

Preface

I started learning functional programming with LISP, and then I learnt Standard ML. I used LISP for several years as my programming language for research, exploiting its functional programming characteristics, and Standard ML for teaching functional programming. This greatly influenced the way I program.

Programming languages are a matter of taste. I like functional programming, recursivity, and immutable objects. I think that this makes programming simpler and fun. In addition, I like LISP for its syntactic simplicity, both for programming and for structuring data. I used LISP regularly in the past, and although I am not using it regularly now, I still use it when I need to make a simple program quickly. I also consider Standard ML a nice language. I like its conciseness, its type inference system, and that it is simple to define algebraic data types.

Later I met other functional programming languages. Haskell, with lazy evaluation and lazy lists, and Scheme, with its continuations.

I have included in my personal list of programming languages Scala, which integrates functional programming into the object-oriented paradigm. It also permits the use of functional programming constructions in the context of big data, with its integration with Spark. In addition, it includes actors as one of its parallel mechanisms. Actors are a high level model that integrates well with object-oriented programming.

This book is an introduction to Scala from this functional programming perspective. The origin of this book is in the notes of a course on *Advanced Programming* we started in 2015–2016 at the University of Skövde. The course belongs to the Master on Data Science and focuses on functional programming using Scala.

I see Scala as an object-oriented programming language that supports rather effectively functional programming. This introduction presents Scala, focusing on this functional programming part. Nevertheless, in order to make concepts clear, and because it is a crucial part of the language, I also include some description of the object-oriented aspect of Scala.

This focus makes me ignore or give less importance to some other characteristics of the language. All programmers know that some problems can be solved from different perspectives. For example, we can define stacks (or lists) by means of an algebraic data type, but also by means of linked cells in memory with pointers. The former follows a functional programming style while the latter a more imperative style. In this book, stressing functional programming, I focus on the definition of abstract data types, recursion, and the like. Less importance is given to variables and to iteration. For a more imperative approach, [9] is a nice alternative.

In the same way, functions are first declared by means of `val`, and our discussion on method declaration `def` is deferred to a latter section. This differs from other books as e.g., [3], (a book that presents Scala as a functional language). For Scala from the object-oriented paradigm, you can consider [8].

The book is by no means a complete description of the language. That is, it does not provide information on all constructions and functions of the language. The Internet offers enough material on line for this. So, I have not written the book with this purpose. Its goal is to provide an introduction (a concise one) of the language from this functional programming perspective. Nevertheless, I believe that the book is self-contained and contains enough material to enable readers to use it to learn the language and eventually use it also as a reference. An index is included for this purpose.

In addition, this text having been prepared for a course on advanced programming and for master students, I discuss and compare Scala's approach with those of other languages. I think that it is good for any programmer, and naturally for any computer engineer, to know different languages and ways to tackle programming problems. It is well known that while some languages are better in some aspects, they are not the simplest for all purposes. In the book, I mention and compare Scala with e.g., Java, Standard ML (SML), and Prolog. The most detailed comparison is in the chapter devoted to algebraic data types. I consider that SML offers a simpler way to define them and this is explained in the text in some detail.

Organization of the Book

The book is divided into eight chapters. The first one is an introduction to functional programming, its main characteristics and languages. The second one presents the basics of the Scala language. The most important concepts seen there are the functions. We also give an overview of lists as well as other types of sequences. We introduce pattern matching. Chapter 3 presents lazy evaluation, which permits us to define infinite lists. At this point we introduce (Chapter 4) the main concepts and definitions related to Scala as an object-oriented programming language. We show how to define classes and methods. We also see traits and packages. Chapter 5 focuses on classes with polymorphic types. Functional programming tries to define functions as generally as possible. Because of that, polymorphism plays an important role. Chapter 6 focuses again on object-oriented aspects. The chapter explains how the object-oriented and the functional elements in Scala interact. We discuss tail-recursive functions that permit an efficient implementation (compilation) of recursive functions into an imperative language. Chapter 7 is devoted to algebraic data types. These types are characteristic of functional programming languages. We explain how to define them in Scala. We also compare their definition with the one offered by Standard ML. The book finishes in Chapter 8 with parallelism in Scala. We focus on two models: parallel collections and actors.

How to Use This book

We expect readers to be programmers using imperative/object-oriented languages. Knowledge of functional programming is not a prerequisite.

As I explained above, this book has been used in our course on advanced programming at the University of Skövde. The content has been used in 10 sessions of 2 hours each. We explained the main concepts (except Chapter 8) and did most of the exercises.

The book has been prepared with examples, exercises, and solutions to permit self-study. We have a web page for this book available under the following URL: <http://www.mdai.cat/scala>.

Programming and programming languages can only be learnt by doing. Therefore it is expected that readers install the language, test the examples, and program themselves in Scala.

Acknowledgments

My first acknowledgment goes to Ulises Cortés, who introduced me to the functional programming paradigm with the LISP programming language around 1990. Then, to the SAIL research group at the University of Skövde, in which I am integrated and which launched the Master on Data Science where this material has been used. Special thanks go to Elio Ventocilla, who read this material in its previous version and gave me useful comments. Last and not least, to the students of the master that used the first version of this material while I was producing it. All errors are, of course, my own.

August 2016

Vicenç Torra

Scala: From a Functional Programming Perspective
An Introduction to the Programming Language

Torra, V.

2016, XIII, 124 p. 7 illus., Softcover

ISBN: 978-3-319-46480-0