Logic and Databases: a 20 Year Retrospective[*]

Jack Minker[1]

Department of Computer Science
Institute for Advanced Computer Studies
University of Maryland. College Park, MD 20742 U. S. A.
minker@cs.umd.edu
http://www.cs.umd.edu/~minker

Dedicated in Honor of Hervé Gallaire and Jean-Marie Nicolas[**]

Abstract. At a workshop held in Toulouse, France in 1977, Gallaire, Minker and Nicolas stated that *logic and databases* was a field in its own right (see [131]). This was the first time that this designation was made. The impetus for this started approximately twenty years ago in 1976 when I visited Gallaire and Nicolas in Toulouse, France, which culminated in a workshop held in Toulouse, France in 1977. It is appropriate, then to provide an assessment as to what has been achieved in the twenty years since the field started as a distinct discipline. In this retrospective I shall review developments that have taken place in the field, assess the contributions that have been made, consider the status of implementations of deductive databases and discuss the future of work in this area.

1 Introduction

As described in [234], the use of logic and deduction in databases started in the late 1960s. Prominent among the developments was the work by Levien and Maron [202, 203, 199, 200, 201] and Kuhns [186, 187, 188] at the Rand Corporation. Green and Raphael [151, 152] were the first to realize the importance

[*] I would like to thank the Organizing Committee of the *Workshop on Logic in Databases*, San Miniato, Italy, 1996 for inviting me to present a lecture at the Workshop. I would like to thank many of my colleagues for their thoughts on the significant developments in the field, including: Robert Demolombe, Hervé Gallaire, Georg Gottlob, John Grant, Larry Henschen, Bob Kowalski, Jean-Marie Nicolas, Raghu Ramakrishnan, Kotagiri Ramamohanarao, Ray Reiter and Carlo Zaniolo. Many of my former and current students and Ph.D. students also contributed thoughts, including Sergio Alvarez, Chitta Baral, José Alberto Fernández, Terry Gaasterland, Parke Godfrey, Jarek Gryz, Jorge Lobo, and Carolina Ruiz. Although many of the views reflected in the paper may be shared by those who made suggestions, I take full responsibility for them. Support for this paper was received from the National Science Foundation under grant number IRI 9300691.

[**] This paper is dedicated in honor of my co-authors, co-editors, colleagues and friends, Hervé Gallaire and Jean-Marie Nicolas, who have helped to make the field of deductive databases a reality.

of the Robinson Resolution Principle [294] for databases. Early uses of logic in databases are reported upon in [235] and are not covered here. Detailed descriptions of many of the accomplishments made in the 1960s can be found in [247].

A major influence of the use of logic in databases was the development of the field of logic programming through the work of Kowalski [180], who promulgated the concept of logic as a programming language, and by Colmerauer and his students who developed the first *Prolog* interpreter [79]. Throughout the paper I will refer to logic programs that are function-free as *deductive databases (DDBs).* I do so since databases are finite structures. Most of the results discussed can be extended to include logic programming.

The impetus for the use of logic in databases came about through meetings in 1976 in Toulouse, France, when I visited Hervé Gallaire and Jean-Marie Nicolas while on sabbatical leave. The idea of a workshop on "Logic and Data Bases" was conceived at that time. It is clear that a number of individuals had the idea of using logic as a mechanism to handle databases and deduction at around that time and they were invited to participate in the workshop. The appearance of the book, edited by Gallaire and Minker, *Logic and Data Bases*[3], [131], was highly influential in the development of the field, as were the books [132, 133] that were a result of two subsequent workshops held in Toulouse. Another influential development was the article by Gallaire, Minker and Nicolas [134], which surveyed work in the field up to the date of the article.

The use of logic in databases was received by the database community with a great deal of skepticism. It is reported that one of the key researchers at the time stated that, "...logic and databases had little to do either with the theory or the practice of databases." It might also be well to recall some of the comments made in the review of the book *Logic and Data Bases*, [155]. The reviewer took issue with the statement in the Foreword to the book, where Gallaire and I called *logic and databases* a *field*. The reviewer believed that a *field* had to satisfy criteria that he set forth and that these criteria were not met by the book. He stated,

> More significant are the conscious efforts of the editors and most of the authors to promote the work in the volume as representing a field – a well-defined area of research, complete with important past achievements and stimulating areas for future work.

At the time the review came out, and in retrospect, I believe that Gallaire and I were correct in our assessment that *logic and databases* is, indeed, a *field*. In [235] 'important past achievements and stimulating areas for future work' are described. The accomplishments I cite in this article, and the papers that appear in this workshop, are testaments to the accuracy of our remarks.

The reviewer also stated,

> The present volume is a collection of papers on a topic, some more interesting than others but none really outstanding...

[3] Nicolas was not one of the editors of the book by his choice. He played a key role in organizing the workshop.

While each individual may have different criteria for what is outstanding, it should be noted that there were several papers in the book that have now become *classics* in our field. I refer to the article by Reiter on the *Closed World Assumption (CWA)* [288] and by Clark on *Negation-as-Failure (NAF)* [75]. Indeed, several other important papers appeared in the book, such as those by Nicolas and Gallaire [257] on theory vs. interpretation, by Kowalski [181] on logic for data description, by Nicolas and Yazdenian [259] on integrity constraints, and several early implementations [64, 172, 231] of *DDBs*.

It is clear that logic has everything to do with the theory of databases, and many of those who were then critical of the field have changed their position. In the remainder of this paper I shall describe what I believe to be the major intellectual developments in the field, the status of commercial implementations and future trends. As will be seen, the field of logic and databases has been very prolific. I apologize in advance to any author whose work I have inadvertently not covered or referenced.

2 Intellectual Contributions of Deductive Databases

In describing the contributions of logic to databases, it is necessary to take note of work in relational databases as specified by Codd [77]. Codd formalized databases in terms of the relational calculus and the relational algebra. He provided a logic language, the relational calculus, and described how to compute answers to questions in the relational algebra and the relational calculus. Both the relational calculus and the relational algebra provide declarative formalisms to specify queries. This was a significant advance over network [76] and hierarchic systems [335], which only provided procedural languages for databases. The relational algebra and the relational calculus permitted individuals who were not computer specialists to write declarative queries, and to have the computer answer the queries. The development of syntactic optimization techniques (see [84, 337, 338] for references on this topic) permitted relational database systems to retrieve answers to queries efficiently and to compete with network and hierarchic implementations. Relational systems have been enhanced to include "views". A view as has been used in relational databases is, essentially, a non-recursive procedure. There are, today, numerous commercial implementations of relational database systems for large database manipulation and for personal computers. Relational databases may be considered a forerunner of logic in databases.

Although relational databases used the language of logic in the relational calculus, it was not formalized in terms of logic. The formalization of relational databases in terms of logic and the extensions that have been developed are the focus of this paper. Indeed, *formalizing databases through logic has played a significant role in our understanding of what constitutes a database, what is meant by a query, what is meant by an answer to a query, and how databases may be generalized for knowledge bases.* It has also provided tools and answers to problems that would have been extremely difficult without the use of logic.

In the remainder of the paper I will focus on some of the more significant aspects that have been contributed by logic in databases. I shall discuss the following contributions:

1. A formalization of what constitutes a database, a query, and an answer to a query.
2. A realization that logic programming extends relational databases.
3. A clear understanding of the semantics of large classes of databases that include alternative forms of negation, and disjunction.
4. An understanding of relationships between model theory, fixpoint theory and proof procedures.
5. An understanding of the properties that alternative semantics may have and their complexity.
6. An understanding of what is meant by integrity constraints and how they may be used to perform updates, semantic query optimization, cooperative answering and database merging.
7. A formalization and solutions to the update and view update problems.
8. An understanding of bounded recursion and recursion, and how they may be implemented in a practical manner.
9. An understanding of the relationship between logic based systems and knowledge base systems.
10. A formalization of how to handle incomplete information in knowledge bases.
11. A correspondence that relates alternative formalisms of nonmonotonic reasoning to databases and knowledge bases.

I will address the area of implementations of *DDBs* in section 3, where commercial developments have not progressed as rapidly as the intellectual developments. I will then discuss some trends and future directions in section 4.

2.1 Formalizing Database Theory

The first formalization of databases in terms of logic was due to Reiter [290] who noted that underlying relational databases there were a number of assumptions that were not made explicit. In particular, he noted that with respect to negation, an assumption was being made that facts not known to be *true* in a relational database, are assumed to be *false*. This assumption is the well-known *closed world assumption (CWA)*, expounded earlier by Reiter in 1978 [288]. A second assumption is the *unique name assumption*, which states that any item in a database has a unique name, and that individuals with different names are not the same. The last assumption is the *domain closure assumption*, which states that there are no other individuals than those in the database.

Having recognized these assumptions, Reiter then formalized relational databases as follows. He stated that a relational database is a set of ground assertions over a language $\mathcal{L}$ together with a set of axioms. The language $\mathcal{L}$ does not contain function symbols. These assertions and axioms are of the following form:

- **Assertions:** $R(a_1, \cdots, a_n)$, where R is an n-ary relational symbol in $\mathcal{L}$, and $a_1 \cdots, a_n$ are constant symbols in $\mathcal{L}$.

- **Unique Names Axiom:** If $a_1, \cdots, a_p$ are all the constant symbols of $\mathcal{L}$, then

$$(a_1 \neq a_2), \cdots, (a_1 \neq a_p), (a_2 \neq a_3), \cdots, (a_{p-1} \neq a_p)$$

- **Domain Closure:** If $a_1, \cdots, a_p$ are all the constant symbols of $\mathcal{L}$, then

$$\forall X((X = a_1) \vee \cdots \vee (X = a_p))$$

- **Completion Axioms:** For each relational symbol R, if $R(a_1^1, \cdots a_n^1), \cdots, R(a_1^m, \cdots, a_n^m)$ denote all facts under R, the completion axiom for R

$$\forall X_1 \cdots \forall X_n (R(X_1, \cdots, X_n) \rightarrow \\ (X_1 = a_1^1 \wedge \cdots \wedge X_n = a_n^1) \vee \cdots \vee (X_1 = a_1^m \wedge \cdots \wedge X_n = a_n^m))$$

- **Equality Axioms:**

$$\forall X\ (X = X)$$
$$\forall X \forall Y\ ((X = Y) \rightarrow (Y = X))$$
$$\forall X \forall Y \forall Z((X = Y) \wedge (Y = Z) \rightarrow (X = Z))$$
$$\forall X_1 \cdots \forall X_n (P(X_1, \cdots, X_n) \wedge \\ (X_1 = Y_1) \wedge \cdots \wedge (X_n = Y_n) \rightarrow P(Y_1, \cdots Y_n))$$

The *completion axiom* is what Clark had proposed as the basis for his Negation-as-Failure rule [75]. Another contribution of logic programs and databases is that:

The formalization of relational databases in terms of logic now permits the definition of a query and an answer to a query to be defined precisely. A query is a statement in the first-order logic language $\mathcal{L}$. Q(a) is an answer to a query, Q(X), over a database DB if Q(a) is a logical consequence of DB.

2.2 Deductive Databases (DDBs)

Relational databases have been shown to be a special case of deductive databases. A deductive database may be considered as a theory, *DDB*, in which the database consists of a set of ground assertions, referred to as the *extensional database (EDB)*, and a set of axioms, referred to as the *intensional database (IDB)*, of the form:

$$P \leftarrow Q_1, \ldots, Q_n, \tag{1}$$

where $P, Q_1, \ldots, Q_n$ are atomic formulae in the language $\mathcal{L}$. Databases of this form are termed *Datalog* databases [337, 338]. The name *Datalog* comes from those in the database community. A *Datalog* database is a particular instance of a more general Horn logic program that permits function symbols in clauses given by Formula (1). The recognition that logic programs are significant for databases was understood by a number of individuals in 1976. Many of them have articles in the book [131]. The generalization permits views to be defined that are recursive.

The recognition that logic programming and databases are fundamentally related has led to more expressive and powerful databases than is possible with relational databases defined in terms of the relational algebra.

That logic programming and *DDBs* are fundamentally related is a consequence of the fact that databases are function-free logic programs. As shown in many papers and in particular, in Gottlob [146], the expressive power of logic programming extends that of relational databases.

Integrity constraints play an important role in database systems. In addition to defining a database in terms of an *EDB* and an *IDB*, it is also necessary to formalize what is meant by an *integrity constraint (IC)*. There are a number of definitions as to what constitutes an *IC*. Kowalski [306, 181] suggests that an integrity constraint is a formula that is *consistent* with the *DDB*, while for Reiter, and Lloyd and Topor [290, 215], an *IC* is a theorem of the *DDB*. Reiter has proposed other possible definitions of integrity constraints in [292, 293]. He states that *IC*s should be statements about the content of a database. *IC*s can be written in a modal logic to use a belief operator to express beliefs that the database must satisfy. He explores Levesque's KFOPCE [198], an epistemic modal logic, as a suitable framework.

In [91], Demolombe and Jones revisit the basic notion of *IC*s in deductive databases. Their view is that *IC*s are statements *true* about the world, whereas a database is a collection of *beliefs* about the world. *IC*s then can be used to qualify certain information in the database as *valid* or *complete*. They define these properties formally in the framework of doxatic logic.

The major use that has been made of integrity constraints in the database community has been in updating to assure that the database is consistent. Nicolas [256] has shown how, using techniques from *DDBs*, improvements can be made to the speed of update. Blaustein [33] has also made contributions to this problem. It has been shown by Reiter [288] that one can query a Horn database with or without integrity constraints and the answer to the query is the same. However, this does not preclude the use of integrity constraints in the query process. While *ICs* do not affect the result of a query, they may affect the efficiency with which this result may be computed. Integrity constraints provide *semantic* information about the data in the database. If a query requests a join for which there will never be an answer because of the system constraints, this can be used to advantage by not performing the query and returning an empty answer set. This avoids performing an unnecessary join on two potentially large relational tables in a relational database system, or performing a long deduction in a *DDB* system. The process of using integrity constraints to constrain a search has been referred to as *semantic query optimization (SQO)* [57, 58]. McSkimin and Minker [227] were the first to use integrity constraints for *SQO* in *DDBs*. Hammer and Zdonik [154] and King [177] were the first to apply *SQO* to relational databases. Chakravarthy, Grant and Minker [57, 58] formalized *SQO* and developed the *partial subsumption algorithm* and method of *residues*. These provide a general technique applicable to any relational or *DDB* that is able to perform *SQO*. Godfrey, Gryz and Minker, [144], show how to apply the technique in a bottom-up approach. Gaasterland and Lobo [128] extended the work to include databases with negation in the body of clauses, and Levy and Sagiv [206] showed how to handle recursive *IDB* rules in *SQO*. The early work on semantic query optimization by [227] may be a forerunner of the work that has been developed on constraint logic programming by Jaffar and Lassez [167]. See Jaffar and Maher for a comprehensive survey of constraint logic programming [166].

A topic related to semantic query optimization is that of *cooperative answering systems*. The objective in a cooperative answering system is to provide information to a user as to why a particular query succeeded or failed. When a query fails, the user, in general, cannot tell why the failure occurred. There may be several reasons: the database currently does not contain information to respond to the user, or there will never be an answer to the query. The distinction could be important to the user. Another aspect related to integrity constraints is that of *user constraints (UCs)*. A user constraint is a formula, that models a user's preferences. It may constrain providing answers to queries in which the user may have no interest (e.g., stating that in developing a route of travel, the user does not want to pass through a particular city), or provide other constraints that may restrict the search. As shown by [130, 129, 122, 125], user constraints, which are identical in form to integrity constraints, can be used for this purpose. While integrity constraints provide the semantics of the entire database, user constraints provide the semantics of the user. User constraints may be inconsistent with the database and hence, a separation of these two semantics is essential. To maintain the consistency of the database, only integrity constraints are relevant. A query may be thought of as the conjunction of the query and the user constraints. Hence, a query can be semantically optimized based both on *ICs* and *UCs*.

With user and integrity constraints, it is possible to develop a system that provides responses to users that inform them as to the reasons why queries succeed or fail [126, 145]. Other features may be built into a system, such as the ability to relax a query given that it fails, so that an answer to a related request may be found. This has been termed *query relaxation* [125]. A survey of work in cooperative answering systems may be found in [123].

Semantic query optimization, user constraints and cooperative answering systems are important contributions both for relational and DDB systems. I believe that they will eventually be incorporated into commercial relational and DDB systems.

Indeed, I cannot imagine a *DDB* developed for commercial systems to be successful if it does not contain both semantic query optimization and cooperative answering capabilities. How can one expect users to understand why deductions succeed or fail if such information is not provided? How can queries doomed to fail because they violate integrity or user constraints be allowed to take up a significant amount of search time if the query cannot possibly succeed? I also believe that these techniques must be incorporated into relational technology. As discussed in section 3, this is beginning to happen. Practical considerations of performing semantic query optimization in a bottom-up approach have been addressed by [144].

2.3 Extended Deductive Database Semantics

The first generalization of relational databases was to permit Horn rules that could be recursive. A Horn rule is one in which the head of a rule is an atom and the body of a rule is a conjunction of atoms. These databases are referred to as deductive databases. Subsequently, other *DDBs* that may contain negated atoms in the body of rules were permitted. In the following sections I discuss the alternative extensions that have been made to *DDBs* and describe why they are significant.

Horn Semantics and Datalog One of the early developments was due to van Emden and Kowalski [340] who wrote a seminal paper that discussed the semantics of Horn theories. I believe that a significant contribution to logic programming and *DDBs* was *the recognition by van Emden and Kowalski that the semantics of Horn databases can be characterized in three distinct ways by model, fixpoint or proof theory. These three characterizations lead to the same semantics.*

Model theory deals with the definition of a collection of models of a database that captures the intended meaning of the database. Fixpoint theory deals with the definition of a fixpoint operator that constructs the collection of all atoms that can be inferred to be *true* from the database. Proof theory deals with the definition of a procedure that finds answers to queries with respect to the database. van Emden and Kowalski [340] also showed that if one considers all Herbrand models of a Horn *DDB*, the intersection is a unique minimal model. The unique minimal model is the same as all of the atoms in the fixpoint, and are the only atoms provable from the theory.

