Bibliography

This is an introductory book which draws on a wide range of sources. Most of the material here is covered in other texts though often with different perspectives and emphases. This bibliography is not exhaustive: rather, it offers a broad cross section of source and supplementary material. References are almost entirely to books on the grounds that these tend to be more discursive than academic papers.

Chapter 1 - Introduction

Backus'[3] influential paper contains a critique of von Neumann computing and arguments for functional programming.

Brief motivational material on functional programming is contained in Sadler and Eisenbach[48], Glaser, Hankin and Till[21], Henderson[30] and Henson[31].

Brady[9] provides an accessable introduction to the theory of computing. There are further accounts in a wide variety of mathematical logic texts, of which Kleene's[36] and Mendelson's[38] are still outstanding.

For related general computing topics not considered further in this book: the denotational approach to programming language semantics is covered by Gordon[22], Schmidt[49], and Stoy[51]; program specification is covered by Cohen, Harwood and Jackson[15], Gehani & McGettrick[20], Hayes[29], Jones[33], and more formally by Turski & Maibaum[54]; program verification is covered by Backhouse[2], Gries[23], Manna[37], and less formally though thoroughly by Bornat[7].

For related functional programming and language topics not considered further in this book: Eisenbach[8, 16, 24] and Glaser, Hankin and Till[21] contain overviews of implementation techniques; SECD machine implementations are discussed by Brady[9], Burge [10], Field and Harrison[18], Henderson[30], Henson[31], and Wegner[55]; combinators and graph reduction are covered thoroughly by Field and Harrison[18] and by Peyton-Jones[43], and, in less detail, by Henson[31] and Revesz[46]; Bird and Wadler[6] cover verification; Field and Harrison[18] cover program transformation and abstract interpretation; Henson[31] covers verification and transformation; Harrison and Khoshnevisan[27] also discuss transformation.

For functional languages not considered further in this book: Field and Harrison[18] is based on Hope and also contains brief discussion of Miranda, Lisp and FP; Peyton-Jones[43] is based on Miranda and contains an introduction by Turner[52]; Bailey[4] covers Hope; Harrison and Khoshnevisan[28] cover FP; Henson[31] covers FP; Glaser, Hankin and Till[21] discuss briefly FP, Hope and KRC; Bird and Wadler contains a brief appendix on Miranda[6]; Revesz[46] discusses briefly FP and Miranda; SASL is covered by Turner[53].

For various imperative languages mentioned here: ALGOL 60[40]; Algol 68[41]; BCPL[47]; C[35]; Pascal[19]; POP-2[11]; Prolog[14] and PS-algol[12]

Chapter 2 - λ calculus

Church's[13] description of the λ calculus is much referenced but little read. Barendregt[5] is the standard reference. Hindley and Seldin[32] is less detailed.

Early descriptions from a computing perspective include Burge[10], and Wegner[55]. Field and Harrison[18], Peyton-Jones[43] and Revesz[46] provide thorough contemporary accounts, as does Stoy[51] though oriented to semantics. Glaser, Hankin and Till[21], and Henson[31] also describe λ calculus.

Pair functions are discussed by Barendregt[5], Field and Harrison[18], Glaser, Hankin and Till[21], Henson[31], Revesz[46] and Wegner[55].

Chapter 3 - Conditions, booleans and integers

Burge[10], Field and Harrison[18], Glaser, Hankin and Till[21], Henson[31], Revesz[46] and Wegner[55] all have accounts of aspects of the material on conditional expressions and booleans. Schmidt[49] and Stoy[51] cover it in exercises.

There are many approaches to representing numbers. The approach here is discussed by Barendregt[5], and Glaser, Hankin and Till[21], and also by Revesz[46] in an exercise. Field and Harrison[18] and Wegner[55] discuss variants. Church's representation is discussed by Barendregt[5], Burge[10], Henson[31], Revesz[46], and Wegner[55].

Chapter 4 - Recursion

Barendregt[5], Brady[9], Glaser, Hankin and Till[21], Henson[31], Peyton-Jones[43], Revesz[46] and Stoy[51] all discuss the derivation of the 'recursion' function. Field and Harrison[18] present it. Schmidt[49] presents it in an exercise. Burge[10] discusses it in terms of combinators.

Kleene[36] Mendelson[38], Peter[42], and Rayward-Smith[45] provide accounts of the construction of arithmetic and comparison operations within recursive function theory. Brief computing oriented accounts are in Burge[10] and Glaser, Hankin and Till[21].

Chapter 5 - Types

The approach to types considered here is a λ calculus implementation of run-time typing with tags. Abelson and Sussman[1] discuss a related Scheme approach to manifest types, and Henderson[30] and Peyton-Jones[43] discuss implementations of typing.

Field and Harrison[18] and Peyton-Jones[43] contain thorough accounts of type checking.

Chapter 6 - Lists and strings

Bird and Wadler[6], Henderson[30], Henson[31] and Revesz[46] contain thorough accounts of lists from a functional programming perspective.

Numerous books on LISP contain material on list processing and mapping functions. Wilensky[57] provides an accessable introduction.

