with basically no references in the text, but it does contain a good bibliography, especially of Russian references with some annotation. There are no exercises or problems.

JULIAN D. COLE
RENSSELAER POLYTECHNIC INSTITUTE
E-mail address: colej@rpi.edu

BULLETIN (New Series) OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 29, Number 2, October 1993

0273-0979/93 $1.00 + $.25 per page

Complexity theory of real functions, by Ker-I Ko. Birkhäuser, Basel, 1991, viii+307 pp., $49.50. ISBN 0-8176-3586-6

A striking aspect of twentieth century mathematics is the emergence of new fields of specialization motivated by the computer. The book under review may be considered in this light. It deals with a specialized topic in theoretical computer science and is written by a computer scientist. However, the mathematical roots of the subject extend back to the pioneering work on computability of the 1930s. Research in this area is still going on today.

We begin with a brief historical account. In the 1930s mathematicians became interested in studying computability in its widest sense. There were many reasons for this, some of which were implicit in the work of Kurt Gödel. For example, mathematicians wanted to understand the notion of a *mathematical proof*. Now a mathematical proof, if written down in detail, is *mechanically checkable*. Thus a definition of the most general type of mechanically checkable procedure was desirable. This led to a study of computability.

In 1936 Alan Turing defined the Turing machine, a digital computer of a very general type [6]. This gave rise to a definition of computability for functions mapping the nonnegative integers N into N. Such a function is computable if its values can be calculated by one of the machines described in [6]. Turing went further and proved the existence of a universal Turing machine—a digital computer which can simulate all others. It is worth noting that the digital computers of today are Turing machines with limited storage capacity. So it is not surprising that a study of computability based on the Turing machine is of interest to computer scientists.

At about the same time, other definitions—by Post, Herbrand/Gödel, Church, etc.—appeared. Several of these were useful in solving problems in computer science. (For example, the Post production, a rule for manipulating strings of symbols, played a role in the theory of programming languages, the work of Noam Chomsky on the grammatical analysis of language, and other topics.) A seminal result was that all of the definitions are equivalent: the functions so defined are called recursive functions. The equivalence provided evidence that the definition of computability which emerged is a good one. (There is additional evidence, which we will not discuss here.) As a consequence of the work on computability, mathematicians obtained a definition of the most general type of proof procedure, and the theory of recursive functions became the theoretical

foundation for computer science. Once again in the history of mathematics, the highly theoretical and the practical are seen in juxtaposition.

The introduction of computability to analysis came much later. In the 1950s Grzegorczyk and Lacombe gave a definition of computability for functions of a real (or complex) variable [1–4]. It turned out that all of these functions are continuous. So the definition does not cover computability for many basic functions of analysis—e.g., the elements of $L^p[0, 1]$ which are step functions with rational jump points and rational values, etc. Nonetheless the work of Grzegorczyk and Lacombe is of considerable importance in computable analysis. A study of computability in a more general setting may be found in Pour-El/Richards [5].

We now turn to the book under review. In doing so, we shift our attention from mathematics to computer science. Ko's work is concerned not merely with computability but with "feasible computability". Although there are many ways to define feasibility, polynomial time computability is of particular interest to the computer scientist. Thus a recursive function f is polynomial time computable ($f \in \mathbb{P}$) if there exists a Turing machine M and a polynomial P_f so that, for each argument n, the machine computes $f(n)$ in at most $P_f(|n|)$ steps. (Here $|n|$ is the length of the input n in a suitable encoding of the nonnegative integers.) Computer scientists also define the term "nondeterministic Turing machine". Denote by $\mathbb{NP}$ the set of recursive functions which can be computed by a nondeterministic Turing machine in polynomial time. One of the most famous open problems in theoretical computer science is the $\mathbb{P} = \mathbb{NP}$ problem: Do the classes $\mathbb{P}$ and $\mathbb{NP}$ coincide? Many well-known combinatorial problems are known to be in $\mathbb{NP}$. The question is whether these problems can be solved in polynomial time by an ordinary Turing machine.

The purpose of Ko's book is to inject polynomial time computability into computable analysis. Ko's definitions are reminiscent of those of Grzegorczyk and Lacombe mentioned earlier. (Thus his "polynomial time computable functions of a real variable" are also continuous.) Since a real number is associated with a sequence of rationals which converges to it, the real becomes a type-1 function. This function maps the nonnegative integers (type-0) into the rationals (also type-0). A function of a real variable becomes a type-2 operator, mapping type-1 functions into type-1 functions. Operations on functions of a real variable become type-3 operators. These definitions were difficult to work with in the 1950s, and they remain so when polynomial time computability is added.

Many of Ko's results are related to conjectures associated with polynomial time computability. For example:

Theorem. *Let*

$$f\colon [0, 1]^2 \to R,$$

$$g\colon g(x) = \max\{f(x, y)\colon 0 \le y \le 1\}.$$

Define the operator Max *by* $\mathrm{Max}(f) = g$. *Then* $P = NP$ *if and only if, for all polynomial-time computable real functions* f, $\mathrm{Max}(f)$ *is polynomial-time computable.*

In addition to maximization, the book has results on integration, differentiation, approximation of functions of a real variable by polynomials, and a variety of other topics. As the author makes quite clear, some of this material is a straightforward adaptation of results and proofs in computable analysis to polynomial time computability (cf. [5]).

We have already remarked that Ko deals with polynomial time computability, whereas Pour-El and Richards [5] investigate computability. There is another difference, a difference in scope. Throughout his book Ko remains concerned with continuous functions of a real variable. Beginning with Chapter 2, Pour-El and Richards deal with computability on a Banach space—which leads them to wave propagation, heat dissipation, computation of eigenvalues and other topics from classical analysis, functional analysis, and physical theory.

For a mathematician there is one disconcerting aspect of Ko's book. There is a proliferation of definitions for a single concept. (This is not the case when dealing with computable analysis.) Consider, for example, the treatment of the Weierstrass approximation theorem. Ko gives three inequivalent definitions for "polynomial time computable sequence of polynomials"—definitions 6.6(b), 8.1(a), and 8.1(b). He dismisses 6.6(b) and proves that the Weierstrass approximation theorem holds for definition 8.1(b) but not for 8.1(a). It would be nice to know whether any of these definitions are useful in solving problems in computer science.

Polynomial time computability is one possible definition for feasible computability. There are others. In the future, some of these may be scrapped and others may be added. It is not clear what role polynomial time computability will eventually play in the emerging discipline of computer science. We will have to wait and see.

References

1. A. Grzegorczyk, *Computable functionals*, Fund. Math. **42** (1955), 168–202.
2. ———, *On the definitions of computable real continuous functions*, Fund. Math. **44** (1957), 61–71.
3. D. Lacombe, *Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles*, I, C. R. Acad. Sci. Paris Sér. I Math. **240** (1955), 2478–2480.
4. ———, *Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles.*, II, III, C. R. Acad. Sci. Paris Sér. I Math. **241** (1955), 13–14, 151–153.
5. M. B. Pour-El and I. Richards, *Computability in analysis and physics*, Springer-Verlag, New York, 1989.
6. A. M. Turing, *On computable numbers with an application to the "Entscheidungsproblem"*, Proc. London Math. Soc. (2) **42** (1936), 230–265; corr. ibid **43** (1937) 544–546.

Marian Boykan Pour-El
School of Mathematics, University of Minnesota
E-mail address: dept.@math.umn.edu