Bibliographie

R. Gleaves: Modula-2 for Pascal Programmers, X+145 pages, Springer-Verlag, New York, 1984

"Modula-2 is a general-purpose language designed primarily for writing software systems. The name Modula is short for modular language.
Modula-2 is the third in a family of languages created by the Swiss computer scientist Niklaus Wirth. The first language, Pascal, was conceived as a teaching language, but achieved widespread use. The second, Modula, was a special-purpose language designed for programming small real-time control systems. Modula-2 emerged as a synthesis of the systems programming capabilities of Modula and the general utility of Pascal."

This book is written for people who know the Pascal language and who wish to learn Modula-2 in terms of their knowledge of Pascal. The book does not offer a complete description of the language; it is intended to give an overview of the language by focussing on the differences from Pascal and by introducing new concepts unique to Modula-2. A major strength of the book lies in its practical approach. The book is divided into three parts and the appendices. The appendices include syntax diagrams and a glossary of Modula-2 terminology. Part 1 introduces language concepts which are unique to Modula-2. These are modules, separately compiled modules, program and subprogram modules, utility modules, module library, low-level programming coroutines and interrupts. Numerous example programs are provided to illustrate the new language concepts. Part 2 discusses the differences between Pascal and Modula-2, Part 3 presents a set of utility modules which are not part of the Modula-2 language. These modules provide the basic programming facilities used by most Modula-2 programs and are part of an existing Modula-2 system.

People who are familiar with Pascal can use this book as a good overview of the language Modula-2.

Gy. Horváth

Machine Learning. An Artificial Intelligence Approach. Edited by R. S. Michalski, J. G. Carbonell, T. M. Mitchell (Symbolic Computation) XI+572 pages, Springer-Verlag, Berlin—Heidelberg—New York—Tokyo, 1984.

We may start with a brief outline of the book.

In Chapter 1, Carbonell, Michalski and Mitchell give an overview of Machine Learning (objectives, taxonomy, historical sketch).

At the Carnegie-Mellon Machine Learning Workshop in July, 1980, Herbert Simon was asked to deliver the keynote address, where he chose to play the role of devil's advocate and ask the question "Why Should Machines Learn?" After dispelling some common myths, Simon concluded with a clarified and more appropriate set of reasons why one ought to pursue machine learning research. Chapter 2 is based almost entirely on that rather controversial keynote address.

In Chapter 3, Dietterich and Michalski analyze some well-known work in concept acquisition from a unified perspective. This part provides a general framework for the comparison of different concept-acquisition systems.

In Chapter 4, Michalski describes a general theory and methodology for the inductive learning of structural descriptions from examples. Various generalization rules are presented and discussed. The methodology developed is illustrated by a problem from the area of conceptual data analysis.

In Chapter 5, Carbonell examines the issue of learning from experience. A general planning and problem-solving paradigm is proposed, based on a computationally-effective model of analogical reasoning.

In Chapter 6, Mitchell, Utgoff and Banerji investigate the issue of acquiring and refining problem-solving heuristics by examining solutions to symbolic integration problems. Like Carbonell's approach, learning is based on past problem-solving experience, but Mitchell at al focus on acquiring heuristics for applying known strategies, rather than generalizing recurring behavior into reusable plans.

In Chapter 7, Anderson examines human problem-solving in the context of providing justifications for geometric proofs. He relies entirely upon a production system framework to encode domain knowledge, learning heuristics and problem-solving strategies.

In Chapter 8, Hayes—Roth investigates the issue of improving flawed or incomplete theories that quide plan formation in a given domain. He presents five heuristic methods and applies them to problem-solving in playing the card game hearts.

In Chapter 9, Lenat focusses on methods for learning from observation and discovery. He analyizes three domains in which heuristics plays a dominant role in guiding a search through the space of possible concepts or processes one may acquire.

In Chapter 10, Langley, Simon and Bradshaw discuss their BACON system and its application to rediscovering some basic laws of chemistry. BACON applies the principles of scientific inquiry first elucidated by Sir Francis Bacon to find the simplest numerical relations that are invariant across sets of measurements. Although not able to design its own experiments, given the unanalyzed results of appropriate chemical experiments, BACON has rediscovered such laws as Guy-Lussac's law and Proust's law of definite proportions.

