yaws, the Erlang-based web server. Background knowledge about web services is assumed, and a number of concepts and configuration possibilities are explained comparing and contrasting with PHP *et al.* The reader might find a bit confusing, strange, or even frustrating that some very simple things are illustrated with basic examples, and right afterwards more interesting and advanced pieces are left to the readers' own extrapolation capabilities. The overall sensation is that it is hard to tell if the writer wants us to follow the book as a hands-on tutorial or manual, since we will find ourselves lacking basic information at some points; but at the same time the amount of source code in examples is so prolific that it is hard to resist the temptation to try and do it oneself.

The book is very ambitious in that it mentions every aspect about web services that one may think of, but in many cases after a few thoughts the author leaves us with one of the most repeated sentences in the volume: "beyond the scope of this book." However, a great point in its favour is the great amount of time and effort that the author has devoted to actually research and select complementary sources of information, which he points to the reader. In Chapter 4, the reader is presented with the first full example, which puts all previously examined pieces together. From then on, we see a series of examples of good practice, again especially focused on explaining key differences with other frameworks such as PHP+Apache. Beside useful pointers to external resources for further information and material, a clear explanation on how to interact with Amazon S3 service is also presented.

Although the first part of the book is heavily oriented to REST-services, Chapter 6 is devoted to websockets, a sort of TCP-socket-like behaviour over HTTP, which represent the opposite philosophy to REST, thus complementing the content so far.

The rest of the book shows specific examples of how to do specific but very common tasks (content upload and streaming, for instance), including a discussion of HTTP client alternatives to be used for testing purposes (CURL, https, etc.). Last but not least, a nice full OTP-flavoured example is included in the last chapter, which in a way is one of the barriers for readers who are not so familiar with web services: a reader cannot see the whole picture until the end, and chapters do not seem to follow a cohesive storyline, they just cover different aspects, which again bring to mind the dichotomy between manual and book.

This book was reviewed in e-book format, which was unfortunately not a very polished edition. Some figures were too small to be read in a 6-inch e-reader, and the verbatim font used for source code sections was not only poorly formatted but also its greyscale colour was too faint to read at times.

Overall, this is a nice book which provides easy reading for professionals, experts in the web-services world that have come across Erlang and want to see that the same things they are used to can be done with Erlang and Erlang-based tools in a very simple and quite similar way to what they already know. It is definitely a volume this reviewer would recommend to read in a digital format, preferably with Wi-Fi capabilities, such is the number of online resources and references that have been meticulously researched and kindly presented. However, this is not a volume for those seeking the basic knowledge to implement a web service, because neither does it give an introduction to what Erlang is capable of nor is it a reference book.

LAURA CASTRO
University of A Coruña

Pearls of Functional Algorithm Design, by Richard Bird, Cambridge University Press, September 2010, £35.00, US $ 60.00. ISBN: 978052151338 (hardback), 286pp
doi: 10.1017/S095679681200041X

When I first started learning Haskell, I did so by adapting a Sudoku implementation into dealing with partial Latin Squares. That implementation was from a Functional Pearl by

Richard Bird, which was not only clearly written – especially for a complete beginner to the language – but also went on to discuss program design, and how to derive a more efficient version of a function from an initial specification.

This Functional Pearl can now be found in Richard Bird's book *Pearls of Functional Algorithm Design*. Prevalent throughout this book is the theme of deriving a working, efficient implementation of an algorithm from an initial simple definition. These derivations take the form of utilising the laws of functional programming to calculate a new version.

For those who are unaware, Functional Pearls are elegant, instructive examples of functional programming. Richard Bird has written many such Pearls – many of which have appeared in this journal – and this book contains 30 of them. About a third of these are new, and those that have previously appeared elsewhere have been polished and improved.

The algorithms and derivations in this book are implemented in the purely functional language Haskell. However, the Pearls do not always use what can be characterised as "typical" Haskell; for example, the very first Pearl "The smallest free number" uses an implementation of the list difference operator "(\\)" that differs slightly in some cases from the definition in the Haskell Report. As such, this needs to be kept in mind when reading this book.

That said, only basic understanding of Haskell is required to be able to read and comprehend the Pearls in this book (the book even provides an introduction to some more advanced aspects of Haskell programming such as utilising the QuickCheck library for testing of functions). It is not necessarily a light reading, and the tone is more formal and academic than the blog posts in which programming techniques are often found today. However, for those that make the effort there is indeed something of beauty to be found in the pages of this book.

For the most part, each Pearl can be read in isolation (though several Pearls are linked). Typically they discuss – and reference – an interesting algorithmic challenge/problem, most of which would probably be unknown to a more practical-oriented programmer as opposed to a computer science researcher. As such, they are not only interesting in the approach taken but also in the breadth and scope that Richard Bird has used to find examples for these Pearls.

That said, some examples – such as the aforementioned Sudoku Pearl – would be more recognisable to the general programming community. Two of the more interesting Pearls cover how two different ways of implementing a specification for string matching lead respectively to the well-known Boyer–Moore and the Knuth–Morris–Pratt algorithms.

The approach taken in this book of design by calculation may not always be followed in everyday programming, but it is a useful technique in a programmer's toolkit. *Pearls of Functional Algorithm Design* provides an excellent guide into this method of algorithm development.

IVAN LAZAR MILJENOVIC
Canberra, Australia
Ivan.Miljenovic@gmail.com

https://doi.org/10.1017/S095679681200041X Published online by Cambridge University Press