On Being a Student of John Reynolds

Frank J. Oles

IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, USA

Abstract

The editors of this volume asked me to put together a description of what it was like to be a Ph.D. student of John C. Reynolds. With the assistance of his other students, this is exactly what I have done. I have asked each of them to contribute their reminiscences on what it was like to work with John, to describe how they came to know John, and to say whatever they would like about how John affected their lives. These contributions will be found below, ordered by the year that each student formally finished his studies. All in all, this piece serves to document the very positive effect that working under John had for those of us lucky enough to work under him.

1 Jerry Schwarz, Oracle Corp. (Thesis: Semantics of Partial Correctness Formalism, December 1974)

I had been working on a thesis under Alan Robinson that involved creating a “theorem checker.” But it was going slowly and I could see no end in sight. I don’t remember exactly how it came about, but at some point I realized that it would be faster to start over on a theoretical topic with John than continue on the programming task under Alan.

The advice I remember most from John was non-technical. He said that the point of a dissertation wasn’t to prove something about the subject, but to prove that the student was persistent enough to prove something about the subject. This turned into sound advice when I was part way through writing up my results and someone else published one of my key theorems.

John seemed to know everyone whose work touched on programming-language semantics. I remember that at one conference I joined John at a table for lunch or dinner with, among others, Edsger Dijkstra. I don’t remember the conversation at all, but I remember that some time afterward John told me I had made a terrible impression.

Preprint submitted to Elsevier
21 December 2006
But what I remember most vividly were the dinners John and Mary gave for visiting speakers. The department generally was invited to a party after dinner, but as one of his students I was frequently invited to the dinner as well. The discussion was inevitably interesting and the musical entertainment provided by John’s piano, played frequently by him and others was always lively.

I join his other graduate students in congratulating John on a long and distinguished career.

2 Nancy J. McCracken, Syracuse University (Thesis: An Investigation of a Programming Language with a Polymorphic Type Structure, June 1979)

Going back to the beginning of my association with John Reynolds, he had just won an ACM award for his paper on his programming language Gedanken at about the time that I was applying to graduate schools. Based on that, I applied to Syracuse University and went there for graduate school, and I am still living here in Syracuse and working at the university!

One of the great things about having John at the university was that he was the colloquium chairman for many years. Between John and Alan Robinson, many interesting and influential computer scientists, such as Tony Hoare and Edsger Dijkstra, visited Syracuse. As graduate students, it was very stimulating to hear all these ideas first hand. An additional benefit of the colloquia was that Mary Reynolds would give dinner parties in the evening after the talk and we were often invited. Both the dinners and the conversation were excellent.

As a thesis advisor, John was full of ideas and enthusiasm and I enjoyed working with him very much on my dissertation topic of the semantics of the polymorphic lambda calculus. I have since gone on to working in the areas of parallel programming languages and now in natural-language processing, but the solid background that I got as a graduate student in programming languages and mathematical and logical models has continued to be a solid underpinning to my work. I am now a Research Professor in the School of Information Studies at Syracuse University and fully enjoying working and learning new ideas.
In 1975, I was at Syracuse University as an Assistant Professor of Mathematics whose anticipated Ph.D. in noncommutative ring theory from Cornell was much delayed. (For a variety of reasons, including my own inability to write things up in a timely fashion, complicated by the failure of my dissertation advisor to get tenure, that Ph.D. never subsequently materialized.) At Syracuse, it is only a slight exaggeration to say that, as the most junior faculty member, I was stuck teaching the courses nobody else wanted. But, two of these courses would change my life. The first course was the History of Mathematics, from which I came to appreciate the importance in the development of many significant areas of mathematics of stimulation by problems from outside areas. The second course was Abstract Mathematics for Engineers, which covered discrete mathematics, elementary automata theory, and some coding theory. The second course opened my eyes to the range of problems of an algebraic character that could be found in computer science. Thus, I felt my desire to do interesting new mathematics could be matched with the need to get my career back on track by starting graduate work in computer science, which is what I proceeded to do.

