PERSPECTIVES IN DEDUCTIVE DATABASES*†‡

JACK MINKER§

▷ I discuss my experiences, some of the work that I have done, and related work that influenced me, concerning deductive databases, over the last 30 years. I divide this time period into three roughly equal parts: 1957–1968, 1969–1978, 1979–present. For the first I describe how my interest started in deductive databases in 1957, at a time when the field of databases did not even exist. I describe work in the beginning years, leading to the start of deductive databases about 1968 with the work of Cordell Green and Bertram Raphael. The second period saw a great deal of work in theorem proving as well as the introduction of logic programming. The existence and importance of deductive databases as a formal and viable discipline received its impetus at a workshop held in Toulouse, France, in 1977, which culminated in the book *Logic and Data Bases*. The relationship of deductive databases and logic programming was recognized at that time. During the third period we have seen formal theories of databases come about as an outgrowth of that work, and the recognition that artificial intelligence and deductive databases are closely related, at least through the so-called expert database systems. I expect that the relationships between techniques from

Address correspondence to Professor Jack Minker, Department of Computer Science, University of Maryland, College Park, Maryland 20742.

Received April 1987; accepted July 1987.

*Invited paper.

†This paper is an expansion of an invited talk presented at the Principles of Database Systems Conference, San Diego, California, 23–25 March 1987.

‡This paper is dedicated to Kent Curtis, Division Director, Division of Computer and Computation Research, of the National Science Foundation. Support from his activity was instrumental in my work on deductive databases. The strength of computer science in the United States is due, in large measure, to his foresight and leadership.

§Supported by Air Force Office of Scientific Research Grant AFOSR-82-0303, Army Research Office (ARO) Grant DAAG-29-85-K-0177, and National Science Foundation grant IRI-8609170.

©Elsevier Science Publishing Co., Inc., 1988
52 Vanderbilt Ave., New York, NY 10017 0743-1066/88/$3.50

formal logic, databases, logic programming, and artificial intelligence will continue to be explored and the field of deductive databases will become a more prominent area of computer science in coming years. ◁

1. INTRODUCTION

This paper is an expansion of an invited talk presented at the Principles of Database Systems (PODS) Conference in March 1987. It is, perhaps, fortuitous that I was invited to lecture, since the year 1987 is the 30th anniversary of my exposure to the field of deductive databases. Of course, in 1957, when I first started to work in this area, neither the field of databases nor the field of deductive databases existed. I will discuss my experiences, some of the work that I have done, and the work that has most influenced me during this time period. Since this is a personal memoir, I do not plan to provide a comprehensive perspective of the field. I hope that those whose work I leave out will not take offense.

2. BEGINNING EXPERIENCES IN DEDUCTION: 1957–1968

In 1957 I was working at the RCA Corporation. At that time RCA was awarded a contract to investigate the possibility of automating work in Army intelligence operations. I was among a group of people who spent time with U.S. Army intelligence experts to learn how they stored their material and performed their work. You must realize that in 1957 the field of computing was still in its early years and only primitive tools were available. I had been exposed to computers a few years earlier, about 1953, and there were no such things as database systems or even general programs to handle files. In 1957 IBM did not even offer peripheral discs or drums on their machines.

I headed a group on this project, termed ACSI-MATIC, to devise a database storage and retrieval system for the Army. Many interesting developments came out of the effort, although it was never adopted by the Army. My coworker and friend Herb Gurk and I were working on a system for Army intelligence personnel to store, retrieve, maintain, and perform inferences on their data. Based on our design, a prototype system was developed [61] which was able to accomplish the following tasks:

(1) process new data automatically into the files,

(2) find chains of related data and test for their consistency,

(3) classify new situations which are recognized as a result of new data,

(4) merge many separate pieces of data into a formal file record or a finished printed report,

(5) accept a variety of interrogations and file maintenance orders easily from the systems analysts and perform required retrieval and processing to produce the desired outputs in useful formats.

The ability to find chains of related data and test for their consistency did indeed perform deduction on data which had reliability estimates associated with them. As far as I know, this may be the earliest and first use of computers to do deduction

and derive revised estimates based on the original reliability values associated with the data. Of course, the deduction that we did was not very sophisticated. However, it was able to perform *modus ponens* and to check for what we called "cycles", i.e., chains of inferences that were of the form

$$A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \cdots \rightarrow A_n \rightarrow A_1.$$

Because of the questionable reliability of the information, the analysts were not willing to accept reasoning based on chains of deductions, but wanted reconfirmation that the chain was closed, that is, a cycle was formed. In a paper with some of my other coworkers at the time we discussed how this was accomplished [119]. Even with such confirmatory information the analysts were not always willing to accept such information as being true. One may characterize the system developed as generative, since deductions were performed at the time of data input.

I left RCA in 1963 and joined a consulting firm, the Auerbach Corporation. Although I did not do work on deduction for several years afterwards, I was intrigued by some efforts being done in the field and read articles that described the work. In particular, there were several places doing work in some aspect of deduction. These were the Rand Corporation, MIT, Computer Corporation of America, Stanford Research Institute, Stanford University, Systems Development Corporation, and the Hughes Aircraft Corporation. In 1965, Robert Simmons [181] wrote an excellent survey article that described work in the development of first-generation question-answering systems. Although his major focus was on natural-language comprehension, he did cover some approaches to inference making. Simmons traces work in question-answering (QA) Systems back to 1959. He wrote an update to his 1965 paper [183] that summarized work during the next five years and covered second-generation QA systems. I believe that his articles were extremely important and had a major impact in the field of QA systems.

Although I left the Auerbach Corporation in 1967 to go to the University of Maryland, I remained as a consultant to them. Jerry Sable and I received a contract from the Rome Air Development Center through the Auerbach Corporation, to perform a study on "relational data systems," as they relate to military applications. The contract was awarded in 1969, and our study was completed in 1970 [122]. In that study we reviewed work being performed on relational systems, with particular emphasis on those that performed some deduction. The study was in line with my research interests at the University of Maryland, as I wanted to go back to doing work on deduction for database systems. I would like to give you an idea of the type of efforts that were taking place at that time. Jerry and I visited the major organizations where work was being performed on relational systems and met with the key individuals who were doing the research. It was only when our final report was nearing completion that we learned of the extremely important paper by Codd [29] in which he discussed the foundations of a theory of relational databases. Codd's work has had a profound influence on databases.

Although Codd was the first to propose a formal theory for databases, based on relations, others had used the concept of a relational database without having specified a formal theory. One such group was at the Rand Corporation, where work was started in 1963 using a relational approach to store data.

I was particularly impressed with the work at the Rand Corporation. Roger Levien and Bill Maron [96–98, 113] developed a system, termed *relational data file*

(RDF), that had an inferential capability, implemented through a language termed INFEREX. An INFEREX program could be stored in the system (as in current systems that store views) and reexecuted if necessary. There were no formal theorem-proving techniques. Rather, the programmer had to specify the reasoning rules via an INFEREX program. The system was also able to handle credibility ratings of sentences in forming deductions. They actually had a working system running (see also [97, 99, 113]). The research was both at the level of implementation and at the theoretical level.

The work by Lary Kuhns [91–93] on RDF was, I thought, particularly important. I believe that Kuhns may have been the first to recognize that there were classes of questions that were, in a sense, not reasonable. For example, consider a database with authors and books listed. The statement "Reichenbach wrote *Elements of Symbolic Logic.*" might be in the database. Whereas the question "What books has Reichenbach written?" is reasonable, most individuals would agree that the question "What books has Riechenbach not written?" or the question "Who did not write *Elements of Symbolic Logic*?" are not reasonable questions. I believe that this is the first time that the issue of negation in queries was explored. Kuhns related the imprecise notion of "a reasonable question" with a precisely defined notion of a *definite formula.*

The notion of definiteness is derived approximately as follows:

Definition. Given a set of sentences S, a dictionary D_S containing known terms, a particular query Q, and an arbitrary name n, Q is said to be *semidefinite* if and only if for any name n the answer to query Q is independent of whether or not D_S contains n. Q is said to be *definite* if and only if Q is semidefinite on every sentence set S.

Robert DiPaola [39, 40], who also was associated with the RDF project, proved that there is no algorithm for determining whether or not a query is definite. His paper may have been the first to use the theory of computing to obtain results on databases. As will be described later, there is continued interest in this subject, explored first by the group at the Rand Corporation.

Kuhns [93] also dealt with the general problem of quantification in query systems; see, for example, [150]. Somewhat related to the work by Kuhns were a number of papers in the late 1950s and early 1960s devoted to a general theory or formalization of questions. See the papers by Aqvist [4], Belnap [11], Carnap [16], Harrah [63], Jespersen [75], and Kasher [78].

Another interesting effort dealing with relations and deduction was made in 1966 by Tom Marill [33–35, 112], who developed a system termed the *relational structure system* (RSS). The system consisted of 12 rules that permitted such capabilities as chaining. He used a *deduction procedure* in what he termed, a "breadth-first-followed-by-depth" manner. Other work during that time was performed by Hugh Love and his colleagues [110, 174, 175] on a system termed the *associative store processor* (ASP).

One cannot leave this beginning period without taking note of work at MIT and Stanford that culminated in a system at the Stanford Research Institute. At MIT, Bertram Raphael [155–157] developed a system termed *semantic information retrieval* (SIR) as part of his doctoral dissertation. SIR had a limited capability with

respect to deduction, using special rules. Raphael subsequently designed and implemented several successors to SIR when he moved to the Stanford Research Institute after completing his degree. One system, QA-1, again had a primitive deduction capability and used list structures to indicate how various kinds of facts might interact. Cordell Green, a doctoral student at Stanford University, also worked at the Stanford Research Institute. Green implemented QA-1 based on the design given in Raphael's dissertation. Green and Raphael [57] were the first to recognize the importance of work being done in automated theorem proving as it related to the problem of deduction in the context of databases. J. Alan Robinson, in 1963 [170], developed the *Robinson resolution principle*, a uniform method for performing automated deduction. Robinson's contribution to theorem proving techniques was significant and led to a great deal of work embodying his concepts. The Robinson resolution principle is a primary method in today's work in automated deduction. Green and Raphael developed a system, QA-2, that had as its base a formal theorem-proving mechanism based on Robinson's work. QA-2 was extended to the system QA-3, which was effectively QA-2 with some added heuristics. Robert Yates participated in the implementation of QA-2 and QA-3. Green and Raphael then extended the system to QA-3.5 [57], which permitted alternative design strategies to be tested within the context of the resolution theorem prover, and to provide a more flexible input language than that of the first-order predicate calculus. I believe that Green and Raphael should be given the credit for starting work in the field of deductive databases. It was their work on the QA systems that showed the viability of implementing deductive databases in a uniform manner.

In the conclusions to our study [121, 122], Sable and I recognized the importance of that work and stated:

> The use in QA-3.5 of a single rule of inference based on developments in formal logic is a significant development and important for plausible inferences that must be specified dynamically.

We further recommended with respect to the area of formal theorem-proving techniques oriented towards information retrieval that

> It is important to support this area since it appears to be extremely promising with regard to providing a general search capability. Considerably more research is required before the tool can be applied in a practical environment.