To find if the negation of a ground atom is *true*, one can subtract, from the Herbrand base (the set of all atoms that can be constructed from the constants and the predicates in the database), the minimal Herbrand model. If the atom is in this set then it is assumed to be *false* and its negation is *true*. Alternatively, answering queries that consist of negated atoms that are ground may be achieved using negation-as-finite failure as described by [288, 75].

The first approaches to answering queries in *DDBs* did not handle recursion and were primarily top-down (or backward reasoning) or top-down using sets [131]. The approach to answering queries in relational database systems was a bottom-up (or forward reasoning) approach, since all answers are usually required and it is more efficient to do so in a bottom-up approach. Pioneering work in handling recursion in *DDBs* was done by Chang [64, 65], McKay and Shapiro [317] and Henschen and Naqvi [156]. Both Chang, and McKay and Shapiro used a bottom-up approach to handle recursion, while Henschen and Naqvi used an iterated top-down approach. This work was followed by a large number of papers on computing recursion including the *naive* and *semi-naive evaluation* *methods* [11]. These methods are based on primitive deduction techniques and are generally inefficient. They perform bottom-up reasoning proceeding from the database. Hence, they do not use constants that may appear in a query to guide the search. Top down reasoning takes into account constants that appear in a query. The *QSQ* method introduced by Vieille [344, 346] and the extension tables method defined by Dietrich and Warren [92] are key representatives of this approach. On the other hand, the renaming of the Alexander [295] and *magic set* [12, 24] methods make use of the constants for bottom-up reasoning. Bry [48] reconciles the bottom-up and top-down methods to compute recursive queries. He shows that the Alexander and magic set methods based on rewriting and the methods based on resolution implement the same top-down evaluation of the original database rules by means of auxiliary rules processed bottom-up. Based on the work by Bry, Brass [43] developed a rewriting method for *Datalog-programs* which simulates *SLD-resolution* more closely than the usual magic set method. It improves upon the method of Ross [299], can save joins and simulates *SLD-resolution* closely. In addition, it was also first shown by Minker and Nicolas [239] that there are forms of rules that lead to *bounded recursion.* That is, the deduction process using these rules must terminate in a finite number of steps. This work has been extended by Naughton and Sagiv [254].

The efficient handling of recursion and the recognition that some recursive cases may inherently be bounded contributes to the practical implementation of deductive databases. An understanding of the relationship between resolution-based (top-down) and fixpoint-based (bottom-up) techniques and how the search space of the latter can be made identical to top-down resolution with program transformation is another contribution of DDBs.

Extended Deductive Databases and Knowledge Bases The ability to develop a semantics for theories in which there are rules with a literal (i.e., an atomic formula or the negation of an atomic formula) in the head and literals with possibly negated-by-default literals in the body of a clause, has significantly expanded the ability to write and understand the semantics of complex applications. Such clauses, referred to as *extended clauses*, are given by:

$$L \leftarrow M_1, \cdots, M_n, not\ M_{n+1}, \cdots not\ M_{n+k}, \tag{2}$$

where L and the M_j, $j = 1, \cdots, (n+k)$ are literals. Databases with such clauses combine both *classical negation* and *default negation* (represented by *not* immediately preceding a literal), and are referred to as *extended deductive databases.* The combining of classical and default negation provides users with greater expressive power.

The development of logic programs where default negation may appear in the body of a clause first appeared in the proceedings of a workshop, held in 1986 [233]. Selected papers from the workshop were published in 1988 [234]. The concept of stratification was discussed first by Chandra and Harel [62] and introduced to logic programs by Apt, Blair and Walker [7], and by Van Gelder [341] who considered stratified theories in which L and the M_j in Formula (2) are atomic formulas and there is no recursion through negation. For such theories they show that there is a unique preferred minimal model, computed from strata to strata. Przymusinski [271] has termed this minimal model to be the *perfect model*. He also extends the concept to *locally stratified* theories.

The theory of stratified databases was followed by that permitting recursion through negation in Formula (2) where the L and M_j are atomic formulae. In the context of *DDBs* they are called *normal deductive databases*. There have been a large number of papers devoted to defining the semantics of these databases. These semantics, drawn from [245], are summarized in the second and third columns of Table 1. The most prominent of these for the Horn case, are the *well-founded semantics* of Van Gelder, Ross and Schlipf [137], and the *stable semantics* of Gelfond and Lifschitz [139]. The well-founded semantics leads to a unique three-valued model. Stable semantics may lead to a collection of minimal models. For some *DDBs* this collection may be empty. Fitting [119] has also defined a three-valued model to capture the semantics of normal logic programs.

There have been a number of normal deductive database semantics that have been developed that have been shown to be equivalent. You and Yuan [352], show that a number of extensions to the stable model semantics coincide: the regular model semantics of You and Yuan [350], the partial stable model of Saccà and Zaniolo [305], the preferential semantics of Dung [103], and a stronger version of the stable class semantics of Baral and Subrahmanian [22]. R-stable models have been proposed by Jakobovits and Vermier [168].

There have been several implementations of the well-founded semantics. Chen and Warren [67] have developed a top-down approach to answer queries in this semantics, while Leone and Rullo [194] have developed a bottom-up method for *Datalog* databases. Sagonas, Swift and Warren [307] show that a fixed order of computation does not suffice for answering queries in the well-founded semantics. They introduce a variant of SLG resolution [332], SLG_{strat} which uses a fixed computation rule to evaluate ground left-to-right dynamically stratified programs. There have been several methods developed for computing answers to queries in stable model semantics. Fernández et al. [117] develop a bottom-up approach based on the concept of *model trees*. Every branch of a model tree is a model of the database, where a node in a tree is an atom that is shared by each branch below that node. Nerode et al. [25] have developed a method based on linear programming. Also, Inoue, Koshimura and Hasigawa [163] have developed a method to compute stable model semantics. Ruiz and Minker, [301], devised a procedure to construct the collection of partial (3–valued) stable models of a *DDB*. Baggai and Sunderraman [10] have developed a bottom-up method to compute the Fitting model.

A further extension of normal deductive databases, proposed by Gelfond and Lifschitz [140] and by Pearce and Wagner [260], permits clauses in Formula (2) where L and M_j are literals and therefore combines classical and default negation in one database. Blair and Subrahmanian [32] use the same idea as in [140, 260] to handle literals in paraconsistent logic programs. The semantics for for normal deductive databases has been described by [244, 245].

Semantics	Horn		Disjunctive	
	Theory	Reference	Theory	Reference
Positive Consequences				
Fixpoint	T_P	[340]	T_P^I	[243]
Model	Least Model	[340]	Min. Models	[232]
			Model-State	[276]
Procedure	SLD	[158]	SLI/SLO	[248, 276]
			Case Based	[286]
Negation				
Theory	CWA	[288]	GCWA	[232, 243]
			WGCWA	[300, 277]
Rule	NAF	[75]	SN-rule	[241]
			NAFFD-rule	[277]
Procedure	SLDNF	[75]	SLONF	[276]
Stratified Programs				
Fixpoint	T_P	[7]	T_P^C	[243]
			T_P^I	[300, 242]
Model	Standard	[7]	Stable State	[242]
	Perfect	[267]	Perfect	[271]
Procedure	SLS	[272]	SLP	[115]
Normal Programs				
	Well-Founded		Strong/Weak Well-F/Stationary	
Fixpoint	I^{∞}	[138]		
Model	$M_{WF}(P)$	[138]	$M_{WF}^{S/W}(P)$	[297]
			M_P	[269]
Procedure	SLS	[296, 268]		
	General Well-Founded		General Disj. Well-Founded	
Fixpoint	I^E	[19]	S^{ED}	[20]
Model	M_P^E	[21]	MS_P^{ED}	[21]
Procedure	SLIS	[20]	SLIS	[269]
	Stable Models		Stable Models	
Fixpoint	T_P^M	[117, 163]	T_P^M	[117, 163]
Model	Stable	[139]	Stable	[273, 274]
Procedure	SLP	[115]	SLP	[115]
Extended Programs				
	Stable Models		Stable Models	
Fixpoint				
Model	Stable	[140, 141]	Stable	[273, 274]
Procedure				
	Stationary		Stationary	
Fixpoint	$\mathcal{S}^{SE}$	[244]	$\mathcal{S}^{SE}$	[244]
Model	Stationary	[3]		
Procedure				
	Arbitrary Semantics SEM		Arbitrary Semantics SEM	
Fixpoint	T_P^E	[244, 245]	T_P^E	[244, 245]
Model	$\mathcal{M}_P^{SEM}$	[244, 245]	$\mathcal{M}_P^{SEM}$	[244, 245]
Procedure	EPP_{SEM}	[244, 245]	EPP_{SEM}	[244, 245]

Table 1. (Taken from [245]) Semantics of Horn and Disjunctive Logic Programs

These notions of default negation have been used as separate ways to interpret and to deduce default information. That is, each application has chosen one of these notions of negation and has applied it to every piece of data in the domain of the application. In [246], Minker and Ruiz define a new class of more expressive DDBs that allow for the combination of several forms of default negation in the same database. In this way different pieces of information in the domain may be treated appropriately. They introduce a new semantics called the *well–founded stable* semantics that characterizes the meaning of DDBs that combine the well–founded and the stable semantics.

Many complexity results have been obtained for deductive databases. Schlipf [313] has written a comprehensive survey article that summarizes the results. Some of these results, taken from [313], are listed in Table 2. A user may wish to determine which semantics to be used based upon the complexity expected to find answers to queries.

The development of the semantics of extended deductive databases that permit a combination of classical negation and multiple default negations in the same deductive database are important contributions. The study of and the development of results in the computational complexity of these databases are important contributions to database theory. They permit wider classes of applications to be developed.

Knowledge bases are important for artificial intelligence and expert system developments. A general way in which one can represent knowledge bases is through logic. All of the work developed for extended deductive databases concerning semantics and complexity apply directly to knowledge bases. Baral and Gelfond [13] have written an article which describes how extended deductive databases may be used to represent knowledge bases. Extended deductive databases, together with integrity constraints permit a wide range of *knowledge bases* to be implemented. Many papers devoted to knowledge bases consider them to consist of facts and rules. Certainly, this is one aspect of a knowledge base, as is the ability to extract proofs. However, integrity constraints supply another aspect of knowledge and differentiates knowledge bases which may have the same rules, but different integrity constraints. I believe that one should *define a knowledge base to consist of an extended deductive database plus integrity constraints.*

Since alternative extended deductive semantics have been implemented, the knowledge base expert should focus on the particular knowledge base problem that is to be developed. His focus should be upon writing the rules and integrity constraints that characterize the database, selecting the particular semantics that meets the needs of the problem and employing a *DDB* system that uses the required semantics.

The field of deductive databases has contributed to providing an understanding of knowledge bases and their implementation.

2.4 Extended Disjunctive Deductive Database Semantics

In databases that have been discussed above, information is definite. However, there are many situations where our knowledge of the world is incomplete. For example, when a null value appears as an argument of an attribute of a relation, the value of the attribute is unknown. Uncertainty in databases may be represented by probabilistic information. See [190, 330, 255]. Another area of incompleteness arises when it is unknown as to which among several facts are *true*, but it is known that one or more are *true*. It is therefore necessary to be able to represent and understand the semantics of theories that include incomplete data. A natural way to extend databases to include incomplete data is to permit disjunctive statements as part of the language. This leads to deductive databases which permit clauses to have disjunctions in their heads. These clauses are represented as,

$$L_1 \vee L_2 \vee \ldots \vee L_m \leftarrow M_1, \ldots, M_n, not\ M_{n+1}, \ldots not\ M_{n+k}, \qquad (3)$$

and are referred to as *extended disjunctive clauses.* A database that consists of such clauses is referred to as an *extended disjunctive deductive database (EDDDB).* The book by Lobo, Minker and Rajasekar [216] describes the theory of disjunctive logic programs and includes several chapters devoted towards disjunctive deductive databases.

In section 2.4, I discuss the semantics of *DDDBs*, where clauses are given by Formula (3), where the literals are restricted to atoms and there is no default negation in the body of a clause. In section 2.4, I discuss the semantics of *EDDDBs*, where there are no restrictions on clauses in Formula (3).

Disjunctive Deductive Databases (DDDBs) As I noted in [236], work in disjunctive theories was pursued seriously after a workshop I organized in 1986 [233]. The field of *disjunctive deductive databases (DDDBs)* started approximately in 1982 with the appearance of the paper by Minker [232], who described how one can answer both positive and negated queries in such databases. For a historical perspective of disjunctive logic programming and *DDDBs*, see [235]. There is a major difference between the semantics of deductive databases and those for *DDDBs*. Whereas *DDBs* usually have a unique minimal model that describes the meaning of the database, *DDDBs* generally have multiple minimal models.

As shown in [232] it is sufficient to answer positive queries over *DDDBs* by showing that the query is satisfied in every minimal model of the database. Thus, in the *DDDB*, $\{a \vee b\}$, there are two minimal models, {{a}, {b}}. The query, a?, is not satisfied in the model $\{b\}$, and hence, it cannot be concluded that a is *true*. However, the query, $(a \vee b)$ is satisfied in both minimal models and hence the answer to the query $\{a \vee b\}$ is *yes*. To answer negated queries, it is not sufficient to use Reiter's *CWA* [288] since, as he noted, from the theory $DB = \{a \vee b\}$, it is not possible to prove a, and it is not possible to prove b. Hence, by the *CWA*, *not a* and *not b* follow. But, $\{a \vee b, not\ a, not\ b\}$ is not consistent. The

Generalized Closed World Assumption (GCWA), [232] resolves this problem by specifying that a negated atom be considered *true* if the atom does not appear in any minimal model of the database. This provides a model theoretic definition of negation. An equivalent proof theoretic definition, also presented in [232], is that an atom a may be considered to be *false* if, whenever $a \vee C$ may be proven from the database, then also C may be proven from the database, where C is an arbitrary positive clause.

For related work on negation in disjunctive theories see Yahya and Henschen [349], Gelfond, Przymusinski and Przymusinska [143], Chan [59, 60], Sakama [308], Suchenek [331], Bossu and Siegel [38], Ross and Topor [300], and Lobo, Rajasekar and Minker [277]. For surveys on negation see the articles by Shepherdson [318], by Apt and Bol [8], by [99] and by Minker [237].

In deductive databases, it is natural for the fixpoint operator to map atoms to atoms. However, for *DDDBs*, it is natural to map positive disjunctions to positive disjunctions. A set of positive disjunctions is referred to as a *state*. A *model state* is a state all of whose minimal models satisfy the *DDDB*. The concept of a state was defined by Minker and Rajasekar [240, 243] as the domain of a fixpoint operator T_P whose least fixpoint characterizes the semantics of a disjunctive logic program P. The operator is shown to be monotonic and continuous, and hence converges in ω iterations. The fixpoint computation operates bottom-up and yields a minimal model state that is logically equivalent to the set of minimal models of the program. The Minker/Rajasekar fixpoint operator is an extension of the van Emden/Kowalski fixpoint operator. If one considers all model states of a *DDDB* and intersects them, then the resultant is a model state, and among all model states it is minimal. Hence, one obtains a unique minimal model in a Horn database, while one obtains a unique model state in a *DDDB*.

Decker [86, 87] develops a fixpoint operator for *DDDBs*. His fixpoint operator reduces to the Minker/Rajasekar fixpoint operator [240, 243]. At each iteration of his operator, Decker finds partial models of the database. In the limit, he obtains the set of minimal models of the database. If one were to take an atom from each minimal model and form a disjunction, the resulting set of all such disjunctions is equivalent to the minimal model state of the *DDDBs*.

Answering queries in *DDDBs* has been studied by a number of individuals. Grant and Minker [238] were among the first to address the problem of computing answers to queries in *DDDBs*. They investigated the case where the database consists exclusively of ground positive disjuncts. Yahya and Henschen [349] developed a deductive method to determine whether or not a conjunction of ground atoms can be assumed *false* in a *DDDB* under the *EGCWA*. The *EGCWA* is an extension of the *GCWA*. Bossu and Siegel [38] developed a deductive method to answer a query by subimplication (a generalization of the *GCWA* that handles databases that have no minimal models). Henschen and Park [157] provide a method to answer *yes/no* questions is a database that consists of an *EDB*, an *IDB* and *ICs* that are all function-free. In addition, they allow negated unit clauses to be part of the database. The axioms in the *IDB* may be recursive. Liu and Sunderraman [212] generalize the relational model to represent disjunctive data. For this purpose they develop a data-structure, called *M-table* to represent the data. The relational algebra is then generalized to operate on *M-tables*. Although their extended algebra is sound, it is not complete. They answer some types of queries with a *maybe*, which may in fact be *true*. Yuan and Chiang [354] have developed a sound and complete query evaluation algorithm for disjunctive databases, that is, for databases that do not contain recursive *IDB* rules. They define an extended algebra in terms of an *extended union, extended intersection, extended projection, extended selection, extended cartesian product* and *extended division* operators.

Fernández and Minker [118] developed the concept of a *model tree*. They show how one can incrementally compute sound and complete answers to queries in *hierarchical DDDBs*. A *DDDBs* is hierarchical if it contains no recursion. In [116] they show how one can develop a fixpoint operator over trees to capture the meaning of a *DDDB* that includes recursion. The tree representation of the fixpoint is equivalent to the Minker/Rajasekar fixpoint [240, 243].

