Computational Models of Natural Language Processing

Bruno G. Bara and Geovanni Guida (Elsevier, New York, 1984, 327 pp., $52)

Teiji Furugori,
University of Electrocommunications, Tokyo

Natural language processing began soon after the advent of computers. Research results during the past thirty years are enormous, or insignificant, depending on how we evaluate them. When we view natural language as strings, we have accomplished much in natural language processing. For example, word processing for Japanese was unthinkable until quite recently. But now one can purchase a portable model for less than $200, thanks to the hardware technology and software development that came from statistical studies depicting Japanese language characteristics. In this respect, we have come far in natural language processing.

Mechanical understanding of natural language, on the other hand, seems as remote as it was in the 1950's. Understanding children's stories, generating an abstract from an article, conducting conversations and the like are still in the infant stage at best, despite significant theoretical progress.

This book deals with research in computer comprehension and production of natural language. The 14-paper collection encompasses linguistic, cognitive, and pragmatic aspects of natural language processing.

The editors’ introductory paper classifies computational models as cognitive, pragmatic, and linguistic. It defines linguistic competence and performance, emphasizing the importance of the former in systems of natural language processing. Other papers in the collection touch upon such topics as parsing, knowledge representation, text structures, language generation, speech acts, and cognitive models of language comprehension and production.

Two papers are on parsing, “A Framework for Integrating Syntax and Semantics” and “Knowledge Representation and Natural Language: Extending the Expressive Power of Proposition Nodes.” They differ from traditional syntactic parsing in that they purport to incorporate semantic processing into a parsing system by manipulating knowledge structure. These models are partially and experimentally implemented.

Four of the papers are concerned with semantic structure. These present various semantic-modeling schemes: a knowledge representation system based on propositions in semantic nets, a linguistic formalism representing the meaning of natural language text, a mental model of language's deep structure with primitives ‘smaller’ than words, and a memory representation and processing requirements for narrative structures.

Papers by Hovy and Schank and by McKeown deal with language generation. Hovy and Schank describe a program that generates sentences from the knowledge structure in one's memory. The sentences are generated according to the speaker's and hearer's models that include their social, emotional, and pragmatic status. McKeown describes a mechanism for text generation based on the focus of attention.

Two papers, “Viewing Parsing as Word Sense Discrimination: A Connecting Approach” and “Towards an Integral Model of Language Competence,” offer cognitive models of language understanding and generation. Structures and processes behind linguistic communication are the focal points here. The first is a computational theory of language comprehension based on a human brain's parallel architecture. The second describes a comprehensive model of language understanding whose components include models of the physical world, interpersonal communication, verbal interaction, speech acts, coherent text, and sentences.

Another paper is on speech acts, presenting a theory on cognitive processes underlying interpersonal actions. And another analyzes topic and focus, and the roles they play in correcting sentences. Finally, there is one on the state of the art in Chinese language processing, perhaps for westerners interested in how computers handle Chinese characters.

The history of natural language processing shows that we need theoretical models to have high-performance systems. Ever since Winograd’s well-known system, arduous attempts have been made to unify all aspects of understanding natural language. Theories of frames, scripts, and schemas are most notable. However, the realization has grown that more and more devices need to be incorporated in natural-language-understanding systems. Hovy and Schank's paper demonstrates that point clearly.

The book focuses on language competence rather than performance. The editors note that “Performance can in fact be measured by mere input output experiments, but a concrete reliability of the system behavior can only be guaranteed by detailed knowledge and evaluation of its internal operation.” The book also tries to unify “the different paradigms of artificial intelligence, linguistics, and cognitive science in the fundamental issues of developing computational models of language competence.”