For details concerning several systems during this time period, see the above survey. Other papers that touched upon the work in deduction during this period are by Kochen [85], Minker and Sable [120], Simmons [182], Salton [173], Ash and Sibley [5], and Montgomery [137].

3. DEDUCTIVE DATABASES: THE FORMATIVE YEARS 1969–1978

The thesis by Cordell Green [58], related work by Green [59, 60], and the work by Green and Raphael [57] were influential in renewing my interest in deductive databases. I was very pleased when Cordell Green was recognized for his work by the ACM in 1985 and given the Grace Murray Hopper Award for his work accomplished before the age of 30. I was proud to have had an opportunity to recommend him for the award, as his work had been so influential to my research.

I thought that it would be essential to become engaged in automated theorem proving, since it was fundamental to work in deductive databases. Fortunately for me, my wife, Rita, was working as a computer professional at the National Institutes of Health in the Division of Computer Research and Technology. Chin Chang and Richard Lee were working in a different group within the same Division. In the spring of 1970 Richard and Chin were writing their book on theorem proving [24] and decided to offer a course at NIH using the book. Rita took the course. After each lecture she would come home with the latest handouts that they produced, and I would go over them diligently and send back corrections and suggestions. They also asked the students to do homework problems. I am pleased to say that Rita was the only one who answered all questions correctly. I am not at liberty to comment on whether she had any assistance.

I was very impressed with the book that they were writing, and I was surprised to learn that it had been turned down by a publisher. I told Richard and Chin that I would speak to my friend and colleague Werner Rheinboldt, who was and still is in charge of accepting computer books for Academic Press. I did, and, of course, Werner was also impressed with the material, and the book was accepted for publication. The book has been a tremendous success both technically and in the number of copies sold.

While I was reading the book by Chang and Lee, I undertook the development of an experimental theorem-proving system, the *Maryland refutation proof procedure systems* (MRPPS), with my students [123]. The MRPPS 2.0 system had a large number of inference systems and search strategies incorporated in it. Gerry Wilson and I [199] reported on a large number of experiments that we performed on this system to evaluate alternative inference systems and search strategies in automated theorem proving. Active experimental research on automated theorem proving was also being conducted by Wos and his group at Argonne National Laboratory [202–204] and by Bledsoe [13] at the University of Texas. Wilson and I drew many of our problems from those used by Wos's group [203] in their experiments. As part of his doctoral dissertation, Dan Fishman [44] investigated the problem of using sets to represent predicates and their arguments. We thought that this would be most useful for databases. As a result, we reported on "PI resolution" [45], an inference system that operated on sets. Indeed, a deductive database system, termed MRPPS 3.0 [126], designed and implemented with Gerry Wilson, James McSkimin, and Alan Aronson, was developed. The work in this area started in approximately 1970.

I was not aware of much other work being done in deductive databases during the period 1968 to 1974. However, in 1974 I attended the International Federation of Information Processing Societies (IFIPS) meeting in Stockholm, Sweden. Two papers impressed me at that time. One was presented by Jean-Marie Nicolas [144], and the other by Robert Kowalski [88]. The paper by Nicolas was on a deductive system, and the one by Kowalski was his seminal article in which he first proposed logic as a programming language. I was pleased to meet Kowalski at the conference, as I was impressed with the work he had been doing on his thesis and work that he had done with Pat Hayes [84] on semantic trees in automated theorem proving. His work on SL resolution [86] and on search strategies [85, 87] were, I thought, significant. Indeed, his work on search strategies generalized the A^* algorithm of Hart, Nilsson, and Raphael [64], and deserves greater recognition. In that paper he showed how search strategies generalize to theorem-proving contexts and introduced

the upwards diagonal search strategy. Gordon VanderBrug and I [193] described relationships among state-space, problem-reduction, and theorem-proving representations of a problem. With respect to his IFIPS paper, I must say that I was skeptical of logic as a programming language because of my work in automated theorem proving. The experiments that Wilson and I had conducted were not encouraging with respect to the time a theorem prover took to complete its search. I mentioned this to Kowalski, and he assured me that there would be no problem, since the complexity of theorem proving was not needed for normal programming problems. I was skeptical about his response at the time. It was not until a few months later, after reading his paper carefully, that I realized that Kowalski had based his remarks on the use of Horn clauses for logic programming. A Horn clause contains at most one positive atom, and when using SL resolution it does not require ancestry resolution. Consequently, a much simpler inference system results. Hill [68] termed this subset of SL resolution that operates on Horn clauses LUSH resolution. LUSH resolution permits an arbitrary literal in a clause to be selected for expansion, and uses neither factoring nor ancestry resolution. When restricted to Horn-clause programs, LUSH is complete and sound. LUSH resolution is the basis of all PROLOG interpreters. LUSH resolution is now referred to as SLD resolution, linear resolution with selection function for definite clauses [2]. A clause is said to be definite if it contains exactly one positive atom. One reason for the complexity associated with SL resolution theorem proving is the need to do ancestry resolution.

I also wrote to Alain Colmerauer to ask for a copy of PROLOG, which, I had learned, was a programming language based on logic. Alas, I did not receive a response from him. Colmerauer [30] was the first to implement a logic programming language. Both Kowalski and Colmerauer deserve credit for founding the field of logic programming. Cordell Green in his thesis and papers that emanated therefrom shows that he too had the basic idea of logic programming and also deserves credit for recognizing that one could use logic as a programming language. Another individual who deserves mention here is Carl Hewitt [67], who developed PLANNER, which in effect is a logic programming language. Kowalski, however, was the visionary who kept doggedly speaking about logic programming until the field became recognized as important.[1]

In addition to work in theorem proving, I was working in deductive databases. In particular, the MRPPS 3.0 [118, 126] system was oriented towards deductive databases, although it had many features useful for general theorem proving. Jim McSkimin, one of my students, and I introduced some concepts into the MRPPS 3.0 system [118]. One was the idea of incorporating a semantic net as part of the unification process, which we called semantic unification. A semantic network, as used in artificial intelligence, was essentially statements of the form $H(x) \leftarrow M(x)$. Hence, a semantic net can be represented easily in logic. Jim [117] also realized that one can compile the axioms of the above type once, in advance, and incorporate them into the unification algorithm. Thus, one could do type checking on arguments dynamically during unification. (See [1] and [151] for more recent work on this

[1] The idea of using logic as a programming language was proposed in 1966 by Louis Hodes [69]. He implemented part of his work and noted that restricted portions of the predicate calculus would be candidates for programming languages. An abstract of the unpublished paper appears in the August 1966 *Communications of the ACM*. Hodes has made the paper available to me.

subject.) As to representing semantic nets, Deliyanni and Kowalski [38] generalized the concept used in artificial intelligence and showed how to represent such networks elegantly in logic. A second useful idea that we introduced was that of using integrity constraints to cut off search in database problems. One should know, without an extensive search, that it is impossible, for example, to find a person who is both the mother and the father of any individual. King [82] in his thesis explored this topic more fully than we had done, for the subject of relational databases. Hammer and Zdonik [62] also used this idea. It was not until the 1980s that I thought about extending and formalizing these ideas. I shall discuss this later in the paper.

In 1976 I had a sabbatical leave and decided to spend some time visiting researchers in Europe. I was particularly interested in visiting Jean-Marie Nicolas in Toulouse. Although I had heard his talk at IFIPS, I had not met him. It seemed to me that he was one of the few people doing work in deductive databases. I was also interested in visiting Bob Kowalski at the Imperial College of London. Both Nicolas and Kowalski were kind enough to invite me to come. Both visits were rewarding for me. I met the very fine group at Toulouse. Hervé Gallaire, whom I had not known previously, was head of the Computer Science Department, and Nicolas was a member, as were Robert Demolombe, Claudine Lassere, K. Yazdanian, and Guy Zanon. I gave several lectures on my research, heard about their work, and had a very fine exchange. Nicolas discussed the work that he was doing with Yazdanian on integrity-constraint checking, and Demolombe discussed some of his work. Guy Zanon, whom I met there, then came to Maryland in 1977 as a student and to work with me. Gallaire and Nicolas asked me what I thought about the idea that they should hold a Workshop in Toulouse on deductive databases. I told them that it was, indeed, a very fine idea and that I would be pleased to participate. I suggested that if there were a workshop, then assuming that the papers were good, a book should be published that included the best papers.

At the Imperial College of London I met with Bob Kowalski and his budding group, and learned about the research in which they were engaged. David H. D. Warren was also visiting Kowalski's group, and I was pleased to meet him. The success of PROLOG as a useful language is due to Warren's [196, 197] development of an efficient interpreter and making it available to others. I also spoke with Keith Clark, who had just joined the faculty at Imperial College. Keith told me about his work on negation and how one could characterize it as the unstated "only if" part of "if and only if" statements. The added "only if" statements effectively complete the database. If we are dealing with a Horn clause program P, then from P one can deduce only positive facts. An interpreter that uses SLD resolution (complete and sound for Horn clauses) can attempt to prove a ground atom and show if it is a logical consequence of the program P. However, it cannot be proved if a query $\neg Q$ is a logical consequence of P, since the union of P with the negation of $\neg Q$, that is, Q, is easily shown to be satisfiable. The rule, *negation as finite failure* is used to avoid this problem. The rule states that if Q is ground and in the finite failure set of P, then $\neg Q$ holds. That is, if the attempt to prove Q from P fails at a finite distance along every possible path, then $\neg Q$ can be assumed to be true. The significance of Clark's [27] result is that he proved that if one augments the program P, together with the "only if" half of each of the clauses, called the *completion of* P [comp(P)],

plus some axioms for equality, then if Q is in the finite failure set of P, then $\neg Q$ is a logical consequence of comp(P). This provides a soundness for the negation-as-finite-failure rule.

Gallaire and Nicolas decided to hold their Workshop on Logic and Databases in Toulouse in November 1977. Nicolas was the primary organizer and selected the individuals who would be invited. I felt strongly that if the Workshop were a success, it would be important to have a book in which the major articles would appear. In that way we would focus attention on what Gallaire, Nicolas, and I all agreed was an extremely important field.

