Books

Artificial Intelligence and Instruction; Applications and Methods, Greg Kearsley, Ed., Addison-Wesley Publishing Company, Reading, MA, 1987. 351 pp. (ISBN 0-201-11654-5).

Read this book! If you are interested in applications of AI, uses of computers in education, or applied psychology, this book will have something of interest for you. It is an interesting collection of previously published work in an area that has come to be called Intelligent Computer-Assisted Instruction (ICAI). As presented in this book, ICAI is an intersection of psychology (how do people think?), education (how do people learn? how does one teach?), and computer science (how can this be implemented on a computer?). ICAI differs from an ordinary computer application in that in addition to knowledge about a specific subject, it also requires a model of what the student knows, and applied knowledge of various teaching methods. ICAI applications are also exceptionally dependent on the quality of the interface between the computer and the student.

A book whose scope spans three disciplines must of necessity omit much of relevance and interest, and the editor acknowledges that fact. However, the book gives one a good feeling for the territory, and the references allow further reading for those whose interest is really stimulated.

The book is divided into five sections: an overview, two sections on instances of ICAI applications, and two sections on writing an ICAI application. The overview does a good job of describing some of the tensions among psychological and educational theories that have affected the emergence of ICAI. The old teaching method appears to center around Skinnerian drill and practice, with statistical measures of performance. The new method seems to require a cognitive model of student knowledge, which can be compared to a body of knowledge represented explicitly in the system.

The book could do more to explore, or at least acknowledge, alternatives to and limitations of the artificial intelligence aspects being applied in the ICAI projects reported in the book. For example, many of the reported student-system dialogs seem impressive, but such natural-language systems can be extremely fragile. This was not in general acknowledged by the authors. I found only one who expressed hope that AI researchers would "improve" the natural-language capability of ICAI systems.

A varied array of ICAI selections are offered. Space does not allow an in-depth review of them all, but the following may give an idea of some of the topics with which the book is concerned.

PROUST attempts to locate errors in elementary computer programs and suggest corrections. With a 70% success rate, it is not very successful at locating errors. The authors did not report on the success rate for non-elementary programs, but from their description I would predict that the success rate degrades very quickly for larger or more sophisticated programs. No data are given indicating how well PROUST helps students learn elementary programming.

The chapter on PROUST is a reprint of an article that appeared in 1985. At the time of original publication, the authors said they were "not yet sufficiently satisfied with PROUST's accuracy to make it generally available to students." However, they had ambitious plans "to build an automated programming course around PROUST" that would "not only correct the students' mistakes, but also suggest additional problems to solve." They reviewed such a completed system as "an important milestone in applying artificial intelligence to teaching programming." One hopes it has been improved in the intervening two

years, since this is one of two programs offered for sale in the book. There is actually a full-page commercial advertisement (yes, major credit cards are accepted).

With the sales blurb in mind, one disappointing aspect of the presentation on PROUST is the limited reference to previous work on finding and correcting programming errors. This does not allow a reader unfamiliar with this sub-branch of artificial intelligence to form a balanced perspective. Such an omission is particularly convenient in this case because a more successful error locator and corrector had previously been produced at the same university (Yale). Ehud Shapiro's *Algorithmic Program Debugging* had success rates much higher than PROUST.

TSEARCH (and its PC implementation Micro-SEARCH) leads students through exercises in reducing mathematical expressions. It can list the operators available at a given stage, list the rules for applying given operators, give the student a hint, and "juxtapose" a student's solution with the one that would be chosen by TSEARCH. No data are given as to the educational effectiveness of this system.

TSEARCH, implemented in LISP on a DEC System 10, is apparently too slow to be very effective. The authors report that "the major disadvantage of TSEARCH was its speed for any nontrivial task." It is not clear whether Micro-SEARCH, implemented in interpreted LISP on an IBM PC, is any faster. One suspects not. Although they did change some of their algorithms, the authors do not give any comparative performance statistics. This lack of performance would not matter much, except that Micro-SEARCH is the other program being offered for sale.

Since the methodology they use in TSEARCH is a straightforward depth-first search algorithm, they would probably be better off implementing Micro-SEARCH in a compiled depth-first language such as Prolog.

One of the more interesting concepts is presented in the chapter on mathematical microworlds. One can get direct physical experience of the concept of "lever" on a playground with an adjustable see-saw, but more complicated mathematical concepts are more difficult to experience. The purpose of a mathematical microworld is to provide a computer playground where the student can get first-hand experience. No instruction is given. As the author says, when "you shoot an arrow at a target and the arrow is wide to the left, the target doesn't yell 'Hey, more to the right.'" Again, it is unfortunate that no data are given on the educational effectiveness of the system.

