Knihy došlé do redakce / (Books received)

ESOP '88 — 2nd European Symposium on Programming, Nancy, France, March 21—24, 1988, Proceedings (*H. Ganzinger, ed.*). (Lecture Notes in Computer Science 300.) Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1988. VI + 381 pages; DM 50,—.

Pattern Recognition — 4th International Conference, Cambridge, U. K., March 28—30, 1988, Proceedings (*J. Kittler, ed.*). (Lecture Notes in Computer Science 301.) Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1988. VII + 668 pages; DM 112,—.

Daniel M. Yellin: Attribute Grammar Inversion and Source-to-source Translation. (Lecture Notes in Computer Science 302.) Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1988. VIII + 176 pages; DM 31,50.

Advances in Database Technology: EDBT '88 — International Conference on Extending Database Technology, Venice, Italy, March 14—18, 1988, Proceedings (*J. W. Schmidt, S. Ceri, M. Missikoff, eds.*). (Lecture Notes in Computer Science 303.) Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1988. X + 620 pages; DM 80,—.

I. M. Kogan: Applied Information Theory. (Studies in Cybernetics 14.) (Translation of: Prikladnaia teoriia informatsii, Radio i Svyaz, Moscow 1979.) Gordon and Breach Science Publishers, New York—London—Paris—Montreux—Tokyo—Melbourne 1988. ix + 466 pages; $ 195,—.

E. M. Vaisbord, V. I. Zhukovskii: Introduction to Multi-player Differential Games and their Applications. (Studies in Cybernetics 15.) (Translation of: Vvedenie v differensialnye igry neskolkikh lits i ikh prilozheniia, Radio i Svyaz, Moscow 1980.) Gordon and Breach Science Publishers, New York—London—Paris—Montreux—Tokyo—Melbourne 1988. viii + 581 pages; $ 190,—.

G. P. Melnikov: Systemology and Linguistic Aspects of Cybernetics. (Studies in Cybernetics 16.) (Translation of: Sistemologiya i yazikovye aspekty kibernetiki, Soviet Radio, Moscow 1978.) Gordon and Breach Science Publishers, New York—London—Paris—Montreux—Tokyo—Melbourne 1988. xiv + 440 pages; $ 198,—.

Kurt Reinschke: Multivariable Control: A Graph-theoretic Approach. (Mathematical Research 41.) Akademie-Verlag, Berlin 1988. 274 pages; M 38,—.

Štefan Neuschl, Jan Blatný, Jiří Šafařík, Jaroslav Zendulka: Modelovanie a simulácie. ALFA — Vydavateľstvo technickej a ekonomickej literatúry, Bratislava a SNTL — Nakladatelství technické literatury, Praha 1988. 424 strán; 205 obr., 4 tab.; cena 33,— Kčs.

Rainer König, Lothar Quäck: Petri-Netze in der Steuerungstechnik. Akademie-Verlag, Berlin 1988. 199 Seiten; 226 Bilder, 10 Tafeln; M 28,—.

Anton Scheber: Dátabázové systémy. ALFA — Vydavateľstvo technickej a ekonomickej literatúry, Bratislava a SNTL — Nakladatelství technické literatury, Praha 1988. 328 strán; 104 obr., 43 tab.; cena 30,— Kčs.

P. M. PARDALOS, J. B. ROSEN

Constrained Global Optimization: Algorithms and Applications

Lecture Notes in Computer Science 268.
Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1987.
VII + 143 pages; DM 27,—.

Global optimization is the area of mathematical programming concerned with the characterization of global minima and maxima of generally nonlinear functions and with the algorithms for

their finding. For usual problem of mathematical optimization — minimization of a function on a compact — standard ways of computation of local minima were developed. There can exist a number of these minima and classical algorithms have no possibility to find the best one. Local minimum of a function will only be global under certain special circumstances, for instance, when the function we are minimizing and the compact set of feasible points are convex. Last years have brought up many algorithms for global finding of general problems of mathematical programming.

The book under review presents results of the increased activity on this subject. This new activity was induced by new prospective ways of solving the problems of combinatorial programming. Authors, renowned specialists in the field of global large-scale optimization, give us overview of the results obtained during last 15—20 years. show us the ideas on which the methods for practical computing are based.

The book starts with the brief summary of the important properties of convex and concave sets and functions in the first chapter. After short explanation of the Kuhn-Tucker conditions in the next chapter, authors describe the problem of polynomially bounded solvability of a quadratic program — a special case of mathematical optimization problem with quadratic objective function.

The third chapter shows us a broad spectrum of difficult (NP-complete, NP-hard) problems of combinatorial programming problems that can be formulated as nonconvex quadratic programs: integer programming, quadratic zero-one programming, assignment problems, min-max problems, to quote the most popular ones.

