Book reviews

Learn You a Haskell for Great Good! A Beginner's Guide, by Miran Lipovaca, No Starch Press, April 2011, ISBN-10: 1593272839; ISBN-13: 978-1593272838, 376 pp.
doi:10.1017/S095679681300004X

The first thing that you will notice about this book is the clunky how-foreigners-speak-English title. And the second is the clunky illustration style. Of course, the word "clunky" demonstrates either my high-mindedness or my pomposity. Either way, these tropes really got in the way of my taking this book as seriously as it deserved.

"Comic" books on programming languages actually have a long pedigree. Kaufman's *A FORTRAN Coloring Book* has a Dr Seuss-ish feel. I used Alcock's *Illustrating Basic* (1977) for many years to teach non-specialists with the BBC Microcomputer. In our own noble discipline, Friedman's *The Little LISPer* (1974) predates both. In turn, this has spawned Friedman and Felleisen's *The Little Schemer* (1998) and *Little MLer* (1998).

These older books have a gently whimsical feel. In contrast, I found *Learn You a Haskell...* considerably more abrasive in tone. Weak puns abound, as do geek culture references, for example to *Mission Impossible* and *Star Trek* and Terry Pratchett and spaghetti westerns. "Stuff" is "cool." The "maximum" function is "awesome." We take a journey to "the top of Monad Mountain." Perhaps, the target readership will warm to this.[1] Luckily, for me at any rate, the author does not maintain this style throughout: most of the book is thoughtful and well-written.

I think the illustrations also lack the charm of those in the earlier books. At best, they are mildly cute. At worst, they are offensive, in particular the scantily clad, curvaceous woman on p. 102 accompanying a phone book example, which only contains female names. Indeed, one of the few other representations of women, among a plethora of men and animals, is the "old lady" on p. 95. Really, sexism is not funny. This is 21st century: gender neutrality should be taken for granted in academic computing.

Now, there is a generic problem with books about specific programming languages, or perhaps a family of problems. Do they assume that the reader already knows how to program in some other language? If so, which language? If not, then do they seek to teach some pedagogy of programming? If so, then which pedagogy?

Learn You a Haskell... is aimed at readers with imperative programming experience in, say C++, Java, or Python. At the start, there is a very brief summary of key functional programming concepts from an imperative perspective. Thereafter, the book dives straight into interactive *GHCi* use and is driven by Haskell constructs. Thus, no clear discipline is offered to marshal an initial myriad of techniques to solve substantial problems. Of course, this is a fault of many functional programming texts, my own included. Overall, though, I think this book might prove hard going for self-study except for experienced programmers.

Nonetheless, the book is well suited to accompany a taught course on Haskell, offering a strong and systematic emphasis throughout on the central notions of types and higher order constructs. Topics in functional programming, such as recursion, type polymorphism,

[1] The publisher, No Starch Press, has as trademark "The Finest in Geek Entertainment," and also publishes *The Manga Guide to Regression Analysis* and *LEGO Heavy Weapons: Building Instructions for Working Replica Guns.*

functional abstraction, and laziness, that are new to students often find hard on first encounter but are treated steadily and sympathetically through generally well-conceived, if occasionally irritating, practical examples to illuminate wider theory.

The book's main strength is the thorough treatment of I/O, monads, monoids, and stateful functional programming, where many introductory Haskell texts are covered just enough to enable basic interaction. The equally substantial coverage of these topics in *Real World Haskell* (O'Sullivan *et al*., 2009) may be better arranged for people who know what they are looking for, and lacks the distracting asides and cartoons, but the *Real World Haskell* examples are dry and have low motivation in comparison to *Learn You a Haskell*....

Frankly, if I had not been sent this book for review, I would have not bought it. But I will have no hesitation in lending it to my students.

References

Alcock, D. (1977) *Illustrating Basic* (*A Simple Programming Language*). Cambridge, UK: Cambridge University Press.

Felleisen, M. & Friedman, D. (1998) *The Little MLer*. Cambridge, MA: MIT.

Friedman, D. (1974) *The Little LISPer*. Cambridge, MA: MIT.

Friedman, D. & Felleisen, M. (1998) *The Little Schemer*. Cambridge, MA: MIT.

Kaufman, R. (1977) *A FORTRAN Coloring Book*. Cambridge, MA: MIT. Available at: http://www.seas.gwu.edu/~kaufman1/FortranColoringBook/ColoringBkCover.html.

O'Sullivan, B., Goerzen, J. & Stewart, D. (2009) *Real World Haskell*. Cambridge, MA: O'Reilly.

GREG MICHAELSON
Heriot-Watt University, Scotland

OCaml from the Very Beginning, by John Whitington, Coherent Press, 2013, £25.99, US $37.99. ISBN-10: 0957671105 (paperback), 204pp.
doi:10.1017/S0956796813000087

The publisher's blurb for this book says that it "will appeal both to new programmers and experienced programmers," and that it is suitable for "an undergraduate or graduate curriculum, and for the interested amateur." That is a lot to expect of a book, and the author's preface is more modest. He explains that the book arose from his experiences working as a tutor at Cambridge for the first-year course taught by Larry Paulson using Paulson's classic text "ML for the Working Programmer." It is reasonable to judge this book, then, on how it might work with first-year undergraduates majoring in a STEM discipline.

Some of these students will have prior programming experience (usually in an imperative language) and some will not. Both groups will be impatient, which accounts for the brevity of this book. It has 16 chapters, some as short as two pages. Each chapter is followed by a half-page of questions and a cumulative summary of the book so far, typeset to fit onto a single page. Hints and solutions to the questions are at the end of the book. The style is concise but not terse; the author takes care to write plainly and clearly. The book tries in this fashion to deal with a reluctant reader, or one who dips into the book in a nonlinear fashion, or to whom English is a foreign language.

The magnitude of the task is made clear in the single page of "Getting Ready," preceding the first chapter, which starts with the expression $1 + 2$ entered into the OCaml REPL. How we are to get to this point is not specified. Of course, a complete description (for each of three major platforms, one with several variants) of how to locate the software, download it, install it, navigate to a working directory using a command-line interface, and start the interpreter would make the book half again as long, and parts of it would be obsolete before the book