Luckily, the School of Computer and Information Science at Syracuse at the time was staffed with a faculty well-suited to work with an exile from pure mathematics. From my point of view, foremost among the department members were Alan Robinson and John C. Reynolds. It turned out that I had been incredibly lucky to stumble, more or less by accident, into such a university in what now seem to be the early days of computer-science education (although not the early days of computer science—neither John nor I are that old).

I quickly discovered that John seemed to be the best match as dissertation advisor because he was already talking about the use of category theory in programming-language semantics, and I was already well grounded in categories, functors, and natural transformations. My recollection is that John had initially been turned on to possibilities in this general area by Dana Scott. I certainly can vouch for the fact that John knew that Cartesian closed categories could be used to explain the semantics of a simply typed lambda calculus. In a course of lectures, John graciously shared with me his initial sketch of how these ingredients might interact in order to give the semantics of an Algol-like language. This was the starting point for my dissertation, which attempted to give an algebraic treatment of a typed lambda calculus that supported implicit conversions between types, an algebraic treatment of general category-theoretic semantics for such a lambda calculus in terms of Cartesian closed categories, and finally an application of this theory to
the simultaneous explication of block structure, higher-order procedures, and implicit conversions in Algol-like languages by using functor categories (i.e., possible-world semantics). This was a dissertation that took me about three years to finish.

I thought I would never get that dissertation done, but in that time and place I was mostly very happy as a perpetual graduate student. While I was working on my dissertation, John and I met about once every two weeks. These meetings, generally about two hours long, were quite wonderful. I would briefly describe what I had been doing, and then I would furiously take notes on whatever John had been recently thinking, which normally wouldn’t have much to do with my dissertation. Often, he worked things out first time right in front of me. At least, it seemed that way. I think I typically only understood about a third of what he said, but I don’t want to place the blame for that on John. John gave me direct exposure to a deep thinker who really loved doing his work, while he was doing it. This was a great inspiration.

Perhaps because I had once been on the Syracuse faculty, and perhaps because I was a bit older than the other students, I was treated very well at Syracuse. John and I were on very good terms. I visited his home many times, although I don’t think I ever told him how terrifying I found his monstrous dog. At my place, I remember him, sitting on the couch after dinner, telling me that he did not restrain himself at the table because of his goal of trying to become a perfect sphere. Happily, it seems he later gave up this unhealthy quest for perfection. In those days, John and his wife, Mary, were often invited to wine tastings organized by Skip Mattson (another CIS professor) and myself. Once John joked that he almost blocked my Ph.D. because I beat him out to the last case of a good inexpensive Italian merlot at Liquor Square. It was only $4.00 a bottle. As a poor graduate student, I figured I was entitled to that case of wine, but, honestly, I didn’t know how badly John wanted it. After I got my Ph.D., I did give John a bottle of cognac.

When I finished my Ph.D. in 1982, John was a great help both in introducing me to the semantics community and in getting my first position as a visiting lecturer at the University of Aarhus in Denmark. And, by introducing me to Jim Thatcher—a pioneer in applying categories to programming-language semantics—he was instrumental in helping me get my current permanent job at IBM Research, where I have been a Research Staff Member since 1983.

Under the pressure of working in an industrial lab, my area of research has drifted away from programming-language semantics in the intervening years. Now I am working on new theories of symbolic pattern recognition with natural-language applications in mind. Nonetheless, my approaches to problems in the mathematical foundations of computer science, whether in semantics, domain theory, non-well-founded set theory, knowledge representation,
or pattern generalization, remain consistent with the algebraic and category-theoretic approaches that I developed while working under John’s direction more than two decades ago, except that now I have returned to more conventional notation! I am grateful to John for help in my intellectual maturation, for sharing some of his best ideas with me, and for many years of assistance and friendship.


I joined Syracuse University as a graduate teaching assistant in 1978. I had just graduated with an undergraduate degree in Electrical Engineering from the Indian Institute of Technology. The only thing I was sure of was that I loved everything theoretical. I loaded up on as many theoretical courses as I could and it was with reluctance that I signed up for John’s introductory course on the Craft of Programming. The fact that the course met from 7:30 to 10:30 PM in almost utter darkness did not help much. But John quickly won me over with the clarity and elegance of his lectures. One startling aspect of the course was the lack of actual programming as most people understand it. In fact, we were honor-bound from running the programs. Each program we submitted was essentially a thought-experiment and we substantiated our “code” as best as we could. The fact that stands out in my mind was my actual experiment after the term. I tried running the programs I had submitted and was impressed with the success rate of the John’s “thoughtful programming” style.

