ENCYCLOPEDIA OF COMPUTER SCIENCE AND TECHNOLOGY

EXECUTIVE EDITORS

Allen Kent *James G. Williams*

UNIVERSITY OF PITTSBURGH
PITTSBURGH, PENNSYLVANIA

ADMINISTRATIVE EDITORS

Rosalind Kent *Carolyn M. Hall*

PITTSBURGH, PENNSYLVANIA

VOLUME 24
SUPPLEMENT 9

MARCEL DEKKER, INC. • NEW YORK and BASEL

KNOWLEDGE AND DATA ENGINEERING

INTRODUCTION

According to Webster's dictionary [*1*], data refers to numerical information suitable for computer processing, while knowledge refers to the sum or range of what has been perceived, discovered, or learned. Knowledge can be considered data at a high level of abstraction and can be processed by a computer when it is represented as data. The distinction between these two concepts in terms of computer processing is usually vague [*2,3*]. For example, a statement that data engineering is a growing field is a piece of knowledge. This statement has to be substantiated by facts (or data) or by algorithms (or programs) that can generate the supporting evidence. Since the amount of data necessary to characterize a piece of knowledge can be infinite in size, so the meaning of a piece of knowledge can be imprecise or uncertain. In general, knowledge can be considered as a compact and sometimes imprecise way of representing a body of data.

Data and knowledge engineering collectively refers to the methods, algorithms, and systems for design, utilization, and maintenance of data and knowledge. This involves (a) methods for automated acquisition and learning of new data and knowledge, (b) modeling, design, access, control, and evaluation of data and knowledge engineering systems, (c) representation, language, and architectural supports, and (d) deployment, evolution, maintenance, and standardization of data and knowledge engineering systems with existing and emerging technologies. The problems involved in the area of knowledge and data engineering are continuously evolving, as new applications arise and emerging technologies become mature.

Different degrees of abstraction of a given problem in knowledge and data engineering may be required, depending on the complexity of the problem involved. Some problems require less abstraction and therefore can be implemented easily in hardware/software. An example of this type of problem is the design of a hardware selection unit for a relational database. Other problems are more complex and require a hierarchy of abstraction to simpler problems before they can be solved. These simpler problems by themselves may not represent realistic, efficient, or realizable implementations. An example is a distributed database that can handle concurrent accesses and updates. The concurrency control problem is extremely complex if all characteristics of the physical distributed system and user behavior have to be considered. To solve this problem, a simplified model of the communication network, processors, and user behavior has to be assumed. A concurrency control algorithm is then developed for the simplified model, and the algorithm is mapped onto the physical distributed system. Since simplifying assumptions have been made in developing the algorithm, the algorithm actually implemented on the physical system may not be totally efficient.

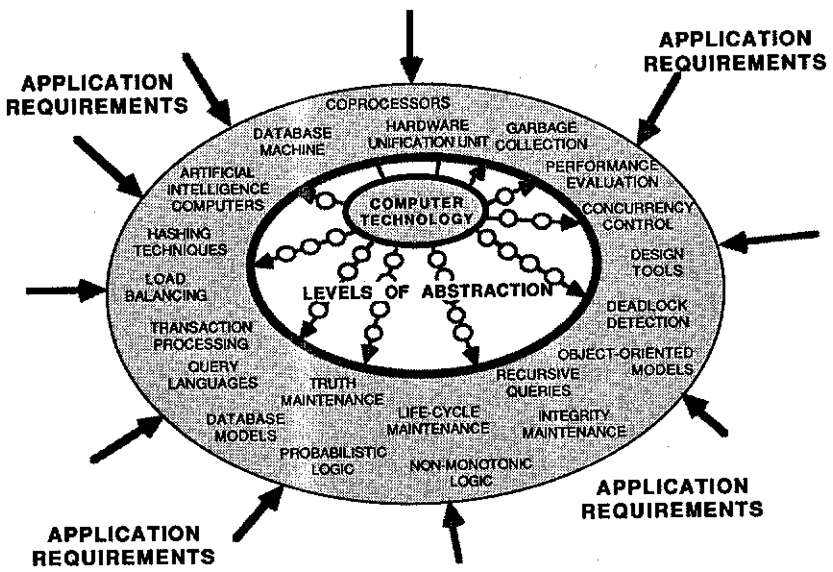

FIGURE 1. Problems of various levels of abstraction in knowledge and data engineering

Figure 1 is a conceptual view of problems in knowledge and data engineering and the relationship to applications and technologies. Examples of problems with varying degrees of abstraction are also illustrated in the figure.