The workshop was successful beyond my imagination. There were many significant talks and contributions. The paper by Nicolas and Gallaire [146] focused on the difference between model theory and proof theory. They demonstrated that the approach that had been taken by the database community was model theoretic, that is, the database represents the truths of the theory. However, in the deductive database approach, the main thrust was proof theoretic. Ray Reiter, whose technical report at Bolt, Beranek and Newman [159] had come out only a few months earlier, presented two talks at the workshop. I thought that both of these papers were significant. His well-known paper on the closed-world assumption (CWA) [161] was one of the two papers. The second paper was on compiling axioms [160]. Reiter noted that if there are no recursive axioms, then one could use a theorem prover to generate a new set of axioms where the head of the axiom was defined in terms of relations in a database. Hence, one could interface with a relational database and not have to use a theorem prover during query operations. I suggested to Reiter that his technical report should be published as a monograph, as I thought it was highly significant. Reiter's paper on the CWA shed light on three major issues: the definition of a query, the definition of an answer to a query, and how one deals with negation. Keith Clark presented his well-known paper on negation and introduced the important concept of "if and only if" conditions that underly the meaning of negation (see the discussion above of Clark's work on negation). Papers by Chin Chang [25], Charles Kellogg et al. [80], and me [126] reported on actual systems that had been developed that performed deductive search. In my paper I described the indexing scheme that we used to access clauses in the MRPPS 3.0 system, as well as its other features. Other important papers that appeared in the workshop and ultimately in the book *Logic and Data Bases* [48] were by Kowalski [89], who discussed the use of logic for data description; Futo, Darvas, and Szeredi [46] on applications of PROLOG to drug data and drug interactions that they were working on in Hungary; Nicolas and Yazdanian's [145] paper on integrity constraints; and the paper by Pirotte [150], who presented a framework for comparing high-level nonprocedural query languages for the relational model of data. There were two other papers that I thought were highly significant. These were by Alain Colmerauer [31] on natural-language processing and by Roussel on PROLOG, both of the Marseilles group. Colmerauer believed that it would be more important for him to publish his paper in a journal, which he subsequently did [32]. Roussel had not prepared a paper for the Proceedings and could not write his paper in time to meet the publication date; hence it did not appear in the book. Few people outside the group that met at Toulouse knew about PROLOG, and Gallaire and I had hoped that a paper would appear in the book, as we thought that a larger community

should be aware of the work. It was left to others like David H. D. Warren, Kowalski, and subsequently the Japanese to focus attention on PROLOG. See [9, 30, 171] for early work on PROLOG by Colmerauer's group at Marseilles.

In the Foreword to the book, Gallaire and I stated,

> The book provides, for the first time, a comprehensive description of the interaction between logic and data bases. It will be seen that logic can be used as a programming language, as a query language, to perform deductive searches, to maintain the integrity of data bases, to provide a formalism for handling negative information, to generalize concepts in knowledge representation, and to represent and manipulate data structures. Thus, logic provides a powerful tool for data bases that is accomplished by no other approach developed to date. It provides a unifying mathematical theory for data bases.

Work that has been accomplished since then bears out these comments. The book helped focus attention on the use of logic for deductive databases, as had not been done earlier.

The formative years ended with the recognition of the significance of deductive databases and logic programming. The Japanese announced their "Fifth Generation Project", [138], whose work was based on the concepts of logic programming. The emphasis of the effort was to develop architectures that would take advantage of logic programming for both sequential and parallel architectures. Their expectation was that artificial-intelligence problems would be made easier to implement on architectures based on logic programming. The director of the Japanese effort, Dr. Kazuhiro Fuchi, had been exposed to PROLOG and was convinced of its significance. There is no doubt in my mind that the Japanese were instrumental in making computer professionals pay greater attention to logic programming and deductive databases. The current explosion of research in these two topics since the Japanese announcement is evidence of this trend.

In the United States, the artificial-intelligence community generally ignored PROLOG and logic programming. The ideas on PROLOG had been developed in Europe, and the Americans apparently believed that logic programming was of interest only to those involved in automated theorem proving. Formal techniques such as theorem proving were of no interest to this community, since procedure invocation was in vogue, and after all, theorem proving was very time consuming. Almost all researchers in the database community ignored deductive databases and logic programming, apparently believing that it had no relevance to "real database work" of either a theoretical or a practical nature. Those who were pushing the field of deductive databases were from artificial intelligence, a subject thought to be "flaky" by many in the database community. I am therefore thankful to the Japanese who saw the light and helped publicize the field.

4. THEORETICAL FOUNDATIONS: 1979–PRESENT

The period from 1979 to the present can be characterized as the era when theoretical foundations of both deductive databases and logic programming were developed. One cannot divorce deductive databases from logic programming, as they are intimately related.

A number of important developments happened simultaneously at a number of different places. The first step in this development was the classic paper by van Emden and Kowalski [191], in which they outlined fixed-point and operational semantics of Horn-clause logic as a programming language. They demonstrated that

fixed-point semantics corresponds to model theory, while operational semantics corresponds to proof theory. Van Emden and Kowalski gave the formal semantics of a definite-clause logic formula, viewed as a program. Their use of the least-model and least fixed-point constructions, as well as the procedural interpretation, have become standard tools in logic-programming theory. They were the first to provide a clearly defined formal declarative semantics, which was shown to be compatible with a fixed-point semantics and a procedural interpretation of a logic formula, viewed as a program. Their ideas also led to the concept of negation. If one takes the Herbrand base and subtracts out the minimal model, then one can say that the atoms that remain are those that can be assumed to be false. This provides a model-theoretic view of negation. Thus, the paper, written from the viewpoint of logic programming, had immediate consequences for databases.

Apt and van Emden [2], in an elegant paper, built upon the work of van Emden and Kowalski [191]. To appreciate their contribution it is necessary to provide the following background. If P is a Horn-clause logic program, the Herbrand base of P is denoted by $B(P)$. One can identify Herbrand interpretations for P and subsets of $B(P)$. The corresponding subset of the Herbrand base is the set of all ground atoms which are true in the interpretation. The set of all Herbrand interpretations of P is a complete lattice under the partial order of set inclusion. The mapping T_P, defined in [191], from the lattice of Herbrand interpretations to itself is given as:

$$T_P(I) = \{ A \in B(P) : A \leftarrow B_1, B_2, \ldots, B_n \text{ is a ground instance of a clause in } P \text{ and } B_1, B_2, \ldots, B_n \in I \}$$

where I is a Herbrand interpretation. The operator T_P is monotonic, and $T_P \downarrow \omega$ is defined as $\bigcap_{n=1}^{n=\omega} T_P^n(B(P))$.

With the above as background, one of the results of Apt and van Emden is that A is in the SLD finite failure set if and only if $A \notin T_P \downarrow \omega$. Lassez and Maher [94] show that the finite failure set is characterized by $\mathrm{FF} = B(P) \setminus T_P \downarrow \omega$, and thus the result of Apt and van Emden is essentially a weak soundness and completeness result for finite failure. It only guarantees the existence of one finitely failed SLD, tree and others may be infinite. Using the concept of fairness, they identified those computation rules which guarantee finitely failed SLD trees, leading to a strong soundness and completeness result on finite failure.

Jaffar, Lassez, and Lloyd [73] then showed that the inference system, SLDNF (selective linear resolution for definite clauses with negation as failure) was complete for ground negated atoms in the case of positive programs. Clark [27] had shown the soundness of the negation-as-finite-failure rule for Horn logic programs P, augmented by comp(P) and equality axioms. Jaffar, Lassez, and Lloyd's contribution was to show that if $\neg Q$ is a logical consequence of comp(P), then Q is in the finite failure set of the program P, which is the completeness result. Hence, a firm theoretical foundation was given to negation for logic programming and deductive databases. See Shepherdson [179, 180] for a comprehensive discussion on negation in deductive databases and logic programming. If one has a negated atom to be solved, then assuming that the atom is ground, if one fails to find a proof for the positive atom, the negated atom can be assumed true. SLDNF therefore provided a proof-theoretic view of finding answers in the presence of negation.

These developments, together with work on fixed-point theory, SLD resolution, and SLDNF resolution, provided the framework for John Lloyd's outstanding

recent book, *Foundations of Logic Programming*. Logic programming now has a firm theoretical foundation, and one can view PROLOG in the light of this theory. The foresight of Colmerauer and his students in developing tools such as the "not" operator and other extralogical features was clearly justified. The depth-first choice and no-occurs check feature also led to efficient, but not complete, implementations.

Ray Reiter [165], during the same period, provided some fundamental insights into database theory. He was the first to propose formal theories of deductive databases that encompassed and generalized the work of Codd. Reiter reinterpreted the conventional model-theoretic perspective on databases in purely proof-theoretic terms. He demonstrated how relational databases can be seen as special theories of first-order logic, where the theories incorporated the following assumptions:

(1) *The domain-closure assumption.* The individuals occurring in the database are all and only the existing individuals.

(2) *The unique-name assumption.* Individuals with distinct names are distinct.

(3) *The closed-world assumption.* The only possible instances of a relation are those implied by the database.

The use of a proof-theoretic approach permitted Reiter to provide a correct treatment of query evaluation for databases that have incomplete information and a class of null values; integrity constraints and their enforcement; and the extension of the relational model to incorporate more real-world semantics such as the representation of events and hierarchies. The significance of Reiter's work is that he focused on the proof-theoretic approach rather than the model-theoretic approach, he gave precise definitions of a number of central issues, and he clarified and extended relational databases to include deductive databases.

In another paper Reiter [167] treated the problem of indexed null values when it is known that there is a value and that it may not be among the given constants in the domain. In this case he showed that one could compute answers in a reasonable way; however, in certain cases one obtains correct answers, but not necessarily all the answers. Hence, he has a sound, but not necessarily complete, theory.

In my own work during this period, several problems were of interest to me. The first was that of a theory for non-Horn clauses corresponding to the one that Reiter had developed for Horn clauses, the second was to give consideration to some aspect of recursive axioms in the theory, the third was to interface a logic language with a database system, the fourth was to take advantage of integrity constraints during the search process, and the fifth was to do work in nonmonotonic reasoning as it relates to databases. I will touch upon some of this work.

With respect to non-Horn clauses, as part of the MRPPS 3.0 system we developed a parenthesized notation [201] to keep track of the proof tree as we were developing the proof. This was necessary because while performing deduction we arbitrarily selected literals for expansion, and we wanted to retain the proof tree along all paths. An arbitrary literal selection was important both for databases and for problem-solving search. It was obvious that a PROLOG depth-first left-to-right strategy would not be a good strategy for the class of problems in which we were interested. For non-Horn sets of axioms, there is a problem with the selection of arbitrary literals. Reiter [158] had shown that it would inhibit a complete search. Kowalski and Kuehner's SL resolution allows a limited selection strategy. Once one

selects a literal for expansion, one must solve that literal before other literals in the same clause can be selected. Guy Zanon suggested that the parenthesized notation that had been developed could be the basis for an inference system that was similar to SL resolution, but it would allow an arbitrary selection strategy and be complete and sound. We were able to devise a new complete and sound inference system called linear resolution with unrestricted selection function based on trees (LUST) [127, 130]. LUST resolution was useful for non-Horn theories in which an open-world assumption (OWA) is made. As defined by Reiter, an open world is one in which no assumptions are made about negation. That is, one has a first-order theory.