I must confess that I was not the most focused of John’s Ph.D. students. I recall Lockwood Morris asking me early on in my career at Syracuse what I intended to do. The one fact I was sure of was that I was going to do a Ph.D. in Computer Science. He seemed shocked that I did not have a specific topic in mind or a specific thesis advisor in mind. (I read a lot of science fiction and the protagonist of “Doorways in the Sand” by Roger Zelazny has always appealed to me. I must have unconsciously emulated his stated goal of permanent studenthood. I spent 10 very happy years in Syracuse.)

I gravitated towards John and took part in all the seminar courses that John offered. In one of them we were reading a bunch of papers. Each student had to select a paper and present the results in class. I don’t recall the exact details but I ended up with one of Girard’s early papers. Two facts stick in my mind. One is that the title was extraordinarily long and included Gödel. The second fact was that the paper was in French and I did not know any more French than I had encountered in Agatha Christie. Prodigious labor ensued and, after my overlong presentation on Girard’s results and the supporting proofs, Nancy
McCracken encouraged me to talk to John about a research topic he could
guide me on.

Once John graciously accepted me as a Ph.D. student, we met every two or
three weeks to share what I had been reading and working on. John would
generously share his ideas on what areas of exploration were open. I thoroughly
enjoyed the exploratory nature of this endeavor and did not pay a whole lot of
attention to staying on a topic and writing up a thesis. I knew it was time to
get real when John moved to Carnegie Mellon. I spent that last year focused on
my research and writing up my thesis. John’s generosity of spirit and luminous
intelligence are the precious aspects of my years at Syracuse that I will never
forget.

I join you all in wishing John the very best on his seventieth birthday. I would
like to thank John for all his help in the academic sphere and John and Mary
for their easy hospitality that has led me to appreciate the finer things in life.

5 Richard H. Connelly, Providence College (Thesis: A Comparison
of Semantic Domains for Interleaving, August 1990, advised by
F.L. Morris after John left Syracuse)

Dr. Reynolds and I met under awkward circumstances. My research topic was
predetermined as part of my agreement with my employer and Dr. Reynolds
graciously accepted me as a student. The topic proved to be difficult and I
am very grateful to Dr. Reynolds for his patience and tolerance as I struggled
with the topic and being an academic.

The rigor and discipline Dr. Reynolds brought to computer science has touched
us all. I am reminded of this each time I try to motivate an undergraduate to
be careful and thorough, and to “think before typing.”

I join everyone in congratulating Dr. John Reynolds on his seventieth birthday
and celebrating his many contributions to Computer Science.

6 Benjamin C. Pierce, University of Pennsylvania (Thesis: Pro-
gramming with Intersection Types and Bounded Polymorphism,
December 1991, coadvised by Robert Harper)

Few things can truly be called perfect. But I vividly recall a time when John
asked me to proofread a final draft of a paper (“The Coherence of Languages
with Intersection Types”) that he was about to send off for publication. I
considered myself a careful writer, and whenever I read one of my own final drafts I always noticed things that could be done better. So, having sweated to incorporate so many of John’s corrections to my papers, I looked forward to turning the tables for once. But to my astonishment, I found myself unable to suggest any improvements after a full day’s toil over this manuscript. Every prime in every line of every proof, every choice of phrasing, every comma—absolutely every detail was optimal.

The distinctive quality of John’s intelligence is power. He covers great distances by small increments—rather in the manner of a steamroller. I remember a graduate seminar on semantics and universal algebra where he spent the first 90-minute lecture entirely on sets and relations. I thought I was going to die of boredom. By the middle of the second lecture, with no apparent shifting of gears, we were talking about natural transformations. For the rest of the semester we were swimming in a sea of abstractions, six or eight layers of definitions deep. My nose came to the surface only often enough for an occasional breath, but John seemed to be walking on solid land. Since the first set of definitions was clear and precise, so was the second layer, and the one built on this, and the next, and so on to seemingly arbitrary depth.

