Knihy došlé do redakce (Books received)

Ferenc Forgó: Nonconvex Programming. Akadémiai Kiadó, Budapest 1988. 188 pages; 23 figs.

B. A. Bublik, V. P. Klimenko, S. B. Pogrebinskij, Ju. S. Fišman: Analitik: čislenno-analitičeskoe rešenie zadač na malych EVM — spravočnoe posobie. Naukova dumka, Kiev 1987. 144 stran; Rbl. 0,75.

Open Problems in Communication and Computation (*Thomas M. Cover, B. Gopinath, eds.*). Springer-Verlag, New York—Berlin—Heidelberg—London—Paris—Tokyo 1987. viii + 236 pages; 28 figs; DM 49,50.

J. L. C. Sanz, E. B. Hinkle, A. K. Jain: Radon and Projection Transform-Based Computer Vision — Algorithms, A Pipeline Architecture, and Industrial Applications. (Springer Series in Information Sciences 16.) Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1988. VIII + 123 pages; 39 figs.; DM 59,—.

Mathematics and Computer Science in Medical Imaging (*M. A. Viergever, A. Todd-Pokropek, eds.*). (NATO ASI Series — Series F.: Computer and Systems Sciences, Vol. 39.) Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1988. VIII + 546 pages; DM 178,—.

Graphs-Grammars and Their Application to Computer Science — 3rd International Workshop, Warrenton, Virginia, USA, December 2—6, 1986 (*H. Ehring, M. Nagl, G. Rozenberg, A. Rosenfeld, eds.*). Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1987. VIII + 609 pages; DM 86,—.

STACS 88 — 5th Annual Symposium on Theoretical Aspects of Computer Science, Bordeaux, France, February 11—13, 1988, Proceedings (*R. Cori, M. Wirsing, eds.*). Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1988. IX + 404 pages; DM 55,—.

Uncertainty in Knowledge-Based Systems — International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Paris, France, June 30—July 4, 1986, Selected and Extended Contributions (*B. Bouchon, R. R. Yager, eds.*). (Lecture Notes in Computer Science 286.) Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1987. VIII + 450 pages; DM 55,—.

ESEC' 87 — 1st European Software Engineering Conference, Strasbourg, France, September 9—11, 1987, Proceedings (*H. K. Nichols, D. Simpson, eds.*). (Lecture Notes in Computer Science 289.) Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1987. XII + 404 pages; DM 55,—.

Tung X. Bu: Co-oP: A Group Decision Support System for Cooperative Multiple Criteria Group Decision Making. (Lecture Notes in Computer Science 290.) Springer-Verlag, Berlin—Heidelberg—New York—Paris—London—Tokyo 1987. XIII + 250 pages; DM 40,50.

ANDREAS ALBRECHT, HERRMAN JUNG, KURT MEHLHORN, Eds.

Parallel Algorithms and Architectures

Proceedings of the International Workshop held in Suhl, GDR, May 25—30, 1987

Lecture Notes in Computer Science 269.
Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1987.
205 pages; DM 36,—.

The volume contains eight invited papers and fifteen contributions presented during the International Workshop on Parallel Algorithms and Architectures held in Suhl, GDR, May 25—30, with the aim to support research activities in the domains in question. Because of the fact that there are no evident differences between the invited and other papers, neither in their form, ordering or length, nor in the way of presentation, let us refer the contributions sequentially according to their order in the volume.

H. Alt, T. Hagerup, K. Mehlhorn and F. P. Preparata describe a non-uniform deterministic simulation of probabilistic random access machines on module parallel computers and on processor network of bounded degree and improve some previous results by Upfal and Wigderson concerning the corresponding computational complexities. R. Dwyer and R. Kannan deduce upper and lower bounds of expected values of certain numerical characteristics of convex hulls of randomly chosen points from polytopes. An overview of dataflow computing models and dataflow computing machines and languages for numerical and non-numerical computations is presented by J. Herath, T. Yuha and N. Saito. E. Koerner, I. Tsuda and H. Shimizu argue in favour of large-scale computational approaches to neural computing which are as tightly as possible related to the known details of neural hardware and function. N. N. Mirenkov deals with parallel algorithms associated with his experience in the parallel language construction for multi-computer systems.