I thought that it would be useful to have a theory concerning negation that worked like Reiter's CWA for Horn theories, but applied to non-Horn theories. I initially tried to develop a proof-theoretic method which seemed to work. In January 1981 I was invited to visit Simon Bolivar University in Venezuela. Phillipe Roussel, who had developed the first PROLOG implementation with Colmerauer, was a visiting faculty member there. We discussed the problem, and he told me about some ideas that he had regarding a model-theoretic approach. In the very short period of time we had, about two days, we sketched some ideas. I then returned to the U.S.A. and worked on the problem, sending Roussel my results. I was finally able to obtain a soundness and completeness proof demonstrating that the model-theoretical and the proof-theoretical methods gave the same results. In the presence of non-Horn clauses, there is no unique minimal model; rather there are a set of minimal models. In the model-theoretic approach, an atom that does not appear in any minimal model is assumed to have its negation true. Thus, in the database that consists of the non-Horn clause $\{ p \vee q \}$, there are three models: $\{ p \}$, $\{ q \}$, and $\{ p, q \}$. Of these, two are minimal, $\{ p \}$ and $\{ q \}$, and neither is contained in the other. Hence, neither p nor q can be assumed to be false, as each appears in a minimal model. In the proof-theoretic approach that I developed, I demonstrated that if one cannot prove $P(a) \vee K$, where K is an arbitrary positive clause and one cannot prove K, then one can assume not-$P(a)$. I termed the method the generalized closed-world assumption (GCWA) [129]. I showed that for function-free clauses the same responses to queries were obtained in the model-theoretic and the proof-theoretic definitions of the GCWA. John Grant and I [135] have developed a method to compute answers to queries in the ground non-Horn case where all clauses (disjuncts of literals) are restricted to consist only of positive atoms. Henschen and his students [66, 205] have also attempted to develop methods to answer queries in databases that comply with the GCWA. Gelfond, Przymusinska, and Przymusinski [52] used the concept of the GCWA to develop an extended closed-world assumption (ECWA). Przymusinski [152, 153] has also shown how one can utilize SL resolution to be able to answer queries in non-Horn databases subject to the GCWA. He terms the modified inference system SLSNF (linear resolution with subsumption based on negation as failure). The basic idea is to use SL resolution on the negation of some positive atom, say $P(a)$, first selecting only negated atoms in a resolvent clause. Resolving away all negative literals leaves at most positive literals. One then uses SL resolution to determine if, starting with a clause containing the negation of the positive atoms, one can find a proof. If one cannot find a proof, then one can assume the negation of the original atom, $P(a)$. For additional work on non-Horn clauses see Bossu and Siegel [14] and Bidoit and Hull [12].

McCarthy [114] had done some important work in artificial intelligence in the area of nonmonotonic reasoning that was of interest to me. I thought that his concept of circumscription might be useful for some aspects of databases. In 1984, Don Perlis and I started to look into this topic. After reviewing McCarthy's paper, we tried to apply his method to a particular problem in databases. We were interested in answering queries in databases where it is known that it is unknown whether or not a particular fact were true. That is, we may know $P(a)$ is true and we may also know that we do not know whether or not $P(b)$ is true. This situation is not handled in relational databases. It also turned out that McCarthy's original concept of circumscription did not handle this case. We wrote a series of papers [132–134] in which we addressed this problem. We were able to show that a slightly different "only if" statement than that specified by Clark could handle the situation, and furthermore that one could modify the database to achieve a Horn theory that could be used to compute answers. The Horn theory is sound, but in a few cases does not give all answers. The interesting aspect of the work was the applicability of circumscription to a problem in databases. Perlis and I [148] were also able to obtain some completeness results for circumscription, and McCarthy [124] had obtained a soundness result for circumscription. Reiter also recognized the possibility of applying circumscription to databases. In [164] he shows how Clark's negation as failure is a consequence of circumscription for Horn clauses, and in [165] he emphasizes the importance of circumscription for database theory. I shall return to circumscription later.

In the area of recursive axioms as part of the Horn theory, Reiter [162] had proposed that one avoids dealing with them by breaking cycles in axioms. Chang [26] was the first to propose that one should use a connection graph to handle some aspect of recursion. Jean-Marie Nicolas and I [128, 129] worked together for a few months on the problem of recursive axioms in the intensional database, while he was on a sabbatical leave at the University of Maryland. We were able to show that there are a number of interesting cases where recursion can be terminated, based on the type of recursive axioms that one has. See related work by Naughton [143], Naughton and Sagiv [142], and Ioannides and Wong [72]. Naqvi and Henschen [65, 140], using a connection-graph approach, have extended the work by Chang to a wider class of recursive axioms. It seems like "magic" that there has been a mushrooming interest in handling recursive axioms, as evidenced by the work of Bancilhon et al. [6], Beery and Ramakrishnan [10], Kifer and Lozinskii [81], and Lozinskii [111]. Bancilhon and Ramakrishnan [7, 8] have performed a comprehensive comparative study on these systems.

At the time of Nicolas's visit, we had completed two books [48, 49] and were working on the third book [51] that came out of workshops at Toulouse. During Nicolas's visit, Gallaire came to Maryland for a few days, and we decided that the time was right to write a comprehensive survey article on deductive databases, since much work had been done on the subject and it was time to put the work in perspective. The survey article we wrote finally came to fruition in 1984 [50]. I believe that the survey article accomplished what we had intended—to make the literature on deductive databases more accessible and better known.

One of the main aspects of relational-database technology is the implementations that have been developed to handle queries and updates. On the other hand, there have been few implementations of large deductive databases. Although useful for

small databases that could reside entirely in main memory [36, 124, 125], it was clear very early that PROLOG was not a language suitable for database applications. There are two major problems. The first is that most PROLOG systems are interpreters and do not work with large databases. The second is the left-to-right search. The paper by Bowen and Kowalski [15] that dealt with the use of metalevel programming was, I thought, significant. It seemed to be a natural extension and gave the promise of being able to modify PROLOG to obtain new control structures within PROLOG. Hence, one need not be restricted to the control structure within PROLOG. To be sure, one pays a price because of the necessity to do a double interpretation. It seemed to me, however, that for databases this would not be a significant penalty, since one could compile axioms once and set them up appropriately, assuming that there were no recursive axioms. With this in mind, I developed a meta-interpreter to interface between PROLOG and databases [17]. This was an early attempt, and more efficient interpreters can be developed (see, for example, [109]). The work on MU-PROLOG [139, 187] avoids a meta-interpreter approach and has special indexing routines for working with large databases. Another effort in the same direction is the work on NAIL! [189]. Other efforts in this area include work by Warren [198], who developed a PROLOG program which could take a query and, using information about indexing and other features, transform the query so that the best literals are placed first. The work by Warren is related closely to the work by Selinger et al. [176], accomplished for System R. Grant and Minker [54–56] have developed a branch-and-bound algorithm to take advantage of the fact that when one compiles queries in a deductive-database context, then a set of conjunctive queries result. These queries generally share relations that have to be searched. One may gain search speed if one optimizes a set of queries, rather than optimizing a single conjunct at a time. See Sellis [177, 178] for related work.

A result of Reiter [161] is that in a Horn database it is not necessary to use integrity constraints during query search. Answers found with the theory without integrity constraints will be the same as with integrity constraints. This does not mean that one should not use integrity constraints during the search, as was noted in the work of King [82], Hammer and Zdonik [62], McSkimin [177], and McSkimin and Minker [118]. As stated earlier, the problem with these approaches was that there was no general mechanism to handle integrity constraints. A formal approach was developed by Chakravarthy in his thesis [19] and in a series of papers with others [18, 20, 21]. The utility of the approach is that it can substantially decrease search time, and there is a once-only penalty to compile the axioms to take advantage of the technique. The core of most expert systems should be a deductive database. The integrity constraints supply the semantics of the domain of application and therefore supply some of the "expertise" for an expert system. It is also of interest to note that the approach can be utilized to provide informative answers to a user. Thus, if a database were about parents, the query, "Who is both the father and the mother of a particular person?" should return the response, "A person cannot be both the father and the mother of another person," rather than the answer that there is no one listed. That is, expert systems should have the capability to provide informative answers. Gal and Minker [47] have shown how one may obtain such informative answers. See [74] and [77] for related approaches to this problem. A somewhat different approach has been taken by Imielinski [71].

It was clear to me (and to many others) in the 1970s that executing programs in parallel was a natural for logic programming. One can see, obviously, how to take advantage of AND/OR parallelism automatically in a logic program. A procedure name with many bodies represents OR parallelism, while a procedure body represents AND parallelism. I was pleased, therefore, that my colleague Chuck Rieger [169] designed the ZMOB parallel architecture, which consisted of 128 Z-80A microprocessors interconnected on a high-speed ring structure. With some of my students (Eisinger, Kasif, Kohli [41, 79]) I designed a parallel inference system (PRISM) that incorporated AND/OR parallelism. PRISM has been implemented on the MCMOB system, a modification to ZMOB that consists of 16 Motorola 68000 processors interconnected on the ZMOB ring structure. Although designed for problem solving and top-down search, PRISM can be used as a deductive database system. A paper describing experiments using PRISM is in preparation.

While these efforts were taking place, Lloyd and Topor, in a series of three papers [105, 106, 108], developed a theoretical basis for deductive database systems which are implemented using a PROLOG system as the query evaluator. They use a typed first-order logic to express data, queries, and integrity constraints. They introduced extended programs and extended goals for logic programming. In contrast to Horn-clause logic, a clause in extended programs can have an arbitrary first-order formula as its body, and similarly for an extended goal. They have provided a definition of an answer to a query being correct with respect to a database and also presented a definition of an integrity constraint being satisfied by a database. In addition they have developed two query evaluation processes and have proved that both are sound and, for definite and hierarchical databases, complete. A database is *hierarchical* if the predicates in the program P can be partitioned into levels so that the definition of level-0 predicates consists solely of unit clauses and the bodies of the clauses in the definition of level-j predicates ($j > 0$) contain only level-i predicates, where $i < j$.

The work of Lloyd and Topor is important. They show that a formalism based on first-order logic provides an expressive environment for modeling databases; that a single formalism may be used, namely first-order logic; and that logic provides a theoretical foundation required for databases.

The work previously described in this section, together with the book by Lloyd [104] on the foundations of logic programming, sets forth major theoretical developments in these two fields that distinguishes the work in this period. There now exist formal theories in both deductive databases and logic programming. Unified results now exist in databases, rather than fragmented results on various topics.

In 1985 I thought that it would be important to bring together researchers from the deductive-database and logic-programming communities. I received a grant from the National Science Foundation and significant support from the University of Maryland Institute of Advance Computer Studies to organize the Workshop on Foundations of Deductive Databases and Logic Programming. The workshop was held during the summer of 1986 in Washington, D.C. Papers at the workshop were highly significant and will, I believe, set the tone for the next period in these two areas [136]. I cannot recall a conference or a workshop where so much work of significance came together at one time, except perhaps at the first Toulouse workshop in 1977. The work that was reported shows the maturity which these two fields have attained. Among the many outstanding papers I would like to cite a few. This does not at all mean that the other papers were not significant.

Negation in logic programming has been troublesome even with the developments that have been cited above. It is interesting that three individuals addressed this problem and others expanded upon it. Important theoretical results were reported on what are now termed stratified databases. A stratified database is one in which one deals with extended or general clauses that have negated atoms in the antecedent of a clause. The consequent of a clause is a single atom. Hence, we are referring to extended Horn clauses (since the antecedent of a Horn clause cannot contain a negated atom). A database is *stratified* if the clauses can be so ordered that if a negated atom appears in the body of a clause, then the definition of the atom (the consequent of a clause) precedes the clause in which the negated atom appears in the body of the clause. A stratified database is said to be "free from recursive negation", since stratification prevents recursion on negation. Theoretical results in this area were obtained by Apt, Blair, and Walker [3] and Van Gelder [192]. Apt et al. develop a fixed-point theory of nonmonotonic operators and apply it to provide a declarative meaning of a general program. They also prove the consistency of Clark's completed model database for stratified programs and clarify some previously reported problems with negation in logic programming. Van Gelder showed that general logic programs with the so-called bounded-term-size property and freedom from recursive negation are "completely classified" by what he refers to as tight tree semantics in which every atom in the Herbrand base of the program either succeeds or fails. As all programs terminate on every input, Van Gelder deals only with a strict subset of what is computable, as opposed to Horn clauses. Shamim Naqvi [141] also recognized the importance of stratified databases. The importance of stratification in databases, in a slightly different context, was noted by Chandra and Harel [23], who defined the class of stratified queries (which they referred to as Class C). They showed that stratified queries are identical with fixed-point queries defined by Chandra and Harel [22]. Two complementary papers were written on this subject. Lifschitz [103] used McCarthy's [115, 116] concept of prioritized circumscription to obtain results with respect to the semantics and minimal model of stratified programs. Thus, again, we see the application of circumscription to databases. For additional work on circumscription see [100–102]. To round out these papers, Przymusinski [152], using a model-theoretic approach initiated in [131], extended the notion of stratified logic programs to deductive databases which allow negative premises and disjunctive consequents. He introduced the concept of *perfect model* of a database and showed that the set of perfect models provides a correct semantics for such a database. He extends and strengthens the results of Apt et al., Van Gelder, and Lifschitz. A paper by Shepherdson [180] provides a comprehensive, excellent survey on negation in deductive databases and logic programming.