Next three chapters (4—6) discuss separately three important ways of solution of the concave quadratic programs from the unifying point of view. In the case, when the feasible region of the mathematical optimization problem is a polyhedron, enumeration methods can be very often used. These ones try to find the solution as some extreme point of this polyhedron by direct comparison with the aid of underestimating functions. Methods of cutting planes try to reduce the feasible region by the hyperplanes — using additional linear constraints — so that no optimal feasible point is excluded. This method, once very popular for integer programming, becomes now very important again. Third important type of methods to which Chapter 6 is devoted — branch and bound techniques — is used very often for exact global minimization of nonconvex problems. These techniques are based on partitioning of the feasible region that is determined by the constraints of the original problem.

Chapter 7 is an introduction into the state of the art of classical problems of bilinear programming and jointly constrained bilinear programming problems.

Special care were devoted to the large scale case of nonconvex quadratic problems. Authors show various possibilities of processing of these problems: reduction to separable quadratic form, using piecewise linear approximations of nonlinear functions.

Most of the parts of the book concentrates on the special type of problem — the one with concave quadratic function. When the quadratic program is indefinite, special methods of partition, decomposition or branch and bound techniques must be used. These methods are described in Chapter 9. Last chapter of the book deals with the test problems for generally constrained global optimization problems. All chapters of the revised book are supplied with the exercises that help to demonstrate the general techniques described.

This book summarizes the basic facts on nonconvex global minimization. I cannot recommend it to the people that want to solve some real-world problems. There is a lack of information on comparison of methods and on implementation on computers. It can be recommended to mathematicians working in some other fields of optimization as a reference book or as a textbook for newcomers to the field of global optimization.

Miroslav Tůma

D. H. PITT, A. POIGNÉ, D. E. RYDEHEARD, Eds.

Category Theory and Computer Science

Edinburgh, U. K., September 7—9, 1987, Proceedings

Lecture Notes in Computer Science 283.
Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1987.
V + 300 pages; DM 45,—.

This volume of Lecture Notes in Computer Science is the proceedings of a conference on Category Theory and Computer Science held at the University of Edinburgh, on 7th—9th September 1987. This conference was arranged as a sequel to that held at the University of Surrey in 1985, whose proceedings were published also in Lecture Notes in Computer Science in volume 240. (This volume was reviewed in Kybernetika *23* (1977), no. 6.)

Twenty five authors took part in sixteen contributions that deal with various aspects of mutual links established inbetween computer science and computer programming on one side and category theory on the other side. The following list of contributions shows both range and large variety of problems discussed in this volume:

G. Rosolini, Categories and Effective Computations.
A.M. Pitts, Polymorphism is Set Theoretic, Constructively.
Th. Coquand, Th. Ehrhard, An Equational Presentation of Higher Order Logic.
S. Kasangian, A. Labella, A. Pettorossi, Enriched Categories for Local and Interaction Calculi.
D. B. Benson, The Category of Milner Processes is Exact.
G. Winskel, Relating Two Models of Hardware.
D. E. Rydeheard, J. G. Stell, Foundations of Equational Deduction: A Categorial Treatment of Equational Proofs and Unification Algorithms.
T. Hagino, A Typed Lambda Calculus with Categorial Type Constructors.
L. S. Moss, J. Mesequer, J. A. Goguen, Final Algebras, Cosemicomputable Algebras, and Degrees of Unsolvability.
G. Bernot, Good Functors ... are Those Preserving Philosophy.
C. Beierle, A. Voss, Viewing Implementations as an Institution.
S. Martini, An Interval Model for Second-Order Lambda Calculus.
E. Robinson, Logical Aspects of Denotation Semantics.
M. Proietti, Connections Between Partial Maps Categories and Tripos Theory.
S. Vickers, A Fixpoint Construction of the p-adic Domain.
J. M. McDill, A. C. Melton, G. E. Stecker, A Category of Galois Connections.

What's the connection between category theory and computer science? Some of the computer scientists know that the category theory is not more than fifty years old discipline that arose in forties originally as an endeavour to give a unified view on various constructs used in such highly abstract fields of mathematics as e.g. algebra, theory of sets, topology, and also in logic. Due to well established links to logic, already at the very beginning of category theory building, the people working in it tried to establish good links also to computer science, but for some years with a little success, especially with respect of gaining some *practical* results. Although the main sense of category theory attempt in computer science lies still in "better understanding" the fundamental notions, the papers included in this volume demonstrate a remarkable shift to rather more "practical" questions. Let us mention e.g. categorial treatments for equational deduction, unification algorithm or equational presentation of higher order logic, the very actual topics in logic programming. Such a shift would be surely welcome by many computer scientists. Thus, the book can be recommended to all computer scientists that are interested in deeper understanding of conceptual background of their work. Most of papers are well readable not only for permanently good lay-out of Lecture Notes.