The above approaches to answering queries in disjunctive databases have the following limitations. Grant and Minker [238] can only compute answers to queries that contain a disjunctive extensional database. Henschen and his students [349, 157, 68] can only answer *yes/no* questions. Liu and Sunderraman [211, 212] provide sound, but not complete answers to queries. To be able to compute sound and complete answers, Yuan and Chiang [354] essentially have to compute the fixpoint of the entire *DDB* to answer each query. Fernández and Minker compute the model tree of the extensional *DDDB* once. To answer queries *IDB* rules may be invoked. However, the models of the extensional disjunctive part of the database do not have to be generated for each query. The approach by Fernández and Minker to compute answers generalizes both to stratified and normal *DDDB*.

Loveland and his students [214, 218, 321, 287, 286] have developed a top-down approach when the database is *near Horn*. They have developed a case-based reasoner that uses *Prolog* to perform the reasoning. This is one of the few efforts that have implemented disjunctive deductive databases. Loveland, Reed and Wilson [219] have introduced a relevancy detection algorithm to be used with *SATCHMO*, developed by Manthey and Bry, for automated theorem proving. Their system, termed *SATCHMORE (SATCHMO with RElevancy)*, improves on *SATCHMO* by limiting the uncontrolled use of forward chaining. Another approach using theorem proving is the work by Stickel on a theorem prover *PTTP (Prolog Technology Theorem Prover)* [327].

A number of individuals have investigated the complexity of answering queries in disjunctive logic programs. For example, Imielinski and Vadaparty [162], Vardi [342] and Imielinski [160]. Chomicki and Subrahmanian [71] discuss the complexity of the Generalized Closed World Assumption. There are some disjunctive theories that are tractable. See Borgida and Etherington [37], and Lobo, Yu and Wong [217] for such cases. See Dalal [82] for a discussion of an algorithm to detect if a disjunctive theory is tractable. A comprehensive description of complexity results for propositional logic programs is given by Eiter and Gottlob

[106]. A summary of complexity results, drawn from [106], is given in Table 3.

The development of model theoretic, fixpoint and proof procedures has placed the semantics of DDDBs on a firm foundation. Methods to handle DDDBs have started to be developed and should eventually enhance implementations. The GCWA and alternative theories of negation have enhanced our understanding of default negation in DDDBs. Complexity results provide an understanding of the difficulties to find answers to queries in such systems.

Extended Disjunctive Deductive Databases Fernández and Minker [114] present a new fixpoint characterization of the minimal models of disjunctive and stratified disjunctive deductive databases. They prove that by applying the operator iteratively, in the limit, it constructs the perfect models semantics (Przymusinski [271]) of stratified disjunctive deductive databases. Given the equivalence between the *perfect models semantics of stratified programs* and *prioritized circumscription* [271] their fixpoint characterization captures the meaning of the corresponding circumscribed theory. Based on these results they present a bottom-up evaluation algorithm for stratified *DDDBs*. This algorithm uses the *model-tree* data structure to represent the information contained in the database and to compute answers to queries. In [112, 113], Fernández and Minker develop the theory of *DDDBs* using the concept of model trees. Work on updates in *DDDBs* is described in [148, 149, 111].

Four alternative semantics were developed for non-stratifiable normal *DDDBs* at approximately the same time: Ross [297], Baral, Lobo and Minker [21, 20] and two other semantics by Przymusinski [269, 273]. Ross termed his semantics the *strong well founded semantics*, Baral, Lobo and Minker defined their semantics to be the *Generalized Disjunctive Well-Founded Semantics (GDWFS)*. They defined a fixpoint operator, and gave a model theoretic and proof theoretic semantics for such *DDDBs*. Przymusinski [273, pag. 466] defines an extension of the *stable model semantics* for normal disjunctive databases. He also defined in [269] the *stationary semantics*. As in the case of normal deductive databases it will be necessary to develop effective bottom-up computing techniques to answer queries in these theories. Some results already exist in this area for some classes of disjunctive databases as noted above.

In addition to the above semantics, a number of other important semantics have been developed. Przymusinski [270] describes a new *semantic framework* for disjunctive logic programs and introduces the *static expansions* of disjunctive programs. The class of static expansions extends both the classes of stable, well-founded and stationary models of normal programs and the class of minimal models of disjunctive programs. Any static expansion of a program P provides the corresponding semantics for P consisting of the set of all sentences logically implied by the expansion. The stable model semantics has also been extended to disjunctive programs [141, 274]. Leone, Rullo and Saccà [195], develop an efficient algorithm for solving the (co-NP-hard decision) problem of checking if a model is stable. The algorithm runs in polynomial space and single exponential time (in the worst case). The algorithm runs in polynomial time on the class of *head-*

cycle free programs (discussed below), and in the case of general disjunctive logic programs limits the inefficient part of the computation only to the components of the program which are not head-cycle free. The D-WFS semantics of Brass and Dix [45] is also of considerable interest as it permits a general approach to bottom-up computation in disjunctive programs.

The last two columns of Table 1 summarize work accomplished in the semantics of *DDDBs*, and is drawn from [245]. See, also, the paper by Eiter, Leone and Saccà [108], in this collection, that summarizes complexity results for partial models for disjunctive databases. They summarize results for partial stable models [274]; maximum stable models (M-stable) which are partial stable models under set inclusion [108, 302]; regular models [351] which are similar in spirit to M-stable models, but based on a weaker concept; and least undefined stable models [108, 302] which are the partial stable models with the minimal degree of undefinedness.

As noted in this and previous sections, there are a large number of different semantics both for extended deductive and extended disjunctive deductive databases. A user who wishes to use such a system is faced with the problem of selecting the appropriate semantics for his needs. Which semantics should be used, and under what circumstances? There have been no guidelines that have been developed. However, many complexity results have been obtained for these semantics. Schlipf [313] and Eiter and Gottlob [106] have written comprehensive survey articles that summarize the complexity results that are known for alternative semantics. Some of these results, taken from [104, 106], are listed in Table 2. A user may wish to determine the semantics to be used based upon the complexity expected to find answers to queries.

In addition to the results for extended disjunctive theories, there is work in investigating tractable cases of disjunctive theories. Ben-Eliyahu and Dechter [26] introduced the concept of a *head-cycle free (HCF)* clause. Let a clause consist of a disjunction of literals. A *dependency graph* G_P is associated with each program P as follows:

- each clause of the form, Formula (2) and each predicate in P is a node.
- there is a positive (negative) arc from a predicate node p to a rule node δ iff p appears positive (negative) in the body of δ, and an arc from δ to p (resp., and also an arc from p to δ) if p appears in the head of δ.

The *positive dependency graph* of P is a subgraph of G_P containing only positive arcs. A directed cycle in G_P is called *negative* if it contains at least one negative arc. A deductive database P is *head-cycle free* if for every two predicate names p and q, if p and q are on a positive directed cycle in the dependency graph G_P then there is no rule in P in which both p and q appear in the head. It is shown in [28] that answers to queries expressed in this language can be computed in polynomial time. Furthermore, the language is sufficiently powerful to express all polynomial time queries. It is further shown in [27] that there is an algorithm that performs, in polynomial time, minimal model finding and minimal model checking if the theory is *HCF*. An efficient algorithm for solving the (co-NP-hard decision) problem of checking if a model is stable in function-free disjunctive logic programs is developed in [195]. They show that the algorithm runs in polynomial time on the class of *head-cycle free programs* and in the case of general disjunctive logic programs, it limits the inefficient part of the computation only to the components of the program which are not head-cycle free.

In addition to work on tractable databases, consideration has been given to approximate reasoning. In such reasoning, one may give up soundness or completeness of answers. Efforts have been developed both for deductive and disjunctive deductive databases by Selman and Kautz [314, 171, 315], who developed lower and upper bounds for Horn (Datalog) databases and compilation methods, by Cadoli [51], who developed computational and semantical approximations, and by del Val [88, 89], who developed techniques for approximating and compiling databases. See also Cadoli [52] for additional references concerning compilation, approximation and tractability of knowledge bases.

A second way to determine the semantics to be used is through their properties. Dix in [93, 94] proposed a large number of criteria that are useful to consider in determining the appropriate semantics to be used. Indeed, Dix developed semantics both for normal [96, 97] and normal disjunctive [95, 44] deductive databases that satisfy some of the properties that he describes. While some of these properties are adaptations and extensions to those developed by Kraus, Lehmann and Magidor [185] to compare nonmonotonic theories, others were newly developed, such as, *relevance*, *partial evaluation* and *modularity*.

One property an arbitrary semantics, *SEM*, might have is that if a tautology is eliminated from a database, then the semantics of *SEM* should not be changed. Table 4 summarizes other useful properties of semantics of deductive databases and specifies for some alternative semantics the properties that they satisfy. This table is adapted from similar tables in [98, 99, 47, 46]. Although complexity results and properties that a semantics satisfy are extremely useful, no generally accepted criteria yet exist as to why one semantics should be used over another. A semantics may have all the properties that one may desire, and be computationally tractable and yet not provide answers that a user expected. Perhaps the best that can be expected is to provide users with complexity results and criteria by which they may decide as to which semantics meets the needs of their problems.

Understanding the semantics of disjunctive theories is related to nonmonotonic reasoning. The field of nonmonotonic reasoning has resulted in several alternative approaches to perform default reasoning [225, 289, 226, 249, 250]. The survey article by Minker [237] and papers by [106, 54] cite results where alternative theories of nonmonotonic reasoning can be mapped into extended disjunctive logic programs and databases. Hence, *DDDBs* may be used to compute answers to queries in such theories. See Cadoli and Lenzerini for complexity results concerning circumscription and closed world reasoning [50, 53]. See also Yuan and You [353] for a description of the relationships between autoepistemic circumscription and logic programming. They use two different belief constraints to

	Propositional		First Order over Herbrand models		First Order no function symb. over Herbrand models	
Semantics	Complexity	Ref.	Complexity	Ref.	Data Complexity	Ref.
Positive Consequences						
Minimal Model	$\mathcal{O}(\|P\|)$	[101, 165]	r.e.–complete	[323, 322, 5]	polynomial in $\|E\|$	[61]
Negation						
CWA	$\mathcal{O}(\|P\|)$	[101, 165]	co–r.e.–complete	[323, 322, 5]	co–r.e.–complete	[61]
Stratified Programs						
Perfect	$\mathcal{O}(\|P\|)$		complete arithmetic	[6]	polynomial in $\|E\|$	[62]
Locally Stratified Programs						
Perfect	$\mathcal{O}(\|P\|)$		Δ_1^1–complete over ω	[31]	N/A	
Normal Programs						
2–valued completion	co–NP–complete	[178]	Π_1^1–complete over ω	[178]	co–NP–complete	[178]
3–valued completion	$\mathcal{O}(\|P\|)$	folklore	Π_1^1–complete over ω	[119]	polynomial in $\|E\|$	[119]
Stable	co–NP–complete	[223]	Π_1^1–complete over ω	[222, 312]	co–NP–complete	[223]
Well–Founded	$\mathcal{O}(\|A\|\|P\|)$	folklore	Π_1^1–complete over ω	[136, 312]	polynomial in $\|E\|$	[138, 136]
Extended Programs						
Stable	co–NP–complete	[245]	Π_1^1–complete over ω	[245]	co–NP–complete	[245]
Well–Founded	$\mathcal{O}(\|A\|\|P\|)$	[245]	Π_1^1–complete over ω	[245]	polynomial in $\|E\|$	[245]

Table 2. (Adapted from [313]) Complexity of Horn Logic Programs

Notation: The complexity results in the above table refer to worst case analysis for skeptical reasoning, i.e. to determining if a given literal is *true* in every canonical model (with respect to a particular semantics) of the program. For logic programs with no function symbols, the data complexity over an $EDB\ E$ is presented. The notation used is the following: $|P|$ denotes the length of the program P; $|A|$ denotes the number of propositional letters in P; $|E|$ denotes the total number of symbols that occur in the $EDB\ E$.

Semantics	Propositional: Complexity	Ref.	First Order over Herbrand models: Complexity	Ref.	First Order no function symb. over Herbrand models: Data Complexity	Ref.
Positive Consequences						
Minimal Models	Π_2^P–complete	[104]			Π_2^P–complete	[107]
Negation						
GCWA	Π_2^P–complete	[104]	Π_2^0–complete	[71]	Π_2^P–complete	[107]
WGCWA	co–NP–complete	[60]				
Stratified Programs						
Perfect	Π_2^P–complete	[104]			Π_2^P–complete	[107]
Locally Stratified Programs						
Perfect	Π_2^P–complete	[104]			Π_2^P–complete	[107]
Normal Programs						
Stable	Π_2^P–complete	[104]			Π_2^P–complete	[107]
Partial Stable	Π_2^P–complete	[104]			Π_2^P–complete	[107]
Extended Programs						
Stable	Π_2^P–complete	[105]			Π_2^P–complete	[107]
Partial Stable	Π_2^P–complete	[245]			Π_2^P–complete	[107]

Table 3. (Taken from [104, 106]) Complexity of Disjunctive Logic Programs (with Integrity Constraints)

define two semantics, the *stable circumscriptive semantics* and *the well-founded circumscriptive semantics*, for autoepistemic theories. The work in [353] and on *static semantics* developed by Przymusinski [270] appear to be related.

Another area to which *DDDBs* have contributed is to the null value problem. If an attribute of a relation may have a null value, where this value is part of a known set, then one can represent this as a disjunction of relations, where, in each disjunction a different value is given to the argument. For papers on the null value problem both in relational and deductive databases, see [29, 30, 78, 147, 150, 161, 210, 213, 291, 343, 355].

There are several significant contributions of *DDDBs*:

1. *Greater expressive power is provided to the user to develop knowledge base systems.*
2. *Alternative concepts of negation have been developed.*

Property	Condition on a Semantics SEM to satisfy the Property							
	Clark's Compl.	GCWA	WGCWA	Perfect	Stable	WFS	D-WFS	Static
Elimination of Tautologies	If a rule $\mathcal{A} \leftarrow \mathcal{B}, \textbf{not } \mathcal{C}$ with $\mathcal{A} \cap \mathcal{B} \neq \emptyset$ is eliminated from a program P, then the resulting program is SEM–equivalent to P.							
	No	Yes	No	Yes	Yes	Yes	Yes	Yes
Generalized Principle of Partial Evaluation (GPPE)	If a rule $\mathcal{A} \leftarrow \mathcal{B}, \textbf{not } \mathcal{C}$, where $\mathcal{B}$ contains an atom B, is replaced in a program P by the n rules $\mathcal{A} \cup (\mathcal{A}^i - \{B\}) \leftarrow ((\mathcal{B} - \{B\}) \cup \mathcal{B}^i), \textbf{not } (\mathcal{C} \cup \mathcal{C}^i)$ where $\mathcal{A}^i \leftarrow \mathcal{B}^i, \textbf{not } \mathcal{C}^i$ $(i = 1, \ldots, n)$ are all rules for which $B \in \mathcal{A}^i$, then the resulting program is SEM–equivalent to P.							
	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Positive/ Negative Reduction	If (1) a rule $\mathcal{A} \leftarrow \mathcal{B}, \textbf{not } \mathcal{C}$ is replaced in a program P by $\mathcal{A} \leftarrow \mathcal{B}, \textbf{not } (\mathcal{C} - C)$ where C appears in no rule head, and (2) a rule $\mathcal{A} \leftarrow \mathcal{B}, \textbf{not } \mathcal{C}$ is deleted from P if there is a fact $\mathcal{A}' \leftarrow$ in P such that $\mathcal{A}' \subseteq \mathcal{C}$, then the resulting program is SEM–equivalent to P.							
	Yes	N/A	N/A	Yes	Yes	Yes	Yes	Yes
Elimination of Non–Minimal Rules	If a rule $\mathcal{A} \leftarrow \mathcal{B}, \textbf{not } \mathcal{C}$ is deleted from a program P if there is another rule $\mathcal{A}' \leftarrow \mathcal{B}', \textbf{not } \mathcal{C}'$ such that $\mathcal{A}' \subseteq \mathcal{A}$, $\mathcal{B}' \subseteq \mathcal{B}$, and $\mathcal{C}' \subseteq \mathcal{C}$, where at least one $\subseteq$ is proper, then the resulting program is SEM–equivalent to P.							
	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
Consistency	If SEM$(P) \neq \emptyset$ for all disjunctive deductive database P.							
	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Indepen- dence	If for every literal l, l is *true* in every $M \in$ SEM(P) iff l is *true* in every $M \in$ SEM$(P \cup P')$ provided that the language of P and P' are disjoint and l belongs to the language of P.							
	No	Yes	Yes	Yes	No	Yes	Yes	Yes

Table 4. (Adapted from [98]) Properties of the semantics of disjunctive deductive databases.

3. *Complexity results have been found for alternative semantics of disjunctive databases including alternative theories of negation.*
4. *Methods have been developed to permit prototype systems to be implemented.*
5. *Disjunctive databases can be used as the computational vehicle for a wide class of nonmonotonic reasoning theories.*

3 Implementation Status of Deductive Databases

The field of deductive databases has made significant intellectual contributions during the past 20 years. However, these have not been matched by implementations that are available in the commercial market. In the early 1970's, when Codd introduced the relational model [77], there were numerous debates in the database community as to the efficacy of such systems relative to network [84] and hierarchic systems [84]. These debates ended when an effective relational system was implemented and shown to be comparable to these systems. Now, some of those individuals who are prominent in relational databases claim that deductive databases are not effective and are not needed. Although I believe otherwise, these comments can be addressed better either when a full commercial implementation of a deductive database is available, or when many of the techniques introduced in deductive databases find their way into relational databases. I believe that both of these are beginning to happen.

In the following subsection I discuss the stages through which implementations of deductive databases have progressed. I describe some of the contributions made during each stage. Following this, I discuss the reasons why I believe that no current systems are commercially marketed and speculate on how this might change.