John’s ability to get to the absolute bottom of things—combined with his nose for which things are worth getting to the bottom of—are what give his results their permanence.

Working closely with him during my Ph.D. was a continuous lesson in what is possible—one that I return to every day and that influences my own work profoundly.

7 Andrzej Filinski, University of Copenhagen (Thesis: Controlling Effects, May 1996)

I was one of John’s Ph.D. students at Carnegie Mellon from September 1989 through December 1995 (co-advised by Bob Harper for the last year). The opportunity to work with John was in fact one of my primary reasons for applying to CMU in the first place: his 1972 paper “Definitional interpreters for higher-order programming languages” was my first exposure to continuations, a concept central to both my M.S. and Ph.D. thesis work.

At CMU, I found that while John’s research focus had largely shifted away from continuations, he still generously let me pursue my own ideas. These eventually crystallized into an investigation of the relationship between monadic and continuation semantics, with the goal of formally establishing that, between them, first-class continuations and mutable state could simulate any
other computational effect. It all boiled down to a correctness argument for about 20 lines’ worth of highly convoluted ML code. Ironically, while I was of course aware of John’s 1974 paper “On the relation between direct and continuation semantics,” I found the domain-theoretic constructions there too forbidding, and instead based my work on a seemingly simpler, abstract approach proposed in a 1985 paper by Meyer & Wand. This worked fine, up to a point, and I actually put together a mostly complete and plausible thesis draft in this style, with just a few warts left to be cleaned up after the defense.

As it happened, John was away for a sabbatical during my last year at CMU, and we communicated mostly by email. Though he did read the succession of slowly converging thesis drafts ahead of the defense, it was only about a week before the event that he—and with him, the rest of the committee—developed serious reservations about the formal treatment. Still, since the problems did not actually appear show-stopping, the defense went ahead as scheduled, although it was atypically grueling. By the end, even though I passed, it was clear that the revisions to the final document would have to be substantial.

Ultimately, the revisions turned out to take almost another year. During that time, John was consistently supportive, and maintained that the overall result was most likely correct; it just needed to be established properly. After a couple of false starts, I eventually realized that the abstract approach had inherent limitations, and I therefore returned to the domain-theoretic treatment from John’s 1974 paper. In the meantime, as Bob Harper pointed out to me, the scariest aspects of relations over reflexive domains had been somewhat tamed by Pitts, and I was able to redo the formal development to everyone’s satisfaction. Much to my relief, the ML code was indeed correct, even in a stronger sense that I had first thought. It was also gratifying to see how all the pieces of the proof now fit cleanly together on a solid foundation, instead of occasionally getting bent out of shape—perhaps even beyond the breaking point—as in the original draft.

After graduation, John has continued to take an interest in my career, written recommendation letters, etc. Still, perhaps paradoxically, what I’m most grateful to him for, is precisely that he made me effectively rewrite my thesis from scratch, after I thought it almost finished. A lesser supervisor might have let it slide, perhaps with some papering over of the holes; but moving on to further work without realizing the crucial importance of grounding all definitions, lemmas, and theorems in a single, well-defined semantic framework, could very well have made me eventually “prove” something demonstrably false. Going over some of John’s articles again with a more experienced eye, I can now better appreciate his constant attention to precision, and strive to emulate it in my own work.
In truth, John Reynolds was a secondary advisor for me. John and my primary advisor, Frank Pfenning, shared me for about two years before I settled into a line of work (ordered linear logic) a bit afield from John’s immediate interests. However, John continued to provide support and keep me as an official student. From time to time I would go to John’s office to update him on my progress, or lack thereof. He would always graciously let me babble on about my latest technical catastrophe and try to give me some useful, general advice gleaned from his many years of formidable research experience. In addition to giving academic support and encouragement, John often invited me to the dinner parties he and Mary gave for various visitors. I quite enjoyed these affairs both for the chance to meet and socialize with John’s colleagues, and for the excellent fare from Mary’s kitchen. My lasting impression of John is of a very kind and supportive man; even though I cannot say that we are good friends, I feel very comfortable around him. Finally, I would like to add that aside from the obvious advantages, to a young computer scientist, of my association with John, I feel that I am a better researcher and person for knowing him.