Petr Jirků

J. W. de BAKKER, A. J. NIJMAN, P. C. TRELEAVEN, Eds.

PARLE — Parallel Architectures and Languages Europe

Eindhoven, The Netherlands, June 15—19, 1987, Proceedings

Volume I: Parallel Architectures. Volume II: Parallel Languages

Lecture Notes in Computer Science 258, 259.
Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1987.
Volume I: XII + 480 pages; DM 60, 50. Volume II: XII + 464 pages; DM 60,50.

The PARLE conference was dedicated to the discussion of problems of theory, design and applications of parallel computer systems. The proceedings contain fifty five papers of participants from Europe and North America. Volume I comprises contributions dealing mostly with the architectural aspects of parallel computing systems while Volume II is oriented rather on languages and other programming-related problems. An important group of papers reflects the state of development of several subprojects of the project 415 (Parallel Architectures and Languages for Advanced Information Processing) of the European Strategic Programme for Research and Development in Information Technology (ESPRIT).

Volume I

Most of the contributions dealing with architectural design of parallel systems have a common feature — a distributed topology of processing elements, no shared memory and a highly connected communications network, usually based on point-to-point links. From this common basis the individual contributors part in their specific directions — e.g. massively parallel systems and Boltzmann machines, systolic computation arrays — or concentrate their attention on various aspects of design, specification and simulation/emulation, such as inter-processor communications, process-to-processor assignment, translation of high-level programs into parallel computer systems, etc. Several papers are dedicated to applications, such as processing graphical data on a systolic screen, simulation of digital logic on transputer networks, signal processing, etc. There are also five papers that overview major subprojects of the ESPRIT project 415 — Parallel Reduction Machine (D. I. Bevan et al.), Delta Driven Computer (R. Gonzales-Rubio et al.), Parallel Inference Machine for First-Order Logic (P. Jorrand), Multi-Level Simulator for VLSI (P. Mehring, E. Aposporidis) and *DOOM*, the Distributed Object-Oriented Machine (E. A. M. Odijk).

Volume II

The papers of this volume cover a wide range of topics from the field of parallel systems programming — analysis, specification, languages, design, modelling, process synchronization and distributed garbage collection. Several languages are described — *Id* for the MIT tagged-token dataflow architecture (Arvind, R. S. Hikhil), *PARLOG*, a language for parallel logic programming (K. L. Clark), *Tempura* for temporal logic programming (R. Hale, B. Moszkowski) and *RUTH* a functional language for real-time programming (D. Harrison). Parallel programming requires also new, specific methods and tools for analysis, specification, design and verification. These problems are often solved using graph rewriting/reduction theory to which several papers are dedicated. The problem of a distributed-system garbage collection is also discussed and the strategy of reference counting seems to be generally accepted.

To summarize, the PARLE conference and its proceedings have reflected the state-of-art of highly parallel computer systems design and programming. While the hardware topology aspects seem to be in the basic principles fairly clear and generally agreed upon (though concrete VLSI realizations still require a good deal of research and development), the programming of such systems remains a principal problem to be solved.

Boris Dědina

H. K. NICHOLS, D. SIMPSON, Eds.

ESEC' 87 — 1st European Software Engineering Conference

Strasbourg, France, September 9—11, 1987, Proceedings

Lecture Notes in Computer Science 289.
Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1987.
XII + 404 pages; DM 55,—.

The continuing progress of the computer hardware technology offers more and more resources and execution power so that software can be more refined, user-friendly and it can use more information to solve remarkably complex problems. At the same time, however, software itself becomes proportionally complicated and thus more difficult to manage in all its life cycle phases, i.e. specification, design, coding and testing, configuration/integration, validation and acceptance. To obtain acceptable efficiency and reliability in software production we need appropriate methodologies, tools and even complete, integrated project/production environments (PSEs). This necessity is the common denominator of all the fourty papers published in the Proceedings of the ESEC '87.

Attempting to make an intuitive statistics of the papers which cover the field of software engineering from various points of view and on different levels of complexity, we can point out the following characteristics and trends:

— The specification/definition phase plays a crucial role and thus directly influences the cost and quality of every larger software project.
— Every medium-to-large project requires a dedicated database as an indispensable tool that serves the user during practically all the phases of the software life cycle.
— The user interface is becoming more friendly, i.e. more graphical tools are used at the foreground with the necessary bindings to the internal textual representation. Moreover the user should not be bothered with strict specification syntax and command language rules where the system can figure out what it is being told.