3.1 Deductive Database Systems

There have been three stages of implementations of deductive database systems. The first stage is pre-1970, the second is 1970-1980 and the third is 1980-present. Each of these stages has contributed towards understanding the problems inherent in developing a deductive database system.

First Stage: Pre 1970s. The work that stands out during this period was that of Levien, Maron and Kuhn [202, 203, 199, 200, 201, 186, 187, 188] at the Rand Corporation who developed a prototype system that demonstrated the feasibility of performing deduction in databases. The second was by Green and Raphael [151, 152] who recognized that the resolution method of Robinson [294] was a uniform procedure based on a single rule of inference that could be used for deductive databases. This was the first general approach to deductive databases. The work at the Rand Corporation was termed *Relational Data File (RDF)* and was started in 1963. A procedural language, *INFEREX*, was used to execute inference routines. Plausible and formal inferencing were both possible in *RDF*. Temporal reasoning was also permitted. The system was implemented on a file consisting of some 55,000 statements. The work by Green and Raphael resulted in a system termed *Question Answering System (QA-3.5)*. It was an outgrowth of Raphael's thesis on *Semantic Information Retrieval (SIR)* [282] that performed deduction. *QA-3.5* was a generalization of this work and included a natural language component. Another system, *Relational Store Structure (RSS)*, was started in 1966. The system performed deduction and was developed by Marrill [80, 81]. The system had twelve deductive rules built into the program. It was also possible to incorporate other deductive rules into the system. The *Association Store Processor (ASP)* developed by Savitt, Love and Troop [311] also performed deduction over binary relational data. The inference rules, specified as relational statements were handled by breadth-first, followed by a depth-first search. These efforts, as well as those cited in [247], were important pre-cursors to deductive databases. In Table 5, adapted from [247], I list some of the capabilities of systems developed in the first stage.

Second Stage: 1970-1980. The *SYNTEX* system built by Nicolas and Syre [258] used logic as the basis for deduction. It was when I heard Nicolas lecture at the IFIP Congress in 1974 that I decided to visit him in Toulouse on my sabbatical leave the following year. It was there that Nicolas, Gallaire and I conceived of holding our "Workshop on Logic and Data Bases." The work by Chang [64], Kellogg, Klahr and Travis [172], and by Minker [231] represent work during the second stage of development of deductive databases. These papers appear in the book by Gallaire and Minker [131]. Table 6 provides a brief summary of some of the features of these systems.

Kellogg, Klahr and Travis developed a system termed *Deductively Augmented Data Management (DADM)*. The system precomputed unifications among premises so as to not have to recompute them during deduction. Variables were typed. Inference plans and database access strategies were created from a premise file without requiring access to database values.

Minker described the *Maryland Refutation Proof Procedure System (MRPPS 3.0)* that performed top-down searches for large databases. It permitted arguments of predicates to contain function symbols and had a knowledge base index to access the data. The system performed deduction and used a typed unification algorithm and a semantic network. The semantic query optimization method described in [227] was incorporated in the system. An answer extraction, natural language mechanism that also provided voice output was part of the system.

Chang described the *DEDUCE 2* system. The system performed deduction over databases. Horn rules were used. These rules were not recursive and were compiled in terms of base relations. Integrity constraints were used to perform semantic query optimization on queries. He also discussed the problems that arise with respect to recursive rules and termination [65].

Third Stage: 1980-Present. A large number of prototype *DDBs* were developed during this period and most of them are described in [280]. I shall briefly discuss several major efforts during this period: work at ECRC led by Jean-Marie Nicolas, work at MCC led by Shalom Tsur and Carlo Zaniolo, work at the University of Wisconsin led by Raghu Ramakrishnan, and work at Stanford led by Jeff Ullman. Each of these efforts were attempts to develop operational and

Name	QA–3.5 [151, 152] Question–Answering System	ASP [311] Association Storing Processor	RDF [202, 200] Relational Data File	RSS [80, 81] Relational Structures System
Organization	Stanford Research Inst.	Hughes Aircraft Corp.	RAND Corp.	Computer Corp. of America
Designers	Raphael, Green & Coles	Savitt, Love & Troop	Levien & Maron	Marill
Computer	PDP 10	IBM 360/65	IBM 7044 & 360/65	IBM 360/75
Pgmg. Lang.	LISP 1.5	Assembly Language	Assembly Language	Assembly Language
Input Language Model	"Near–natural" language model based on simple transformations and context–free grammar	Stylized input form & a procedural language	Stylized input forms analyzed by FOREMAN Language	"Near–natural" language model based on matching sentence templates
Syntactic Analysis Technique	Earley algorithm for context–free grammar	N/A	N/A	Match of sentence against stored templates
Semantics Analysis Technique	Semantics stack built during syntax analysis phase	N/A	N/A	"Pattern $\Rightarrow$ action" operation invoked as a result of template match
Intermediate Language	First–order predicate calculus	Binary relations	Binary relations	n–ary relations ($n \leq 7$ as implemented)
Data Structures	LISP chained list structures	Relational statement elements randomized (coded) and replicated statements stored under each element	Files quadruplicated and ordered by statement number and three elements	Statement elements are hash–coded and "open" statements linked to corresponding "closed" statements
Inference Procedures	Formal theorem proving by Robinson Resolution Procedure	Inference rules specified as relational statements are handled by "breath–first–followed–by–depth"	"Plausible" inference rules are specified in a procedural language called INFEREX	Twelve general rules of deductive logic are used
Output Language	"Near–natural" language generated in a synthesis phase	Relational statements	Relational statements	"Near–natural" language generated from n–ary relational statements

Table 5. (Adapted from [247]) First Stage DDB Implementations

Name	MRPPS 3.0 [231] Maryland Refutation Proof Procedure System	DADM [172] Deductively Augmented Data Management	Deduce 2 [64]
Organization	University of Maryland	System Development Corp.	IBM San Jose
Designers	Minker, McSkimin, Wilson & Aronson	Kellogg, Klahr, & Travis	Chang
Computer	UNIVAC 1108		
Pgmg. Lang.	SIMPL		
Input Language Model	Multi–sorted Well–Formed formulae	Primitive Conditional Statements and Natural Language	DEDUCE [63] (based on Symbolic Logic)
Intermediate Language	Clausal Form	Primitive Conditional Statements	DEDUCE
Data Structures	Semantic Networks, Knowledge Base Index	Predicate Array, Premise Array, Semantic Network, Predicate Connection Graph [183, 319]	Connection Graph
Inference Procedures	SL–resolution [184] and LUSH–resolution [159]	Connection Graph	Connection Graph
Output Language	Natural Language Voice and English [263]	Primitive Condition Statements	DEDUCE
Features	Semantic Query Optimization, Multi–Sorted Variables, No Recursion, Non–Horn Clauses, Clauses need not be Function–Free, Relations not in First Normal Form	Semantic Query Optimization, Multi–Sorted Variables, No Recursion	Semantic Query Optimization

Table 6. Second Stage DDB Implementations

possibly commercial *DDBs*. They contributed significantly to both the theory and implementation of *DDBs*. Detailed descriptions of contributions made by these systems and others may be found in [280]. I briefly describe some highlights here. In Table 7, I list some of the capabilities of systems developed in the third stage. Those listed are taken from [280].

Implementation efforts at ECRC on *DDBs* started in 1984 and led to the study of algorithms and prototypes: deductive query evaluation methods (QSQ/SLD and others) [344, 345, 346, 192]; integrity checking (Soundcheck) [85]; the deductive database system EKS(-V1) by Vieille and his team [347]; hypothetical reasoning and integrity constraints checking [348]; and aggregation through recursion [193]. The EKS system used a top-down evaluation method and was released to ECRC shareholder companies in 1990. ECRC was also engaged in research on the development of a deductive object-oriented deductive database.

Implementation efforts at MCC on a *DDB* started in 1984 and emphasized bottom-up evaluation methods [336] and query evaluation using such methods as seminaive evaluation, magic sets and counting [12, 24, 303, 304], semantics for stratified negation and set-grouping [23], investigation of safety, the finiteness of answer sets, and join order optimization. A system, termed *LDL* was implemented in 1988, and released with refinements during the period 1989-1991. It was among the first operational *DDBs* that was widely available. It was distributed to universities and shareholder companies of MCC.

Implementation efforts at the University of Wisconsin on the *Coral DDBs* started in the late 1980s. Bottom-up and magic set methods were implemented. The system, written in *C* and *C++*, is extensible and provides aggregation modularly stratified databases. *Coral* supports a declarative language, and an interface to C++ which allows for a combination of declarative and imperative programming. The declarative query language supports general Horn clauses augmented with complex terms, set-grouping, aggregation, negation, and relations with tuples that contain universally quantified variables. CORAL supports a wide range of evaluation strategies, and automatically chooses an efficient evaluation strategy. Users are permitted to guide query optimization by selecting from among alternative control choices. *Coral* provides imperative constructs such as update, insert and delete rules. Disk-resident data is supported using the *EXODUS* storage manager, which also provides transaction management in a client-server environment. *Coral* is available over the net and has been disseminated widely in the academic community.

Implementation at Stanford University on a *DDBs* started in 1985 on a system called *NAIL! (Not Another Implementation of Logic!)*. The results of this work led to the first paper on recursion using the Magic Sets method [12], done in collaboration with the group at MCC. Other contributions were aggregation in logical rules, and theoretical contributions to negation: stratified negation by Van Gelder [341], well-founded negation [138] and modularly stratified negation [298]. A language, called *GLUE* [261, 251] was developed. *GLUE* is a language for logical rules that has the power of *SQL* statements, together with a conventional language that permits the construction of loops, procedures and modules.

Implementations of DDBs in the first, second and third stages of their development have demonstrated the feasibility and practicality of the technology. Tools and techniques have been developed to produce efficient DDBs.

3.2 Prospects for Commercial Implementation of DDBs [4]

One might address why, after 20 years that theoretical research was started in deductive datases, no commercial systems exist. To place this statement in perspective, it is well to recall that it required approximately twelve years for relational systems to become a reality. Additionally, as Jeff Ullman has stated on a number of occasions, deductive database theory is more more subtle than relational database theory. Nevertheless, there have been many prototypes, starting from the 1960s until the present, as described in section 3.1. However, none of the systems in [280] are likely to become commercial products with possibly two exceptions, *Aditi* [281] and *VALIDITY* [120, 209], developed at the Bull Corporation. According to a personal communication from Ramamohanarao, the leader of the *Aditi* effort, *Aditi* is perhaps one year from being a commercial product. Whether or not it will become a product remains to be seen. I believe that it will depend upon moving the system from a university setting to industry. Implementors and applications specialists will be required, rather than university researchers.

At the Bull Corporation, Jean-Marie Nicolas has headed up an effort with Laurent Vieille and their colleagues, to develop a deductive database system that integrates object-oriented features. The effort has been on-going for about 4 years and appears to be entering the market place. There are indications that the effort will be moved from the Bull Corporation to a new company that will be responsible for its maintenance, marketing and improvements. They have been developing a system called *VALIDITY*, which is an outgrowth of the work at *ECRC*, and may be less than one year away from being a commercial product. Indeed, in a personal communication with Jean-Marie Nicolas, there are indications that *VALIDITY* is already in use by some selected applications partners. The *VALIDITY* team has written a summary state of the art report on deductive and deductive object-oriented databases [333].

There are several reasons why we may not have had commercial systems earlier. Most of the prototype systems that have been developed have been at universities. Unless there is commercial backing for the venture, universities are not in a position either to develop or to support the maintenance that is required for large system developments. Those systems that have been developed in research

[4] Revisions made to this section subsequent to the Workshop on Logic in Databases, held in San Miniato, Italy, July 1-2, 1996, incorporate comments made in the panel session, *Deductive Databases: Challenges, opportunities and Future Directions*, by Arno Siebes, Shalom Tsur, Jeff Ullman, Laurent Vieille and Carlo Zaniolo, and in a personal communication by Jean-Marie Nicolas. I am responsible for any views expressed here.

Name	Developed	Recurs.	Negation	Aggregat.	Upd.	ICs	Optimizat.	Storage	Interfaces
Aditi [339]	U. Melbourne	General	Stratified	Stratified	No	No	Magic Sets, Seminaive	EDB, IDB	Prolog
COL [1]	INRIA	?	Stratified	Stratified	No	No	None	Main memory	ML
Concept-Base [169]	U. Aachen	General	Locally Stratified	No	Yes	No	Magic sets, Seminaive	EDB only	C, Prolog
CORAL [279]	U. Wisconsin	General	Modular Stratified	Modular Stratified	No	No	Magic sets, Seminaive, Context factoring, Projection pushing	EDB, IDB	C,C++, Extensible
EKS–V1 [347]	ECRC	General	Stratified	General	Yes	Yes	Query–subquery, left/right linear	EDB, IDB	Persistent Prolog
DECLARE [174]	MAD Intelligent Systems	General	Locally Stratified	General	No	No	Magic sets, Seminaive, Projection pushing	EDB only	C, Common Lisp
LDL, LDL++, Salad [69]	MCC	General	Stratified	Stratified	Yes	No	Magic sets, Seminaive, left/right linear, Projection pushing, "Bushy depth–first"	EDB only	C, C++, SQL
LOGRES [49]	Polytech. of Milan	Linear	Inflation. Semant.	Stratified	Yes	Yes	Seminaive, Algebraic Xforms	EDB, IDB	INFORMIX
NAIL/ GLUE [251]	Stanford U.	General	Well–Founded	Glue only	Glue only	No	Magic sets, Seminaive, Right–linear	EDB only	None
Starburst [253]	IBM Almaden	General	Stratified	Stratified	No	No	Magic sets, Seminaive (variant)	EDB, IDB	Extensible

Table 7. (Taken from [280]) Existing Implementations of DDBs

organizations controlled by consortia (ECRC and MCC) have not had the full backing of the consortia members. Second, the implementation effort to develop a commercial product was vastly underestimated by some of the organizations. There is a large investment that must be made to develop a deductive database that will both compete with relational technology and extend it. It should be emphasized that according to industry standards, an investment in the order of $30 to $50 million dollars is required to develop and place in the market a database system, whatever technology it relies upon. Furthermore, researchers tend to change their interests rather than to consolidate their work and invest into technology transfer towards industry to favor development of commercial software. Third, until recently, there has not been a convincing demonstration of a large commercial problem that requires a deductive database. This may have been a reason why the *LDL* system at *MCC*, and the *ECRC* developments were terminated. However, today, there appear to be a large number of applications that could take advantage of this technology as evidenced by the book edited by Ramakrishnan, "Applications of Logic Databases," [278]. In addition, Levy, Mendelzon, Sagiv and Srivastava [204] study the problem of computing answers to queries using materialized views and note that this work is related to applications such as Global Information Systems, Mobile Computing, view adaptation and maintaining physical data independence. In [205] they describe how deductive databases can be used to provide uniform access to a heterogeneous collection of more than 100 information sources on the World Wide Web. The work by Pulido [275] on *STARBASE*, an automatic method that reorders literals in the body of a rule so that the next literal to be processed is guaranteed to be the most instantiated, might be useful to consider in conjunction with this work. Fourth, none of the university researchers tried to obtain venture capital to permit them to build a product outside of the university. Efforts by some from MCC to obtain venture capital did not succeed. The *VALIDITY* effort to develop a system in an industrial setting is a prospect as described above.

Does lack of a commercial system at this date forebode the end of logic and databases? I believe that such a view would be naive. First, there still is the prospect, as noted above, that we will have a commercial deductive database system. Second, when one considers the fact that it took a good twelve years before relational database technology entered the market place from the late 1960s, when research into relational systems started (see [77]), there is no need to be alarmed. Third, as the following developments portend, relational databases are starting to incorporate techniques stemming from research in deductive databases.

Indeed, many of the techniques introduced within *DDBs* are finding their way into relational technology. The new SQL standards for relational databases are beginning to adopt many of the powerful features of deductive databases. Although there are always delays and discrepancies between specifications of SQL standards and their effective support by commercial systems, this is a good sign. In the SQL-2 standards (also known as SQL-92) [229], a general class of integrity constraints called *asserts* allow for arbitrary relationships between tables and views to be declared. These constraints exist as separate statements in the database, and are not attached to a particular table or view. This extension is powerful enough to express the types of integrity constraints generally associated with deductive databases. However, it should be noted that only the full SQL-2 standard includes assert specifications. The intermediate SQL-2 standard, which is the basis for most current commercial implementations, does not include asserts. For most current commercial implementations, asserts are not included. At the present time, the relational language for the next generation *SQL*, *SQL3* will provide an operation called *recursive union* that directly supports recursive processing of tables [228]. As noted in [228],

> The use of the recursive union operator allows both linear (single-parent, or tree) recursion and non-linear (multi-parent, or general directed graph) recursion. This solution will allow easy solutions to bill-of-material and similar applications.

Linear recursion is currently a part of the client server of *IBM*'s *DB2* system. It appears that they are using the *magic sets* method to perform linear recursion. There are indications that the *ORACLE* database system will also support some form of recursion.

A further development is that some form of *semantic query optimization* is beginning to be incorporated in relational databases. In some instances, such as *DB2*, cases are recognized when only one answer is to be found and the search is terminated. In other systems, it appears that equalities and other arithmetic constraints are being added to optimize the search. It will not be too long, in my view, before join elimination will be considered and introduced to relational technology. One can now estimate when it will be useful to eliminate a join [144]. The tools and techniques already exist and it is merely a matter of time before the users and the system implementers have them as part of their database systems.

Another technology that is available for commercial use is that of cooperative databases. The tools and techniques exist, as evidenced by the *COBASE* [73, 74] and *CARMIN* [127] work. With the introduction of recursion and semantic query optimization techniques into relational database technology, it will be necessary to provide users with cooperative responses so they understand why certain queries fail or succeed, as may be the case. It will also permit queries to be relaxed when the original query fails. This will permit reasonable, if not logically correct answers to be provided to users. Since user constraints may be handled in the same way that integrity constraints are handled, we will see relational systems that incorporate the needs of individual users into a query, as represented by their constraints.