In other work reported at the workshop, Paris Kanellakis [76] presented a significant survey on logic programming and parallel complexity. The logic-programming problems addressed are related to query optimization for deductive databases and to fast parallel execution of primitive operations in logic-programming operations, such as fixpoint operators, term unification, and term matching. Sagiv [172] showed how one can optimize a class of function-free logic programs. Rodney Topor presented a paper (Topor and Sonenberg [188]) that shed light on the problems that Kuhns first studied in the 1960s. They introduce and study a class of "domain-independent" databases: databases for which the set of correct answers to a query is independent of the domains of variables in database clauses. They prove

that every "allowed" stratified database is domain independent and that every domain independent stratified database has an equivalent allowed database. See Vardi [194] for work on the decision problem related to the work by Kuhns. Kowalski and Sadri [90] proposed an extension to SLDNF for checking constraints in deductive databases. They achieve the effect of the simplification methods of Nicolas [147], Lloyd and Topor [107], and Decker [37] in their work. In addition to his work on intelligent answers noted earlier, Imielinski [70, 71] addresses three different types of incomplete information: (1) existentially quantified statements, (2) existentially quantified statements with range-coupled existential quantifiers, and (3) arbitrary disjunctive information.

I am currently editing a book entitled *Foundations of Deductive Databases and Logic Programming* that will consist of refereed papers drawn from the workshop.

5. FUTURE DIRECTIONS

We have seen a tremendous spurt of research in the areas of deductive databases and logic programming during the past few years. These developments are an outgrowth of work that was started in automated theorem proving. It led to foundational work in deductive databases and in logic programming. It has led to the clarification and handling of negation, null values, indefinite databases, integrity-constraint checking, and syntactic and semantic optimization of database programs, as well as other developments. Work from artificial intelligence dealing with nonmonotonic reasoning and circumscription are also seen to have played an important role in these developments.

Today, a great deal of consideration is being given to the concept of expert systems. Whatever one considers an expert system to be, it is clear that deductive databases and logic programming will play a prominent role. To achieve capabilities for expert systems such as the ability to review a proof tree and the ability to handle fuzzy data and interactive responses, two basic methods may be used. These are to develop meta-interpreters [184–186] or to use a logic language directly [28]. The ability to work with large masses of extensional and intensional axioms will require interfacing logic programs with database technology. An effective amalgamation has not yet been achieved. However, it is clear that this can be accomplished and is only a matter of time and funding.

There remains a great deal that still has to be clarified about databases themselves. Handling updates and deletions efficiently and intelligently has yet to be achieved. Effective computation methods are required for alternative database classes, such as databases that are non-Horn, contain null values, or are not function-free. The work by Imielinski [70, 71] to obtain approximate answers to queries is of interest here.

I believe, moreover, that deductive databases will have a major impact on artificial intelligence (AI). Significant progress has been made in databases only since theoretical foundations started to be developed. I refer here to the work of Codd and the work of those who have developed the field of dependencies described by Ullman [190], as well as in deductive databases as described here. The AI community must move towards theories that describe phenomena. They can no longer rely on programs that illustrate techniques and whose theory is somehow embedded in a program. It is extremely difficult to understand a theory that consists

of a program and to abstract out broader concepts. Promising work here is being done by McCarthy [114] on circumscription, and by Reiter on nonmonotonic reasoning [163] and more recently on his theory of medical diagnosis [166]. For a comprehensive survey on nonmonotonic reasoning see [168].

The reason for the centrality of deductive databases is due to the need in all of the work to have facts and intensional statements about the world. Thus, part of the theory must deal with the nature of the database. Integrity constraints effectively describe the semantics of the database. In AI belief systems we will have to describe the semantics of the users, since they represent the beliefs that the users have about the world, which may or may not be the same for each user, and for that matter may or may not be the same as that of the database. Thus, work on belief systems in AI will also be important for databases. Interesting work in this connection is being done by Gelfond and Przymusinska [53], Fagin and Halpern [42, 43], Levesque [95], Perlis [149], and Vardi [195].

It is interesting to note that work in theorem proving that was denigrated because of its "complexity" has been the basis for the theories described here. This does not mean that pure theorem proving was the only tool used. But certainly, it was the primary tool that was the basis of the work. Logics other than variations of first-order logic, such as higher-order logics, may be important in future developments. However, this remains to be seen. For references related to other logics, see [180].

The subjects of deductive databases and logic programming are important, viable, and thriving disciplines still growing toward their prime.

It seems to me that the fields of databases, logic programming, deductive databases, artificial intelligence, and expert systems will move towards one another. Formalisms and techniques developed in each of these areas will assist the others. Science builds on theories. Theories developed for deductive databases and logic programming will, therefore, be built upon to further developments in the above subjects.

The following grants supported this work: AFOSR 82-0303, ARO DAAG-29-85-K-0177, and NSF IRI-8609170. I would like to express my appreciation to John Grant, Jorge Lobo, Don Perlis, Arcot Rajasekar, and Deepak Sherlekar for their suggestions regarding the paper. I especially thank my wife, Rita G. Minker, for her comments on the paper, for being so supportive of my work, and for sharing her life with me.

In preparing the paper for publication, I have tried to assure the accuracy of the early historical information and I asked many individuals for comments. I am grateful to Krzysztof Apt, Ashok Chandra, Maarten van Emden, Hervé Gallaire, Cordell Green, Larry Henschen, Lary Kuhns, Jean-Louis Lassez, M. E. (Bill) Maron, Jean-Marie Nicolas, Ray Reiter, and Rodney Topor for sending me their comments. I have incorporated most of their suggestions in the text. I am, of course, responsible for any errors that may remain.

REFERENCES

1. Ait-Kaci, H. and Nasr, R., LOGIN: A Logic Programming Language with Built-In Inheritance, *J. Logic Programming* 3(3):185–215 (Oct. 1986).
2. Apt, K. R. and van Emden, M. H., Contributions to the Theory of Logic Programming, *J. Assoc. Comput. Mach.* 29:841–862 (1982).

3. Apt, K. R., Blair, H. A., and Walker, A., Towards a Theory of Declarative Knowledge, in: J. Minker (ed.), *Proceedings of the Workshop on Foundations of Deductive Databases and Logic Programming*, Washington, 18–22 Aug. 1986, pp. 546–623.
4. Aqvist, L., A New Approach to the Logical Theory of Interrogatives: Part 1, Analysis, Univ. of Uppsala, Sweden, 1965.
5. Ash, W. L. and Sibley, E. H., TRAMP: An Interpretive Associative Processor with Deductive Capabilities, in: *Proceedings of the 1968 ACM National Conference*, Princeton, N.J., 1968, pp. 143–156.
6. Bancilhon, F., Sagiv, Y., and Ullman, J., Magic Sets and Other Strange Ways to Implement Logic Programs, in: *Proceedings of the 5th ACM SIGMOD-SIGACT News Symposium on Principles of Database Systems*, 1986.
7. Bancilhon, F., and Ramakrishnan, R., An Amateur's Introduction to Recursive Query Processing Strategies, in: *Proceedings of ACM SIGMOD '86*, Washington, 28–30 May 1986, pp. 16–52.
8. Bancilhon, F. and Ramakrishnan, R., Performance Evaluation of Data Intensive Logic Programs, in: J. Minker (ed.), *Proceedings of the Workshop on Foundations of Deductive Databases and Logic Programming*, Washington, 18–22 Aug. 1986, pp. 284–314.
9. Battani, G. and Meloni, M., Interpreteur du Language de Programmation PROLOG, Internal Technical Report, Groupe d'Intelligence Artificielle, Univ. d'Aix-Marseille II, 1973.
10. Beeri, C. and Ramakrishnan, R., On the Power of Magic, in: *Proceedings on Principles of Database Systems*, San Diego, Mar. 1987, 269–283.
11. Belnap, N. D., An Analysis of Questions: Preliminary Report, TM-128/000/00, System Development Corp., 3 June 1963.
12. Bidoit, N. and Hull, R., Positivism vs. Minimalism in Deductive Databases, in: *Proceedings of the ACM SIGACT-SIGMOD News Symposium on the Principles of Database Systems*, Cambridge, Mass., 1986, pp. 123–132.
13. Bledsoe, W. W., Splitting and Reduction Heuristics in Automatic Theorem Proving, *Artificial Intelligence* 2(1):55–77 (1971).
14. Bossu, G. and Siegel, P., Saturation, Nonmonotonic Reasoning and the Closed-World Assumption, *Artificial Intelligence* 25(1):13–63 (Jan. 1985).
15. Bowen, K. and Kowalski, R. A., Amalgamating Language and Metalanguage, in: K. Clark and S. A. Tarnlund (eds.), *Logic Programming*, Academic, London, 1982, pp. 153–172.
16. Carnap, R., *Meaning and Necessity (Enlarged Edition)*, Univ. of Chicago Press, Chicago, 1956.
17. Chakravarthy, U. S., Minker, J., and Tran, D., Interfacing Predicate Logic Languages and Relational Databases, in: *Proceedings of the 1st Logic Programming Conference*, France, Sept. 1982.
18. Chakravarthy, U. S., Fishman, D. H., and Minker, J., Semantic Query Optimization in Expert Systems and Database Systems, in: L. Kerschberg (ed.), *Expert Database Systems*, Benjamin Cummings Pub. Co., 1986, 659–675.
19. Chakravarthy, U., Semantic Query Optimization in Deductive Databases, Ph.D. Thesis, Univ. of Maryland, College Park, Aug. 1985.
20. Chakravarthy, U. S., Grant, J., and Minker, J., Semantic Query Optimization: Additional Constraints and Control Strategies, in: L. Kerschberg (ed.), *Proceedings on Expert Database Systems*, Charleston, Apr. 1986, pp. 259–269.
21. Chakravarthy, U. S., Grant, J., and Minker, J., Foundations of Semantic Query Optimization for Deductive Databases, in: J. Minker (ed.), *Proceedings of the Workshop on Foundations of Deductive Databases and Logic Programming*, Washington, D.C., 18–22 Aug. 1986, pp. 67–101.
22. Chandra, A. and Harel, D., Structure and Complexity of Relational Queries, *J. Comput. System Sci.* 25:99–128 (1982).