B. Monien and O. Vornberger report on the design and analysis of multiprocessor systems used to solve combinatorial optimization problems from the area of operations research and artificial intelligence. A. Apostolico, C. S. Iliopoulos and R. Paige suggest a parallel algorithm which computes the coarsest refinement of a partition of a finite set S of n elements with respect to a function f on S; the algorithm requires $O(n)$ processors, $O(\log n)$ time and $O(n \log n)$ space in the worst case. A linear-time systolic algorithm for computing the visibility polygon and for trapezoidal partition or triangulation of a polygonal region which may contain holes is presented by T. Asano and H. Umeo. A brief description of a new recursive layout computing system for automated design of integrated circuits can be found in the contribution by L. Budach, E. G. Giessmann, H. Grassmann, B. Graw and C. Meinel. R. Creutzburg solves an interesting problem to design parallel memories to acess trees. An exact definition of systolic systems and several basic notions related to them is suggested by D. De Baer and J. Paredaens. The contribution by K. Diks deals with parallel recognition of outerplanar graphs.

Considering a network of communicating finite state machines which exchange massages over channels, D. Ferment and B. Rozoy discuss the distributed termination problem in the more general context of asynchronous environment and show this problem to be undecidable. Then they investigate superimposing algorithms to detect termination and examine the case of faulty processors. M. Gössel and B. Rebel deal with memories for parallel subtree-access. A more technically oriented paper by S. Matsui, Y. Kato, S. Terramura, T. Tanaka, N. Mohri, A. Maeda and M. Nakanishi offers a multimicroprocessor LISP machine with parallel garbage collector. W. Rytter writes on optimal parallel transformations of regular expressions to nondeterministic finite automata and this author, in cooperation with R. Giancarlo, present also another paper on optimal parallel parsing of bracket languages. D. Uhlig investigates reliable networks built from unreliable gates and proposes a method how to build networks which are, for almost all

Boolean functions, of a very small probability of error and of almost minimal complexity (cost). I. Vrťo proves nearly optimal lower and upper bounds on the AT^2-complexity of the problem of selecting the kth smallest from a set of integers.

F. Wächter's contribution deals with optimization of special permutation networks using simple algebraic relations. C. D. Thomborson, L. L. Deneen and G. M. Shute propose an algorithm for computing a rectilinear Steiner minimal tree on n points in $2^{O(\sqrt{(n)}\log n)}$ time and $O(n^2)$ space, so improving the $2^{O(n)}$ time required by current algorithms. Finally, the last papers in the volume are that by Ch. K. Yap, overviewing the possibilities of parallelization in computational geometry, and the paper by A. Kanagaya, R. Nakazaki and M. Umemura on a co-operative programming environment for a back-end type sequential inference machine.

The limited extent of invited papers as well as of contributions (8—10 pages in average) make the form of presentation very comprimed and close to that of extended abstracts, so that, in many cases, a full comprehension of all details will be possible only for a reader with a rather rich preliminary knowledge in the domain in question. Also the orientation of papers varies from rather philosophical ones and those of very abstract and theoretical nature till more technically and hardware oriented contributions. The volume offers an insight into the topic achievements of contemporary theory and applications of parallel algorithms and can be recommended to specialists in these domains. It is a pity that the editors have not achieved a more unified form of presentation of particular contributions, the differences in typing, spacing, headlines, etc. make the first impression of the volume not so high as typical of other LNCS volumes.

Ivan Kramosil

D. BJØRNER, C. B. JONES, M. MAC AN AIRCHINNIGH, E. J. NEUHOLD, Eds.

VDM'87, VDM — A Formal Method at Work

Proceedings of VDM-Europe Symposium 1987, Brussels, Belgium, March 23—26, 1987

Lecture Notes in Computer Science 252.
Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1987.
IX + 422 pages; DM 55,—.

VDM (the Vienna Development Method) is a formal method for designing software systems. The result of a VDM design process is a series of specifications, starting with an abstract specification of the functions the desired system is intended to perform. Each specification in this series is more concrete and closer to the implementation than the previous one. VDM is equipped with a suitable notation (known as "metalanguage" META-IV) for documenting design.

Including P. Lucas's introductory paper, 22 contributions can be found inside the volume, and they are divided into 10 groups. Studying papers from groups 1—4 the reader is assumed to be familiar with the general concepts of VDM, the other papers are self-contained.