The early years of computing research were dominated by information processing. Loosely speaking, information refers to bits that are stored in computer memories. These bits include data, software, and knowledge represented in data and software. In von Neumann computers, for example, data and programs are stored in the same memory areas. Therefore, data can sometimes be interpreted as programs, while programs are sometimes considered as data. Since knowledge can be treated as a general class of data, its characterization may be uncertain or imprecise. Knowledge can be represented in computers as data or software, or as a mixture of both.

With the growing complexity of applications, it was soon discovered that techniques for designing and processing software were quite different from techniques for processing data, and that techniques for acquiring and managing knowledge were different than techniques for data processing and information processing. This discovery led to the development of structured programming, database processing [4,5], and artificial intelligence [6,7] in the 1960s and 1970s. Early data and knowledge processing systems were small in scale: modeling, design, and management could be handled by one or a few experts.

The ever-increasing complexity of applications and information processing systems in the 1980s and the increasing need to capture and manage abstract knowledge require the collective effort of a large number of experts. Data and knowledge can no longer be handled by individuals alone, and its management must be treated as an engineering discipline.

Data and knowledge engineering systems are feasible because of recent advances in technology and computer architecture. Existing technologies, such as VLSI, VHSIC and fiber optics; and emerging technologies, such as lightwave technologies, sea of gates, three-dimensional VLSI, and superconductivity, promise faster computers, more memory, and higher networking bandwidth. Research on multiprocessing and distributed processing also allow faster and parallel computers to be utilized efficiently.

In this article we provide an overview of the research and development directions in knowledge and data engineering. We classify research problems and approaches in this area, and discuss future trends. We do not attempt to survey all related work and references in the area, as the scope of work in this area is extremely broad. Further, because knowledge and data engineering have been applied in many areas, it is not possible to provide a complete discussion of each application.

CHARACTERIZATION OF KNOWLEDGE/DATA ENGINEERING PROBLEMS

Solutions to problems in knowledge and data engineering applications depend on the characteristics of these problems. In this section, problems are classified according to three attributes: (a) completeness of knowledge or data in the environment, (b) accuracy of knowledge or data available in the environment, and (c) knowledge about the objective and/or constraints of the problem. Table 1 shows the eight possible combinations of these three attributes and offers examples for each class. Possible techniques in solving problems in each class are also indicated.

Knowledge/data available in the environment can be complete or incomplete. When it is complete, no additional knowledge is needed to solve the problem. In contrast, when it is incomplete, heuristics must first be applied to find a complete set of knowledge/data. In some cases, the amount of knowledge may not be very large or unbounded, and heuristics must be applied to define a restricted set of knowledge/data that can be used in solving the problem. For example, in proving a theorem, it may not be possible to define all of the necessary axioms. As a result, the original problem has to be heuristically modified to prove the theorem based on the *available* axioms.

The knowledge/data available in the environment may be exact or inexact. When it is exact, it can be represented in numerical or logical form. Data/knowledge is inexact when the number of possible cases is infinite, and it is impossible to enumerate or represent all of them. In such cases, before the problem can be solved, heuristics must first be applied to either define a finite number of possibilities or redefine the meaning of exactness so that what is available can be treated as exact.