23. Chandra, A. and Harel, D., Horn Clause Queries and Generalizations, *J. Logic Programming* 2(1):1–15 (Apr. 1985).
24. Chang, C. L. and Lee, R. C. T., in: *Symbolic Logic and Mechanical Theorem Proving*, Academic, New York, 1973.
25. Chang, C. L., DEDUCE 2: Further Investigations of Deduction in Relational Databases, in: H. Gallaire and J. Minker (eds.), *Logic and Databases*, Plenum, New York, 1978, pp. 201–236.
26. Chang, C. L., On Evaluation of Queries Containing Derived Relations, in: H. Gallaire, J. Minker, and J. Nicolas (eds.), *Advanced in Database Theory*, Vol. 1, Plenum, New York, 1981, pp. 235–260.
27. Clark, K. L., Negation as Failure, in: H. Gallaire and J. Minker (eds.), *Logic and Data Bases*, Plenum, New York, 1978, pp. 293–322.
28. Clark, K. L. and McCabe, F. G., A Language for Implementing Expert Systems, in: P. Hayes, D. Michie, and Rao (eds.), *Machine Intelligence 10*, Ellis Horwood, 1982, pp. 455–470.
29. Codd, E. F., A Relational Model of Data for Large Shared Data Banks, *Comm. ACM* 13(6):377–387 (June 1970).
30. Colmerauer, A., Kanoui, H., Pasero, R., and Roussel, P., "Un Système de Communication Homme-Machine en Francais, Technical Report, Univ. d'Aix-Marseille, II, Marseille, 1973.
31. Colmerauer, A., An Interesting Natural Language Subset, in *Proceedings of the Workshop on Logic and Databases*, Toulouse, 1977.
32. Colmerauer, A., Un Sous-Ensemble Intéressante du Francais, *RAIRO Inform. Théor.* 4:309–336 (1979).
33. Relational Structures Research, Computer Corp. of America, Contract DA18-119-AMC-03409(X), F/R18-119-36-00326(X), 6 Apr. 1966–5 July 1967.
34. Fundamentals of Relational Structures, ESD-Tech. Rep.-68-404, Computer Corp. of America, Contract No. AF 19 (628)-5939, Electronic Systems Divisions, Air Force Systems Command, United States Air Force, L. G. Hanscom Field, Bedford, Massachusetts, 3 Sept. 1969.
35. Relational Structures Applications Research, DAAB03, Computer Corp. of America, 5 May 1967–31 Mar. 1969.
36. Dahl, V., On Database Systems Development through Logic, *ACM Trans. Database Systems* 7(1):102–123 (Mar. 1982).
37. Decker, H., Integrity Enforcement on Deductive Databases, in: *Proceedings of EDS 86*, South Carolina, 1986, 271–285.
38. Deliyanni, G. and Kowalski, R. A., Logic and Semantic Networks, *Comm. ACM* 22(3):184–192 (1979).
39. DiPaola, R. A., The Recursive Unsolvability of the Decision Problem for the Class of Definite Formulas, *J. Assoc. Comput. Mach.* 16(2):324–327 (Apr. 1968); Rand Corp.
40. DiPaola, R. A., The Relational Data File and the Decision Problem for Classes of Proper Formulas, in: J. Minker and S. Rosenfeld (eds.), *Proceedings of the Symposium on Information Storage and Retrieval*, College Park, 1–2 Apr. 1971.
41. Eisinger, N., Kasif, S., and Minker, J., Logic Programming: A Parallel Approach, in: *Proceedings of the Logic Programming Conference*, Marseilles, France, Sept. 1982, pp. 71–77.
42. Fagin, R., Halpern, J. Y., and Vardi, M. Y., A Model-Theoretic Analysis of Knowledge, in: *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science*, West Palm Beach, 1984, pp. 268–278.
43. Fagin, R. and Halpern, J. Y., Belief, Awareness and Limited Reasoning, *Proceedings of the 9th International Conference on Artificial Intelligence*, Los Angeles, 1985, pp. 491–501.
44. Fishman, D. H., Experiments with a Resolution-Based Deductive Question-Answering System and a Proposed Clause Representation for Parallel Search, Ph.D. Thesis, Dec. 1973

45. Fishman, D. H. and Minker, J., PI-Representation: A Clause Representation for Parallel Search, *Artificial Intelligence* 6(2):103–127 (1975).
46. Futo, I., Darvas, F., and Szeredi, P., The Application of PROLOG to the Development of QA and DBM Systems, in: H. Gallaire and J. Minker (eds.), *Logic and Data Bases*, Plenum, New York, 1978, pp. 347–375.
47. Gal, A. and Minker, J., A Natural Language Database Interface that Provides Cooperative Answers, in: *Proceedings of the Second Conference on Artificial Intelligence Applications*, Florida, 11–13 Dec. 1985, 352–357.
48. Gallaire, H. and Minker, J., *Logic and Databases*, Plenum, New York, Apr. 1978.
49. Gallaire, H., Minker, J., and Nicolas, J., *Advances in Database Theory*, Vol. 1, Plenum, New York, 1981.
50. Gallaire, H., Minker, J., and Nicolas, J., Logic and Databases: A Deductive Approach, *ACM Comput. Surveys* 16(2):153–185 (June 1984).
51. Gallaire, H., Minker, J., and Nicolas, J., *Advances in Database Theory*, Vol. 2, Plenum, New York, 1984.
52. Gelfond, M., Przymusinska, H., and Przymusinski, T., The Extended Closed World Assumption and its Relation to Parallel Circumscription, in: *Proceedings of the ACM SIGACT News–SIGMOD Symposium on Principles of Database Systems*, 1986, pp. 133–139.
53. Gelfond, M. and Przymusinska, H., Negation as Failure: Careful Closure Procedure, *Artificial Intelligence* 30(3):273–286 (Dec. 1986).
54. Grant, J. and Minker, J., Optimization in Deductive and Conventional Relational Data Base Systems, in: H. Gallaire, J. Minker, and J. Nicolas (eds.), *Advances in Data Base Theory*, Vol. 1, Plenum, New York, 1981, pp. 195–234.
55. Grant, J. and Minker, J., On Optimizing the Evaluation of a Set of Expressions, *Internat. J. Comput. Inform. Sci.* 11:179–191 (1982).
56. Grant, J. and Minker, J. A Set Optimizing Algorithm, in: *Proceedings of the Conference on Information Sciences and Systems*, New Jersey, 1982, pp. 259–263.
57. Green, C. C. and Raphael, B., The Use of Theorem-Proving Techniques in Question-Answering Systems, in *Proceedings of the 23rd National Conference ACM*, Washington, 1968.
58. Green, C. C., The Application of Theorem Proving to Question-Answering Systems, Tech. Rep. CS 138, Ph.D. Thesis, Computer Science Dept., Stanford Univ., June 1969, ARPA Order No. 457, RADC Contract F30602-69-C-0056.
59. Green, C. C., Application of Theorem Proving to Problem Solving, in: D. E. Walker and L. M. Norton (eds.), *Proceedings of the International Conference on Artificial Intelligence*, Washington, 1969, pp. 219–240.
60. Green, C. C., Theorem Proving by Resolution as a Basis for Question-Answering Systems, in: B. Meltzer and D. Michie (eds.), *Machine Intelligence 4*, Edinburgh U.P., New York, 1969, pp. 183–205.
61. Gurk, H. and Minker, J., The Design and Simulation of an Information Processing System, *J. Assoc. Comput. Mach.* 8(2):260–270 (Apr. 1961).
62. Hammer, M. T. and Zdonik, S. B., Knowledge-Based Query Processing, in: *Proceedings of the 6th International Conference on Very Large Data Bases*, New York, 1–3 Oct. 1980, pp. 137–147.
63. Harrah, D., *Communication: A Logical Model*, MIT Press, Cambridge, Mass., 1963.
64. Hart, P. E., Nilsson, N. J., and Raphael, B., A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Trans. Systems Cybernet.* SSC-4(2):100–107 (1968).
65. Henschen, L. J. and Naqvi, S. A., On Compiling Queries in Recursive First-Order Databases, *J. Assoc. Comput. Mach.*, 31(1):47–85 (Jan. 1984).
66. Henschen, L. J. and Park, H., Compiling the GCWA and Indefinite Databases, in: J. Minker (ed.), *Workshop on Foundations of Deductive Databases and Logic Programming*, 22–28 Aug. 1986.

67. Hewitt, C. E., PLANNER: A Language for Proving Theorems in Robots, in: *First International Joint Conference on Artificial Intelligence*, Washington, 1969, pp. 295–301.
68. Hill, R., LUSH Resolution and its Completeness, DCL Memo No. 78, School of Artificial Intelligence, Aug. 1974.
69. Hodes, L., Programming Languages, Logic and Cooperative Games, presented at Symposium for Symbolic and Algebraic Manipulation, Mar. 1966.
70. Imielinski, T., Automated Deduction in Databases with Incomplete Information, in: J. Minker (ed.), *Proceedings on the Foundations of Deductive Databases and Logic Programming*, Washington, 18–22 Aug. 1986, pp. 242–283.
71. Imielinski, T., Intelligent Query Answering in Rule Based Systems, in: J. Minker (ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan-Kaufman, Washington, 1987.
72. Ioannidis, Y. E. and Wong, E., An Algebraic Approach to Recursive Inference, in: L. Kerschberg (ed.), *Proceedings of the First International Conference on Expert Database Systems*, Apr. 1986, pp. 209–223.
73. Jaffar, J., Lassez, J., and Lloyd, J. W., Completeness of the Negation as Failure Rule, in: *Proceedings Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, 8–12 Aug. 1983, pp. 500–506.
74. Janas, J. M., On the Feasibility of Informative Answers, in: H. Gallaire, J. Minker, and J. Nicolas (eds.), *Advances in Database Theory*, Vol. 1, New York, 1978, pp. 397–414.
75. Jespersen, O., *The Philosophy of Grammar*, Norton, New York, 1965.
76. Kanellakis, P., Logic Programming and Parallel Complexity, in: *J. Minker (ed.), Proceedings of the Workshop on Foundations of Deductive Databases and Logic Programming*, Washington, 18–22 Aug. 1986, pp. 629–657.
77. Kaplan, S. J., Cooperative Responses from a Portable Natural Language Query System, *Artificial Intelligence* 19(2):165–187 (Oct. 1982).
78. Kasher, A., Data-Retrieval by Computer: A Critical Survey, in: M. Kochen (ed.), *The Growth of Knowledge*, Wiley, New York, 1967, pp. 292–324.
79. Kasif, S., Kohli, M., and Minker, J., PRISM: A Parallel Inference System for Problem Solving, in: *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, West Germany, 8–12 Aug. 1983, pp. 544–546.
80. Kellogg, C., Klahr, P., and Travis, L., Deductive Planning and Pathfinding for Relational Data Bases, in: H. Gallaire and J. Minker (eds.), *Logic and Data Bases*, Plenum, New York, 1978, pp. 179–200.
81. Kifer, M. and Lozinskii, E., Query Optimization in Logic Databases, Technical Report, SUNY at Stony Brook, June 1985.
82. King, J. J., QUIST: A System for Semantic Query Optimization in Relational Databases, in: *Proceedings of the 7th International Conference on Very Large Data Bases*, New York, 9–11 Sept. 1981, pp. 510–517.
83. Kochen, M., Automatic QA of English-Like Questions about Simple Diagrams, *J. Assoc. Comput. Mach.* 16:26–48 (Jan. 1969).
84. Kowalski, R. A. and Hayes, P. J., Semantic Trees in Automatic Theorem-Proving, in: B. Meltzer and D. Michie (eds.), *Machine Intelligence 4*, Edinburgh U.P., 1968, pp. 87–101.
85. Kowalski, R. A., Search Strategies for Theorem Proving, in: B. Meltzer and D. Michie (eds.), *Machine Intelligence 5*, Edinburgh U.P., New York, 1969, pp. 179–200.
86. Kowalski, R. A. and Kuehner, D., Linear Resolution with Selection Function, in *Artificial Intelligence*, Vol. 2, 1971, pp. 227–260.
87. Kowalski, R. A., AND-OR Graphs Theorem Proving Graphs and Bidirectional Search, in: B. Meltzer and D. Michie (eds.), *Machine Intelligence 7*, Edinburgh U.P., New York, 1972, pp. 167–194.
88. Kowalski, R. A., Predicate Logic as a Programming Language, in: *Proceedings of IFIP 4*, Amsterdam, 1974, pp. 569–574.
89. Kowalski, R. A., Logic for Data Description, in: H. Gallaire and J. Minker (eds.), *Logic and Data Bases*, Plenum, New York, 1978, pp. 77–102.

