Lattner, C. (2010) *The LLVM Compiler Infrastructure*. http://llvm.org.

Leijen, D. & Meijer, E. (2001). *Parsec: Direct style Monadic Parser Combinators for the Real World*. Technical Report UU-CS-2001-35. Department of Computer Science, Utrecht University. http://www.cs.uu.nl/~daan/parsec.html.

Nielson, F., Nielson, H. R. & Hankin, C. (2005) *Principles of Program Analysis*. 2nd printing edn. Springer Verlag.

Parr, T. (2007) *The Definitive Antlr Reference: Building Domain-Specific Languages*. The Pragmatic Bookshelf. http://www.pragprog.com/titles/tpantlr/the-definitive-antlr-reference.

Parr, T. (2009) *Language Implementation Patterns*. The Pragmatic Bookshelf. http://www.pragprog.com/titles/tpdsl/language-implementation-patterns.

Swierstra, S. D., Rodriguez, A., Middelkoop, A., Baars, A. I. & et al., A. Loeh. (2010) *The Haskell Utrecht Tools (hut)*. http://www.cs.uu.nl/wiki/HUT/WebHome.

JURRIAAN HAGE
s.j.thompson@ukc.ac.uk

Foundations of F# Robert Pickering, Apress, 2007 ISBN 10: 1-59059-757-5 doi:10.1017/S0956796810000110

The fact that Microsoft has created a functional programming language on top of the .NET platform enables a very large community of programmers to experiment with functional programming in an environment that they are familiar with. If they decide that functional programming is of value for them, it will be a big advantage that it can be deployed on a platform that is known and trusted. Especially in bigger commercial companies it is considered quite an investment to start operating a new platform, and the fact that F# runs on the .NET platform is really going to make a difference there. As an extra bonus there is the huge set of supporting tools and libraries that programmers will have access to.

Since F# can be downloaded from Microsoft's website for free, all that is needed for a .NET programmer to give it a try is a good description of the language and the way it can be put to use. For most people it will be difficult to get going without something like a book they can hold in their hand, read on the train, put little slips into as bookmarks etc – just the online manuals simply aren't enough. "Foundations of F#" by Robert Pickering seems to be aimed at this category of users.

After a short introduction, the book dedicates three chapters on what could indeed be called the "foundations" of F#. The chapters describe the Functional, Imperative and Object Oriented aspects of the language in a fairly formal way. It even starts with a list of the keywords, which maybe a bit too fundamental for practical purposes. Fortunately the rest of these chapters find a good balance between concise description of the language, some examples, and some general comments on functional programming.

In general the book doesn't waste much time explaining concepts like variable scope, or why you might want to use a relational database. The information in its 385 pages is relevant for a professional programmer who understands the principal concepts of computer programming. At the same time it doesn't assume that the reader understands typical functional programming constructs like currying. As a consequence, the level is suited for experienced programmers who would like to start with a functional programming language.

The examples provide good bits of functional programming "jargon", but they can sometimes be a bit difficult to follow. The use of variable names that are sometimes rather short and cryptic doesn't help here. In a similar vein, I am not sure that it is such a good idea to use the ‘light’ notation throughout the book. This notation uses indentation to capture the structure of the program. This may be very convenient for a programmer who understands the language, but a novice might benefit from a bit more explicit structure.

Since F# runs on the Microsoft .NET platform, an important aspect of the language is the interfacing with this platform. The book describes, for example, how to create a dynamic web page with ASP.NET web forms, or how to access a database using ADO.NET. The array of topics that are covered in this way is rather large, which result in sections that can only provide an idea of what can be accomplished. It won’t really tell you what ADO.NET or ASP.NET is (or maybe just enough to understand what it is about), so in general you shouldn’t expect to be able to use this in practice without some prior experience with the .NET platform.

The book includes a section on “Language Oriented Programming”, an excellent idea (and one of the few sections after the first three that isn’t very .NET specific). The examples might have been more interesting, but never the less this section may provide valuable ideas for programmers new to functional programming.

A final word of caution: the book dates from 2007, and by now it is a bit behind current developments. This means that some of the examples need a bit of tweaking to work on the current version of the platform, and a few of the new features of F# (such as support for parallel programming) are not covered.

WILLEM DE JONG
w.a.de.jong@gmail.com