In Chapter 11, Michalski and Stepp investigate the problem of the automated construction of taxonomies of observed events in a manner that is meaningful to a human. They present an algorithm that implements the "conceptual clustering" operation and demonstrate its utility for the tasks of formulating descriptions of plant diseases from obsereved symptoms and taxonomizing Spanish songs in a manner meaningful to a musicologist. In contrast with statistical clustering techniques, the conceptual clustering algorithm produces characteristic descriptions of the concepts defined by each cluster.

In Chapter 12, Mostow discusses the process of learning by taking advice. Declaratively stated advice must be transformed into operational procedures effective in a given task domain. Mostow focusses on the general issue of providing advice for a heuristic search mechanism, as applied to playing the game of hearts.

In Chapter 13, Haas and Hendrix investigate the issue of automatically extending a natural language interface by acquiring domain semantics, dictionary entries and syntactic patterns from the user. The most significant aspect of their KLAUS system is that the user need not be a computational linguist, but rather is guided by the system into providing exemplary information that is later transformed into effective grammar and dictionary representations.

In Chapter 14, Rychener provides a retrospective analysis of the instructable production system project, in which many different instructional techniques for learning by being told were tried, different organizations of the knowledge were considered, and different problem-solving strategies were investigated. He concludes his chapter with an analyzis of the organizational and instructional principles that a production-system based instructional learner should adhere to in order to maximize his chances for successful knowledge acquisition.

In Chapter 15, Quinlan presents a method for generating efficient decision trees for classifying given exemplars, and applies his method to the analysis of king-and-rook versus king-and-knight chess endgames.

In Chapter 16, Sleeman investigates the application of machine learning to inter models of students learning algebra. An interesting aspect of Sleeman's work is that the teacher, in order to be effective, must learn to adapt to the student's needs, indicating that machine learning can help to make computer-assisted human education more effective.

This is a very well-written book. The chapters have a unified style and progression. From the first few chapters the reader not familiar with this field may acquire a general understanding. The second part of the book gives an excellent state of the art of machine learning.

J. Csirik

Model Program Committee of the IEEE Computer Society, "Model Program in Computer Science and Engineering", Committee Report, VI + 154 pages, IEEE Computer Society Press, 1983.

"In late 1981 the Educational Activities Board of the IEEE Computer Society reviewed the curriculum recommendations made in 1977. This evaluation clearly indicated that major changes had taken place which were not included in the earlier curriculum recommentations and that a major revision was needed. The committee established to consider this problem not only recommended that the 1977 model curriculum report be updated, but that it be expanded to cover all areas defining an exellent-caliber undergraduate program in the computer area. Therefore, in addition to the curriculum, this report addresses guidelines for the development of faculty, administration, and material resources.

The Model Program has the following goals:

— to provide an overview of the desirable features of undergraduate academic programs in computer science and engineering;
— to provide a standard of comparison that can be used to guide the development of new programs or the modification and upgrading of established programs;
— to provide an interpretation of Accreditation Board for Engineering and Technology (ABET) criteria for minimum program standards, particularly for departments seeking ABET accreditation;
— to establish a set of standards that can be used to define 'Target of Exellence' programs;
— to define the computer science and engineering aspects of a curriculum in a way that allows flexibility to meet the requirements of individual institutions;
— to provide guidance to academic administrators concerning the level of commitment needed to support a program."

The report consists of five sections, appendices and references. Section A is the introduction. A set of program criteria specifies requirements for computer science and engineering, discussed in Section B. Section C deals with all aspects of an undergraduate program in computer science and engineering. The main part of the report is the description of those 28 subject areas that constitute the core of a computer science and engineering curriculum. Subject areas are organized as collections of modules. Each module includes:

— the specification of the teaching purpose,
— the indications of the prerequisite modules,
— the concepts relevant to the module,
— references.