Although the relational database community may ignore that the features that they are adding to their systems are outgrowths of developments in deductive databases, we should not be concerned. We should be pleased that our work in developing logic-based theories is now coming to fruition and has also helped in traditional relational database systems.

Two significant developments have taken place in the implementation of commercial DDBs. The first is the incorporation of techniques developed in DDDBs into relational technology. Recursive views that use the magic set technique for implementation are being permitted and methods developed for semantic query optimization are being applied. The second is the development of a deductive object–oriented DBs, VALIDITY, that appears to be in commercial use; and the development of the Aditi DDBs that is scheduled for commercial use in approximately one year. It remains to be seen how long one can make patches to relational technology to simulate the capabilities of deductive databases systems.

4 Emerging Areas and Trends

In the previous sections we have shown that there are many theories for negation both in extended deductive databases and in disjunctive deductive databases. There have been many different semantics proposed based on these theories. We understand a great deal about negation, except for how and when one should use a given theory. This will be a major area of confusion when users become cognizant of what is available for them to use. Much more work has to be done if the areas of implementation and application are to catch up to the intellectual developments that have taken place over the past 20 years. We have saturated the market for alternative theories of semantics and should move to more fertile topics. Unless we do so, funding for logic and databases will start to wane, as I believe that it has in the United States. That does not mean that we should abandon all research in theories of negation and alternative semantics, but we have to take stock of what we have accomplished and make it more accessible for users.

The role of logic will be of increasing importance due to the need to handle highly complex data (partly due to the advances in networking technology and the reduction of cost in both processing time and primary, secondary and tertiary memory). This data will require more complex models of data access and representation. Advances will require formal models of logic rather than ad-hoc solutions. Below, I shall briefly mention a number of fertile areas for further exploration. This is not intended to be an exhaustive listing of important areas to investigate.

Temporal databases, which deal with time, needs additional research. It is important for historical databases, real-time databases and other aspects of databases. Work in this area has been done by Snodgrass [326, 324, 325], Chomicki [70], and Wolfson and Sistla [320]. A paper on applying transition rules to such databases for integrity constraint checking appears in this collection [224].

Transactions and updates have not had sufficient attention. There exist semantic models of updates [109, 149, 111] that assure that views and data are updated correctly. Considerably more work is required in the area of transactions which require sequences of updates. Long duration transaction models and workflow management are areas that require work. In emerging applications of database systems, transactions are viewed as sequences of nested, and most probably interactive sub-transactions that may sparsely occur over long periods of time. In this scenario new complex transaction systems must be designed. Logic-based transaction systems will be essential to assure that an appropriate and correct transaction is achieved. Work in this area is being done by Bonner and Kifer [34, 35], Lin and Reiter [208, 207], and others [179, 4, 72, 110, 135, 220].

Active databases consist of data that protects its own integrity and describes the database semantics. Active rules are represented by the formalism, *Event-Condition-Action (ECA)* [334] and denote that whenever an event E occurs, if condition C holds, then trigger action A. It has a behavior of its own beyond passively accepting statements from users or applications. Upon recognition of certain events, it invokes commands and monitors and manages itself. It may invoke external actions that interact with systems outside the database. Indeed, it may activate a potentially infinite set of triggers. Although declarative constraints are provided for such systems, the *ECA* formalism is essentially procedural in nature. Zaniolo has noted the need for declarative semantics of triggers [356, 357]. He has developed a unified semantics for active and deductive databases and has shown how active database rules relate to transaction-conscious stable model semantics. Work reported upon in this collection by [18] proposes a first step towards characterizing active databases. Additional work is required to develop a clear semantics, sound implementations and a better understanding of complexity issues in active databases. Work in the situation calculus and Datalog extensions apply here.

Data mining and inductive inference deal with finding generalizations that may be extracted from a database or a logic program. Such generalizations may be integrity constraints that must be *true* with respect to the database, or generalizations that may be *true* of the current state, but may change if there are updates. It will be up to the database administrator to determine which of the generalizations are integrity constraints, and which are applicable only to the current state and must be checked at update time. Semantic query optimization can handle either of these two cases, and inform the user as to which constraint may be applicable to a query. As demonstrated in the work of Muggleton and De Raedt [252], logic programming may be used to find inductive inferences. The book edited by Piatetsky-Shapiro and Frawley [262] covers work on knowledge data mining. Laurent and Vrain [191], in this collection, discuss how to couple deductive databases and inductive logic programming to learn query rules for optimizing databases with update rules.

Integrating production systems with deductive databases is future work that is needed. A formal approach to performing such an integration and to developing the semantics of rule-based systems will result. Since rule-based systems have played a role in active systems, work in this area will benefit active database efforts. Related work in this area is described in [153, 221, 284, 285, 283]

Logical foundations of object-oriented deductive databases is needed. In many ways object-oriented databases relate to hierarchic and network systems. It is essential that such systems have a formal theory and a semantics. This is difficult as there does not appear to be a formal definition of an object-oriented database.

Efforts have been undertaken by Kifer and his co-workers [175] to develop a formal foundation for object-oriented databases. Extensive work is required to develop semantic query optimization techniques, a formal theory of updating and all of the tools and techniques developed for deductive databases.

Description logics are concerned with restricting knowledge representation so that deduction over a knowledge base is tractable, but is still powerful enough to represent the knowledge in a natural manner. KRYPTON was an early such hybrid knowledge representation system which combined a functional knowledge representation with a specialized connection graph theorem prover [40, 41]. KL-ONE is perhaps the best known knowledge representation system based on a description logic [42]. CLASSIC is a modern successor of KL-ONE which allows for the definition of complex objects and moves towards database support [36, 39]. In deductive databases, representational power is also limited to allow for more tractable deduction (via specific proof procedures). Some of these limits are a restriction to Horn clauses, no logical terms, no existential quantification, and so forth. Research in deductive databases has sought to extend the representational power some, yet preserve tractability to the greatest extent possible: for example, disjunctive deductive databases allow for general clauses and the addition of null values allow for a type of existential quantification. The topics of deductive databases and description logics have remained somewhat distinct. However, their goals are quite similar. As the representation power improve in each, and better more efficient proof procedures are developed, we should see a convergence of these areas.

Heterogeneous databases are of considerable interest. The objective here is to integrate into one system multiple databases that do not necessarily share the same data models. There is the need for a common logic-based language for mediation and a formal semantics of such databases. Work on the *Hermes* system by Subrahmanian [328], the *Tsimmis* system by Garcia-Molina and his colleagues [66], and the work by Ramakrishnan on [230] are illustrative of work on this subject. It is not unlikely that heterogeneous databases will also be required to handle textual data. Kero and Tsur [173], in this collection, describe the $\mathcal{IQ}$ system that uses a deductive database $\mathcal{LDL}++$ to reason about textual information. Language extensions for the semantic integration of deductive databases is proposed by [9] in this collection. The language allows *mediators* to be constructed, using a set of operators for composing programs and message passing features.

Multi-media databases [329] is an emerging area for which new data models are needed. These databases have special problems such as manipulating geographic databases; picture retrieval where a concept orthogonal to time may appear in the database: space; and video databases, where space and time are combined. Temporal and spatial reasoning will be needed. Logic will play a major role in the development of query languages for these new data models, will permit deductive reasoning, and a formal semantics will provide a firm theoretical basis for them.

Combining databases is a topic related to both heterogeneous and multimedia systems. Here one is trying to combine databases that share the same integrity constraints and schema. Such databases arise in distributed systems work and in combining knowledge bases. In addition to handling problems that arise because the combined databases may be inconsistent, one has to handle priorities that may exist among individual facts. A formal treatment of this subject started with Minker and his co-workers [15, 16, 17, 266, 265, 264].

Integrity constraints, semantic query optimization and constraint logic programming are related topics. Semantic query optimization uses constraints in the form of integrity constraints to prune the search space. These integrity constraints introduce equalities, inequalities, and relations into a query to help optimize the search [58]. Constraint logic programming introduces domain constraints. These constraints may be equalities, inequalities, and may even be relations [167, 166]. Constraint databases and constraint-intensive queries are required in many advanced applications. In constraint databases, constraints can capture spatial and temporal behavior which is not possible in existing databases. The relationships between these areas need to be explored further and applied to deductive databases. Spatial databases defined in terms of polynomial inequalities are investigated by [189], who consider termination properties of *Datalog* programs.

Abductive reasoning is the process of finding explanations for observations in a given theory. Selman and Levesque [316] developed an approach using logic, which is described in Kakas, Kowalski and Toni [170]. Given a set of sentences T (a theory), and a sentence G (an observation), the abductive task can be characterized as the problem of finding a set of sentences Δ (abductive explanation for G) such that

(1) $T \cup \Delta \models G$,
(2) $T \cup \Delta$ is consistent.
(3) Δ is minimal with respect to set inclusion [316, 170, 54].

A comprehensive survey and critical overview of the extension of logic programming to perform abductive reasoning (abductive logic programming) is given in [170]. They outline the framework of abduction and its applications to default reasoning; and introduce an augmentation theoretic approach to the use of abduction as an interpretation for negation-as-failure. They show that abduction has strong links to extended disjunctive logic programming. Abduction is shown to generalize negation-as-failure to include not only negative but also positive hypotheses, and to include general integrity constraints. They show that abductive logic programming is related to the justification based truth maintenance system of Doyle [102]. Inoue and Sakama [164] develop a fixpoint semantics for abductive logic programs in which the belief models are characterized as the fixpoint of a disjunctive program obtained by a suitable program transformation. For a summary of complexity results on abductive reasoning and nonmonotonic reasoning see the excellent survey by Cadoli and Schaerf [54].

High-level robotics is an area of active research. In approaches taken to this problem, logic has played a significant role. Knowledge bases are used in high level robotics to solve problems in cognition required to plan actions for robots and to deal with multiple agents in complicated environments. Work in deductive and disjunctive databases relates to this problem. In some instances a robot may have several options that can be represented by disjunctions. Additional information derived either by alternative information derived from sensors may serve to disambiguate the possibilities. There are several groups engaged in this research: the University of Toronto [196, 197], the University of Texas at El Paso [14], the University of Texas at Austin [142] and the University of Linkoping [309, 310]. Kowalski and Sadri [182], discuss a unified agent architecture that combines rationality with reactivity in this collection, and relates to this topic.

Applications of deductive database techniques will be important. Greater emphasis is required to apply deductive database technology to realistic problems. Deductive databases have been shown to be important both for relational and deductive systems on such topics as semantic query optimization [56], cooperative answering [124], Global Information Systems and Mobile Computing [204, 83].

Implement Deductive (DDB) and Object-Oriented Deductive Databases (DOOD). It is important to develop a commercial *DDB* or *DOOD* system. The deductive model of databases is beginning to take hold as evidenced by the textbook, "Foundations of Databases," by Abiteboul, Hull and Vianu [2]. The prospect that the *Aditi* [281] system may be available in a year is indication that progress is being made in developing a commercial *DDBS*. The merger of object-oriented and deductive formalisms is taking place as illustrated by the proceedings of the DOOD conference series [176, 90, 55, 209]. This series was established in 1989 at the initiative of Shojiro Nishio, together with Serge Abiteboul, Jack Minker and Jean-Marie Nicolas. That the *VALIDITY* [121, 120], system is probably in use by customers at the present time is an indication that the object-oriented and deductive formalisms are soon to be available commercially. Additional features will be required for commercial systems such as cooperative answering [123] and arithmetic and aggregate operators, as described in this collection by Dobbie and Topor [100].

It is clear from the above, that logic and databases can contribute significantly to a large number of exciting new topics. Hence, the field of logic and databases will continue to be a productive area of research and implementation.

5 Summary

In section 1, I discussed the significant pre-history of the field of *logic and databases*. I noted the work by Levien, Maron and Kuhns [202, 203, 199, 200, 201, 186, 187, 188], by Green and Raphael [151, 152], and referred to my paper [234], where the pre-history of the field is discussed. In sections 2 and 3, I discussed the major accomplishments that have taken place in this field in the 20 years since 1976. Among these accomplishments are the extension of relational databases, the development of the semantics and complexity of these alternative databases, permit knowledge base systems to be represented and developed, and nonmonotonic reasoning systems to be implemented. In section 4, I discussed many exciting new areas that will be important in the near and long-term future. *It is clear that the field of logic and databases has had a significant pre-history before 1970, and a well-defined area of research, complete with past achievements and continued future areas of work.*

In these past twenty years we have seen logic and databases progress from a fledgling field, to a fully developed, mature field. The new areas that I have cited that need further investigation show that we have not nearly exhausted the work in this field. I envision that many more workshops will be held on this topic. *Logic and databases has contributed to the field of databases being a scientific endeavor rather than an ad-hoc collection of techniques. We understand what constitutes a database, a query, an answer to a query, and where knowledge has its place.* I look forward to the next twenty years of this field. I hope that I will have an opportunity, then, to look back and see a field that has accomplished much, and is still vibrant. To remain vibrant, we will have to take on some of the new challenges, rather than to be mired in the semantics of more exotic databases. We will have to address implementation issues and we will have to be able to provide guidance to practitioners who will need to use the significant developments in logic and databases.

References

1. S. Abiteboul and S. Grumback. A rule-based language with functions and sets. *ACM Transactions on Database Systems*, 16(1):1–30, 1991.
2. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley Publishing Comp, 1995.
3. J. Alferes and L. Pereira. On logic program semantics with two kinds of negation. In K. Apt, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 574–588, Washington, D.C. USA, Nov 1992. The MIT Press.
4. P. Ammann, S. Jajodia, and I. Ray. Using formal methods to reason about semantics-based decompositions of transactions. In *Proc. of 21st VLDB Conference*, pages 218–227, 1995.
5. H. Andreka and I. Nemeti. The generalized completeness of horn predicate logic as a programming language. *Acta Cybernetica*, 4(1):3–10, 1978.
6. K. Apt and H. Blair. Arithmetic classification of perfect models of stratified programs. *Fundamenta Informaticae*, XIII:1–18, 1990. With addendum in vol. XIV: 339-343. 1991.
7. K. Apt, H. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann Pub., Los Altos, California, 1988.
8. K. Apt and R. Bol. Logic programming and negation: a survey. *Journal of Logic Programming*, 19/20:9–71, May/June 1994.
9. P. Asirelli, C. Renso, and F. Turini. Language extensions for semantic integration of deductive databases. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 425–444, July 1-2 1996. Also in this collection.
10. R. Bagai and R. Sunderraman. Bottom-up computation of the fitting model for general deductive databases. *Intelligent Information Systems*, 6(1):59–75, Jan. 1996.

11. F. Bancilhon. Naive evaluation of recursively defined relations. In M. Brodie and J. Mylopoulos, editors, *On Knowledge Base Management Systems–Integrating Database and AI Systems*, pages 165–178. Springer-Verlag, 1986.
12. F. Bancilhon, D. Maier, Y. Sagiv, and J. Ullman. Magic sets and other strange ways to implement logic programs. In *Proc. ACM Symp. on Principles of Database Systems*, March 1986.
13. C. Baral and M. Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 19/20:73–148, July 1994.
14. C. Baral, M. Gelfond, and A. Provetti. Representing Actions: Laws, Observations and Hypothesis. *Journal of Logic Programming (to appear)*, 1996.
15. C. Baral, S. Kraus, and J. Minker. Combining multiple knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 3(2):208–220, July 1991.
16. C. Baral, S. Kraus, J. Minker, and V. Subrahmanian. Combining knowledge bases consisting of first order theories. Technical Report CS-2531, Dept of Computer Science, University of Maryland, College Park Md 20742, 1990.
17. C. Baral, S. Kraus, J. Minker, and V. Subrahmanian. Combining default logic databases. *Journal of International Information Systems*, 1994. To appear.
18. C. Baral and J. Lobo. Formal characterization of active databases. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 195–215, July 1-2 1996. Also in this collection.
19. C. Baral, J. Lobo, and J. Minker. Generalized well-founded semantics for logic programs. In M. E. Stickel, editor, *Proc. of Tenth Internatinal Conference on Automated Deduction*, pages 102–116, Kaiserslautern, Germany, July 1989. Springer-Verlag.
20. C. Baral, J. Lobo, and J. Minker. Generalized disjunctive well-founded semantics for logic programs: Procedural semantics. In *Proceedings of the Fifth International Symposium on Methodologies for Intelligent Systems*, pages 456–464, Knoxville TN, USA, 1990.
21. C. Baral, J. Lobo, and J. Minker. Generalized disjunctive well-founded semantics for logic programs: Declarative semantics. In *Proceedings of the Fifth International Symposium on Methodologies for Intelligent Systems*, pages 465–473, Knoxville TN, USA, 1990.
22. C. Baral and V. Subrahmanian. Stable and extension class theory for logic programs and default theory. *Journal of Automated Reasoning*, pages 345–366, 1992.
23. C. Beeri, S. Naqvi, O. Shmueli, and S. Tsur. Set constructors in a logic database language. *Journal of Logic Programming*, 10 (3&4), 1991.
24. C. Beeri and R. Ramakrishnan. On the power of magic. *Journal of Logic Programming*, 10(3/4):255–300, 1991.
25. C. Bell, A. Nerode, R. Ng, and V. Subrahmanian. Implementing stable model semantics by linear programming. In *Proceedings of the 1993 International Workshop on Logic Programming and Non-monotonic Reasoning*, June 1993.
26. R. Ben-Eliyahu and R. Dichter. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial intelligence*, 12:53–87, 1994.
27. R. Ben-Eliyahu and L. Palopoli. Reasoning with minimal models: efficient algorithms and applications. In *Proc. of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, pages 39–50, 1994. Full paper submitted for journal publication.
28. R. Ben-Eliyahu, L. Palopoli, and V. Zemlyanker. The expressive power of tractable disjunction. In W. Wahlster, editor, *ECAI96. 12th European Conference on Artificial Intelligence*, pages 345–349, 1996.

