https://doi.org/10.1007/s00165-021-00561-4
The Author(s) © 2021
Formal Aspects of Computing (2021) 33: 1279–1280

Formal Aspects of Computing

Book Review

Foundations of programming languages

by Kent D. Lee
Second Edition. Undergraduate Topics in Computer Science, Springer 2017, ISBN 978-3-319-70789-1, pp. 1–367

Reinhard Wilhelm[1]

[1] Saarland University, Saarland Informatics Campus, Campus E1 3, Room 405, 66123 Saarbrücken, Germany

I must confess that I have co-authored a book bearing the same title [LMW89] and the first compiler-design book treating the same three language paradigms as the book to be reviewed [WS10]. Thus, I might appear a bit preoccupied, in particular since both books are not cited.

The purpose of this book, according to the author, is *to introduce you to three styles of programming languages by using them to implement a non-trivial programming language*. He starts to realize this purpose by providing Chapter 2 on Syntax including bits on grammars and automata and on parsing and lexing tools. Chapter 3 introduces a virtual machine, JCoCo, tailored towards implementing Python, and shows how a non-trivial subset of Python is translated into JCoCo bytecode. Chapter 4 describes an implementation of JCoCo in Java, intended to teach essentials of object-oriented programming. Chapter 5 introduces functional programming with a short excursion into the lambda-calculus, normal forms, and reduction orders. SML is used as example of a functional language. The author mixes up referential transparency with a case of non-terminating recursion. SML is explained going through the language features and by implementing a prefix calculator. Chapter 6 describes an SML compiler by giving bytecode sequences for SML language constructs in the same way it was done in Chapter 3. Chapter 7 introduces Prolog. Defining the important notion of unification to be *simply a list of substitutions for variables* is only half the truth. As applications of Prolog the author shows how to do parsing and type inference in Prolog. All chapters are accompanied by examples and exercises with solutions.

The strengths of the book lie in the massive amount of quite interesting and relevant material in Chapters 2 to 8. The weakness of the book is the colloquial and often imprecise writing style. The author is not only imprecise and sometimes plainly wrong, he fails to assign the right weight to concepts of different relevance. For example in Section 1.4.3, which introduces virtual machines. He gives the same weight to the concept of compilation of source code to the intermediate code of a virtual machine, the organizational issue of whether the generated virtual-machine code is externally stored like in Java or *internally buried* like in Python, and the facts that the Java compiler is called *javac* and the names of bytecode files end in an *.class* extension.

The book has a lot of colorful graphics to explain concepts. However, there is no legend defining the semantics of the graphics elements, e.g. arrows. In fact, they don't have a consistent meaning reducing the explanatory value of the figures.

Another complaint concerns the author's treatment of the history of computing. His knowledge appears to be quite cursory, and he doesn't seem to have spent much effort on improving this. He surprises the reader by viewing the Norwegian mathematicians Abel and Lie as forefathers of Computer Science. Many others,

Correspondence to: R. Wilhelm, e-mail: wilhelm@cs.uni-saarland.de

mathematicians, logicians, philosophers could serve this role better. What about Ramon Lull and Gottfried-Wilhelm Leibniz? Why doesn't he mention Ada Lovelace as the first programmer? Austrian-American logician Kurt Gödel is missing in his treatment of undecidability. Konrad Zuse is not mentioned as implementor of the first programmable, fully automatic digital computer. The first relevant machine in the class of stack-based architectures, the Borroughs B5000, an Algol60-machine, introduced in 1961, is not mentioned. Instead, he assigns to Hewlett Packard mainframes, said to have appeared in the 1960s, this pioneering role. However, the HP3000 was only introduced in 1972. The invention of object-oriented languages is first attributed to Wirth and Stroustrup. Fortunately, some pages later, SIMULA 67 and their designers Ole-Johann Dahl and Kristen Nygaard are given some credit. At the end of the book there is a short bibliography. Five of the fifteen entries point to sources for photographs of programming-language pioneers, one to an interview and one to a press release. An enthusiastic reader is left alone on his search for further reading material.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

[LMW89] Jacques Loeckx, Kurt Mehlhorn, Reinhard Wilhelm (1989) Foundations of Programming Languages. Wiley, London
[WS10] Reinhard Wilhelm, Helmut Seidl (2010) Compiler Design—Virtual Machines. Springer, Berlin

Accepted 24 July 2021
Published online 16 August 2021