Papers in this book are highly theoretical and exploratory. Since they are forerunners in the field and none of them is easy to read, the book is not suitable for undergraduate courses. In addition, trying to abstract concrete techniques from the papers for implementation of a particular system is likely to be disappointing. However, the book is highly valuable and recommended for those seriously engaged in natural language understanding.
Analysis of Speedup in Distributed Algorithms

John P. Fishburn (UMI Research Press, Ann Arbor, Mich., 1984, 118 pp., $39.95)

John D. Holt,
California State University at Fresno

Research into the design and analysis of parallel algorithms has yielded a wealth of literature and some significant results. Although current technology is still unable to benefit from many of these results, no doubt the accumulating research will eventually serve it well.

This book contributes some interesting variations of known algorithms and in-depth analyses and comparisons of their performance and efficiency. It is unusual to find such highly technical content in book form and in such a small book. This appears to be a consequence of its origin — the author's doctoral dissertation — and of an unfortunate adherence to dissertation format.

Previous research in relevant areas is encapsulated in a four-page chapter. Fishburn summarizes the significant work in the application of distributed algorithms to sorting techniques, numerical methods, and graph theory. This summary is almost too compact, and the relevance of individual previous results is not clear at this point. Some indication of the improvements about to be encountered would whet the reader's appetite for the three chapters that follow.

The next and longest chapter (one third of the book) is devoted to an interesting description of parallel alphabeta search algorithms. The author discusses and analyzes the basic algorithm and several implementation variants and presents new theorems, establishing time and space bounds on parallel traversal of different arrangements of the search tree. He compares the principal parallel algorithms — parallel alpha-beta and mandatory-work-first — to complete the chapter.

The lack of an end-of-chapter overview of the significance of the author's work left me with the same sense of disappointment experienced by the reader of a good mystery novel who finds the last page missing.

Fishburn uses the Dirichlet problem in examining the properties of locally defined iterative methods. He compares the speed of the Jacobi, Gauss-Seidel, and Successive Over-Relaxation methods and analyzes several parallel algorithms for solving the Dirichlet problem with the Jacobi method.

He considers various processor communication topologies that provide an interesting representation of load distributions. In a style consistent with the rest of the book, this chapter also comes to an abrupt end. An informative conclusion that reviews the principal ideas and results of the chapter would be much more satisfying.

The final investigative chapter discusses quotient-network and large-network algorithms. Large-network algorithms assume as many processors as there are points in the problem space, while quotient-network models are based on restricted single-instruction, multiple-data architectures. A particularly intriguing section of this chapter describes a procedure for emulating large networks by smaller networks with similar communication architectures.

The book concludes with a brief chapter summarizing the content of each of the three main chapters. Some of the material included here would have made the earlier chapters more complete. Readers may find it useful to read this last chapter before tackling the three chapters preceding it. An appendix proposes three optimization strategies for serial alpha-beta searching.

Obtaining the most from this book requires thorough familiarity with the previous work in this area. The significant results relevant to the author's treatise are stated too briefly to serve as an adequate review. Without this background, the reader may have to apply substantial effort to verify some of the steps in the analytical expositions and the theorems.

One might expect to see material such as this presented as three research papers in an appropriate journal, where it would receive the exposure it deserves. The author's results are significant and do contribute to the field of parallel algorithms. However, the book might have broader appeal and could be more instructive if the dissertation format had not been so rigidly followed.

Verification and Validation of Real-Time Software

W.J. Quirk, ed. (Springer-Verlag, New York, 1985, 245 pp., $29.50)

Raymond J. Rubey, Softech

Verification is a term that, when applied to software, has at least two distinct meanings. This reflects the gap between the academic computer science culture and the commercial software development culture.

The academic meaning, exemplified by the book Formal Methods of Program Verification and Specification (Berg et al., Prentice-Hall, Englewood Cliffs, N.J., 1982), describes the process in which mathematically rigorous techniques are used to show or "prove" that a program matches its specification.

The commercial meaning, exemplified by the book Software Verification and Validation: Realistic Project Approaches (Deutsch, Prentice-Hall, Englewood Cliffs, N.J., 1982), describes a process that employs a great many practical and intuitive techniques to discover errors made in going from one phase of a program's development to the next.