90. Kowalski, R. and Sadri, F., An Application of General Purpose Theorem Proving to Database Integrity, in: J. Minker (ed.), *Proceedings of the Workshop on Foundations of Deductive Databases and Logic Programming*, Washington, 18–22 Aug. 1986, pp. 477–517.

91. Kuhns, J. L., Answering Questions by Computer: A Logical Study, RM-5428-PR, Rand Corp., Dec. 1967.

92. Kuhns, J. L., Logical Aspects of Questions Answering by Computer, presented at Third International Symposium on Computer and Information Sciences, Miami Beach, Fla., 1969, P-4251.

93. Kuhns, L., Quantification in Query Systems, in: J. Minker and S. Rosenfeld (eds.), *Proceedings of the Symposium on Information Storage and Retrieval*, 1–2 Apr. 1971, pp. 81–93.

94. Lassez, J. and Maher, M. J., Closure and Fairness in the Semantics of Programming Logic, *Theoret. Comput. Sci.* 29:167–184 (1984).

95. Levesque, N. J., A Logic of Implicit & Explicit Belief, in: *Proceedings of the National Conference on Artificial Intelligence*, 1984, pp. 198–202.

96. Levien, R. and Maron, M. E., *Relational Data File: A Tool for Mechanized Inference Execution and Data Retrieval*, Rand Corp., Dec. 1965.

97. Levien, R. E., Relational Data File II: Implementation, in: *Proceedings of the Annual National Colloquium on Information Retrieval*, Washington, 1967, pp. 225–241.

98. Levien, R. E. and Maron, M. E., A Computer System for Inference Execution and Data Retrieval, *Comm. ACM*, 10:715–721 (Sept. 1966).

99. Levien, R. E., Relational Data File: Experience with a System for Propositional Data Storage and Inference Execution, RM-5947-PR, Rand Corp., Apr. 1969.

100. Lifschitz, V., Closed World Databases and Circumscription, *Artificial Intelligence* 27(2):229–235 (Nov. 1985).

101. Lifschitz, V., Computing Circumscription, in: *Proceedings of the Ninth International Joint Conference on Artificial Intelligence, Los Angeles*, 1985, *pp.* 121–127.

102. Lifschitz, V., On the Satisfiability of Circumscription, *Artificial Intelligence* 28(1):17–27 (1986).

103. Lifschitz, V., On the Declarative Semantics of Logic Programs with Negation, in: *Proceedings on the Foundations of Deductive Databases and Logic Programming*, Washington, 18–22 Aug. 1986, pp. 420–432.

104. Lloyd, J. W., *Foundations of Logic Programming*, Springer, 1984.

105. Lloyd, J. W. and Topor, R. W., Making PROLOG More Expressive, *J. Logic Programming* 1(3):225–240 (Oct. 1984).

106. Lloyd, J. W. and Topor, R. W., A Basis for Deductive Databases Systems, *J. Logic Programming* 2(2):93–109 (July 1985).

107. Lloyd, J. W., Sonenberg, E. A., and Topor, R. W., Integrity Constraint Checking In Stratified Databases, Tech. Rep. 86/5 Univ. of Melbourne, 1986; *J. Logic Programming*, to appear.

108. Lloyd, J. W. and Topor, R. W., A Basis for Deductive Database Systems II, *J. Logic Programming* 3(1):55–67 (Apr. 1986).

109. Lobo, J. and Minker, J., A Metainterpreter to Semantically Optimize Queries in Deductive Databases, UMIACS #8721, Computer Science Center Tech. Rep. 1861, Univ. of Maryland, College Park, June 1987.

110. Love, H. H. and Rutman, R. A., ASP User's Manual Association Storing Processor Interpreter Program, Revised, Report FR # 70-11-275, Hughes Aircraft Co., Fullerton, Calif., Jan. 1970.

111. Lozinskii, E. L., Evaluating Queries in Deductive Databases by Generating, in: *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985, pp. 173–177.

112. Marill, T. and Murray, H., Computer Comprehension of Natural Language; a Progress Report on the Relational Structure System, unpublished report Computer Corp. of America, 17 Jan. 1969.
113. Maron, M. E., Relational Data File Design I: Design Philosophy, in: G. Schecter (ed.), *Information Retrieval*, Washington, 1967, pp. 211–223.
114. McCarthy, J., Circumscription—A Form of Nonmonotonic Reasoning, *Artificial Intelligence* 13(1 and 2):27–39 (1980).
115. McCarthy, J., Applications of Circumscription to Formalizing Common Sense Knowledge, in: *Proceedings of the AAAI Workshop on Non-monotonic Reasoning, New Paltz, N.Y.*, 1984, pp. 295–323.
116. McCarthy, J., Applications of Circumscription to Formalizing Common Sense Knowledge, *Artificial Intelligence* 28(1):89–116 (1986).
117. McSkimin, J. R., Techniques for Employing Semantic Information in Question-Answering Systems, Ph.D. Dissertation, Univ. of Maryland, College Park, 1976.
118. McSkimin, J. R. and Minker, J., The Use of a Semantic Network in Deductive Question-Answering Systems, in: *Proceedings of IJCAI 5*, 1977, pp. 50–58.
119. Minker, J., Shindle, W. E., Miller, L., and Reed, W. G., A Multi-level File Structure for Information Processing, in: *Proceedings of the Western Joint Computer Conference NJCC No. 17*, San Francisco, May 1960, pp. 53–59.
120. Minker, J. and Sable, J., File Organization and Data Management, in: *Annual Review of Information Science and Technology*, Vol. 2, New York, 1967.
121. Minker, J. and Sable, J. D., Relational Data System Study, Final Report, AUER-1776-Tech. Rep.-1, Auerbach Corp., July 1970.
122. Minker, J. and Sable, J., Relational Data System Study, Final Report, RADC-Tech. Rep.-70-180, Rome Air Development Center, Sept. 1970.
123. Minker, J., Fishman, D. H., and McSkimin, J. R., The Q^* Algorithm—a Search Strategy for a Deductive Question-Answering System, *Artificial Intelligence* 4:225–243 (1973).
124. Minker J., Performing Inferences over Relational Databases, in: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1975, pp. 79–91.
125. Minker, J., Search Strategy and Selection Function for an Inferential Relational System, *Trans. Data Base Systems* 3(1):1–31 (Mar. 1978).
126. Minker, J., An Experimental Relational Data Base System Based on Logic, in: H. Gallaire and J. Minker (eds.), *Logic and Data Bases*, Plenum, New York, Apr. 1978, pp. 107–147.
127. Minker, J. and Zanon, G., LUST Resolution: Resolution with Arbitrary Selection Function, Tech. Rep. 736, Univ. of Maryland, Feb. 1979.
128. Minker, J. and Nicolas, J., On Recursive Axioms in Relational Databases, Tech. Rep. 1119, Univ. of Maryland, July 1981.
129. Minker, J., On Indefinite Databases and the Closed World Assumption, in: *Lecture Notes in Computer Science 138*, Springer, 1982, pp. 292–308.
130. Minker, J. and Zanon, G., An Extension to Linear Resolution with Selection Function, *Inform. Process. Lett.* 14(3):191–194 (13 June 1982).
131. Minker, J. and Nicolas, J., On Recursive Axioms in Deductive Databases, *Inform. Systems* 7(4):1–15 (1982).
132. Minker, J. and Perlis, D., Applications of Protected Circumscription, in: *Proceedings of the Conference on Automated Deduction*, California, May 1984.
133. Minker, J. and Perlis, D., Protected Circumscription, in: *Proceedings of the Workshop on Non-monotonic Reasoning*, New Paltz, N.Y., 17–19 Oct. 1984, pp. 337–343.
134. Minker, J. and Perlis, D., Computing Protected Circumscription, *J. Logic Programming* 2(4):235–249 (Dec. 1985).
135. Minker, J. and Grant, J. Answering Queries in Indefinite Databases and the Null Value Problem, in: P. Kanellakis, (ed.), *Advances in Computing Research*, 1986, pp. 247–267.

136. J. Minker, *Proceedings of Workshop on Foundations of Deductive Databases and Logic Programming*, 18–22 Aug. 1986.

137. Montgomery, C. A., Automated Language Processing, in: C. A. Cuadra (ed.), *Annual Review of Information Science and Technology*, Vol. 4, Chicago, 1969, pp. 145–174.

138. Moto-Oka, T., Challenge for Knowledge Information Processing Systems (Preliminary Report on Fifth Generation Computer Systems), in: *Proceedings of the International Conference on Fifth Generation Computer Systems*, 1981, p. 1–85.

139. Naish, L. and Thom, J. A., The MU-PROLOG Deductive Database, Tech. Rep. 83/10, Univ. of Melbourne, 1983.

140. Naqvi, S. A. and Henschen, L. J., Performing Inferences over Recursive Data Bases, in: *Proceedings of the 1st Annual National Conference on Artificial Intelligence*, Stanford, Conn., Aug. 1980, pp. 263–265.

141. Naqvi, S. A., A Logic for Negation in Database Systems, in: J. Minker (ed.), *Proceedings of Workshop on Foundations of Deductive Databases and Logic Programming*, Washington, 18–22 Aug. 1986, pp. 378–387.

142. Naughton, J. F. and Sagiv, Y., A Decidable Class of Bounded Recursions, in: *Proceedings of the Sixth ACM SIGACT News–SIGMOD–SIGART Symposium on Principles of Database Systems*, San Diego, Calif., 23–25 Mar. 1987, pp. 227–236.

143. Naughton, J. F., One-Sided Recursions, *Proceedings of the Sixth ACM SIGACT News–SIGMOD–SIGART Symposium on Principles of Database Systems*, San Diego, Calif., 23–25 Mar. 1987, pp. 340–348.