Chapter 7 - Composite values and trees

The use of lists to represent composite values is discussed implicitly in numerous books on LISP.

Henderson[30] and Queinnec[44] discuss accumulation variables.

Abelson and Sussman[1] and Shapiro[50] discuss the list representation of binary trees in Scheme and LISP respectively.

Bird and Wadler[6] provides a thorough account of trees from a functional language perspective.

Chapter 8 - Evaluation

Barendregt[5] and Hindley and Seldin[32] contain formal details of reduction. There are more accessable approaches in Brady[9], Burge[10], Field and Harrison[18], Glaser, Hankin and Till[21], Henson[31], Peyton-Jones[43], Revesz[46], Stoy[51], and Wegner[55]. Note that Barendregt, Burge, Field and Harrison, Hindley and Seldin, and Wegner only name one Church-Rosser theorem.

Bird and Wadler[6] discuss evaluation models, time and space efficiency and algorithm design techniques.

Brady[9] and Minsky[39] contain details of the halting problem for Turing machines.

Abelson and Sussman[1], Bird and Wadler[6], Field and Harrison[18], Glaser, Hankin and Till[21], Henderson[30], Henson[31], Peyton-Jones[43] and Revesz[46] all discuss aspects of lazy evaluation.

Bird and Wadler[6] provides a thorough account of infinite lists.

Chapter 9 - Functional programming in Standard ML

Wikstrom[56] provides thorough coverage of SML programming. Harper, MacQueen and Milner[25] describes SML informally and gives details of I/O and modules. (The informal description and the I/O details are duplicated in Wikstrom). Harper, Milner and Tofte[26] is a first version of a formal semantics for SML.

Bird and Wadler[6] provide much additional material which is relevant to SML programming including discussion of concrete and abstract types.

Chapter 10 - Functional programming and LISP

Steele[34] is the COMMON LISP 'bible'. Wilensky[57] is an good introduction. Winston and Horn[58] take an Artificial Intelligence perspective.

Abelson and Sussman[1] is the standard reference for SCHEME and is based around an introductory computer science course. Dybvig[17] is a more traditional programming language text.

References

1. H. Abelson, G. J. Sussman, and J. Sussman, *Structure and Interpretation of Computer Programs*, MIT Press, Cambridge, Massachusetts, (1985).

2. R. C. Backhouse, *Program Construction and Verification*, Prentice-Hall, Englewood Cliffs, N.J., (1986).

3. J. W. Backus, "Can programming be liberated from the von Neumann style? A functional style and its algebra of programs." *Communications of the ACM*, **Vol. 21**, (8), pp. 613-641, (August 1978).

4. R. Bailey, "An Introduction to Hope," in *Functional Programming: Languages, Tools & Architectures*, ed. S. Eisenbach, Ellis Horwood, Chichester, (1987).

5. H. P. Barendregt, *The Lambda Calculus: Its Syntax and Semantics*, North-Holland, Amsterdam, (1981).

6. R. Bird and P. Wadler, *Introduction to Functional Programming*, Prentice-Hall, New York, (1988).

7. R. Bornat, *Programming from First Principles*, Prentice-Hall, Englewood Cliffs N.J., (1987).

8. B. Boutel, "Combinators as Machine Code for Implementing Functional Languages," in *Functional Programming: Languages, Tools & Architectures*, ed. S. Eisenbach, Ellis Horwood, Chichester, (1987).

9. J. M. Brady, *The Theory of Computer Science: A Programming Approach*, Chapman and Hall, London, (1977).

10. W. H. Burge, *Recursive Programming Techniques*, Addison-Wesley, Reading, (1975).

11. R. M. Burstall, J. S. Collins, and R. J. Popplestone, *Programming in POP-2*, Edinburgh University Press, Edinburgh, (1977).

12. R. Carrick, J. Cole, and R. Morrison, "An Introduction to PS-algol Programming," PPRR 31, Dept. of Computational Science, University of St Andrews, St Andrews, Scotland, (1986).

13. A. Church, *The Calculi of Lambda Conversion*, Princeton University Press, Princeton, N.J., (1941).

14. W. F. Clocksin and C. S. Mellish, *Programming in Prolog*, Springer-Verlag, Berlin, (1981).

15. B. Cohen, W. T. Harwood, and M. I. Jackson, *The Specification of Complex Systems*, Addison-Wesley, Wokingham, (1986).

16. M. Cripps, T. Field, and M. Reeve, "An Introduction to ALICE: a Multiprocessor Graph Reduction Machine," in *Functional Programming: Languages, Tools & Architectures*, ed. S. Eisenbach, Ellis Horwood, Chichester, (1987).

17. R. K. Dybvig, *The SCHEME Programming Language*, Prentice-Hall, Englewood Cliffs, New Jersey, (1987).

18. A. Field and P. Harrison, *Functional Programming*, Addison-Wesley, Wokingham, (1988).

19. W. Findlay and D. A. Watt, *Pascal: An Introduction to Methodical Programming*, Pitman, London, (1981).

20. N. Gehani and A. D. McGettrick, *Software Specification Techniques*, Addison-Wesley, Wokingham, (1986).

21. H. Glaser, C. Hankin, and D. Till, *Principles of Functional Programming*, Prentice-Hall, Englewood-Cliffs, N.J., (1984).