29. J. Biskup. A formal approach to null values in database relations. In H. Gallaire, J. Minker and J.M. Nicolas, editors, *Advances in Data Base Theory*, volume 1, pages 299–341. Plenum Press, New York, 1981.
30. J. Biskup. A foundation of Codd's relational maybe-operations. University Park, 1981. Pennsylvania State Univ.
31. H. Blair, W. Marek, and J. Schlipf. The expressiveness of locally stratified programs. Technical report, Mathematical Sciences Institute, Cornell University, 1992. Available as technical report 92-8.
32. H. Blair and V. Subrahmanian. Paraconsistent logic.
33. B. Blaustein. *Enforcing database assertions: techniques and applications.* PhD thesis, Harvard University, Computer Science Dept., Cambridge, Mass., Aug. 1981. Ph.D. dissertation.
34. A. Bonner and M. Kifer. Transaction logic programming. In D. S. Warren, editor, *Logic Programming: Proc. of the 10th International Conf.*, pages 257–279, 1993.
35. A. Bonner and M. Kifer. Concurrency and communication in transaction logic. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 153–172, July 1-2 1996. Also in this collection.
36. A. Borgida, R. J. Brachman, D. L. McGuiness, and L. A. Resnick. CLASSIC: A structural data model for objects. *ACM SIGMOD Record*, 18(2):58, June 1989. Also published in/as: 19 ACM SIGMOD Conf. on the Management of Data, (Portland OR), May–Jun 1989.
37. A. Borgida and D. Etherington. Hierarchical knowledge bases and efficient disjunction. In *Proceedings of the First International Conference on Principle of Knowledge Representation and Reasoning (KR-89)*, pages 33–43, Toronto, Ontario, CANADA, 1989.
38. G. Bossu and P. Siegel. Saturation, nonmonotonic reasoning and the closed-world assumption. *Artificial Intelligence*, 25(1):13–63, Jan. 1985.
39. R. Brachman, A. Borgida, D. McGuinness, P. Patel-Schneider, and L. Resnick. The Classic knowledge representation system of KL-ONE: The next generation. In *International Conference on Fifth Generation Computer Systems*, pages 1036–1043, ICOT, Japan, 1992.
40. R. J. Brachman, R. E. Fikes, and H. J. Levesque. KRYPTON: A functional approach to knowledge representation. *IEEE Computer*, 16(10):67–73, Oct. 1983.
41. R. J. Brachman, V. P. Gilbert, and H. J. Levesque. An essential hybrid reasoning system: Knowledge and symbol level accounts of KRYPTON. In *International Joint Conference on Artificial Intelligence*, pages 532–539, Aug. 1985.
42. R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, pages 171–216, Aug. 1985.
43. S. Brass. Sldmagic — an improved magic set technique. In B. Novikov and J. Schmidt, editors, *Advances in Database and Information Systems - ADBIS'96*, Sept 1996.
44. S. Brass and J. Dix. Disjunctive Semantics based upon Partial and Bottom-Up Evaluation. In L. Sterling, editor, *Proceedings of the 12th Int. Conf. on Logic Programming, Tokyo*, pages 199–213. MIT Press, June 1995.
45. S. Brass and J. Dix. A general approach to bottom–up computation of disjunctive semantics. In J. Dix, L. Pereira, and T. Przymusinski, editors, *Nonmonotonic Extensions of Logic Programming*, pages 127–155. Lecture Notes in Computer Science 927. Springer-Verlag, 1995.
46. S. Brass and J. Dix. Characterizations of the Disjunctive Stable Semantics by Partial Evaluation. *Journal of Logic Programming*, forthcoming, 1996.

47. S. Brass and J. Dix. Characterizing D-WFS: Confluence and Iterated GCWA. In L. Pereira and E. Orlowska, editors, *JELIA '96*, LNCS 1111, Berlin, 1996. Springer.
48. F. Bry. Query evaluation in recursive databases: bottom-up and top-down reconciled. *Data and Knowledge Engineering*, 5:289–312, 1990.
49. F. Cacace, S. Ceri, S. Crespi-Reghizzi, and R. Zicari. Integrating object-oriented data modeling with a rule-based programming paradigm. In *Proc. of ACM SIGMOD Conference on Management of Data*, May 1990.
50. M. Cadoli. The complexity for model checking for circumscrioptive formulae. *Information Processing Letters*, 44:113–118, Oct 1992.
51. M. Cadoli. Semantical and computational aspects of Horn approximations. In *Proc. of IJCAI-93*, pages 39–44, 1993.
52. M. Cadoli. Panel on "Knowledge compilation and approximation": terminology, questions, references. In *Proc. of the Fourth Int. Symp. on Artificial Intelligence and Mathematics, AI/Math-96*, pages 183–186, January 3-5 1996.
53. M. Cadoli and M. Lenzerini. The complexity of closed world reasoning and circumscription. *J. Comp. and Syst. Sci.*, 43:165–211, 1994.
54. M. Cadoli and M. Schaerf. A survey of complexity results for non-monotonic logics. *Journal of Logic Programming*, 13:127–160, 1993.
55. S. Ceri, K. Tanaka, and S. Tsur, editors. *Proc. of the 3rd Int. Conf. on Deductive and Object-Oriented Databases - DOOD'93*, December 1993. In LNCS 760, Springer-Verlag, Heidelberg, Germany.
56. U. Chakravarthy, J. Grant, and J. Minker. Logic-based approach to semantic query optimization. *ACM Transactions on Database Systems*, 15(2):162–207, June 1990.
57. U. S. Chakravarthy, J. Grant, and J. Minker. Foundations of semantic query optimization for deductive databases. In J. Minker, editor, *Proc. Workshop on Foundations of Deductive Databases and Logic Programming*, pages 67–101, Washington, D.C., Aug. 1986.
58. U. S. Chakravarthy, J. Grant, and J. Minker. Logic based approach to semantic query optimization. *ACM Transactions on Database Systems*, 15(2):162–207, June 1990.
59. E. Chan. A possible world semantics for non-Horn databases, 1989. University of Waterloo, Canada.
60. E. Chan. A possible world semantics for disjunctive databases. *IEEE Trans. Data and Knowledge Eng.*, 5(2):282–292, 1993.
61. A. Chandra and D. Harel. Structure and complexity of relational queries. *Journal of Computer System Sciences*, 25:99–128, 1982.
62. A. Chandra and D. Harel. Horn clause queries and generalizations. *Journal of Logic Programming*, 2(1):1–15, Apr. 1985.
63. C. Chang. DEDUCE—a deductive query language for relational databases. In C. Chen, editor, *Pattern Recognition and Artificial Intelligence*, pages 108–134. Academic Press, New York, 1976.
64. C. Chang. Deduce 2: Further investigations of deduction in relational databases. In H. G. J. Minker, editor, *Logic and Databases*, pages 201–236. Plenum, New York, 1978.
65. C. Chang. On evaluation of queries containing derived relations. In H. G. J. M. J.-M. Nicolas, editor, *Advances in Database Theory, Volume 1*, pages 235–260. Plenum Press, New York, 1981.

66. S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proceedings of IPSJ Conference, Tokyo, Japan*, October 1994. Available via anonymous FTP from host db.stanford.edu, file /pub/chawathe/1994/tsimmis-overview.ps.
67. W. Chen and D. Warren. A goal-oriented approach to computing well founded semantics. In K. Apt, editor, *Proceedings of the joint international conference and symposium on logic programming*, Washington, D.C., Nov. 1992.
68. S. Chi and L. Henschen. Recursive query answering with non-Horn clauses. In E. Lusk and R. Overbeek, editors, *Proc. 9^{th} International Conference on Automated Deduction*, pages 294–312, Argonne, IL, May 1988.
69. D. Chimenti, R. Gamboa, R. Krishnamurthy, S. Naqvi, S. Tsur, and C. Zaniolo. The LDL system prototype. *IEEE Transactions on Knowledge and Data Engineering*, 2(1):76–90, 1990.
70. J. Chomicki. Efficient checking of temporal integrity constraints using bounded history encoding. *ACM Transactions on Database Systems*, 20(2):111–148, May 1995.
71. J. Chomicki and V. Subrahmanian. Generalized closed world assumption is π_2^0-complete. *Information Processing Letters*, 34:289–291, May 1990.
72. P. Chrysanthis and K. Ramamritham. Synthesis of extended transaction models using ACTA. *ACM, TODS*, 19(3):450–491, 1994.
73. W. W. Chu, Q. Chen, and A. Hwang. Query answering via cooperative data inference. *Journal of Intelligent Information Systems*, 3:57–87, 1994.
74. W. W. Chu, Q. Chen, and M. A. Merzbacher. CoBase: A cooperative database system. Studies in Logic and Computation 3, chapter 2, pages 41–73. Clarendon Press, Oxford, 1994.
75. K. L. Clark. Negation as Failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.
76. CODASYL. *CODASYL Data Base Task Group April 71 Report.* ACM, New York, 1971.
77. E. Codd. A relational model of data for large shared data banks. *Comm. ACM*, 13(6):377–387, June 1970.
78. E. F. Codd. Extending the database relational model to capture more meaning. *ACM Trans. Database Syst.*, 4(4):397–434, 1979.
79. A. Colmerauer, H. Kanoui, R. Pasero, and P. Roussel. Un systeme de communication homme-machine en francais. Technical report, Groupe de Intelligence Artificielle Universitae de Aix-Marseille II, Marseille, 1973.
80. Computer Corporation of America. Relational structures research. Technical report, Computer Corporation of America, August 5 1967. For Contract Period April 6, 1966-July 5, 1967.
81. Computer Corporation of America. Relational structures applications research. Technical report, Computer Corporation of America, July 11 1969. For Contract Period May 5, 1967-March 31, 1969.
82. M. Dalal. Some tractable classes of disjunctive logic programs. Technical report, Rutgers University, 1992.
83. S. Dar, H. Jagadish, A. Levy, and D. Srivastava. Answering queries with aggregation using views. In *Proceedings of the 22nd International Conference on Very Large Databases, VLDB-96*, Sept 1996. To appear.
84. C. Date. *An Introduction to Database Systems, Sixth Edition.* Addison-Wesley Publishing Comp, 1995.

85. H. Decker. Integrity enforcement on deductive databases. In *Proc. 1st Int. Conf. on Expert Database Systems*, April 1986.
86. H. Decker. On alternative models, fixpoints and consistency of disjunctive databases, May 1991.
87. H. Decker. On the declarative, operational and procedural semantics of disjunctive computational theories. In *Proceedings of the* 2^{th} *International Workshop on the Deductive Approach to Information Systems and Databases*, Aiguablava, Spain, Sept. 1991. Invited paper.
88. A. del Val. Tractable databases: how to make propositional unit resolution complete through compilation. In *Proc. of KR-94*, pages 551–561, 1994.
89. A. del Val. An analysis of approximate knowledge compilation. In *Proc. of IJCAI-95*, 1995.
90. C. Delobel, M. Kifer, and Y. Masunaga, editors. *Proc. 2nd Int. Conf. on Deductive and Object-Oriented Databases - DOOD'91*. Springer-Verlag, Heidelberg, Germany, December 1991.
91. R. Demolombe and A. J. Jones. Integrity constraints revisited. *Journal of the IGPL (Interest Group in Pure and Applied Logics): An Electronic Journal on Pure and Applied Logic*, 4(3):369–383, 1996.
92. S. Dietrich and D. S. Warren. Extension tables: memo relations in logic programming. In *Proc. Symp. on Logic Programming*, pages 264–273, San Francisco, Ca, 1987.
93. J. Dix. Classifying Semantics of Logic Programs. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *Logic Programming and Non-Monotonic Reasoning, Proceedings of the first International Workshop*, pages 166–180, Cambridge, Mass., July 1991. Washington D.C, MIT Press.
94. J. Dix. A Framework for Representing and Characterizing Semantics of Logic Programs. In B. Nebel, C. Rich, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR '92)*, pages 591–602. San Mateo, CA, Morgan Kaufmann, 1992.
95. J. Dix. Classifying Semantics of Disjunctive Logic Programs. In K. R. Apt, editor, *LOGIC PROGRAMMING: Proceedings of the 1992 Joint International Conference and Symposium*, pages 798–812, Cambridge, Mass., November 1992. MIT Press.
96. J. Dix. A Classification-Theory of Semantics of Normal Logic Programs: I. Strong Properties. *Fundamenta Informaticae*, XXII(3):227–255, 1995.
97. J. Dix. A Classification-Theory of Semantics of Normal Logic Programs: II. Weak Properties. *Fundamenta Informaticae*, XXII(3):257–288, 1995.
98. J. Dix. *Disjunctive deductive databases: theoretical foundations and operational semantics*. PhD thesis, Institut für informationssysteme abteilung wissensbasierte systeme, Technische Universität Wien, Sept 1995. Habilitation Thesis.
99. J. Dix. Semantics of Logic Programs: Their Intuitions and Formal Properties. An Overview. In A. Fuhrmann and H. Rott, editors, *Logic, Action and Information – Essays on Logic in Philosophy and Artificial Intelligence*, pages 241–327. DeGruyter, 1995.
100. G. Dobbie and R. Topor. Arithmetic and aggregate operators in deductive object-oriented databases. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 399–407, July 1-2 1996. Also in this collection.
101. W. Dowling and J. H. Gallier. Linear time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 1:267–284, 1984.
102. J. Doyle. Truth Maintenance System. *Artificial Intelligence*, 13, 1980.

103. P. Dung. Negations as hypothesis: an abductive foundation for logic programming. In *Proc. of the 8th International Conference on Logic Programming*, 1991.
104. T. Eiter and G. Gottlob. Complexity aspects of various semantics for disjunctive databases. In *Proceedings of the Twelfth ACM SIGART–SIGMOD–SIGART Symposium on Principles of Database Systems (PODS-93)*, pages 158–167. ACM Press, May 1993.
105. T. Eiter and G. Gottlob. Complexity results for disjunctive logic programming and application to nonmonotonic logics. In D. Miller, editor, *Proceedings of the International Logic Programming Symposium ILPS'93*, pages 266–278, Vancouver, Canada, October 1993. MIT Press.
106. T. Eiter and G. Gottlob. On the computation cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence*, 15(3-4):289–323, Dec. 1995.
107. T. Eiter, G. Gottlob, and H. Mannila. Adding Disjunction to Datalog. In *Proceedings of the Thirteenth ACM SIGACT SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '94)*, pages 267–278, May 1994.
108. T. Eiter, N. Leone, and D. Saccà. The expressive power of partial models for disjunctive databases. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 261–280, July 1-2 1996. Also in this collection.
109. R. Fagin, J. Ullman, and M. Vardi. On the semantics of updates in databases. In *Proc. Senth ACM SIGACT/SIGMOD Symposium on Principles of Database Systems*, pages 352–365, 1983.
110. A. Farrag and M. Ozsu. Using semantic knowledge of transactions to increase concurrency. *ACM, TODS*, 14(4):503–525, 1989.
111. J. Fernández, J. Grant, and J. Minker. Model theoretic approach to view updates in deductive databases. *Journal of Automated Reasoning*, 1996. To appear.
112. J. Fernández and J. Minker. Semantics of disjunctive deductive databases. In *Proceedings of the International Conference on Database Theory*, pages 332–356, 1992. (Invited Paper).
113. J. Fernández and J. Minker. Theory and algorithms for disjunctive deductive databases. *Programmirovanie*, N 3:5–39, 1993. (also appears as University of Maryland Technical Report,CS-TR-3223, UMIACS-TR-94-17,1994. Invited Paper in Russian).
114. J. Fernández and J. Minker. Bottom-up computation of perfect models for disjunctive theories. *Journal of Logic Programming*, 25(1):33–51, October, 1995.
115. J. A. Fernández and J. Lobo. A proof procedure for stable theories. Submitted to the Journal of Logic Programming, 1993.
116. J. A. Fernández, J. Lobo, J. Minker, and V. Subrahmanian. Disjunctive LP + integrity constraints = stable models semantics. In L. Lakshmanan, editor, *Proceedings of the ILPS'91 Workshop on Deductive Databases*, pages 110–117, San Diego, California, Oct. 1991. Extended Version presented at the Second International Symposium on Artificial Intelligence.
117. J. A. Fernández, J. Lobo, J. Minker, and V. Subrahmanian. Disjunctive LP + integrity constraints = stable model semantics. *Annals of Mathematics and Artificial Intelligence*, 8(3–4):449–474, 1993.
118. J. A. Fernández and J. Minker. Bottom-up evaluation of Hierarchical Disjunctive Deductive Databases. In K. Furukawa, editor, *Logic Programming Proceedings of the Eighth International Conference*, pages 660–675. MIT Press, 1991.
119. M. Fitting. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming*, 2:295–312, 1985.