The problem of a complete, integrated software environment is dealt with by several papers though — quite naturally — from different standpoints and using different means. For example SADL (F. Poncet), PantaPM (H. Oswald), ECLIPSE, ProMod (P. Hruschka) and Eiffel (B. Meyer), to name just some. Moreover the last two systems mentioned have been in use at several sites over the world. Some of the projects described in the Proceedings are also very closely related to these topics, though their approach may be viewed as somewhat less complex, namely the Programming environment for Setl (V. Douzoeau-Gouge et al.), Adele program database (N. Belkhatir, J. Estublier), FORTUNE (D. Mullin). Other group of papers contributes to the problem of higher quality software production with dedicated tools or toolsets, such as the Expert Directed Graphical Editor (W. Tichy, F. J. Newbery), the specification language ASDL (M. L. Christ-Neumann, H. W. Schmidt), "SPECIF-X: A Tool for CASE" (M. Lissandre, B. De Vaulx) and "Ten 15: An Abstract Machine for Portable Environments" (I. F. Currie et al.).

There are also papers that, though based on concrete projects, have a more general impact. The analysis of software re-use based on 25 large NASA projects (R. W. Selby), the discussion of exception handling in industrial software engineering (C. Knabe), "Strengths and Weaknesses of Program Verification Systems" (D. Craigen) and "Requirements Analysis for a Database Administration Support Environment" (H. M. Robinson, J. M. Emms).

To summarize the wide range of the presented ideas and experiences, it is evident that the endeavour of the present software engineering community is aimed at an integrated software project/production environment based on a reasonably loose syntax-driven specification language, a deductive database system and a user-friendly (though not expert-hostile), preferably graphical, interface.

The ESEC '87 as well as its Proceedings have undoubtedly contributed to approach this goal. From the formal point of view I only see the division of the fourty papers into twelve sections somewhat overfragmented and, in a few cases, even misleading. *Boris Dědina*

G. GOOS, J. HARTMANIS, Eds.

TEX for Scientific Documentation

Proceedings of the Second European Conference on TEX, Strasbourg, France, June 19—21, 1986

Lecture Notes in Computer Science 236.
Springer-Verlag, Berlin—Heidelberg—New York—London—Paris—Tokyo 1986.
204 pages; DM 36,—.

TEX is a wordprocessing system for scientific documentation designed by D. Knuth at Stanford University. Originally it was developed for writing mathematical texts on non-graphic terminals. This is an heritage which it carries: compared with the WISIWYG class of wordprocessing systems ('What You See Is What You Get'), TEX uses a special formatting language in which the document must be prepared. For comparison, similar job do the *ms* and *me* packages for the UNIX *troff* formatter.

In the competition with the new generation of WISIWYG systems, as e.g. CHI-WRITER which have a high level of user comfort, the recent TEX has developed into a semi-interactive tool using the multiwindow environment with a reasonably fast feedback between typing a page in the TEX language and seeing its effect in another window as a simulated printer output. However, the user must still use the non-graphic window for typing his formulas in an uneasy way.

A lot of work on TEX has been done at University of California, Berkeley. A strong UNIX tradition at Berkeley — "everything can be imported if you just have the right preprocessor" — has lead to VORTEX, LATEX, and BIBITEX enhancements. The paper "An Improved User Environment for TEX" by P. Chen et al. is based on an idea of a screen previewer used for simulation of a printed document on the terminal screen.

Some other presented papers deal also with certain form of making interactive the originally batch-oriented TEX software. Wolfgang Appelt describes one such attempt using the PERQ workstation, or L. Gallot in his ASHTEX uses another form of previewer. However, the authors of the EasyTEX version, E. Chrisanti, A. Formigoni and P. La Bruna, go still farther: They have built a formula editor which directly transforms keyboard strokes into the (invisible) TEX commands, thus being most close to WISIWYG systems.

Two papers by J. Ferguson concentrate on adding national language capability to TEX. Similar problems had R. Wonneberger who describes development of Greek and Hebrew fonts for theological manuscripts. The use of a METAFONT design tool for the creation of user defined fonts described by R. Southall is a common solution in such situations.

The last group of papers concerns the document processing environment: in Karlsruhe, by A Brüggemann-Klein, Peter Dolland, and Alois Heinz, which unites personal computers, workstations and mainframes using a common TEX language for scientific documents, or in Aachen, which H. W. Petersen names 'a document factory'. Chen et al. in "The VORTEX Document Preparation Environment" explains the organization of TEX document libraries in Berkeley.

The reviewed book is mainly interesting in showing that in many different (mostly university) environments the similar job had to be done to solve problems with high quality publishing of scientific documentation. In reading this book, the reader's goal should be to eliminate this effort and to use the gained experience in the right choice of a most productive desktop publishing system. The possible reader could be a scientist, software engineer or a magazine editor.

Petr Pavlík