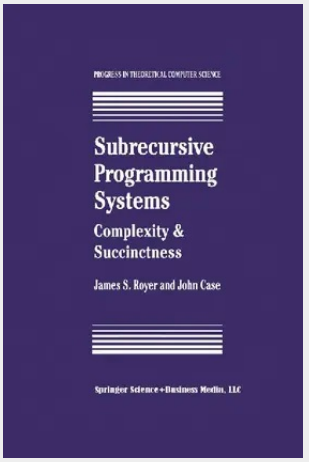
Royer / Case

Subrecursive Programming Systems

Complexity & Succinctness



1.1. What This Book is About This book is a study of • subrecursive programming systems, • efficiency/program-size trade-offs between such systems, and • how these systems can serve as tools in complexity theory. Section 1.1 states our basic themes, and Sections 1.2 and 1.3 give a general outline of the book. Our first task is to explain what subrecursive programming systems are and why they are of interest. 1.1.1. Subrecursive Programming Systems A subrecursive programming system is, roughly, a programming language for which the result of running any given program on any given input can be completely determined algorithmically. Typical examples are: 1. the Meyer-Ritchie LOOP language [MR67,DW83], a restricted assem bly language with bounded loops as the only allowed deviation from straight-line programming; 2. multi-tape 'lUring Machines each explicitly clocked to halt within a time bound given by some polynomial in the length ofthe input (see [BH79,HB79]); 3. the set of seemingly unrestricted programs for which one can prove 1 termination on all inputs (see [Kre51,Kre58,Ros84]); and 4. finite state and pushdown automata from formal language theory (see [HU79]). lOr, more precisely, the collection of programs, p, ofsome particular general-purpose programming language (e.g., Lisp or Modula-2) for which there is a proof in some par ticular formal system (e.g., Peano Arithmetic) that p halts on all inputs.

1.1. What This Book is About This book is a study of ¿ subrecursive programming systems, ¿ efficiency/program-size trade-offs between such systems, and ¿ how these systems can serve as tools in complexity theory. Section 1.1 states our basic themes, and Sections 1.2 and 1.3 give a general outline of the book. Our first task is to explain what subrecursive programming systems are and why they are of interest. 1.1.1. Subrecursive Programming Systems A subrecursive programming system is, roughly, a programming language for which the result of running any given program on any given input can be completely determined algorithmically. Typical examples are: 1. the Meyer-Ritchie LOOP language [MR67,DW83], a restricted assem bly language with bounded loops as the only allowed deviation from straight-line programming; 2. multi-tape 'lUring Machines each explicitly clocked to halt within a time bound given by some polynomial in the length ofthe input (see [BH79,HB79]); 3. the set of seemingly unrestricted programs for which one can prove 1 termination on all inputs (see [Kre51,Kre58,Ros84]); and 4. finite state and pushdown automata from formal language theory (see [HU79]). lOr, more precisely, the collection of programs, p, ofsome particular general-purpose programming language (e.g., Lisp or Modula-2) for which there is a proof in some par ticular formal system (e.g., Peano Arithmetic) that p halts on all inputs.

106,99 €
99,99 € (zzgl. MwSt.)

Lieferfrist: bis zu 10 Tage

Artikelnummer: 9780817637675
Medium: Buch
ISBN: 978-0-8176-3767-5
Verlag: Birkhäuser
Erscheinungstermin: 01.08.1994
Sprache(n): Englisch
Auflage: 1. Auflage 1994
Serie: Progress in Theoretical Computer Science
Produktform: Gebunden
Gewicht: 1210 g
Seiten: 253
Format (B x H): 160 x 241 mm

Kundenservice Verlag Dr. Otto Schmidt KG
Neumannstraße 10, 40235 Düsseldorf | kundenservice@otto-schmidt.de | 0221-93738-997 (Inland)