P. Lucas introduces the volume by presenting the origins of VDM. The history of VDM is traced by highlighting the important ideas that contributed to its present form and status. The emphasis is on ideas more than on events, persons and institutions.

The first group of papers is entitled "VDM Experience" and contains three papers describing the experience of using VDM to develop real industrial software systems, e.g. software for telecommunication administration "TAT-8", electronic mail system "Mailman", description of semantics of the programming language ADA, development of machines-independent compiler for concrete source languages (Pascal, C-BASIC, C). Several shortcommings that prevent widespread use of VDM in large-scale industrial developments are discussed, some limitations of existing methods are noted giving pointers for further development of the technology.

Papers from the groups "Use", "Development Methods" give concrete example of using VDM and META-IV as software-development tools.

Chapter four entitled "VDM Environments" contains three papers. K. B. Jones discusses the experience and issues of building systems to support the use of VDM. The MULE system is an example of an environment giving support in the syntactic generation of formal objects, such as specifications. The IPSE 2.5 system is an attempt to produce an industrial scale system to support the use of formal methods over the whole system of a software development life cycle. In M. Haß's paper a method using VDM and META IV for systematic semi-automated compiler development is outlined. Then the construction of META IV compiler sophisticated enough to extend the method to a fully automated one is described. S. Prehn discusses the transition to a new "second generation" formal method RAISE (Rigorous Approach to Industrial Software Engineering). Basic problems of VDM solved by RAISE are the lack of powerful, computerized tools, the lack of a well designed structuring mechanism and of concurrency.

Section 5, 7 and 10 entitled "Foundations I", "Foundations II" and "Tutorial Papers" contain 7 papers dealing especially with three topics: basic mathematical concepts and notations underlying META IV and VDM; data types; two basic techniques used in VDM, i.e. Data Rectification and Program Decomposition. A. Blike's extensive paper is devoted to the methodology of using denotational techniques in software design. The syntax is derived from denotations. Both denotations and syntax constitute many-sorted algebras, denotational semantics is a homomorphism between them. The construction of a denotational model of software system is regarded as a derivation of a sequence of algebras. M. Mac an Airchinnigh describes the relationship between some concrete algebraic structures and META-IV data types tuple, set, map, tree and the use of these data types for specification of objects and their operations. Axiomatic approach to types used in VDM specifications based on Scott domain theory is given by B. Q. Monahan. Papers of C.B. Jones and D. Andrews provide example of two basic steps of software design using VDM-Data Reification (the process of translating abstract data types into data structures which can be implemented in a programming language) and Program Decomposition (the process of translating pre- and post-conditions into executable code).

Sections 6 "Specifications" and 9 "A Case Study" contain two papers. C. Minkowitz and P. Henderson present a formal definition of an object-oriented environment derived from Smalltalk architecture. It is sufficient to design an applications environment based on AI object-oriented architectures or to define such difficult concepts as inheritance, message passing and state change. K. D. Jones presents an outline of a formal description of a non-conventional machine architecture (The Manchester Dataflow Machine) written in VDM extended for non-determinate environments.

The reviewed book can be recommended to program designers, specialists working in theoretical computer science and software industry.

Jan Šindelář

FRED KRÖGER

Temporal Logic of Programs

EATCS Monographs on Theoretical Computer Science 8.
Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1987.
VIII + 148 pages; DM 68,—.

Temporal logic is a logic of propositions whose truth and falsity may depend on time. It is a branch of modal logic. Temporal logic of programs provides a framework for examination of execution sequences of programs and offers logical apparatus for the formal description and analysis of dynamic program properties. In the 1980's, temporal logic of programs has received much attention and has been succesfully applied in particular in the field of parallel programs. It is still an important area of the present-day research.

The book under review presents temporal logic of programs with discrete linear time and modal operators $\circ$ (nexttime), $\Box$ (always), $\Diamond$ (sometime or eventually), atnext and the until operator.

The latter two operators are binary i.e. connecting two formulas, E.g. the formula *A* atnext *B* is interpreted as "*A* will hold at the next time point that *B* holds" or *A* until *B* reads as "A holds at all following time points up to a time point at which *B* holds".