22. M. J. C. Gordon, *The Denotational Description of Programming Languages: An Introduction*, Springer-Verlag, New York, (1979).

23. D. Gries, *The Science of Programming*, Springer-Verlag, New York, (1981).

24. C. Hankin, D. Till, and H. Glaser, "Applicative Languages and Data Flow," in *Functional Programming: Languages, Tools & Architectures*, ed. S. Eisenbach, Ellis Horwood, Chichester, (1987).

25. R. Harper, D. MacQueen, and R. Milner, "Standard ML," ECS-LFCS-86-2, LFCS, Dept. of Computer Science, University of Edinburgh, Edinburgh, Scotland, (March 1986).

26. R. Harper, R. Milner, and M. Tofte, "The Semantics of Standard ML, Version 1," ECS-LFCS-87-36, LFCS, Dept. of Computer Science, University of Edinburgh, Edinburgh, Scotland, (August 1987).

27. P. Harrison and H. Khoshnevisan, "A Functional Algebra and its Application to Program Transformation," in *Functional Programming: Languages, Tools & Architectures*, ed. S. Eisenbach, Ellis Horwood, Chichester, (1987).

28. P. Harrison and H. Khoshnevisan, "An Introduction to FP and the FP Style of Programming," in *Functional Programming: Languages, Tools & Architectures*, ed. S. Eisenbach, Ellis Horwood, Chichester, (1987).

29. I. Hayes, *Specification Case Studies*, Prentice-Hall, Englewood Cliffs, N.J., (1987).

30. P. Henderson, *Functional Programming: Application and Implementation*, Prentice-Hall, Englewood Cliffs, N.J., (1980).

31. M. C. Henson, *Elements of Functional Languages*, Blackwell, Oxford, (1987).

32. J. R. Hindley and J. P. Seldin, *Introduction to Combinators and λ Calculus*, Cambridge University Press, Cambridge, England, (1987).

33. C. B. Jones, *Systematic Software Development Using VDM*, Prentice-Hall, Englewood Cliffs, N.J., (1986).

34. G. L. Steele Jr., *Common LISP: The Language*, Digital, (1984).

35. B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, Englewood Cliffs, N.J., (1978).

36. S. C. Kleene, *Introduction to Metamathematics*, North-Holland, Amsterdam, (1952).

37. Z. Manna, *Mathematical theory of Computation*, McGraw-Hill, New York, (1974).

38. E. Mendelson, *Introduction to Mathematical Logic*, D. Van Nostrand, Princeton, New Jersey, (1964).

39. M. Minsky, *Computation: Finite and Infinite Machines*, Prentice-Hall, London, (1972).

40. P. Naur and et. al., "Revised Report on the Algorithmic Language ALGOL 60," *Communications of the ACM*, **Vol. 6**, (1), (1963).

41. F. G. Pagan, *A Practical Guide to Algol68*, Wiley, London, (1976).

42. R. Peter, *Recursive Functions*, Academic Press, New York, (1967).

43. S. L. Peyton-Jones, *The Implementation of Functional Programming Languages*, Prentice-Hall, Englewood Cliffs, N.J., (1987).

44. C. Queinnec, *LISP*, Macmillan, London, (1983).

45. V. J. Rayward-Smith, *A First Course in Computability*, Blackwell, Oxford, (1986).

46. G. Revesz, *Lambda-Calculus, Combinators and Functional Programming*, Cambridge University Press, Cambridge, (1988).

47. M. Richards, "C. Whitby-Strevens," *BCPL - The Language and its Compiler*, Cambridge University Press, Cambridge, (1982).

48. C. Sadler and S. Eisenbach, "Why Functional Programming?" in *Functional Programming: Languages, Tools & Architectures*, ed. S. Eisenbach, Ellis Horwood, Chichester, (1987).

49. D. A. Schmidt, *Denotational Semantics: A Methodology for Language Development*, Allyn and Bacon, Boston, (1986).

50. S. C. Shapiro, *LISP: An Interactive Approach*, Computer Science Press, Rockville, Maryland, (1986).

51. J. E. Stoy, *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*, MIT Press, Cambridge, Massachussets, (1977).

52. D. Turner, *An Introduction to Miranda*. (in Peyton-Jones 1987)

53. D. Turner, *SASL Language Manual*, Dept. of Computational Science, University of St Andrews, St Andrews, Scotland, (December 1976).

54. W. M. Turski and T. S. E. Maibaum, *The Specification of Computer Programs*, Addison-Wesley, Wokingham, (1987).

55. P. Wegner, *Programming Languages, Information Structures and Machine Organisation*, McGraw-Hill, London, (1971).

56. A. Wikstrom, *Functional Programming Using Standard ML*, Prentice-Hall, London, (1987).

57. R. Wilensky, *Common LISPcraft*, Norton, New York, (1986).

58. P. H. Winston and B. K. P. Horn, *LISP*, Addison-Wesley, Reading, (1984).