PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 40, NO. 2, PP. 155–156 (1996)

BOOK REVIEW

James S. Royer and John Case:

Subrecursive Programming Systems
Complexity & Succinctness

Progress in Theoretical Computer Science Series,
Birkhäuser Boston 1994

This review has to address basically the following three questions: for whom this book is written, what it is about and how I liked it. However, I must be most careful with the latter, as I am obviously *not* an ideal reader of this work. By an ideal reader here I mean a researcher or a graduate student in the mathematical theory of computation. Alternatively, it is possible to read this book with *some* background plus strong motivation, but without being familiar with recent developments in the field, it is difficult to appreciate the subtleties of the text.

The general subject of the work of Royer and Case is programming systems, which is an abstraction for programming languages; with highlight focused on subrecursive programming systems. These are, roughly said, such programming languages for which the result of running any given program any given input can be determined algorithmically. Obviously not every programming system is subrecursive; it is a well-known result that given an arbitrary program and its input, it is algorithmically undecidable wether the pro even halts. (The case is somewhat different if we equip our programs with built-in clocks that bring them to a halt anyway; indeed, clocking is widely used throughout the book as a means to obtain subrecursive programming systems.) In the introductory chapter the authors present several examples of subrecursive programming systems to point out that many important theoretical models and most existing programming languages fall into this class.

The work is divided into two major parts. Part I of the book develops a toolkit for work with subrecursive programming systems. The discussion starts with the very basics, Turing Machines and general complexity theory results. Even so, the text is by no means elementary, nor is it a first course in computation theory. Such a departure, however, has the advantage of making the monograph self-contained and allows the authors to develop an appropriate system of notations and basic definitions for their later studies. For example, they give a detailed treatment of clocked programming systems, a concept which was in wide use prior to this work, yet never formally defined.[1] The authors then demonstrate the power of their toolkit in alternative proofs of some standard prior results, before they embark in Part II on their study of pro succinctness tradeoffs.

[1]It always involves some level of risk to use the word 'never' in this context. It is known to me that thorough research may possibly reveal that the history of the concept dates back to the ancient Greeks.

If one side of the coin is complexity, then succinctness is the other side. One may expect that computational power and program size are complementary to each other. In their investigation the authors conclude that this is indeed the case. They show how well-known hierarchies of complexity correspond, on the other hand, to hierarchies of program succinctness. Most of their theorems are stated with broad generality; for instance, most results are valid with almost no respect to the measure of program size. An interesting result is the following: the price for program succinctness is not always in the form of loss of efficiency. As one may guess, quadratic-time programs for a given function can be more succinct than linear-time ones. But there exists some function for which the more succinct quadratic-time program is in fact linear-time – only that the linear time bound is impossible to prove!

While one could well place the text to the far end of any 'practical vs. theoretical' scale, a window for more practical concerns opens on page 173 : 'The inspiration ... came from a 1982 discussion Case had with Peskin's mathematical biology group at Courant Institute... It was suggested that the genetic programs guiding embryonic development were probabilistic. It occurred to Case that that might happen naturally in evolution if probabilistic programs were more succinct than ordinary ones – random genetic search might find succinct programs that worked before insuccinct ones. '

Each part of the text requires hard work on the side of the reader, and the authors are fully aware of this. Throughout the book they make constant effort not to get lost in the technical detail and to keep proofs within palatable bounds, even when this is easier said than done. Parts of some proofs are therefore omitted or are left to the references; by some other theorems, the proofs are limited to representative special cases. Definitions and major theorems are often accompanied by short explanations. Widespread references are used to put parts of the work into a wider context, while the introductory chapter provides an overview of the path followed. Yet the conflict between the language of the mathematical argument and human mind remains, to an extent that sometimes brings the authors to ironical remarks: 'We have not explicitly constructed these programs, but we estimate that an explicit construction *could* be carried out by a suitably trained person in less than a day,'[2] and a footnote adds: 'Given, of course, suitable motivation – like a threat of execution.' (page 184) The motto of the work: 'When I struggle to be terse, I end by being obscure', is another expression of this conflict.

The book concludes with an outlook to further, unsolved problems and an Appendix with exercises for the interested reader. Here the reader can also find sample solutions for selected exercises.

P. Szabó
Technical University Budapest, Hungary

[2] Horace, *Ars poetica*

INDEX