For example, the birthdays of people in a group are points in a time spectrum, which must first be converted into real numbers of finite precision before the youngest person can be found. In a second example, recognizing an object in a given image involves incomplete data, because there is an infinite number of possible orientations and features of the object concerned. To solve the problem, it is necessary to identify a finite and reasonable number of features of the object and heuristically match them with those found in the image. As another example, finding points with the lowest energy in a bounded amount of space is a problem with incomplete and inexact data. Although there is a finite number of particles in such a space, it is not possible to examine the energy value of each. Measurements of a finite degree of precision will be made for a reasonable number of points, and interpolations will

TABLE 1. Characterization of Knowledge and Data Engineering Problems

Completeness	Exactness	Objective/ Constraints	Examples	Processing Techniques
Complete	Exact	Well–defined	Querying a relational database	Parallel processing, combinatorial searches
		Ill–defined	Querying a statistical database	Heuristics must first be applied to define objective/constraints Learning algorithms is one possible heuristic
	Inexact	Well–defined	Finding the youngest person in a group; recognizing an object in a given image	Heuristics must first be applied to find better data/knowledge or restrict data to a fixed degree of precision
		Ill–defined	Recognizing language in a voice pattern	See comment above for parts concerning ill–defined objective/ constraints or inexact data

Incomplete or unbounded	Exact	Well–defined	Proving a theorem	Heuristics must first be applied to find a complete set of data/knowledge; if this is not possible, heuristics must be used to define a restricted set of knowledge/data that can be used in solving the problem
		Ill–defined	Deciding where to process a task to minimize the average response time	See comments above for parts concerning incomplete data or ill–defined objective/constraints
	Inexact	Well–defined	Finding points with the lowest energy in a bounded space	See comments above for parts concerning incomplete or inexact data
		Ill–defined	Predicting changes in weather	See comments above for parts concerning incomplete or inexact data or ill–defined objective/constraints

be made for intermediate points. Heuristics are, therefore, applied to redefine the meaning of exactness and completeness in the original problem.

After knowledge/data used in solving the problem is defined or restricted, the problem can be solved by finding a solution in the given solution space. This can be represented as the objective of what the solutions must achieve and the constraints that the solutions must satisfy.

An objective of a problem may be well-defined or ill-defined. A well-defined objective can be represented exactly in terms of the measurable parameters of the problem environment so that it is possible to compare the quality of one solution with another. An ill-defined objective may involve either parameters that cannot be expressed in measurable terms or an unknown relationship among measurable parameters. As a result of these conditions, it is not possible to compare the quality of alternative solution. An ill-defined objective must first be heuristically transformed into a well-defined one before the problem can be solved. This may involve restricting consideration to measurable parameters and defining the objective as a function of these parameters.

The constraints that the solution must satisfy may also be either well-defined or ill-defined. In a well-defined set of constraints, it is possible to enumerate all candidate solutions. In an ill-defined set of constraints, a potentially infinite solution space is defined. Problems with ill-defined constraints must first be heuristically transformed so that all constraints are defined precisely in terms of measurable parameters of the environment.

The querying of a statistical database is an example of a problem with an ill-defined objective because one can only obtain probabilistic rather than precise answers to queries. The recognition of natural language from a voice pattern is a problem with inexact data and ill-defined objective because the inputs are represented as analog signals that are prone to errors, and because there are an infinite number of possible pronunciations of a word. In these conditions, the constraints of the solution space cannot be defined precisely. For this problem, only a finite number of possible pronunciations can be tested. The decision of where a task should be performed in a distributed computer system is another problem with an ill-defined objective because the relationship between the objective (minimizing the response time) and the measurable parameters (workload statistics) is unknown. A heuristic function relating the response time and the measurable quantities must first be defined before the problem can be solved. Finally, the prediction of weather is a problem which may involve an infinite amount of inexact input data and ill-defined objectives and constraints.

RESEARCH ON KNOWLEDGE AND DATA ENGINEERING

The goal of research in knowledge and data engineering is to study and design systems to support knowledge and data engineering applications. Approaches to achieving this goal range from theoretical analysis, mathematical modeling, simulations, and prototyping, to complete system integration.

The design process of a knowledge and data engineering system is extremely complex and iterative. It is not possible to classify all of the variations of this process without oversimplifying it. However, the resulting design has notable characteristics depending on its starting point, which usually dictate the nature of the final result and can be used as a classification scheme for results in this area. The starting point can be either application driven or technology driven.