The first two chapters introduce the pure (linear time) propositional temporal logic with its formal semantics and a proof system. Chapter III extends the propositional logic to the first-order temporal logic Chapter IV describes the temporal semantics of certain classes of parallel programs. Chapters V—VII are concerned with verification of various program properties such as invariance, precedence, liveness, partial and total correctness and termination. The development of the basic proof procedures is illustrated by various examples elaborated in detail. A discussion of special methods for sequential programs is also included. Hoare's logic is embedded in temporal logic of programs.

The book presents material from many sources. It concentrates on what is sometimes called the Manna-Pnueli temporal logic of programs and has already reached some sort of mature state. Other interesting lines of contemporary research within the broad general topic, e.g. the temporal logic of branching time or interval logic are only referred to by bibliography. To reach a certain level of mathematical conciseness, the book restrict itself to a special class of programs. It may serve as a textbook or a reference for active researchers or newcomers to the field of temporal logic.

Petr Štěpánek

KLAUS WEIHRAUCH

Computability

EATCS Monographs on Theoretical Computer Science 9.
Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1987.
X + 517 pages; DM 98,—.

The book under review is divided into three consecutive parts. Part 1 "Basic Concepts of Computability" is devoted to the introduction and discussion of the basic properties of computability on natural numbers and on words. As a very general concept flowcharts and machines are defined. Then as special cases different computability models are introduced and proved to be equivalent (register machines, μ-recursive functions, WHILE-programs; tape machines and stack machines). Recursive and recursively enumerable sets are defined and their basic properties are investigated. Standard numbering of unary computable number functions is defined. Some classical concepts and results on computability and recursion theory are introduced and investigated (universal Turing machine theorem, smn-theorem, halting problem, Post's Correspondence Problem, world problem for Semi-Thue systems). After studying Part 1 the reader should have gained a precise intuition of computability, decidability and recursive enumerability which is independent of specific computational models.

Part 2 "Type 1 Recursion Theory" contains the standard recursion theory on the natural numbers. The formalism is based on axioms of a Blum complexity measure. Numberings are introduced as a tool transferring computability from numbers to other denumerable sets, many-one and one-one reducibility of numberings are investigated. A general principle for defining effective numberings is proposed (an extension of Church's thesis). Relativized computability, decidability and enumerability are studied from a general point of view. The recursive and the recursively enumerable sets are studied in more detail, Rice theorem is proved. Creativity and productivity are further focal points. The computable ordinal numbers are introduced and investigated and a standard numbering of this set is defined. Recursion theory is applied to some questions in logic, where the abstract concepts of creativity, productivity and recursive inseparability obtain very interesting interpretations. Excursions to relativized computability, the Kleene hierarchy and to axiomatic computational complexity complete Part 2.

Part 3 "Type 2 Theory of Constructivity and Computability" has no model in the previous literature. It presents a unified systematic theory of constructivity and computability on sets with the cardinality (not greater than that) of the continuum. Such sets author calls Type 2 sets, theory of computability on Type 2 sets he calls Type 2 recursion (or computability) theory. In Part 3 computability and continuity on different Type 2 sets is studied, Type 2 computability on $B = N^N$ (sequences of natural numbers) is taken as a basis. A representation of a certain class of continuous functions from B to B by elements of B is introduced as a counterpart of the standard numbering studied in Parts 1, 2. Effectivity of representations is discussed in detail. The theory of representations is applied for studying constructivity and computability in Analysis. As an instructive example the determination of zeros of continuous functions on the real numbers is discussed in detail.

Topology and the theory of algebraic cpo's play the fundamental role in Type 2 theory. Topological aspects of this theory can be interpreted as a general theory of constructivity on Type 2 sets.

As a broad class of effective representations the admissible representations of T^0-spaces with countable basis are defined and investigated. The definitions are especially applicable to separable metric spaces.

The theory of cpo's has been developed for purposes of programming language semantics. It also constitutes an interesting general model for studying Type 2 computability and constructivity. Cpo's are used as domains for the semantics (the denotational semantics). As application in theory of semantics the problem of solving recursive equations (domain equations) using cpo's is developed.

Many aspects of Type 2 theory are formally similar to recursion theory on natural numbers developed in Part 2. The relation of these theories is studied.

The revised book presents probably the first monography on Type 2 theory and because of its topic subject as well as of its level of presentation deserves the highest appreciation. It can be recommended to mathematicians working in theoretical computer science, to program designers and to everybody interested in constructive mathematics.

Jan Šindelář