A novel program for philosophers to study computer science

David Hemmendinger
Union College

For three years in the mid-1980s, Wright State University (WSU), near Dayton, Ohio, had a fellowship program that brought people holding Ph.Ds in philosophy to campus for an intensive 15-month M.S. degree program in computer science. It enrolled thirteen philosophers over the three years, eleven of whom completed the program.

A little background is in order. I had a Ph.D. in philosophy, which I taught for over a decade. In 1981 began studying computer science because I found it interesting and because I hoped that it would lead to steadier employment. I found that computer-science departments frequently had difficulty in hiring Ph.Ds in computing, and in some cases, were happy to hire people with Ph.Ds in other subjects if they knew enough computer science to teach it. Wright State hired me after two quarters of study, at the same time that I entered its M.S. program, and they gave me the hardest undergraduate course to teach, on concurrent and real-time programming.

In the year after WSU hired me, the department continued to try to hire Ph.Ds in computing, with no success — one problem was that with no Ph.D. program itself, the department didn't have Ph.D. students to work on faculty research projects. I seemed to have worked out well, and so the department decided to try to hire another philosopher. The department chair and I went to a national philosophy conference and hiring meeting, where we interviewed about twelve candidates. I found it interesting that despite our different backgrounds — he had a Ph.D. in electrical engineering — we agreed completely on our ranking of candidates. We hired our first choice, who began studying computer science in the summer before starting work and then, as I had, enrolled in the M.S. program while also teaching two courses a quarter. She completed the program and then went to a liberal-arts college to teach computer science in 1985.

In 1984, the department chair, having seen the quality of philosophy Ph.D. holders who had applied for our job, proposed offering an intensive M.S. program for philosophy Ph.Ds, some of whom might choose to join computer science departments, he thought. He managed to get money from the university to try such a program. Our proposal said:

> Computer science departments have a shortage of professors; well-qualified Ph.D.'s in philosophy are unable to find rewarding academic jobs, and many of these people have skills that would readily transfer to computer science. There are also a growing number of philosophers who find that solid grounding in computer science would be valuable in their philosophical work. We propose a special M.S. program in computer science to meet the needs of these two groups. It would prepare them for employment in computer science, would enable those who continue to work in philosophy to apply detailed knowledge of computer science to their work, and would support new interdisciplinary work.

We offered grants of tuition and $5,000 stipends to four students who enrolled in the summer of 1984. While the stipend was low, it was somewhat more than average college room and board fees; that is, it was manageable by people still accustomed to living like a student. One of the four was on sabbatical, and the others were under- or unemployed. Three completed the program, and the fourth, who didn't enjoy the intensive study, continued to take some undergraduate courses. In addition a, student with an M.S. in philosophy, who was not eligible for our program, enrolled in the regular computer science graduate program.

We advised the prospective students to learn as much Pascal programming as they could before coming to us in June. During the summer, they took three undergraduate courses: data structures, PDP-11 assembly-language programming, and formal-language theory. These courses were supplemented by a tutorial that I taught, which addressed difficulties that the students

encountered, together with material to tie the courses together. One of the latter topics was recursive-descent parsing of simple languages. Although many beginners find recursion difficult, these students had a background in formal logic and found it quite natural.

In the fall, the students took the real-time programming course that was required of all undergraduate majors and M.S. students. Students commonly regarded it as a *rite de passage*; they wrote in assembly language to build device-drivers and application programs, coordinated by a multi-tasking kernel that we had written. They had to pass an oral exam on the final program, which required them to explain the program operation with the aid of an octal dump. This was quite different from anything that the students had encountered, and it was impressive that they could handle the course after rather brief preparation.

The remainder of the students' program was not prescribed, but generally included courses on programming-language theory and artificial intelligence, along with options that included operating systems, natural-language processing, logic programming, compiler design, and sequels to some of these courses. They also participated in an informal seminar with several faculty members and other graduate students. The seminar had initially been about functional languages and their implementation, but, in part to take advantage of the philosophers' knowledge, it turned to natural-language processing, knowledge representation, and other topics in cognitive science.

The three students who completed the program all obtained positions in CS departments either immediately or subsequently, though I believe that only one remained in the field some years later; he has been a prolific author of textbooks on introductory programming and data structures.