In an application-driven or a topdown approach, research is focused on first identifying and refining the requirements of the problem. The knowledge necessary for solving the problem is then acquired, and algorithms are designed and mapped to physical systems. In this approach, the capabilities of the resulting system are a good match for the requirements of the application. However, in some instances, the requirements are specified in such a way that a system cannot be physically realized. In this case the design process has to be iterated through repeated refinement of the requirements and designs.

In contrast, a technology-driven or a bottom-up approach determines technological capabilities and limitations first. These limitations lead to the design of realizable systems on which the applications can be mapped. Tradeoffs on performance are then evaluated, and one of the candidates is selected as the target system. The problem with this approach is that the resulting design may not be a perfect match for the application requirements. The design process has to be iterated by modifying the design until the application requirements are satisfied.

Research on data and knowledge engineering addresses the theory, analysis, design, development, evaluation, and maintenance of new data and knowledge management techniques, methodologies and systems that can support specialized applications in a cost-effective manner. It also emphasizes a better understanding of problem-solving methods with respect to the applications concerned. Related issues in accomplishing these goals include studies on the theoretical aspects, design tools and methodologies, design tradeoffs, representation and programmability, algorithms and control, reliability and fault tolerance, and designs using existing and emerging technologies. In the following text, we discuss each of these issues by stating its goals and showing some representative examples. It must be pointed out that all of these issues must be considered in designing a complete working system.

Figure 2 depicts a hierarchical organization of studies in knowledge and data engineering. The problem to be solved may be specified in an imprecise form and must be successively refined into a form that can be implemented physically. Applicable methods for such refinement include synthesis, analysis, optimization, verification, validation, testing, and maintenance. Representation on a higher level is influenced more by the applications, while representations on a lower level are influenced by technological constraints. Table 2 shows examples of problems that are studied in each issue.

Theory

Theoretical studies involve abstracting the problem domain and studying the properties and limitations of the abstract problem. Such an abstraction is often necessary to transform the original problem, which is too complex to be studied in its entirety, into a simpler and more manageable form. Moreover, because the abstract model may cover a number of different problems in the class, the results developed will be applicable to these problems as well. Theoretical results can be used to guide the selection of algorithms and design tradeoffs.

Programmability and Representation

This issue involves choosing or designing the appropriate knowledge and data representation schemes and developing the necessary hardware and software support mechanisms. A choice must be made between using an existing representation scheme, such as one that is declarative, procedural, functional, or object-oriented; or designing a new representation scheme. The scheme may have to take into account uncertain, fuzzy, or incomplete knowledge and data in the application, and may need to support data and knowledge engineering applications in a sequential or parallel environment.