From the collection of subject areas, many different curricula can be generated. Subsection C. 4 shows several curricula implementations and how subject areas map into specific courses. The subject areas are the following: lecture component: fundamentals of computing, data structures, system software and software engineering, computing languages, operating systems, logic design, digital systems design, computer architecture, interfacing and communication; laboratory component: introduction to computing laboratory, software engineering laboratory, project laboratory; advanced subject areas: software engineering, digital design automation, theory of computing, database systems, advanced computer architecture, design and analysis of algorithms, fault-tolerant computing, performance prediction and analysis, computer graphics, VLSI system design, translator writing systems, computer communications networks, system laboratory, artificial intelligence, advanced operating systems. Section D provides faculty conciderations for computer science and engineering programs. Section E discusses computing, laboratory and library resource requirements of computer science and engineering programs. There are more than 400 entries in the references.
This report is recommended for those people engaged in educational activities in computer science and engineering.

Gy. Horváth

Ada: ®Language, compilers and bibliography. Ed. by M. W. Rogers (The Ada Companion Series), Cambridge University Press, 1984.

Ada is a programming language designed in accordance with requirements defined by the United States Department of Defense. It is a modern algorithmic language with the usual control structures, and with the ability to define types and subprograms. It also serves the need for modularity, whereby data, types and subprograms can be packaged. It treats modularity in the physical sense

® Ada is a registered trademark of the U.S. Government, Ada Joint Program Office as well, with a facility to support separate compilation. In addition to these aspects, the language covers real-time programming, with facilities to model parallel tasks and to handle exceptions. It also covers systems programming; this requires precise control over the representation of data and access to system-dependent properties. Finally, both application-level and machine-level input-output are defined.

This book is divided into two sections differing in the colour of the pages. The white section contains the reference manual for the Ada programming language (ANSI/MIL—STD—1815A—1983). Some helps are added to facilitate reading of this section, which are not parts of the standard definition: Glossary, Syntax Summary, Implementation-Dependent Characteristics and Index. For example, the Index includes an entry for each technical term or phrase that is defined in the reference manual.

Guidelines for Ada compiler specification and selection are given in the green section. The purpose of this guide is to list the characteristics of an implementation that should be taken into account in the specification or selection of an Ada compiler. The section ends with a selective bibliography for Ada, with 20 aspects, including more than 460 items. This well-structured bibliography lists the principle works on Ada, covering all aspects from the history and evolution of the language to the latest thinking on the many features combined for the first time in Ada.

This book is a member of the Ada Companion Series. "This definitive new series aims to be the guide to the Ada software industry for managers, implementors, software producers and users. It will deal with all aspects of the emerging industry: adopting an Ada strategy, conversion issues, style and portability issues, and management. To assist the organised development of an Ada-orientated software components industry, equal emphasis will be placed on all phases of life cycle support."

This is a reference volume that should never be far from the workbench of the serious software engineer or programmer using Ada.

K. Dévényi

Kurt Melhorn, Data Structures and Algorithms, Vol. 1: Sorting and Searching, XII+336 pages, Vol. 2: Graph Algorithms and NP-Completeness, XII+260 pages, Vol. 3: Multi-dimensional Searching and Computational Geometry, (EATCS Monographs on Theoretical Computer Science), Springer-Verlag, Berlin—Heidelberg—New York—Tokyo, 1984.

The rapid development of computer architectures and the ever increasing complexity of computer applications have produced a revolution in mathematics. The concept of an efficient algorithm has become the centre of investigations in a large part of computer science. These 3 volumes not only give a collection of efficient algorithms, but explain what efficiency means and what principles are used to construct and analyse efficient algorithms.

Volume 1 consists of 3 chapters. Chapter I starts by introducing a machine model of computation and complexity measures. This is followed by a discussion on basic data structures: queues, stacks, lists, etc. Chapter II gives an extensive treatment of sorting algorithms. General sorting methods include heapsort, quicksort and mergesort. Specific sorting methods are presented for sorting words and reals. Chapter III is devoted to one-dimensional searching techniques, such as digital search trees, hashing, weighted and balanced trees.

Volume 2 contains 3 chapters. In Chapter IV graph representations and various graph algorithms are dealt with. The algebraic interpretation of path problems on graphs leads to efficient matrix multiplication algorithms in Chapter V. Chapter IV provides a study of NP-completeness. After a series of well-known NP-complete problems, methods for solving them are investigated.

