advanced Study, and on weeks when only one of us was in Princeton, we would report back to each other over Skype.

HoTT has matured somewhat in the intervening decade, but I still remember its (and my) salad days fondly. I am eternally grateful to Bob for his constant support and advice

over the years, and for his patience when it became clear that my projects were considerably more complex and prolonged than we originally hoped. Beyond the fact that lambda conquers all, what will always stick with me is Bob's advice to develop a point of view: I always try to start a project by articulating exactly *why* it's worthwhile, and keep this perspective close at hand even when I'm deep in the technical details.

15. Evan Cavallo, Stockholm University
(Thesis: Higher Inductive Types and Internal Parametricity for Cubical Type Theory, February 2021)

I met Bob Harper via email in the early years of my undergraduate degree at Carnegie Mellon. I was ostensibly a student of physics, but I had become enamored first with Lisp and then Haskell and wanted to know how I could learn more. I imagine it was hard for him to know what to do with me, but he introduced me to his student Carlo Angiuli, tried to involve me in projects, and checked in on me from time to time. I remember one meeting where he pointedly asked me when I was going to figure out what I wanted to do, what I wanted from him—questions I had a hard time answering. When I made the decision to work under him for my PhD, I think neither of us was confident we'd be able to get on the same wavelength. I was enthusiastic, but also willful and scatterbrained. It took time to carve something recognizable as a researcher out of me.

With Bob, the story is paramount. In my undergraduate days, he was well-known among the students for his bombastic and mind-expanding lectures. As his teaching assistant and then doctoral student, I saw he was forever looking for ways to improve them, to drill down to the essence. Each year he'd reconstruct and reinvent his lesson plans, and in our weekly meetings he'd often take me on a tour through his latest perspective. When we sat down to plan out a paper, the first questions were the story-theoretical: what would be the hook? What was the climax? We didn't always agree at first what the story ought to be, but Bob impressed upon me the importance of working through it together, of putting in the effort to hammer out something that satisfied us both.

I am grateful to Bob for many things: for introducing me to the wild worlds of programming languages and type theory, for always knowing which researcher of yesteryear had the solution to our problem, for being my guide to the research community, for taking a sincere interest in my personal well-being. Most of all I am thankful to him for insisting that I get my story straight.

16. Jonathan Sterling, Aarhus University
(Thesis: First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory, October 2021)

It is hard to describe my first encounters with Bob in 2014—who blazed with enthusiasm for the unity of constructive mathematics and computer programming. This is a man who inspires strong reactions, and my own reaction was to fill myself up with his ideas and his ethos through his online lectures, blog posts, papers, and our correspondence. Bob asked

me to be his student in July of 2015, opening up an opportunity to me that would unfold into the most fulfilling five years of my life.

My fondest memories are of Bob doing what he does best—telling a story. Let me paint the tableau. We are in Bob's office, or talking through a pipe during the pandemic. I have brought some incomprehensible mathematics to Bob and he is thinking about it. Then he starts to unfurl a narrative that explains why this work matters, what it does, how it connects to our research program, and how it fulfills a dream from the classical literature—and all the while I am furiously taking notes. "That's the paper," he says. That's how I remember my time with Bob.

Derek Dreyer
Max Planck Institute for Software Systems
Saarland Informatics Campus E1.5
66123 Saarbrücken, Germany
(e-mail: dreyer@mpi-sws.org)

Benjamin C. Pierce
University of Pennsylvania
Dept. of Computer & Information Science
3330 Walnut Street
Philadelphia, PA, 19104-6389, USA
(e-mail: bcpierce@cis.upenn.edu)