There is much more. The STEAMER project is used to discuss the importance of interactive graphics in a training situation. The GUIDON and NEOMYCIN expert systems illustrate the assertion that expert systems by themselves cannot effectively be used as educational tools, because they lack any sort of teaching strategy. There are four other ICAI-authoring systems discussed. They each may be of some help, but it is difficult to imagine how any of the ICAI instances described could have been constructed using any of the ICAI-building tools presented in the last two chapters.

The book as a whole seemed to spend an inordinate amount of time discussing whether ICAI was indeed something new or just repackaged CAI. Not having an academic stake in this turf-battle, I was more interested in the effectiveness of computer-assisted education, and whether it can be improved. It is disappointing not to be able to find *any* data on the educational effectiveness of the reported systems.

Reading this book reminded me of a conversation with my son's primary-school principal. She said she did not think that computers could ever be effective educational tools. When I asked her why she thought so, she replied: "Computers can't give hugs." I think she might be on to something.

Walter Wilson
IBM Thomas J. Watson
Research Center
Yorktown Heights
New York

Data Structure: Form and Function, Harry F. Smith, Harcourt Brace Jovanovich Publishing Company, Inc., San Diego, CA, 1987. 732 pp. (ISBN 0-15-516820-7).

Data structure and algorithms are fundamental to computer science. Harry Smith has written a book that introduces the concepts to the novice and explains the advanced topics to the more experienced reader. By reading the introductory information in each chapter, the novice can gain an understanding of the field. The advanced reader can skim the introductory material and then study the advanced topics

in depth. The sections in each chapter that are considered advanced are marked.

The first chapter, Preliminaries, reviews the mathematics used throughout the book. The mathematics is not complex, and the review has two advantages. First, it refreshes the reader's memory of specific mathematical notation that may have been forgotten. Also, the sample algorithms are written in Pascal. Although an overview of Pascal is also included, the reader need not know Pascal to understand the algorithms in this book. It, is however, helpful if the reader has some familiarity with one of the many high-level programming languages.

Chapters 2 through 8 are about individual data structures, which include arrays and sets, records, lists, queues and stacks, trees, graphs, and strings. Again, these chapters start with an introduction of the concept and evolve to detailed discussions of the particular data structures. After the various data structures are described, a chapter on structure and complexity helps answer such questions as How do we choose a good implementation for a data structure? Why are there so many different data structures? Are any data structures more fundamental than the others? The last chapters concern searching, managing primary and secondary memory, and sorting.

From the first chapter we are introduced to two sections that are included in all the chapters: Bibliographic Notes and Reference to Terminology. The Reference to Terminology section is something that is not common in advanced-level texts. Terms introduced in the chapter are listed, along with the page where they are introduced; this is useful as a kind of index as well as a review of the chapter. The Bibliographic Notes annotate the references listed in the bibliography that pertain to the chapter. The extensive bibliography has the most significant reference highlighted. Each chapter provides a series of exercises that add to the reader's understanding of the topic, and the intermediate or advanced-level exercises are marked.

Harry Smith has designed this book to take the reader from clearly written introductory topics to the advanced topics in the field. The topics flow logically from one to another. Within each chapter, ideas flow from introductory to advanced. The many examples used throughout the book are clearly written and easy to understand. The text is easy to read. Although this book was primarily designed for courses in upper-level data structures, it is an excellent reference book for the professional programmer. This book could be read profitably by either students or practitioners of computer science.

Gloria J. Hasslacher
ISG Business Systems
Franklin Lakes
New Jersey

SNA: IBM's Networking Solution, James Martin with Kathleen Kavanagh Chapman, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1987. 380 pp. (ISBN 0-13-815143-1).

Merlin the Magician is generally believed to have had extraordinary powers and, therefore, influence in the court of King Arthur. There is, however, a minority opinion concerning Merlin's true prowess, that he was just a clever enunciator of incantations. Merlin was never really a true magician—he merely "knew all the magic words." James Martin and Kathleen Kavanagh Chapman have disproved the assertion that only an SNA Merlin could divine the murky, convoluted, mystical world of Systems Network Architecture. *SNA: IBM's Networking Solution* provides a clear, concise (yet complete) description of the architectural concepts, protocols, functions, and product implementations which comprise the SNA environment.

It is apparent that the authors are sensitive to the natural trepidation of many readers as they attempt to fathom the publicized complexities of SNA. Information is presented in short precise paragraphs, illustrations are simple yet meaningful, and each chapter has a crisp summary. In the overall layout, each major paragraph and subparagraph is titled, lists of items comprising a function or service are separately boxed, and figures are simple yet informative; this contributes to the almost primer look of the information being conveyed. Even when high-level concepts give way to detailed descriptions of frame and control field formats, command flows, and protocols, a sense of straightforward simplicity remains.

The book is divided into two main sections. The first half concentrates on SNA fundamentals and constructs a conceptual framework to support the more detailed information presented in the second half. As such, it serves the reader interested in understanding definitions and rationale for familiar SNA entities, as well as one interested in the specific details of formats, protocols, and control blocks.