120. O. Friesen, G. Gauthier-Villars, A. Lefebvre, and L. Vieille. Applications of deductive object-oriented databases using del. In R. Ramakrishnan, editor, *Applications of Logic Databases*. Kluwer Academic Publishers, 1995.
121. O. Friesen, A. Lefebvre, and L. Vieille. VALIDITY: Applications of a dood system. In *Proc. 5th Int. Conf. on Extending Database Technology - EDBT'96 (LNCS 1057)*, Avignon, France, March 1996. Springer Verlag.
122. T. Gaasterland. *Cooperative Answers for Database Queries*. PhD thesis, University of Maryland, Department of Computer Science, College Park, 1992.
123. T. Gaasterland, P. Godfrey, and J. Minker. An overview of cooperative answering. *Journal of Intelligent Information Systems*, 1(2):123–157, 1992. Invited paper.
124. T. Gaasterland, P. Godfrey, and J. Minker. An overview of cooperative answering. *Journal of Intelligent Information Systems*, 1(2), 1992.
125. T. Gaasterland, P. Godfrey, and J. Minker. Relaxation as a platform for cooperative answering. *Journal of Intelligent Information Systems*, 1:293–321, 1992.
126. T. Gaasterland, P. Godfrey, J. Minker, and L. Novik. A cooperative answering system. In A. Voronkov, editor, *Proceedings of the Logic Programming and Automated Reasoning Conference*, Lecture Notes in Artificial Intelligence 624, pages 478–480. Springer-Verlag, St. Petersburg, Russia, July 1992.
127. T. Gaasterland, P. Godfrey, J. Minker, and L. Novik. A cooperative answering system. In *Proceedings of the Logic Programming and Automated Reasoning Conference*, St. Petersburg, Russia, July 1992.
128. T. Gaasterland and J. Lobo. Processing negation and disjunction in logic programs through integrity constraints. *Journal of Intelligent Information Systems*, 2(3), 1993.
129. T. Gaasterland and J. Minker. User needs and language generation issues in a cooperative answering system. In P. Saint-Dizier, editor, *ICLP'91 Workshop: Advanced Logic Programming Tools and Formalisms for Language Processing*, pages 1–14, INRIA, Paris, France, June 1991.
130. T. Gaasterland, J. Minker, and A. Rajasekar. Deductive database systems and knowledge base systems. In *Proceedings of VIA 90*, Barcelona, Spain, Oct. 1990.
131. H. Gallaire and J. Minker, editors. *Logic and Databases*. Plenum Press, New York, Apr. 1978.
132. H. Gallaire, J. Minker, and J.-M. Nicolas, editors. *Advances in Database Theory*, volume 1. Plenum Press, 1981.
133. H. Gallaire, J. Minker, and J.-M. Nicolas, editors. *Advances in Database Theory*, volume 2. Plenum Press, 1984.
134. H. Gallaire, J. Minker, and J.-M. Nicolas. Logic and databases: A deductive approach. *ACM Computing Surveys*, 16(2):153–185, June 1984.
135. H. Garcia-Molina. Using semantic knowledge for transaction processing in a distributed database. *ACM, TODS*, 8(2):186–213, 1983.
136. A. V. Gelder. The alternating fixpoint of logic programs with negation. In *In Eighth ACM Symposium on Principles of Database Systems*, pages 1–10, 1989. Available from UC Santa Cruz as UCSC-CRL-88-17.
137. A. V. Gelder, K. Ross, and J. Schlipf. Unfounded Sets and Well-founded Semantics for General Logic Programs. In *Proc. 7^{th} Symposium on Principles of Database Systems*, pages 221–230, 1988.
138. A. V. Gelder, K. Ross, and J. Schlipf. The well-founded semantics for general logic programs. *Journal of The association for Computing Machinery*, 38(3):620–650, July 1991.

139. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, editors, *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pages 1070–1080, Seattle, WA. USA, Aug. 1988. The MIT Press.
140. M. Gelfond and V. Lifschitz. Logic programs with classical negation. In D. Warren and P. Szeredi, editors, *Proceedings of the Seventh International Conference on Logic Programming*, pages 579–597, Jerusalem, Israel, June 1990. The MIT Press.
141. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
142. M. Gelfond and V. Lifschitz. Representing actions and change by logic programs. *Journal of Logic Programming*, 17(2,3,4):301–323, 1993.
143. M. Gelfond, H. Przymusinska, and T. Przymusinski. The extended closed world assumption and its relation to parallel circumscription. *Proc. Fifth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 133–139, 1986.
144. P. Godfrey, J. Gryz, and J. Minker. Semantic query evaluation for bottom-up evaluation. In *Proc. ISMIS96*, June 1996. University of Maryland Technical Report, CS-TR-3558, UMIACS-TR-95-109.
145. P. Godfrey, J. Minker, and L. Novik. An architecture for a cooperative database system. In W. Litwin and T. Risch, editors, *Proceedings of the First International Conference on Applications of Databases*, Lecture Notes in Computer Science 819, pages 3–24. Springer Verlag, Vadstena, Sweden, June 1994.
146. G. Gottlob. Complexity and expressive power of disjunctive logic programming (research overview). In M. Bruynooghe, editor, *International Logic Programming Symposium ILPS'94*, pages 23–42, Ithaca, NY, USA, Nov 1994. The MIT Press.
147. J. Grant. Incomplete information in a relational database. In *Proc. Fund Inf III*, pages 363–378, 1980.
148. J. Grant, J. Gryz, and J. Minker. Updating disjunctive databases via model trees. Technical Report CS-TR-3407, UMIACS-TR-95-11, Department of Computer Science, University of Maryland, College Park, MD 20742, Feb. 1995.
149. J. Grant, J. Horty, J. Lobo, and J. Minker. View updates in stratified disjunctive databases. *Journal Automated Reasoning*, 11:249–267, March 1993.
150. J. Grant and J. Minker. Answering queries in indefinite databases and the null value problem. In P. Kanellakis, editor, *Advances in Computing Research: The Theory of Databases*, pages 247–267. 1986.
151. C. Green and B. Raphael. Research in intelligent question answering systems. *Proc. ACM 23rd National Conference*, pages 169–181, 1968.
152. C. Green and B. Raphael. The use of theorem-proving techniques in question-answering systems. *Proc. 23rd National Conference ACM*, 1968.
153. H. Groiss. A formal semantics for a rule-based language. In *IJCAI Workshop on Production Systems and their Innovative Applications*, 1993.
154. M. Hammer and S. Zdonik. Knowledge-based query processing. *Proc. 6th International Conference on Very Large Data Bases*, pages 137–147, Oct. 1980.
155. D. Harel. Review number 36,671 of Logic and Data Bases by H. Gallaire and J. Minker. *Computing Reviews*, 21(8):367–369, August 1980.
156. L. Henschen and S. Naqvi. On compiling queries in recursive first-order databases. *J.ACM*, 31(1):47–85, Jan. 1984.
157. L. Henschen and H. Park. Compiling the GCWA in Indefinite Databases. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*,

pages 395–438. Morgan Kaufmann Pub., Washington, D.C., 1988.
158. R. Hill. Lush resolution and its completeness. Technical Report DCL Memo 78, Department of Artificial Intelligence, University of Edinburgh, August 1974.
159. R. Hill. Lush resolution and its completeness. Technical Report DCL Memo 78, Department of Artificial Intelligence, University of Edinburgh, August 1974.
160. T. Imielinski. Incomplete deductive databases. *Annals of Mathematics and Artificial Intelligence*, 3:259–293, 1991.
161. T. Imielinski and W. Lipski. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.
162. T. Imielinski and K. Vadaparty. Complexity of query processing in databases with OR-objects. In *Proc. 7^{th} ACM SIGACT/SIGMOD Symposium on Principles of Database Systems*, pages 51–65, Philadelphia, Pennsylvania, March 29-31 1989.
163. K. Inoue, M. Koshimura, and R. Hasegawa. Embedding negation as failure into a model generation theorem prover. In D. Kapur, editor, *Proceedings of the Eleventh International Conference on Automated Deduction*, pages 400–415, Saratoga Springs NY, USA, June 1992. Springer-Verlag.
164. K. Inoue and C. Sakama. A fixpoint characterization of abductive logic programs. *Journal of Logic Programming*, 27(2):107–136, May 1996.
165. A. Itai and J. A. Makowsky. On the complexity of Herbrand's theorem. Technical report, Dept. of Computer Science, Israel Institute of Technology, Haifa, 1982.
166. J. Jaffar and M. Maher. Constraint logic programming:a survey. *Journal of Logic Programming*, 19-20:503–581, May-July 1994.
167. J. Jafffar and J.-L. Lassez. Constraint logic programming. In *Proc. of the 14^{th} ACM Symposium on Principles of Programming Languages*, pages 111–119, Münich, Germany, Jan 1987.
168. H. Jakobovits and D. Vermier. R-stable models for logic programs. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 251–259, July 1-2 1996. Also in this collection.
169. M. Jeusfeld and M. Staudt. Query optimization in deductive object bases. In G. Vossen, J. C. Feytag, and D. Maier, editors, *Query Processing for Advanced Database Applications*. Morgan-Kaufmann, 1993.
170. A. C. Kakas, R. A. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 6(2):719–770, 1993.
171. H. Kautz and B. Selman. Forming concepts for fast inference. In *Proc. of AAAI-92*, pages 786–793, 1992.
172. C. Kellogg, P. Klahr, and L. Travis. Deductive planning and pathfinding for relational data bases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 179–200. Plenum Press, New York, 1978.
173. B. Kero and S. Tsur. The $\mathcal{IQ}$ system: a deductive database information lens for reasoning about textual information. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 377–395, July 1-2 1996. Also in this collection.
174. W. Kiebling and H. Schmidt. DECLARE and SDS: Early efforts to commercialize deductive database technology. 1993.
175. M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of ACM– to appear*, 1993.
176. W. Kim, J.-M. Nicolas, and S. Nishio, editors. *Proc. 1st Int. Conf. on Deductive and Object-Oriented Databases - DOOD'89*. North-Holland Publishing Co., Amsterdam, The Netherlands, 1990, December 1990.

177. J. King. Quist: A system for semantic query optimization in relational databases. *Proc. 7th International Conference on Very Large Data Bases*, pages 510–517, Sept. 1981.
178. P. Kolaitis and C. Papadimitriou. Why not negation by fixpoint? *JCSS*, 43:125, 1991.
179. H. Korth and G. Speegle. Formal aspects of concurrency control in long duration transaction systems using the NT/PV model. *ACM, TODS*, 19(3):492–535, 1994.
180. R. Kowalski. Predicate logic as a programming language. *Proc. IFIP 4*, pages 569–574, 1974.
181. R. Kowalski. Logic for data description. In H. G. J. Minker, editor, *Logic and Data Bases*, pages 77–102. Plenum Press, New York, 1978.
182. R. Kowalski and F. Sadri. Towards a unified agent architecture that combines rationality with reactivity. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 131–150, July 1-2 1996. Also in this collection.
183. R. A. Kowalski. A proof procedure using connection graphs. *Journal of the ACM*, 22(4):572–595, Oct. 1975.
184. R. A. Kowalski and D. Kuehner. Linear Resolution with Selection Function. *Artificial Intelligence*, 2:227–260, 1971.
185. S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic Reasoning, Preferential Models and Cumulative Logics. *Artificial Intelligence*, 44(1):167–207, 1990.
186. J. Kuhns. Answering questions by computer: A logical study. Technical report, The Rand Corporation, Dec. 1967.
187. J. Kuhns. Logical aspects of questions answering by computer. *Third International Symposium on Computer and Information Sciences*, Dec. 1969.
188. J. Kuhns. Interrogating a relational data file: Remarks on the admissibility of input queries. Technical report, The Rand Corporation, Nov. 1970.
189. B. Kuipers, J. Paredaens, M. Smits, and J. V. den Bussche. Termination properties of spatial Datalog programs. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 95–109, July 1-2 1996. Also in this collection.
190. L. Laksmanan and N. Shiri. A parametric approach to deductive databases with uncertainty. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 55–73, July 1-2 1996. Also in this collection.
191. D. Laurent and C. Vrain. Learning query rules for optimizing databases with update rules. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 173–192, July 1-2 1996. Also in this collection.
192. A. Lefebvre and L. Vieille. On deductive query evaluation in the DedGin* system. In W. Kim, J.-M. Nicolas, and S. Nishio, editors, *1st Int. Conf. on Deductive and Object-Oriented Databases*, December 1989.
193. L. Lefebvre. Towards and efficient evaluation of recursive aggregates in deductive databases. In *Proc. 4th Int. Conf. on Fifth Generation Computer Systems (FGCS)*, June 1992. extended version in. New Generation Computing 12, Ohmsha Ltd. & Springer-Verlag, 1994.
194. N. Leone and P. Rullo. The safe computation of the well-founded semantics for logic programming. *Information Systems*, 17(1):17–31, Jan. 1992.
195. N. Leone, P. Rullo, and F. Scarcello. Stable model checking for disjunctive programs. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 281–294, July 1-2 1996. Also in this collection.
196. Y. Lesperance, H. Levesque, F. Lin, D. Marcu, R. Reiter, and R. Scherl. A logical approach to high level robot programming – a progress report. In *Working notes*

of the 1994 AAAI fall symposium on Control of the Physical World by Intelligent Systems, New Orleans, LA, November 1994.

197. H. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. Scherl. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming*, 1996. To appear.
198. H. J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23:155–212, Mar. 1984.
199. R. Levien. A computer system for inference execution and data retrieval. *Comm. ACM*, 10:715–721, 1967.
200. R. Levien. Relational data file ii: Implementation. *Proc. Third Annual National Colloquium on Information Retrieval*, pages 225–241, May 1967.
201. R. Levien. Relational data file: Experience with a system for propositional data storage and inference execution. Technical report, The Rand Corporation, Apr. 1969.
202. R. Levien and M. Maron. *Relational Data File: A Tool for Mechanized Inference Execution and Data Retrieval.* The Rand Corporation, Dec. 1965.
203. R. Levien and M. Maron. A computer system for inference execution and data retrieval. 10(11):715–721, Nov. 1967. Received September, 1966.
204. A. Levy, A. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS-95*, May 1995.
205. A. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of the 22nd VLDB Conference*, 1996.
206. A. Levy and Y. Sagiv. Semantic query optimization in datalog programs. In *Principles of Database Systems 1995 (PODS95)*, pages 163–173, 1995.
207. F. Lin and R. Reiter. How to progress a database II: The STRIPS connection. Technical report, Dept of Computer Science, University of Toronto, 1993.
208. F. Lin and R. Reiter. How to progress a database (and Why) I: Logical foundations. In *KR94*, pages 425–436, 1994.
209. T.-W. Ling, A. Mendelzon, and L. Vieille, editors. *Proc. 4th Int. Conf. on Deductive and Object-Oriented Databases - DOOD'95.* Springer-Verlang, December 1995. LNCS 1013, Heidelberg, Germany.
210. W. Lipski. On databases with incomplete information. volume 28, pages 41–70. ACM, New York, 1981.
211. K.-C. Liu and R. Sunderraman. Indefinite and maybe information in relational databases. *ACM Transactions on Database Systems*, 15(1):1–39, 1990.
212. K.-C. Liu and R. Sunderraman. On representing indefinite and maybe information in relational databases: A generalization. In *Proceedings of IEEE Data Engineering*, pages 495–502, Los Angeles, Feb. 1990.
213. Y. Liu. Null values in definite programs. In S. Debray and M. Hermenegildo, editors, *Proceedings of North American Conference on Logic Programming*, pages 273–288, Austin, Texas, Oct. 1990. MIT Press.
214. J. Lloyd. *Foundations of Logic Programming.* Springer–Verlag, second edition, 1987.
215. J. W. Lloyd and R. W. Topor. A basis for deductive database systems. *Journal of Logic Programming*, 2(2):93–109, July 1985.
216. J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming.* MIT Press, 1992.

217. J. Lobo, C. Yu, and G. Wang. Computing the transitive closure in disjunctive databases. Technical report, Department of Electrical Engineering and Computer Science, University of Illinois at Chicago, 1992.
218. D. Loveland. Near-Horn prolog. In J. Lassez, editor, *Proc. 4^{th} International Conference on Logic Programming*, pages 456–459, 1987.
219. D. Loveland, D. Reed, and D. Wilson. Satchmore: Satchmo with relevancy. Technical report, Duke University, Durham, North Carolina, USA, April 1993.
220. B. Ludascher, W. May, and G. Lausen. Nested transactions in a logical language for active rules. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 217–242, July 1-2 1996. Also in this collection.
221. C. Maindreville and E. Simon. Modeling non-deterministic queries and updates in deductive databases. In *Proc. of VLDB*, 1988.
222. V. W. Marek, A. Nerode, and J. Remmel. The stable models of a predicate logic program. In K. Apt, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 446–460, Washington D.C., USA, Nov 1992. The MIT Press.
223. V. W. Marek and Truszczyński. Autoepistemic logic. *Journal of the ACM*, 38(3):588–619, 1991.
224. C. Martin and J. Sistac. Applying transition rules to bitemporal deductive databases for integrity constraint checking. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 111–128, July 1-2 1996. Also in this collection.
225. J. McCarthy. Circumscription - a form of non-monotonic reasoning. *Artificial Intelligence Journal*, 13:27–39, 1980.
226. McDermott and J. Doyle. Non-monotonic logic i. *Artificial Intelligence Journal*, 13:41–72, 1980.
227. J. McSkimin and J. Minker. The use of a semantic network in deductive question-answering systems. *Proc. IJCAI 5*, pages 50–58, 1977.
228. J. Melton. An SQL3 snapshot. In *Twelfth International Conference on Data Engineering*, pages 666–672, 1996.
229. J. Melton and A. R. Simon. *Understanding the New SQL: A Complete Guide.* Morgan Kaufmann, San Mateo, California, 1993.
230. R. Miller, Y. Ioannidis, and R. Ramakrishnan. Translation and integration of heterogeneous schemas: Bridging the gap between theory and practice. *Information Systems*, 19(1):3–31, Jan. 1994.
231. J. Minker. Search strategy and selection function for an inferential relational system. *Transactions on Data Base Systems*, 3(1):1–31, Mar. 1978.
232. J. Minker. On indefinite databases and the closed world assumption. In *Proceedings of the Sixth Conference on Automated Deduction*, pages 292–308, 1982. Also in: *Lecture Notes in Computer Science* 138, pages 292-308. Springer Verlag, 1982.
233. J. Minker, editor. *Proceedings of Workshop on Foundations of Deductive Databases and Logic Programming*, Aug. 1986.
234. J. Minker, editor. *Foundations of Deductive Databases and Logic Programming.* Morgan-Kaufmann, 1988.
235. J. Minker. Perspectives in deductive databases. *Journal of Logic Programming*, 5:33–60, 1988.
236. J. Minker. Toward a foundation of disjunctive logic programming. In *Proceedings of the North American Conference on Logic Programming*, pages 121–125. MIT Press, 1989. Invited Banquet Address.

