he wishes to make which do not fall into an easily defined scheme. However there is plenty of practical advice and information, on questions which many books choose to ignore. Wexler is certainly not afraid to give pragmatic do's and don’ts derived from his experience with a real system.

Overall this is one of the better occam 2 books. It would be of most benefit to those students who are intimidated by a more formal treatment. In fact the only thing which is intimidating about it is the price.

Peter C. CAPON
Department of Computer Science
University of Manchester
Manchester, UK


According to its preface, this book is intended for a first course in program verification systems, and consistent and complementary definitions of programming languages. We may be glad that the emphasis on program correctness in the USA is growing. Unfortunately, there are reasons to be sceptical about program verification. At the end of the introduction, the author himself states modestly: “I do not believe . . . that large-scale program verification . . . is practical in the near future.”

The idea of program verification is to verify the program at the moment it has been completed. This is far too late. In programming methodology, it is nowadays well known that programs are derived; the program and its proof are developed hand in hand, and usually the proof is slightly ahead. Of course, this proof is a human product. Therefore, mechanical assistance may be necessary to certify the steps, to support the development process, to preserve consistency, to provide version control. Such mechanical assistance requires a profound knowledge of programming logics, but programming logics of a certain kind, namely those that are most adequate for human program derivation. This leads to the conclusion that a book on program verification must have the same semantical basis as books on programming methodology. Of course, the rules must be sound, but the builder or user of a verification system need not be aware of the soundness proof of certain well-known rules. Indeed, a fairly high level of abstraction seems a prerequisite for the effective application of a programming methodology. For this reason, it would be highly inadequate to treat total correctness of while-programs by translation to flowchart programs.

This is what happens in this book, however. After 50 pages of mathematical preliminaries and 40 pages on partial correctness of while-programs, we get 80 pages on total correctness of flowchart programs. Then total correctness of while-programs
is treated in 90 pages, using, among other concepts, a translation into flowchart programs. Finally, total correctness of tail-recursive procedures is treated in 50 pages by means of a translation into **while**-programs.

Of course, the soundness proof of the rules for total correctness of **while**-programs with respect to the operational semantics is not trivial and, in constructing a proof, one might be tempted to use a translation into flowchart programs. The treatment in the book, however, seems to convey a different message: in order to really understand **while**-programs we must understand flowchart programs first.

When the author occasionally enters the field of programming methodology (e.g. in Section 1.5), he looks for an invariant after the repetition (guard and body) has been written. By now, it should be known that one obtains better programs and easier proofs if one begins by looking for an invariant and uses the invariant to obtain the repetition.

In the chapters on total correctness of **while**-programs, the author uses Hoare triples, translation into flowchart programs, operational semantics and natural deduction. It seems to be all sound and possibly complete, but it leads to an inefficient way of reasoning. The chapter on tail-recursive procedures uses denotational semantics, cpo's and least fixpoints. Since the semantics is completely deterministic, a procedure specification must contain a complete list of the values of the output variables in terms of the values of the input variables. This seems an undesirable restriction in real programming.

Despite all its formality, the book shows a certain lack of abstraction. For instance, Exercise 13 of Chapter 4 reads: “Modify the syntax, operation semantics, and Hoare axiomatization of this chapter to permit the use of more than one stack.”

The treatment of the subject is sound, but this approach should not be used anymore. We must hope for better books to sell the ideas of program correctness.

Wim H. Hesselink

*University of Groningen
Groningen, Netherlands*

---


This massive (over 650 pages) book has been compiled to commemorate the 50th Anniversary of the publication of Alan Turing’s paper, “On Computable Numbers, with an Application to the Entscheidungsproblem.” In words of Rolf Herken, the volume’s editor,

... it presented the invention of the *universal* Turing machine. It is the publication of this idea that will presumably be acknowledged as marking sub specie aeternitatis the beginning of the “computer age.”
She easily gets whatever she wants, no matter whether it’s a small wish like a new dress or a more ambitious goal. Marina is 7 years older than me. Right after graduation she became a lecturer at the local university and began writing a PhD thesis. Every wish has its time frame and its way of realization. There is no need to hurry; sooner or later your wish will definitely come true. Pay attention to everything that happens in your life and don’t waste your chance to realize your dreams when it comes into your life. Live right here and right now. Enjoy your life and believe that one day your most sacred wish is going to come true. However, after this episode I received so many flowers that I didn’t even have enough vases for them. At first I didn’t understand the situation at all. he wishes to make which do not fall into an easily defined scheme. However there is plenty of practical advice and information, on questions which many books choose to ignore. Wexler is certainly not afraid to give pragmatic dos and don’ts derived from his experience with a real system. Overall this is one of the better occam 2 books. It would be of most benefit to those students who are intimidated by a more formal treatment. In fact the only thing which is intimidating about it is the price. Peter C. CAPON. Department of Computer Science. University of Manchester Manchester, UK. Programming C He regrets the fact that he can no longer compete. D He wishes more people were involved in the sport. 15 What would be a good introduction to this article? A. For each question, choose the correct answer. There are three extra sentences which you do not need to use. A new life. I used to work as a college lecturer in the north of England, running photography courses. It wasn’t a bad job and I really liked my students, but I began to feel tired of doing the same thing every day. 16 x xx. I’d always loved travelling, so one weekend I typed “international volunteering” into an internet search engine. At the top of the results page was the opportunity to go and stay on an island in the Indian Ocean, thousands of miles away, and help to protect the bea
The questions which must be answered are not easy ones. Verbs which are not normally used in the continuous, e.g. be, own, possess can be used in participle phrases: X Yolanda is rich and is owning six homes. Y Yolanda is rich, owning six homes. 1B. Participle subjects. In many cases, participle phrases do not contain a subject. The subject of the main clause (in bold below) usually acts as the subject of the participle phrase: Moaning with pain, the victim was examined by a young doctor. 2 Participle phrases are an efficient way of giving more information about a noun and can often replace a defining relative clause. Compare these examples: The man is very noisy. Verbs 12 Question tags after negative statements 13 Question tags after affirmative statements 14 Question tags: mixed 15 Auxiliaries followed by full or bare infinitive 16 Auxiliaries: mixed 17 have: possessive 18 have: various uses 19 The have + object + past participle construction 20 be 21 it is/there is 22 can and be able 23 may 24 must and have to 25 must. not and need not 26 need not and don't have to etc. 27 must, can't and needn't with the perfect infinitive. 8. into a dustbowl. 31 . . . people think that . . . lead is . . . heaviest metal, but . . . gold is heavier. 32 Our air hostess said, ' . . . rack is only for . . . light articles. . . . heavy things such as . . . bottles must be put on.