The conceptual framework is constructed by means of the following chapters, which give a first-pass, high-level view of SNA:

Network Architectures: The value of architectures in general is discussed. Public vs. User Network Architectures are reviewed and the objectives and significance of SNA are presented.

SNA Concepts: A description is presented of how network users are served by the components of an SNA network, SNA network configurations, SNA paths and network addresses, SNA communication links, and SNA sessions.

SNA Functional Layers: The major functional layers of SNA (Data Link Control, Path Control, Transmission Control, Data Flow Control, and Function Management) are introduced, with emphasis on the functions they perform as opposed to the specifics of their formats and protocols. In addition, the critical notion of adjacent as well as peer-to-peer communication between layers is discussed.

Routing and Data Flow: A close look is taken at the SNA network to see how data are routed through the system and how data flow is controlled. Explicit routes, virtual routes, and class of service are defined to provide routing options. The discussion of data flow introduces the various control headers (Request/Response Header, Transmission Header, SDLC Header/Trailer) necessary to enable messages to pass between two network users via the functional layers of the SNA nodes.

SNA Services: Services related to SNA sessions in operation in the network are described, as well as how those services are distributed among the various SNA functional layers. In addition, services aimed at the overall management of the network and its resources are identified. Finally, end-user-oriented services, including presentation and application-to-application facilities, are described.

SNA and the OSI Model: This section provides the finest description of the OSI Model I have encountered to date, as well as a comparison between SNA and OSI layer definitions. The similarities and differences are clearly described and, most important, the idea that the OSI model and SNA can be viewed as complementary is validated.

SNA Program Products: The key software products used to implement SNA networks are reviewed. Included are Telecommunications Access methods (ACF/TCAM, ACF/VTAM, ACF/VTAM-Entry), Network Control Programs (ACF/NCP, Network Terminal Option), Application Subsystems (CICS/VS, IMS/VS, DPPX/DMS, ACF/TPF, TSO, VSPC, VM/VCNA), and Network Management programs (NCCF, NPDA, NLDM, NetView). In each case a short description of the function provided by the product enables the reader to map the product to the applicable SNA entity or capability.

Advanced Program-to-Program Communication (APPC): The Logical-Unit-to-Logical-Unit (LU-LU) session is examined, starting with a description of the numerous existing LU Types. LU 6.2 is introduced to meet the requirements of the distributed transaction processing environment and to provide a base for general-purpose program-to-program communication.

Logical Unit 6.2 Implementation: The authors have recognized the significance of LU 6.2 in current and future SNA environments, to the extent that a separate chapter is devoted to its implementation. The functional capabilities of LU 6.2 supporting conversations between distributed transaction programs, including data mapping and synchronization verbs, are described. In addition, this chapter provides a definition of the LU 6.2 protocol boundary, which enables using products to represent LU 6.2 verbs to their applications programs in a product-defined programming language or syntax. The APPC facility provides the basis for two key extensions to SNA: Office Automation Architecture and SNA Distribution Services.

Office Automation Architecture: This section addresses Document Content as well as Document Interchange Architecture. Emphasis is placed on descriptions of document distribution and library services, the key elements of the Office Automation implementation. The role of Document Content Architecture in handling editable and/or final-form documents is also described.

SNA Distribution Services: SNA was initially designed to provide synchronous communication between two network users in session and actively exchanging information. Office systems communication must allow for asynchronous communication. The office system must be able to receive documents, distribute them through the network, queue them, and later deliver them upon request of the recipient. SNA Distribution Services provides this asynchronous dis-

tribution capability. This section clearly describes how office products and other applications can make use of names, distribution directories, routing by service level, and underlying LU 6.2 services to provide asynchronous communication facilities to their users.

SNA Network Interconnection (SNI): This final conceptual section describes the difficulties in interconnecting existing networks and the SNI gateway approach as a solution.

The final half of the text takes a second, much more detailed pass through the SNA elements previously described. Here control block formats, control header bit-level descriptions, and flow sequences are used to describe the layers and services within SNA. Included are major chapters on the Path Control Network (Data Link Control, SDLC Frames/Data Flow/Polling Path Control Services), Transmission and Data Flow Control, and Function Management (End-User Services, FM Header, Session Services, Session Termination and Outage Notification, Configuration Services, Switched Links, Maintenance and Management Services). Each section is complete and continues the clear presentation style shown in earlier chapters, even when detailed and potentially confusing volumes of information are conveyed.

SNA: IBM's Networking Solution is a correct, clear, and refreshingly understandable reference and textbook on SNA.

William A. Bernstein
IBM Research Triangle Park
North Carolina

Note—The Editor assigns reviews of books that might interest our readers. Reviews are signed, and opinions are those of the reviewers.