The goal of both processes is the same: error-free software. Their means of achieving this goal is quite different.

To determine what a book on verification is about, one must determine which meaning the author intends. One clue is if the alliterative and sometimes redundant word "validation" is appended to the word verification. If so, the author will be probably be describing practical and intuitive techniques rather than a formal mathematical proof process.

Verification and Validation of Real-Time Software is a small bridge between the two meanings. It describes practical, intuitive, nonrigorous, and applied techniques for developing error-free software and the less directly applicable formal specification, correctness proof, and modeling techniques.

Quirk's book is not a how-to book for any of the methods described. (For that, I'd recommend the books mentioned earlier.) Rather, it should be considered an assessment of the state of the art and an index to further reading on a wide range of software reliability concerns.

The book covers five main topics: software life cycle, structural analysis and proof, systematic testing, statistical testing, and simulation and system validation. All topics are treated somewhat superficially, but they are cross-referenced to an excellent bibliography. There is no index.

In the software life cycle discussion, the role of the program specification in achieving reliable software is discussed with brief descriptions of verification or validation.
activities that may be appropriate at each development phase. Recent research involving some of the more formal specification languages (such as PSL/PA) and automated support tools is cited. However, not enough details of this work are presented to assess its merit.

The section on structural analysis and proof covers a wide range of topics in summary form, including modeling with petri nets, Hoare/Floyd-based proving techniques, software fault-free analysis, and inspection and walkthrough review techniques.

This section is indicative of the approach taken in the book. A wide range of methods is described in summary fashion, some very practical and subjective (such as code walkthroughs) and others offering promise only after further substantial progress is achieved (for example, the axiomatic approach to the correctness proof for parallel programs).

The systematic testing section contains very brief treatments of path testing, domain testing, mutation analysis, and structured testing strategies (such as bottom-up and top-down). The use of metrics to measure test coverage and tools to automate some test activities are also briefly described.

The most original contributions of the book appear in the section on statistical testing. Statistical testing aims to show that the probability of a software failure falls below a certain limit. It is intended for complex applications where systematic testing is impractical. The probability of failure is a function of the extent to which the dependencies and properties of a program’s behavior have been tested.

Whether this approach for determining the required number of test cases for a given reliability will result in realistic test strategies has not, as yet, been demonstrated. However, this section provides an interesting viewpoint.

The section on simulation and system validation is very general, in large part because of the application-dependent nature of simulation and system validation.

Although the book has the term “real-time” in its title, it includes little discussion of specific real-time topics. In many of the contexts where real-time is discussed, only the complexity of and need for reliability in real-time applications is considered.

With the exception of the section on statistical testing, most of the book ignores the very difficult real-time problems such as deadline constraints, fault tolerance and recovery, interacting synchronous and asynchronous functions, man-machine interfaces, and dynamic allocation of system resources.

Edited technical books like this one often turn out to be collections of overlapping papers from a conference workshop. It is pleasant to report that this is not the case here: all the separate authors’ contributions are well-coordinated and consistent.

The authors are affiliated with several European organizations, and I found it interesting that there is little difference between their concerns and solutions and current practice in the United States.

This book will interest individuals concerned about the verification problem, either from the academic computer science viewpoint or from the commercial software development viewpoint.

Readers having one of these viewpoints will be exposed to a little bit of the other.

---

**Modula-2 Special Issue**

The software community is witnessing a migration away from high-level languages like Fortran, APL, and Cobol toward languages like Ada, C, Lisp, and Prolog that are more appropriate to their applications. Modula-2 is one of these "new wave" languages, but its elegance and versatility could make it the programmer’s first choice.

Modula-2 will be the focus of the November issue of IEEE Software. In addition to feature articles, this issue will include a special section on new Modula-2 products. To be included in this review, send new product announcements by October 1, 1986 to:

IEEE Software
Attn.: Modula-2
10662 Los Vaqueros Circle
Los Alamitos, CA 90720