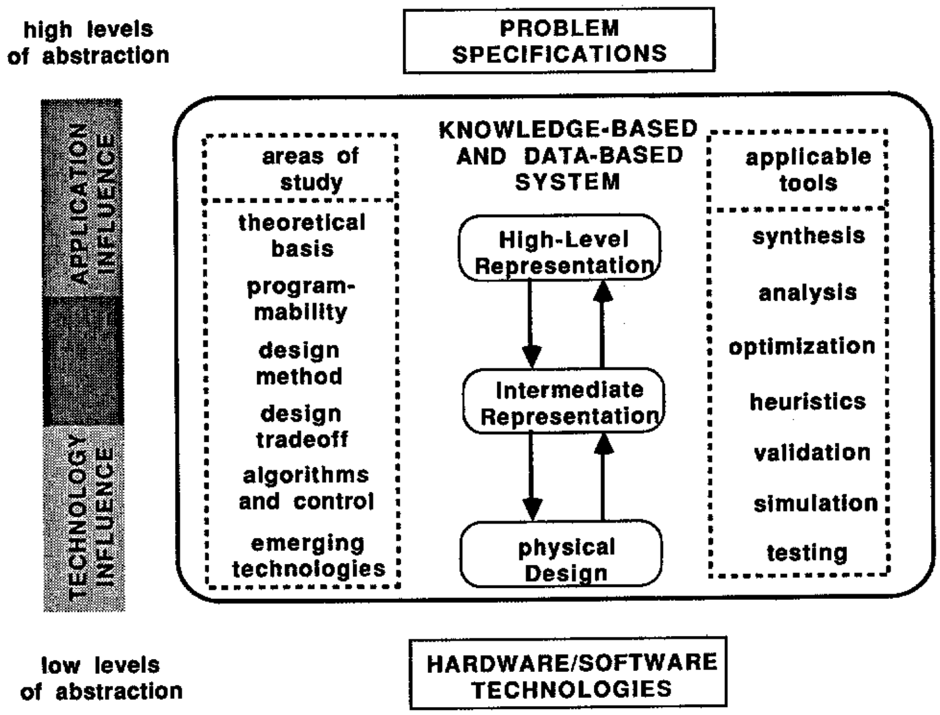

FIGURE 2. Design of knowledge-based and data-based systems.

Many of the issues of representation should be considered early in the design process when the problem semantics are still available. An essential issue to consider is specifying a minimal amount of information to allow control to be carried out efficiently in an implementation. This knowledge should preferably be specified in a form that is independent of the computer architecture on which the knowledge and data management system is implemented. Detection of this information for efficient control may have to be delayed until run time when more accurate knowledge about the resources is available.

Design Tools and Methodologies

The design of a knowledge-based and database system is a complex task, which can be greatly simplified by using computer-aided design tools. Systematic methods must be used for analyzing requirements, representing specifications, partitioning problems into a manageable size, designing algorithms, mapping algorithms into physical data and knowledge management subsystems, and maintenance of these systems. The process is iterative and stops when problem requirements, such as flexibility, extendibility, reconfigurability, user friendliness, reliability, and evolvability, and technological limitations, such as chip size, bandwidth, and packaging, are met.

TABLE 2. Examples of Data and Knowledge Engineering Issues

Theoretical Basis
Recursive queries
Database models [8–10]
Logic databases [11,12]
Logic and problem solving [13]
Nonmonotonic database and knowledge base [14]
Programmability and Representation
Truth maintenance [15]
Object-oriented models [16]
Query, design, and visualization languages
Knowledge and data representation schemes [17]
Design Methods and Tools
Design methodology
Life-cycle maintenance
Computer-aided design tools [18]
Design Tradeoffs
Standardization
Performance evaluation techniques and tools
Integrated voice, image, and data processing [19]
Intensional versus extensional data processing [20]
Algorithms and Control
Testing
Load balancing
Hashing techniques
Deadlock detection
Integrity maintenance
Transaction processing
Reconfiguration strategies
Concurrency control [21]
Integrity and consistency checks
Query processing and optimization [22]
File partitioning, replication, and allocation [23]
Garbage collection and virtual memory management
Fault detection, isolation, checkpointing, and recovery
Emerging Technologies
Optical computing system
Artificial intelligence computers [24]
Database and knowledge-base coprocessor

Design Tradeoffs

Design tradeoffs for data and knowledge management systems involve tradeoffs on hardware and software implementations, representation schemes, use of software languages, computational power of processors, communication bandwidth, storage capacity, cost, design of specialized functional units, operating system environment, and use of hardware technologies. A judicious selection of the components to be used in the system must be made in order to obtain a high-performance system with acceptable cost.

Algorithms and Control

This issue involves the design and evaluation of algorithms and control strategies for efficient and reliable operations in data and knowledge management systems. It also involves methods to acquire, test, store, access, and maintain reliable data and knowledge for a given application.

The design process is usually hampered by imprecise or incomplete knowledge at design time, which necessitates developing and using heuristics for control. These heuristics may underutilize resources and cause anomalous behavior with respect to resources. In the latter case, increasing the resources available to a system may result in worse performance by the heuristics (with respect to the objective).

Another problem in the design of algorithms is that they depend largely on the characteristics of the application involved and the experience of the designers. Automated methods for designing and evaluating algorithms are not available at this time. Computer-aided tools, such as automated learning methods, may be very useful in adapting existing algorithms to new applications.

Emerging Technologies

This issue involves exploration of the limitations and impact of emerging technologies on the design, evaluation, programmability, and application span of data and knowledge engineering systems. Emerging technologies, such as optical processing [25], artificial neural networks [26], high bandwidth optical links, new packaging technologies [27], large memory devices, and new architectural concepts, may impact the ways that design tradeoffs are performed and algorithms are developed. They may also result in different design methods, problem-solving strategies, and control algorithms for future data and knowledge engineering systems and impact the evolution of current systems.