My colleagues and I thought that the program had been successful. As it turned out, the university had accidentally deposited funding for the first year twice, so we had enough to run it a second year. We advertised the program to linguistic departments too, and one linguist was interested, but she decided not to apply to it. We offered grants to four more philosophers, two of whom were on sabbatical from small colleges that wanted to start computing programs (one had been acting president of his college). This group also did well. One of them, with no prior programming experience, became the best programmer in the group and went to an AT&T lab in Ohio, along with another non-Ph.D. philosopher who got a regular M.S. degree. The last went on to teach computing and then joined NASA, where he has been for many years.

I don't recall where our funding came for a third year. It was less than during the first two, but we offered five stipends. One participant didn't complete the program, being interested in continuing his work on cognitive science instead. Of the other four, one went to a CS department and also became an author of computer science texts. Three remained in philosophy departments, though one taught CS at WSU for a year. At that time, the computing accreditation board began to ask programs to include a course on computing and social issues, and the department asked him to develop its course. He not only did, but went on to co-author a well-regarded book on the subject, one of the first of its type.

Program participants wrote master's theses during their second summer. One, who had written his Ph.D. dissertation on the philosophy of history, wrote a thesis on some problems of compiling the Ada language. Other theses that I supervised were on semantic networks, synthesizing programs from logic specifications, natural-language processing with a categorial grammar, and using Prolog for teaching formal logic. Two of these theses yielded journal articles, and another participant was the senior editor of a book on computational linguistics.

For eleven of thirteen participants to complete this intensive program despite having little prior computing experience, was impressive. Studying philosophy includes some work on formal logic, which was probably helpful, but it would be too simple to say that philosophical problem-solving was a major factor (philosophical problems often don't get solved!). Research in any discipline requires analytical thinking and problem solving in terms appropriate to the discipline,

and could help to prepare one to study computing. But the participants were evidently capable and motivated. One of them wrote recently that he'd found the program "extremely rigorous, challenging, and exhausting".

There were several reasons for the program's ending in 1987. The WSU CS department was starting a Ph.D. program and focused its energies on it. As we expected, once the program started, it became easier to hire Ph.Ds in computer science and computer engineering. It also appeared to us, though without much analysis, that undergraduate CS departments were generally finding it a little easier to hire Ph.Ds in computing, so that graduates of our philosophers' program were finding it more difficult to get jobs in CS departments

Our program achieved several things. It brought some talented people into work in computer science and gave others a broader range of teaching and research skills, enriching their work in philosophy. The presence of these people benefited WSU graduate program by giving us a group of scholars capable of doing advanced work in computer science. The faculty who worked with them agreed that the contribution they made in courses through their philosophical training has by far offset their relative lack of experience in computing. As an unintended consequence, the program also influenced at least one other CS department to emulate it, according to the philosopher in the department who started it.

Although the program was demanding, it had a lighter moment, too. The "dining philosophers" is a synchronization problem that Dijkstra introduced to illustrate deadlock prevention. Five philosophers sit around a table, with a bowl of spaghetti in the middle and a fork between each pair of philosophers, which represent independent processes. The philosophers alternate between thinking and eating, and to eat, a philosopher must acquire the two forks on either side. Unlike most real philosophers, they don't communicate. The problem is to avoid the deadlock that would result from each philosopher taking the fork to the left (or right) at once, and generally, to avoid starvation as well, which would be literal in this example. There are a variety of standard solutions to the problem, using synchronization primitives such as semaphores. Because one of the goals of the WSU program was to help philosopher get jobs and hence avoid starvation, it seemed appropriate to offer a new solution to the problem. To that end, I made single-fork deadlock-free solutions to give graduates of our program, as illustrated in Figure 1. In return, the third group of participants presented me with a gardening fork, inscribed with the names of of the philosophers who'd studied with us, which I still use.

Figure 1. Dining philosophers deadlock-free spaghetti fork

About the author

David Hemmendinger is an emeritus professor of computer science at Union College, Schenectady, NY, where he taught courses on programming languages, computer architecture, parallel processing, and the history of computing. He has studied the history of mathematics and of science, and for the past two decades, the history of computing. He also co-edited the fourth edition of the Encyclopedia of Computer Science (John Wiley & Sons, 2003) with Tony Ralston and Ed Reilly. Hemmendinger has a Ph.D. in philosophy from Yale University and an M.S. in computer science from Wright State University, Dayton, Ohio. He is on the editorial board of the

Annals. Contact him at hemmendd@union.edu.