JFP **35**, e6, 3 pages, 2025.
doi:10.1017/S0956796825000024

Book Review

Review of "Haskell in Depth" by Vitaly Bragilevsky, Manning Publications, 2021

"Real World Haskell," by Bryan O'Sullivan, Don Stewart, and John Goerzen, was one of the first serious attempts at demonstrating the utility of Haskell in a practical industrial setting. It largely succeeded, though it's now quite outdated. "Haskell in Depth," by Vitaly Bragilevsky, is a worthy successor, and it goes much further in showing how high-powered theory and advanced type-system features can be put to practical uses.

One of the best features of the book is the extensive collection of almost 200 accompanying examples, ranging from tiny examples illustrating a single concept or operation all the way up to larger applications with multiple modules. All examples are described in the text as well as freely accessible in an associated GitHub repository, with `.cabal` and `stack.yaml` files for easily trying out the examples oneself. As of this writing, the examples have been updated to work with GHC up through version 9.6, and the author plans to continue keeping them updated in the future.

To say that the examples are well chosen is an understatement. For one thing, they do indeed feel like "real-world" examples that are not too contrived; I can easily imagine using many of the examples as a reference the next time I need to accomplish something similar. Indeed, reading some of the examples gave me the confidence to use features and libraries I haven't used much before, such as benchmarking and concurrency. Somewhat miraculously, each example tends to use only a few new bits and pieces (operators, language extensions, and so on) relative to previous examples, and those new bits and pieces are explained just-in-time. The reader never feels overwhelmed with an example that requires many new features all at once, and yet the examples still feel like natural ways to illustrate the concepts and features being discussed. As a teacher myself, I understand how fiendishly difficult this is to pull off, and how much work must have gone into selecting, developing, and revising the examples. Most readers would not notice or think about how much work went into the examples, precisely because of that work—the examples just seem to flow by naturally.

The print quality of my copy was good, other than a few crooked pages near the beginning of the book. The binding, on the other hand, is one of those annoying, cheap, distressingly prevalent paperback bindings with a bunch of pages just glued directly into a cardboard spine. When you're near the beginning or the end of the book, the pages on one side aren't heavy enough to keep the binding from springing the book shut, so the book won't stay open. I assume the pages will also start falling out with moderate use. But I guess it keeps costs down.

This is definitely not an introductory Haskell book, and indeed, it does not pretend to be; the first few pages clearly explain that the book is not intended for beginners. To get the most out of this book, you should already have a solid grasp of Haskell basics such as recursive function definitions, algebraic data types, type inference, type classes, the standard Prelude, and the Functor-Applicative-Monad hierarchy. However, the book still sometimes seems confused about its intended audience, especially in the first few chapters. For example, Chapter 2 tries to explain the Functor, Applicative, and Monad classes, but the explanations are so terse that anyone who does not already understand the concepts will not learn much, whereas those who do already understand the concepts will be bored. A few of the other introductory chapters suffer from similar problems. Of course, each reader's experience will be different, and "understanding monads" is not a binary proposition; perhaps these quick explanations may be more helpful for beginner-to-intermediate Haskellers than I realize. After all, it never really hurts to read another presentation of a deep concept.

Overall, Chapters 1–5 introduce some basic Haskell features and libraries—of course, via good examples. Chapter 3 is an especially nice reference for anyone wanting to put together their first practical Haskell application; it walks through the process of developing an entire application to do some data analysis, importing a bunch of libraries and explaining everything along the way.

Chapter 6 on monad transformers and Chapter 7 on error handling are good, though I was surprised to see no mention of algebraic effects and effect handlers. I can understand that algebraic-effects libraries might not be battle-tested enough to cover in detail, but it seems somewhat outdated not to at least mention such an active area of research and development, especially one that is so relevant in a real-world setting. In contrast, I was impressed at how relevant and up-to-date the chapter on testing (Chapter 8) seemed, and the breadth of technologies it was able to cover (tasty, hspec, Hedgehog, doctest, Liquid Haskell, hlint). I did not expect to learn anything new from that chapter, but I was pleasantly surprised!

Chapter 9 gives an introduction to GHC's runtime system, explaining how closures, thunks, and data structures are represented at run time. This is not something I knew much about previously, so I definitely learned some things from the lucid presentation. There is also an informative worked example of how GHC does optimization via inlining, rewrite rules, foldr/build fusion, and so on. Chapter 10, on benchmarking and profiling, was also a very helpful guide, one I will certainly return to the next time I need to profile some code. (It was amusing to see an internal detail of my `split` package show up as the culprit behind bad performance in a toy IP-filtering application!)

Chapter 12, on metaprogramming, covers both deriving strategies and Template Haskell. I'm not fond of the hand-wavy way in which `deriving via` is explained—given the previous explanations of `newtype` and roles, the explanation could have easily been much more precise. But otherwise the chapter is solid.

Chapters 11 and 13 are about fancy type stuff; since this part of Haskell is changing relatively quickly, some of it feels slightly outdated, but overall the explanations are good. I especially like the example of building an elevator-control DSL that is correct by construction, using fancy type features to track things like the current floor and whether the doors are open—at the type level.

Chapters 14–16 are sort of a grab bag of advanced features, but they contain some very cool stuff. Chapter 14 is particularly mind-expanding, as it introduces the idea of streaming libraries by building one from the ground up!

Overall, this is a solid book to learn from and one that you will keep returning to as a reference. The examples alone are worth their weight in monads. Anyone aspiring to write real-world Haskell code—whether personally or professionally—should have this book on their shelf!

Competing interests

The author declares none.

Brent Yorgey
Hendrix College
Conway, AR 72032, USA
(*e-mail:* yorgey@hendrix.edu)