144. Nicolas, J. and Syre, J., Natural Question-Answering and Automatic Deduction in System SYNTEX, in: *Proceedings of IFIP Congress 1974*, Amsterdam, 1974, pp. 595–599.

145. Nicolas, J. and Yazdanian, K., Integrity Checking in Deductive Databases, in: H. Gallaire and J. Minker (eds.), *Logic and Databases*, Plenum, New York, 1978, pp. 325–599.

146. Nicolas, J. and Gallaire, H., Data Base: Theory vs. Interpretation, in: H. Gallaire and J. Minker (eds.), *Logic and Data Bases*, Plenum, New York, 1978, pp. 33–54.

147. Nicolas, J., Logic for Improving Integrity Checking in Relational Data Bases, *Acta Inform.* 18(3):227–253 (1982).

148. Perlis, D. and Minker, J., Completeness Results for Circumscription, *Artificial Intelligence* 28(1):29–42 (1986).

149. Perlis, D., Circumscribing with Sets, *Artificial Intelligence* 31(2):201–211 (Feb. 1987).

150. Pirotte, A., High Level Data Base Query Languages, in: H. Gallaire and J. Minker (eds.), *Logic and Data Bases*, New York, 1978, pp. 409–436.

151. Porto, A., Semantic Unification for Knowledge Base Deduction, in: J. Minker (ed.), *Proceedings of the Workshop on Foundations of Deductive Databases and Logic Programming*, Washington, 18–22 Aug. 1986, pp. 102–117.

152. Przymusinski, T. C., On the Semantics of Stratified Deductive Databases, in: J. Minker (ed.), *Proceedings of the Workshop on Foundations of Deductive Databases and Logic Programming*, Washington, 18–22 Aug. 1986, pp. 433–443.

153. Przymusinski, T., A Query Answering Algorithm for Circumscriptive Theories, in: *Proceedings of the ACM SIGART International Symposium on Methodologies for Intelligent Systems*, Knoxville, Tenn., Oct. 1986, pp. 85–93.

154. Przymusinski, T., Query Answering in Circumscriptive and Closed World Theories, in: *Proceedings of the American Association for Artificial Intelligence '86*, Philadelphia, Aug. 1986, pp. 186–190.

155. Raphael, B., SIR: A Computer Program for Semantic Information Retrieval, MAC-TR2, Project MAC, Ph.D. Thesis, MIT, June, 1964.

156. Raphael, B., SIR: A Computer Program for Semantic Information Retrieval, in: *Proceedings of the AFIPS 1964 Fall Joint Computer Conference*, Vol. 26, Pt. 1, Spartan Books, New York, June 1964, pp. 577–589.

157. Raphael, B., A Computer Program for Semantic Information Retrieval, in: M. Minsky (ed.), *Semantic Information Processing*, 1968, pp. 33–134.

158. Reiter, R., Two Results on Ordering for Resolution with Merging and Linear Format, *J. Assoc. Comput. Mach.* 18:630–646 (Oct. 1971).

159. Reiter, R., An Approach to Deductive Question-Answering, Tech. Report 3649, Bolt, Beranek and Newman, Inc., Cambridge, 1977.

160. Reiter, R., Deductive Question-Answering on Relational Data Bases, in: H. Gallaire and J. Minker (eds.), *Logic and Data Bases*, Plenum, New York, 1978, pp. 149–177.

161. Reiter, R., On Closed World Data Bases, in: H. Gallaire and J. Minker (eds.), *Logic and Data Bases*, Plenum, New York, 1978, pp. 55–76.

162. Reiter, R., On Structuring a First-Order Database, in: *Proceedings of the Second Canadian Society for Computer Science National Conference*, July 1978.

163. Reiter, R., A Logic for Default Reasoning, *Artificial Intelligence* 13(1 and 2):81–132 (Apr. 1980).

164. Reiter, R., Circumscription Implies Predicate Completion (Sometimes), in: *Proceedings of the American Association for Artificial Intelligence National Conference*, Pittsburgh, 1982, pp. 418–420.

165. Reiter, R., Towards a Logical Reconstruction of Relational Database Theory, in: M. L. Brodie, J. L. Mylopoulos, and J. W. Schmit (eds.), *On Conceptual Modelling*, Springer, New York, 1984, pp. 163–189.

166. Reiter, R., A Theory of Diagnosis from First Principles, Technical Report 187/86, Univ. of Toronto, 1986.

167. Reiter, R., A Sound and Sometimes Complete Query Evaluation Algorithm for Relational Databases with Null Values, *J. Assoc. Comput. Mach.* 33(2):349–370 (Apr. 1986).

168. Reiter, R., Nonmonotonic Reasoning, *Ann. Rev. Comput. Sci.*, to appear.

169. Rieger, C., Bane, J., and Trigg, R., ZMOB: A Highly Parallel Multiprocessor, Tech. Rep. 911, Univ. of Maryland, College Park, 1980.

170. Robinson, J. A., A Machine-Oriented Logic Based on the Resolution Principle, *J. Assoc. Comput. Mach.* 12, No. 1 (Jan. 1965).

171. Roussel, P., *PROLOG: Manuel de Référence et d'Utilisation*, Group d'Intelligence Artificielle, Marseille, 1975.

172. Sagiv, Y., Optimizing Datalog Programs, in: *Proceedings of the Workshop on Foundations of Deductive Databases and Logic Programming*, Washington, 18–22 Aug. 1986, pp. 136–162.

173. Salton, G., Automated Language Processing, in: C. A. Cuadra (ed.), *Annual Review of Information Science and Technology*, Chicago, 1968, pp. 169–199.

174. Savitt, D. A., Love, H. H., and Troop, R. E., Association Storing Processor, Tech. Rep. RADC-Tech. Rep.-67-258, Vol. II, Final Report, AD 818530, Defense Documentation Center, Fullerton, Calif., June 1967.

175. Savitt, D. A., Love, H. H., and Troop, R. E., ASP: A New Concept In Language and Machine Organization, in: *1967 Spring Joint Computer Conference*, 1967, pp. 87–102.

176. Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., and Price, T. G., Access Path Election in a Relational Database Management System, in: *Proceedings of the ACM-SIGMOD International Conference on Management of Data*, New York, 30 May–1 June 1979, pp. 23–34.

177. Sellis, T., Optimization of Extended Relational Database Systems, Ph.D. Thesis, Memorandum UCB/Electronics Research Lab. M86/58, Univ. of California at Berkeley, July 1986.

178. Sellis, T., Global Query Optimization, in: *Proceedings of the 1986 ACM-SIGMOD International Conference on Management of Data*, Washington, May 1986.

179. Shepherdson, J. C., Negation as Finite Failure: A Comparison of Clark's Completed Data Base and Reiter's Closed World Assumption, *J. Logic Programming* 1(15):51–79 (June 1984).

180. Shepherdson, J. C., Negation in Logic Programming, in: J. Minker (ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan-Kaufman, 1987.

181. Simmons, R. F., Answering English Questions by Computer: A Survey, *Comm. ACM* 8(1):53–70 (Jan. 1965).
182. Simmons, R. F., Automated Language Processing, in: C. A. Cuadra (ed.), *Annual Review Series*, Interscience, New York, 1966, pp. 137–169.
183. Simmons, R. F., Natural Language Question-Answering Systems: 1969, *Comm. ACM* 13(1):15–30 (Jan. 1970).
184. Sterling, L. CS84-17, Weizmann Inst. of Science, Israel, 1984.
185. Sterling, L. and Latle, M., An Explanation Shell for Expert Systems, Technical Report 125-85, Center for Automation and Intelligent Systems Research, Cleveland, 1985.
186. Sterling, L., Meta-interpreters: The Flavors of Logic Programing, in: J. Minker (ed.), *Proceedings of the Workshop on Foundations of Deductive Databases and Logic Programming*, Washington, 18–22 Aug. 1986, pp. 163–175.
187. Thom, J. A., Naish, L., and Ramamohanarao, K., A Superjoin Algorithm for Deductive Databases, in: J. Minker (ed.), *Proceedings of the Workshop on Foundations of Deductive Databases and Logic Programming*, Washington, 18–22 Aug. 1986, pp. 118–135.
188. Topor, R. and Sonenberg, E. A., On Domain Independent Databases, in: J. Minker (ed.), *Proceedings of the Workshop on Foundations of Deductive Databases and Logic Programming*, Washington, 18–22 Aug. 1986, pp. 403–419.
189. Ullman, J. D., Implementation of Logical Query Languages for Databases, *ACM Trans. Database Systems* 10(3):289–321 (Sept. 1985).
190. Ullman, J. D., Database Theory—Past and Future, in: *Proceedings on the Principles of Database Theory*, 23–25 Mar. 1987.
191. van Emden, M. H. and Kowalski, R. A., The Semantics of Predicate Logic as a Programming Language, *J. Assoc. Comput. Mach.* 23:733–742 (1976).
192. Van Gelder, A., Negation as Failure Using Tight Derivations for General Logic Programs, in: J. Minker (ed.), *Proceedings of the Workshop on Foundations of Deductive Databases and Logic Programming*, Washington, 18–22 Aug. 1986, pp. 712–732.
193. VanderBrug, G. and Minker, J. State-Space, Problem-Reduction and Theorem Proving —Some Relationships, *Comm. ACM* 18(2):107–115 (1975).
194. Vardi, M. Y., The Decision Problem for Database Dependencies, *Inform. Process. Lett.* 12(5):251–254 (1981).
195. Vardi, M. Y., On Epistemic Logic and Logical Omniscience, in: *Proceedings on the Theoretical Aspects of Reasoning about Knowledge*, 1986, pp. 293–305.
196. Warren, D. H. D., Implementing PROLOG, Res. Rep. 39, 40, Univ. of Edinburgh, 1977.
197. Warren, D. H. D., Pereira, L. M., and Pereira, F., PROLOG—The Language and its Implementation Compared with LISP, *SIGART Newsletter*, Aug. 1977, pp. 109–115.
198. Warren, D. H. D., Efficient Processing of Interactive Relational Database Queries Expressed in Logic, in: *Proceedings of the 7th International Conference on Very Large Data Bases*, New York, 9–11 Sept. 1981, pp. 272–281.
199. Wilson, G. A. and Minker, J., Resolution, Refinements and Search Strategies—A Comparative Study, Tech. Rep. 470, Univ. of Maryland, 1976.
200. Wilson, G. A. and Minker, J., Resolution Refinements and Search Strategies—A Comparative Study, *IEEE Trans. Comput.* C-25(8):782–800 (Aug. 1976).
201. Wilson, G. A. and Minker, J., A Note on Answer Extraction in Resolution Based Systems, *Internat. J. Comput. Inform. Sci.* 6(3):179–192 (1977).
202. Wos, L. T., Robinson, G. A., Carson, D. F., and Shalla, L., The Concept of Demodulation in Theorem Proving, *J. Assoc. Comput. Mach.* 11:698–709 (1964).
203. Wos, L. T., Unpublished notes, approximately 1965.
204. Wos, L. T., Carson, D. F., and Robinson, G. A., Efficiency and Completeness of the Set of Support Strategy in Theorem Proving, *J. Assoc. Comput. Mach.* 12:687–697 (1965).
205. Yahya, A. and Henschen, L. J., Deduction in non-Horn Databases, *J. Automated Reasoning* 1:141–160 (1985).