FUTURE TRENDS

Improvements in knowledge and data engineering systems can come from faster technologies, better representation schemes, more efficient algorithms, automated software design methods, and better hardware/software architectures integrated with the available technologies.

Emerging Technologies

The basis for any computer system is the technology in which it is implemented. The design of a system is often driven by its cost, hence, the fastest technologies, subject to cost constraints, are used. New technologies may give higher performance, but often are prohibitively expensive. These improvements will likely bring a two to three orders of magnitude improvement in computational speeds in the next decade.

Knowledge Representations

Many new representation schemes have evolved in the recent past. These schemes may feature tools for knowledge and data capture and management. However, they are usually not directed toward any specific applications and many have to be modified or extended to tailor to the applications and computational environment. Another major problem is the lack of an overall technique to guide the evaluation and selection of a representation scheme. Research in this area could prove extremely valuable. Learning techniques for incorporating new knowledge about application domains into current solutions in a knowledge-intensive application may also impact knowledge and data engineering.

Algorithms

Research in the area of application-specific algorithms will have the greatest potential for speeding the solution of the given application. The development of new and improved algorithms for an application can be seen as finding alternative ways to incorporate knowledge about the application domain and the technology into the computer solution.

The design of better algorithms and knowledge representation schemes is an important complement to the tremendous potential offered by emerging technologies. Extension in the limit of processing power by new technologies is valuable, especially for real-time systems. However, the two to three orders of magnitude speed improvement offered by these technologies in the next decade will not greatly impact the size or type of knowledge and data engineering applications that are addressed today. Many of these applications involve huge search spaces of an exponential size; two to three orders of magnitude increase in computational speed will do little to extend the size of a solvable instance of such a problem [*28*].

Table 3 shows, for a given problem, the extension in problem size that can be solved by faster or parallel computers in the same amount of time as a single computer solving the same problem and assuming that linear speedup is possible. It illustrates that, for more complex algorithms, faster computers and parallel processing alone are only useful to improve the turnaround time of algorithms that can already be evaluated on existing computer systems.

TABLE 3. Extension in Problem Size Due to Parallel Processing Assuming Linear Speedup and the Same Amount of Time as Sequential Processing

	Number of processors		
Algorithm complexity	1	2	N
N	N	2N	N^2
N^2	N	$\sqrt{2N}$	$N^{1.5}$
N^k	N	$2^{1/k}N$	$N^{1+1/k}$
2^N	N	N+1	$N+\log_2 N$

Software Architecture

Software architecture is an integral part of a knowledge and data management system. Generation of new software environments, tools, and languages will probably rely on amalgamation of known representation techniques and software design methodologies. Software development systems and automated intelligent assistants represent prime areas for advancement of data and knowledge engineering applications. The problems of verification and validation and continuous maintenance of programs, knowledge, and data are important related topics.

Hardware Architecture

As with software, hardware architectures are often based on known design techniques such as parallel processing and pipelining. Innovation for new architectural concepts may be made possible by new and emerging technologies, which affect the cost effectiveness of previous designs.

New hardware architectures are best utilized for operations that the computer performs frequently. Counter to intuition, identification of these tasks is very difficult. Operations may be instructions, parts of instructions, groups of instructions, or frequently recurring tasks. Identification of new and valid areas for development of new hardware architectures is an important area of research.

System Design

System-level design is often based on an overall design philosophy and may contain, for example, a mix of different computational models, and general and special-purpose functional units. An important driving force is the cost effectiveness of the design and its technological feasibility. A major difficulty, however, lies in integrating designs with different knowledge representation schemes, software architectures and hardware components, and evolving systems as technologies change.

ACKNOWLEDGMENT

Research supported by National Science Foundation Grant MIP 88–10584.

