Functional programming in education – Introduction

Functional programming is becoming accepted as an integral part of the computer science curriculum in higher education; indeed in a growing number of places it is the *first* programming paradigm which students learn.

While readers of this journal need no convincing of the value of functional programming, others — including our students and colleagues — certainly do. One of the most valuable contributions of these papers is to collect and discuss reactions of students to learning functional programming. They range from the syntactic (a mystification with application as juxtaposition), to the conceptual (a confusion between definition and assignment; the problem of interpreting type error messages), and the psychological (the perception that functional languages are 'difficult' or 'irrelevant').

For our colleagues, the papers contain arguments in favour of functional programming, and documentary evidence that teaching functional programming can be more successful than using more conventional languages, both for computing specialists and students in other disciplines.

Resources are also important. There is an increasing number of textbooks offering a variety of approaches. Some of these are discussed in the papers, and others in the reviews (Thompson, 1992) (Wadler, 1992). Equally useful are case studies: some can be found in textbooks, but it would be exciting to see more sharing of the work which is undoubtedly going on, perhaps by means of ftp archives?

One area of difficulty which many may face is the integration of functional programming into the wider curriculum. This helps convince students of its relevance, also allowing them to reinforce their skills in a different context. Nonetheless, it is too early to estimate how much functional programming courses will affect their contexts.

Amongst the individual papers, Lambert, Lindsay and Robinson describe their course using Miranda as a first programming language, complete with details of assignments and laboratory work, as well as informal feedback from the course. Molyneux has introduced functional programming to non-specialists as their *only* programming language, contrasting it with 'fourth generation'-style tools like spreadsheets. His paper surveys the enterprise, as well as providing a small case study from business management.

Joosten, van den Berg and van der Hoeven have also introduced a course for first-time programmers, and amongst a wealth of material they have a useful analysis of common student errors as well as a comparative study of imperative *versus* functional programming ability. Harrison's survey gives evidence of the widespread use of functional languages in teaching, providing as well a list of contact addresses.

The issue also contains two general papers describing functional programming environments. Augustsson describes an interactive LML system, itself written in LML. Runciman, Toyn and Firth discusses two interactive environments, Glide and Starship, which support transformational development of lazy functional programs. Together they show that functional language environments have the ability to emulate the best.

Simon Thompson and Philip Wadler

References

Thompson, S. 1992. Comparative review of functional programming textbooks (Bailey, Bird & Wadler, Holyer, Paulson, Reade, Sokolowski, Wikstrom.), *Computing Reviews*, May (CR Number 9205-0262).

Wadler, P. 1992. Banishing errors, *Times Higher Education Supplement*, 21 February. (Reviews *ML for the working programmer* by Paulson, *The definition of Standard ML* by Milner, Tofte, and Harper, *Commentary on Standard ML* by Milner and Tofte, and *Applicative high order programming: the Standard ML perspective* by Sokolowski.)