

The Fun of Programming

Edited by

Jeremy Gibbons and Oege de Moor



Contents

Preface	vii
1 Fun with binary heap trees	1
Chris Okasaki	
1.1 Binary heap trees	1
1.2 Maxiphobic heaps	4
1.3 Persistence	6
1.4 Round-robin heaps	7
1.5 Analysis of skew heaps	10
1.6 Lazy evaluation	12
1.7 Analysis of lazy skew heaps	15
1.8 Chapter notes	16
2 Specification-based testing with QuickCheck	17
Koen Claessen and John Hughes	
2.1 Introduction	17
2.2 Properties in QuickCheck	18
2.3 Example: Developing an abstract data type of queues	20
2.4 Quantifying over subsets of types	25
2.5 Test coverage	30
2.6 A larger case study	32
2.7 Conclusions	39
2.8 Acknowledgements	39
3 Origami programming	41
Jeremy Gibbons	
3.1 Introduction	41
3.2 Origami with lists: sorting	42
3.3 Origami by numbers: loops	49
3.4 Origami with trees: traversals	52
3.5 Other sorts of origami	56
3.6 Chapter notes	60

4	Describing and interpreting music in Haskell	61
	Paul Hudak	
4.1	Introduction	61
4.2	Representing music	61
4.3	Operations on musical structures	67
4.4	The meaning of music	70
4.5	Discussion	78
5	Mechanising fusion	79
	Ganesh Sittampalam and Oege de Moor	
5.1	Active source	79
5.2	Fusion, rewriting and matching	85
5.3	The MAG system	89
5.4	A substantial example	98
5.5	Difficulties	101
5.6	Chapter notes	103
6	How to write a financial contract	105
	Simon Peyton Jones and Jean-Marc Eber	
6.1	Introduction	105
6.2	Getting started	106
6.3	Building contracts	108
6.4	Valuation	116
6.5	Implementation	123
6.6	Operational semantics	127
6.7	Chapter notes	128
7	Functional images	131
	Conal Elliott	
7.1	Introduction	131
7.2	What is an image?	132
7.3	Colours	135
7.4	Pointwise lifting	137
7.5	Spatial transforms	139
7.6	Animation	141
7.7	Region algebra	142
7.8	Some polar transforms	144
7.9	Strange hybrids	147
7.10	Bitmaps	148
7.11	Chapter notes	150
8	Functional hardware description in Lava	151
	Koen Claessen, Mary Sheeran and Satnam Singh	
8.1	Introduction	151
8.2	Circuits in Lava	152
8.3	Recursion over lists	153

8.4	Connection patterns	155
8.5	Properties of circuits	157
8.6	Sequential circuits	160
8.7	Describing butterfly circuits	162
8.8	Batcher's mergers and sorters	166
8.9	Generating FPGA configurations	170
8.10	Chapter notes	175
9	Combinators for logic programming	177
	Michael Spivey and Silvija Seres	
9.1	Introduction	177
9.2	Lists of successes	178
9.3	Monads for searching	179
9.4	Filtering with conditions	182
9.5	Breadth-first search	184
9.6	Lifting programs to the monad level	187
9.7	Terms, substitutions and predicates	188
9.8	Combinators for logic programs	191
9.9	Recursive programs	193
10	Arrows and computation	201
	Ross Paterson	
10.1	Notions of computation	201
10.2	Special cases	208
10.3	Arrow notation	213
10.4	Examples	216
10.5	Chapter notes	222
11	A prettier printer	223
	Philip Wadler	
11.1	Introduction	223
11.2	A simple pretty printer	224
11.3	A pretty printer with alternative layouts	228
11.4	Improving efficiency	233
11.5	Examples	236
11.6	Chapter notes	238
11.7	Code	240
12	Fun with phantom types	245
	Ralf Hinze	
12.1	Introducing phantom types	245
12.2	Generic functions	248
12.3	Dynamic values	250
12.4	Generic traversals and queries	252
12.5	Normalisation by evaluation	255
12.6	Functional unparsing	257

vi

12.7	A type equality type	259
12.8	Chapter notes	262

Bibliography	263
---------------------	------------

Index	273
--------------	------------

Preface

Functional programming has come of age: it is now a standard course in any computer science curriculum. Ideas that were first developed in the laboratory environment of functional programming have proved their values in wider settings, such as generic Java and XML. The time is ripe, therefore, to teach a second course on functional programming, delving deeper into the subject. This book is the text for such a course.

The emphasis is on the fun of programming in a modern, well designed programming language such as Haskell. There are chapters that focus on applications, in particular pretty printing, musical composition, hardware description, and graphical design. These applications are interspersed with chapters on techniques, such as the design of efficient data structures, interpreters for other languages, program testing and optimisation. These topics are of interest to every aspiring programmer, not just to those who choose to work in a functional language. Haskell just happens to be a very convenient vehicle for expressing the ideas, and the theme of functional programming as a *lingua franca* to communicate ideas runs throughout the book.

The prerequisites for this material are covered in any introductory course on functional programming. In fact, it is a seamless sequel to courses that are based on *An introduction to functional programming using Haskell* by Richard Bird [15]. Throughout the text, references to that book are made by the abbreviation 'IFPH'. The present volume could also be used as a sequel to other introductory books, however. All that is expected of the reader is a working knowledge of higher-order functions and polymorphism, and an understanding of lazy evaluation. Many of the chapters in this book are accompanied by software, which can be found on the website

<http://web.comlab.ox.ac.uk/oucl/publications/books/fop>

As the book is adopted for courses by others, we shall be happy to add links to further teaching materials.

This book was produced to celebrate the work of Richard S. Bird on his sixtieth birthday. For many years, Richard has led the development of functional programming, in particular in the area of synthesising programs from

specifications. Apart from these research contributions, he educated many generations of programmers through his textbooks. When the question of a festschrift came up, it was immediately evident that it should be a textbook of lasting value, written by his friends in the research community. Above all, we hope it conveys Richard's sense of fun in the subject, which has delighted us all. For this reason we have borrowed the title of one of Richard's own lectures: the *Fun of Programming*. Many happy returns, Richard!

We would like to thank our editors at Palgrave, Tracey Alcock, Esther Thackeray and Rebecca Mashayekh, for their efficient help in the production of this book. Andres Löh gave sterling help with thorough last-minute reviewing.

*Jeremy Gibbons
Oege de Moor
Oxford, September 2002*