REFERENCES

1. H. Webster, *Webster's II New Riverside University Dictionary*, Riverside Publishing Co., Boston, 1984.
2. "Special Issue on Data Engineering," *IEEE Computer, 19*(1) (1986).
3. G. Widerhold, "Knowledge and Database Management," *Software, 1*(1), 63–73 (1984).
4. J. D. Ullman, *Principles of Database Systems*, Computer Science Press, New York, 1984, pp. 211–267.
5. G. Wiederhold, *Database Design*, 2nd ed., McGraw-Hill, New York, 1983.
6. A. Barr and E. A. Feigenbaum, *The Handbook of Artificial Intelligence*, Vols. 1, 2, and 3, William Kaufmann, Los Altos, CA, 1981, 1982.
7. J. McCarthy, "Program with Common Sense," in *Mechanization of Thought Processes*, Her Majesty's Stationery Office, London, 1959, pp. 75–84.

8. M. Brodie and J. Schmidt (eds.), "Final Report of the ANSI/X3/SPARC DBS-SG Relational Database Task Group," *SIGMOD Rec., 12*(4) (1982).
9. E. Codd, "Relational Model of Data for Large Shared Data Banks," *Comm. ACM, 13*(6), 377–387 (1970).
10. E. F. Codd, "Extending the Database Relational Model to Capture More Meaning," *ACM Trans. Database Sys., 4*(4), 397–434 (1979).
11. H. Gallaire and J. Minker (eds.), *Logic and Databases*, Plenum Press, New York, 1978.
12. R. Kowalski, *Logic for Problem Solving*, North-Holland, Amsterdam, 1979.
13. J. Robinson, *Logic for Problem Solving*, North-Holland, Amsterdam, 1979.
14. D. G. Bobrow and P. J. Hayes (eds.), "Special Issue on Nonmonotonic Logic," *Artif. Intell., 13*(1 & 2) (1980).
15. J. Doyle, "A Truth Maintenance System," *Artif. Intell., 12*(3), 231–272 (1979).
16. T. Rentsch, "Object Oritented Programming," *SIGPLAN Notices, 17*(9), 51–57 (1982).
17. M. Minsky, "A Framework for Representing Knowledge," in *The Psychology of Computer Vision*, P. H. Winston (ed.), McGraw-Hill, New York, 1975.
18. C. V. Ramamoorthy, S. Shekhar, and V. Garg, "Software Development Support for AI Programs," *Computer, 20*(1), 30–42 (1987).
19. N. Q. Duc and E. K. Chew, "ISDN Protocol Architecture," *IEEE Comm., 23*(1), 15–22 (1985).
20. P. S. Rosenbloom, J. E. Laird, J. McDermott, A. Newell, and E. Orciuch, "R1-Soar: An Experiment in Knowledge-Intensive Programming in a Problem Solving Architecture," *IEEE Trans. Pattern Analysis Machine Intell., PAMI-7*, 561–569 (1985).
21. P. A. Bernstein and N. Goodman, "Concurrency Control in Distributed Database Systems," *Comput. Surv., 13*(2), 185–221 (1981).
22. A. R. Hevner and S. B. Yao, "Query Processing in Distributed Database Systems," *Transact. Software Eng., SE-5*(3), 177–187 (1979).
23. B. W. Wah, "File Placement on Distributed Computer Systems," *IEEE Computer, 17*(1), 23–32 (1982).
24. B. W. Wah and G.-J. Li (eds.), *Tutorial on Computers for Artificial Intelligence Applications*, IEEE Computer Society Press, New York, 1986.
25. L. C. West, "Picosecond Integrated Optical Logic," *IEEE Computer, 20*, 34–47 (1987).
26. Defense Advances Research Project Agency, *DARPA Neural Network Study*, Lincoln Laboratory, MIT, Lexington, MA, 1988.
27. J. F. McDonald, H. J. Greub, R. H. Steinvorth, B. J. Donlan, and A. S. Bergendahl, "Wafer Scale Interconnections for GaAs Packaging-Applications to RISC Architecture," *IEEE Computer, 20*(4), 21–35, (1987).
28. B. W. Wah, G. J. Li, and C. F. Yu, "Multiprocessing of Combinatorial Search Problems," *IEEE Computer, 18*(6), 93–108 (1985).

BENJAMIN W. WAH