Volume 3 has 2 chapters. Chapter VII deals with multi-dimensional searching, and Chapter VIII explores computational problems and their solutions in geometry.

Chapter IX, which is included in all 3 volumes, gives an orthogonal overview of the main algorithmic paradigms.

The volumes are written in a readable and lucid style. Except for Chapter IX, each chapter ends with exercises and bibliographic notes.

The material is self-contained. Thus, the books can be recommended to everyone interested in the subject. A large proportion of the discussion is about very recent results, which ensures that even experts will find them interesting reading.

Z. Ésik

Interactive Programming Environments (Editors: D. R. Barstow, H. E. Shrobe, E. Sandewall), XII+610 pages, McGraw-Hill Book Company, 1984.

This book provides a collection of selected papers on interactive programming environments. Some of the papers have been publ!shed previously. The papers are divided into five sections. "The first section includes papers which describe the motivations and characteristics of interactive programming. The second section includes papers dealing with specific environments for particular languages and situations. The third section is concerned with specific issues which arise in many different environments. The fourth section includes descriptions of experimented systems which test the role of artificial intelligence in interactive programming environments. The final section includes papers concerned with the future of interactive programming environments."

The contents of the book are the following:

Section 1: Perspectives on Interactive Programming Environments. T. Winograd: Breaking the Complexity Barrier (Again); B. A. Sheil: Power Tools for Programmers; E. Sandewall: Programming in an Interactive Environment: The Lisp Experience

Section 2: Modern Interactive Programming Environments. W. Teitelman, L. Masinter: The Interlisp Programming Environment; T. Teitelbaum, T. Reps: The Cornell Program Synthesizer: A Syntax — Directed Programming Environment; J. Wilander: An Interactive Programming System for Pascal; V. Donzeau-Gouge, G. Huet, G. Kahn, B. Lang: Programming Environments Based on Structured Editors: The MENTOR Experience; A. Goldberg: The Influence of an Object-Oriented Language on the Programming Environment; B. W. Kernighan, J. R. Mashey: The UNIX Programming Environment; T. Cheatham, J. Townley, G. Holloway: A System for Program Refinement

Section 3: Aspects of Interactive Programming Environments. W. J. Hansen: User Engineering Principles for Interactive Systems; W. Teitelman: Automated Programmering: The Programmer's Assistant; W. Teitelman: A Display-Oriented Programmer's Assistant D. R. Barstow: A Display-Oriented Editor for Interlisp; R. M. Stallman: EMACS: The Extensible, Customizable, Self-Documenting Display Editor; R. D. Greenblatt, T. F. Knight, Jr., J. Holloway, D. A. Moon, D. L. Weinreb: The LISP Machine; T. A. Dolotta, R. C. Haight, J. R. Mashey: UNIX Time-sharing System: The Programmer's Workbench; A. I. Wasserman: Software Tools in the User's Software Engineering Environment; I. P. Goldstein, D. G. Bobrow: A Layered Approach to Software Design; J. W. Goodwin: Why Programming Environments Need Dynamic Data Types; E. Sandewall, C. Strömberg, H. Sörensen: Software Architecture Based on Communicating Residential Environments.

Section 4: Artificial Intelligence in Interactive Programming Environments.
C. Rich, H. E. Shrobe: Initial Report on a Lisp Programmer's Apprentice; R. C. Waters: The Programmer's Apprentice: Knowledge Based Program Editing; E. Kant, D. R. Barstow: The Refinement Paradigm: The Interaction of Coding and Efficiency Knowledge in Program Synthesis

Section 5: The Future of Interactive Programming Environments.
T. Winograd: Beyond Programming Languages; J. N. Buxton, L. E. Druffel: Rationale for Stoneman; S. E. Fahlman, S. P. Harbison: The Spice Project; D. E. Barstow, H. E. Shrobe: From Interactive to Intelligent Programming Environments

The book covers a broad field of interactive programming environments. The focus is mainly on interactive programming environments, and the related traditional areas of computer sciences are also discussed (software engineering, programming languages, artificial intelligence). The book presents a clear picture of the experiments performed in the 1970s, and all people working in this fiedl may benefit from these papers.

J. Csirik