237. J. Minker. An overview of nonmonotonic reasoning and logic programming. *Journal of Logic Programming*, 17(2, 3 and 4):95–126, November 1993.
238. J. Minker and J. Grant. Answering queries in indefinite databases and the null value problem. In P. Kanellakis, editor, *Advances in Computing Research*, pages 247–267. JAI Press, 1986.
239. J. Minker and J.-M. Nicolas. On recursive axioms in deductive databases. *Information Systems*, 7(4):1–15, 1982.
240. J. Minker and A. Rajasekar. A Fixpoint Semantics for Non-Horn Logic Programs. Technical Report CS-TR-1869, Department of Computer Science University of Maryland, College Park, 1987.
241. J. Minker and A. Rajasekar. Procedural interpretation of non-Horn logic programs. In E. Lusk and R. Overbeek, editors, *Proceedings of the Ninth International Conference on Automated Deduction*, pages 278–293, Argonne, IL. USA, May 1988.
242. J. Minker and A. Rajasekar. Disjunctive logic programming. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems*, pages 381–394, 1989. (Invited Lecture).
243. J. Minker and A. Rajasekar. A fixpoint semantics for disjunctive logic programs. *Journal of Logic Programming*, 9(1):45–74, July 1990.
244. J. Minker and C. Ruiz. On extended disjunctive logic programs. In J. Komorowski and Z. Raś, editors, *Proceedings of the Seventh International Symposium on Methodologies for Intelligent Systems*, pages 1–18. Lecture Notes in AI. Springer-Verlag, June 1993. (Invited Paper).
245. J. Minker and C. Ruiz. Semantics for disjunctive logic programs with explicit and default negation. *Fundamenta Informaticae*, 20(3/4):145–192, 1994. Anniversary Issue edited by H. Rasiowa.
246. J. Minker and C. Ruiz. Mixing a default rule with stable negation. In *Proceedings of the Fourth International Symposium on Artificial Intelligence and Mathematics*, pages 122–125, Jan. 1996.
247. J. Minker and J. Sable. Relational data system study. Technical Report RADC-TR-70-180, Rome Air Development Center, Air Force Systems Command, Griffiss Air Force Base, New York, September 1970. Auerbach Corporation Report AD 720-263.
248. J. Minker and G. Zanon. An Extension to Linear Resolution with Selection Function. *Information Processing Letters*, 14(3):191–194, June 1982.
249. R. Moore. Possible-world semantics for autoepistemic logic. In *Proceedings of AAAI Workshop on Non-Monotonic Reasoning*, pages 396–401, New Paltz, 1984.
250. R. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence 25*, pages 75–94, 1985.
251. S. Morishita, M. Derr, and G. Phipps. Design and implementation of the Glue-Nail database system. In *Proc. ACM-SIGMOD'93 Conf.*, pages 147–167, May 1993.
252. S. Muggleton and L. D. Raedt. Inductive logic programming: theory and methods. *Journal of Logic Programming*, 19/20:629–679, May/July 1994.
253. I. Mumick, S. Finkelstein, H. Pirahesh, and R. Ramakrishnan. Magic is relevant. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, May 1990.
254. J. Naughton and Y. Sagiv. A decidable class of bounded recursions. *Proc. of the Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 227–236, Mar. 1987.

255. R. Ng and V. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1993.
256. J.-M. Nicolas. Logic for improving integrity checking in relational databases. *Acta Informatica*, 18(3):227–253, Dec. 1979.
257. J.-M. Nicolas and H. Gallaire. Data base: Theory vs. interpretation. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 33–54. Plenum Press, New York, 1978.
258. J.-M. Nicolas and J.-C. Syre. Natural question - answering and automatic deduction in system syntex. *Proc. IFIP Congress 1974*, pages 595–599, 1974.
259. J.-M. Nicolas and K. Yazdanian. Integrity checking in deductive databases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 325–599. Plenum, New York, 1978.
260. P. Pearce and G. Wagner. Logic programming with strong negation. In P. Schroeder-Heister, editor, *Proceedings of the International Workshop on Extensions of Logic Programming*, pages 311–326, Tübingen, FRG, Dec. 1989. Lecture Notes in Artificial Intelligence, Springer -Verlag.
261. G. Phipps, M. Derr, and K. Ross. Glue-Nail: A deductive database system. In *Proc. ACM-SIGMOD'91 Conf.*, May 1991.
262. G. Piatetsky-Shapiro and W. J. Frawley, editors. *Knowledge Discovery in Databases*. AAAI Press and MIT Press, Menlo Park, California, 1991.
263. P. Powell. Answer-Reason extraction in a parallel relatioanal data base system. Master's thesis, Department of Computer Science, University of Maryland, College Park, MD 20742, 1977.
264. S. Pradhan. Combining datalog databases using priorities. In *Advances in Data Management '94*, pages 355–375. Tata-McGraw Hill, India, 1995.
265. S. Pradhan and J. Minker. Combining datalog databases using priorities. *Journal of Intelligent & Cooperative Information Systems*, 1995.
266. S. Pradhan, J. Minker, and V. Subrahmanian. Combining databases with prioritized information. *Journal of Intelligent Information Systems*, 4(3):231–260, May 1995.
267. T. Przymusinski. Perfect model semantics. In R. Kowalski and K. Bowen, editors, *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pages 1081–1096, Seattle, WA. USA, Aug. 1988. The MIT Press.
268. T. Przymusinski. Every logic program has a natural stratification and an iterated fixed point model. In *Proceedings of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principle of Database Systems*, pages 11–21, Philadelphia, PA. USA, 1989.
269. T. Przymusinski. Stationary semantics for disjunctive logic programs and deductive databases. In S. Debray and M. Hermenegildo, editors, *Proc. of the North American Conference on Logic Programming*, pages 40–62, Austin, Texas, Oct. 1990.
270. T. Przymusinski. Static semantics for normal and disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, 14:323–357, 1995. Festschrift in honor of Jack Minker.
271. T. C. Przymusinski. On the declarative semantics of deductive databases and logic programming. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, chapter 5, pages 193–216. Morgan Kaufmann Pub., Washington, D.C., 1988.
272. T. C. Przymusinski. On the Declarative and Procedural Semantics of Logic Programs. *Journal of Automated Reasoning*, 5:167–205, 1989.

273. T. C. Przymusinski. Extended stable semantics for normal and disjunctive programs. In D. Warren and P. Szeredi, editors, *Proceedings of the* 7^{th} *International Logic Programming Conference*, pages 459–477, Jerusalem, 1990. MIT Press. Extended Abstract.
274. T. C. Przymusinski. Stable semantics for disjunctive programs. *New Generation Computing*, 9:401–424, 1991.
275. E. Pudilo. Database query evaluation with the STARBASE method. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 335–354, July 1-2 1996. Also in this collection.
276. A. Rajasekar, J. Lobo, and J. Minker. Skeptical reasoning and disjunctive programs. In *Proceedings of First International Conference on Knowledge Representation and Reasoning*, pages 349–357. Morgan-Kaufmann, 1989.
277. A. Rajasekar, J. Lobo, and J. Minker. Weak generalized closed world assumption. *Journal of Automated Reasonig*, pages 293–307, 1989.
278. R. Ramakrishnan. *Applications of Logic Databases.* Kluwer Academic Publishers, 1995.
279. R. Ramakrishnan, D. Srivastava, and S. Sudarshan. CORAL—control, relations and logic. In L.-Y. Yuan, editor, *Proceedings of the 18*th *International Conference on Very Large Databases*, pages 238–250, Vancouver, Canada, Aug. 1992.
280. R. Ramakrishnan and J. Ullman. A survey of research on deductive database systems. *Journal of Logic Programming*, 23(2):125–149, May 1995.
281. K. Ramamohanarao. An implementation overview of the Aditi deductive database system. In *Lecture Notes in Computer Science 760, Third International Conference, DOOD'93*, pages 184–203, Phoenix, AZ, December 1993. Springer-Verlag.
282. B. Raphael. A computer program for semantic information retrieval. In M. Minsky, editor, *Semantic Information Processing*, pages 33–134. MIT Press, 1968.
283. L. Raschid. A semantics for a class of stratified production system programs. *Journal of Logic Programming*, 21(1):31–57, 1994.
284. L. Raschid and J. Lobo. A semantics for a class of non-deterministic and causal production system programs. *Journal of Automated Reasoning*, 12:305–349, 1994.
285. L. Raschid and J. Lobo. Semantics for update rule programs and implementation in a relational database management system. *ACM Transactions on Database Systems*, 1996. To appear.
286. D. Reed, D. Loveland, and B. Smith. An alternative characterization of disjunctive logic programs. In *Proceedings of the International Logic Programming Symposium*, Cambridge, Massachusetts, 1991. MIT Press. Also: Technical Report, CS-1991-11. Dept. of Computer Science. Duke University.
287. D. W. Reed and D. W. Loveland. A comparison of three prolog extensions. Technical Report CS-1989-8, Department of Conputer Science, Duke University, Durham, North Carolina 27706, March 1990. To appear in Journal of Logic Programming.
288. R. Reiter. On closed world data bases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 55–76. Plenum, New York, 1978.
289. R. Reiter. A logic for default reasoning. *Artificial Intelligence Journal*, 13:81–132, 1980.
290. R. Reiter. Towards A Logical Reconstruction of Relational Database Theory. In M. Brodie, J. Mylopoulos, and J. Schmit, editors, *On Conceptual Modelling*, pages 163–189. Springer–Verlag Pub., New York, 1984.

291. R. Reiter. A sound and sometimes complete query evaluation algorithm for relational databases with null values. *J.ACM*, 33(2):349–370, Apr. 1986.
292. R. Reiter. On integrity constraints. In M. Y. Vardi, editor, *Proceedings of the Second Conference on the Theoretical Aspects of Reasoning about Knowledge*, pages 97–111, San Francisco, California, Mar. 1988. Morgan Kaufmann Publishers, Inc.
293. R. Reiter. On asking what a database knows. In J. Lloyd, editor, *Computational Logic*, Basic research Series. Springer-Verlag Publishers, 1990. DG XIII Commission of the European Communities.
294. J. Robinson. A machine-oriented logic based on the resolution principle. *J.ACM*, 12(1), Jan. 1965.
295. J. Rohmer, R. Lescoeur, and J.-M. Kerisit. The Alexander method: a technique for the processing of recusive axioms in deductive databases. *New Generation Computing*, 4(3), 1986.
296. K. Ross. A procedural semantics for well-founded negation in logic programs. In *Proceedings of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principle of Database Systems*, Philadelphia, PA. USA, 1989.
297. K. Ross. Well-founded semantics for disjunctive logic programs. In *Proc. of the first International Conference on Deductive and Object Oriented Databases*, pages 352–369, Kyoto, Japan, Dec. 1989.
298. K. Ross. Modular stratification and magic sets for datalog programs with negation. In *Proc. ACM Symp. on Principles of Database Systems*, April 1990.
299. K. Ross. Modular acylicity and tail recursion in logic programs. In *Proc. of the Tenth ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (PODS'91)*, pages 92–101, 1991.
300. K. Ross and R. Topor. Inferring negative information from disjunctive databases. *Journal of Automated Reasoning*, 4(2):397–424, December 1988.
301. C. Ruiz and J. Minker. Computing stable and partial stable models of extended disjunctive logic programs. In J. Dix, L. Pereira, and T. Przymusinski, editors, *Nonmonotonic Extensions of Logic Programming*, pages 205–229. Lecture Notes in Computer Science 927. Springer-Verlag, 1995.
302. D. Saccà. The expressive power of stable models for bound and unbound DATALOG queries. *The Journal of Computer and System Sciences*, 1996. To appear.
303. D. Sacca and C. Zaniolo. On the implementation of a simple class of logic queries. In *Proc. ACM Symp. on Principles of Database Systems*, March 1986.
304. D. Sacca and C. Zaniolo. The generalized counting method for recursive logic queries. *Theoretical Computer Science*, 62, 1988.
305. D. Saccà and C. Zaniolo. Stable models and non-determinism to logic with negation. In *Proc. Workshop on Logic Programming and Nonmonotonic Reasoning*, pages 87–101, 1991.
306. F. Sadri and R. Kowalski. Database integrity. chapter 9, pages 313–362. Morgan Kaufmann Publishers, 1988.
307. K. Sagonas, T. Swift, and D. Warren. The limits of fixed-order computation. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 355–374, July 1-2 1996. Also in this collection.
308. C. Sakama. Possible model semantics for disjunctive databases. In *Proc. First International Conference on Deductive and Object Oriented Databases*, pages 337–351, 1989.
309. E. Sandewall. Features and fluents: A systemetic approach to the representation of knowledge about dynamical systems. Technical report, Institutionen for datavetenskap, Universitet och Tekniska hogskolan i Linkoping, Sweden, 1992.

310. E. Sandewall. The range of applicability of some non-monotonic logics for strict inertia. *Journal of Logic and Computation*, 4(5):581–616, Oct. 1994.
311. D. Savitt, H. Love, and R. Troop. ASP: A new concept in language and machine organization. In *1967 Spring Joint Computer Conference*, pages 87–102, 1967.
312. J. Schlipf. The expressive powers of the logic programming semantics. *JCSS*, 1990. A preliminary version appeared in Ninth ACM Symposium on Principles on Database Systems, pages 196-204, 1990.
313. J. Schlipf. Complexity and undecideability results for logic programming. *Annals of Mathematics and Artificial Intelligence*, 15(3-4):257–288, Dec 1995.
314. B. Selman and H. Kautz. Knowledge compilation using Horn approximations. In *Proc. of AAAI-91*, pages 904–909, 1991.
315. B. Selman and H. Kautz. Knowledge compilation and theory approximation. *Journal of the ACM*, 1996. To appear.
316. B. Selman and H. Levesque. Abductive and default reasoning: A computational core. In *Proceedings AAAI-90*, pages 343–348, Boston, MA, 1989.
317. S. Shapiro and D. McKay. Inferences with recursion. In *Proceedings of the 1st Annual National Conference on Artificial Intelligence*, 1980.
318. J. Shepherdson. Negation in Logic Programming. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 19–88. Morgan Kaufman Pub., 1988.
319. S. Sickel. A search technique for clause interconnectivity graphs. *IEEE Transactions on Computers*, C-25(8):823–835, Aug. 1976.
320. P. Sistla and O. Wolfson. Temporal conditions and integrity constraint checking in active database systems. In *Proceedings of the 1995 ACM SIGMOD International Conferenece on Management of Data*, San Jose, CA, 1995. ACM Press.
321. B. Smith and D. Loveland. A simple near-Horn Prolog interpreter. In R. Kowalski and K. Bowen, editors, *Proc. 5^{th} International Conference and Symposium on Logic Programming*, pages 794–809, Seattle, Washington, Aug. 1988.
322. R. Smullyan. On definability by recursion (abstract). *Bull, AMS62*, page 601, 1956.
323. R. M. Smullyan. Elementary formal system (abstract). *Bull, AMS62*, page 600, 1956.
324. R. Snodgrass. The temporal query language TQuel. *ACM Trans. Database Syst.*, 12(2):247–298, June 1987.
325. R. Snodgrass, editor. *Data Engineering.* IEEE Computer Society, December 1988. Special issue on temporal databases.
326. R. Snodgrass and E. McKenzie. Research concerning time in databases. *SIGMOD Record*, 15(4):19–52, December 1986.
327. M. Stickel. A PROLOG technology theorem prover: Implementation by an extended PROLOG compiler. *Journal of Automated Reasoning*, 4(4):353–380, 1988.
328. V. Subrahmanian, S. Adali, A. Brink, R. Emery, J. Lu, A. Rajput, T. Rogers, and R. Ross. Hermes: A heterogeneous reasoning and mediator system, 1994. Available from (http://www.cs.umd.edu/projects/hermes/overview/paper.
329. V. Subrahmanian and S. Jajodia. *Multimedia Database Systems.* Springer Verlag, 1995.
330. V. Subrahmanian and C. Ward. A deductive database approach to planning in uncertain environments. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases (LID'96)*, pages 77–92, July 1-2 1996. Also in this collection.
331. M. A. Suchenek. Minimal models for closed world databases. In Z. Ras, editor, *Proc. of ISMIS 4*, pages 515–522. Elsvier Sience Publishing Co. Inc., 1989.

332. T. Swift and D. Warren. An abstract machine for SLG resolution: definite programs. In *Proc. of the 1994 ILPS*, pages 633–652. The MIT Press, 1994.
333. T. V. Team. Summary state of the art on deductive and deductive object-oriented databases - dood. Technical report, Bull Corporation, April 1996. Groupe Bull report (limited distribution).
334. X. A. I. Technologies. HIPAC: a research project in active, time-constrained databases. Technical Report 187, Xerox Advanced Information Technologies, 1989.
335. D. Tsichritzis and F. Lochovsky. Hierarchic data-base model: A survey. *ACM Computing Surveys*, 8(1):67–103, March 1976.
336. S. Tsur and C. Zaniolo. LDL: A logic-based data-language. In *Proceedings of the 12th VLDB Conf*, August 1986.
337. J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume I.* Principles of Computer Science Series. Computer Science Press, Incorporated, Rockville, Maryland, 1988.
338. J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume II: The New Technologies.* Principles of Computer Science Series. Computer Science Press, Incorporated, Rockville, Maryland, 1989.
339. J. Vaghani, K. Ramamohanarao, D. B. Kemp, and P. J. Stuckey. Design overview of the Aditi deductive database system. In *Proc. of the 7th Intl. Conf. on Data Engineering*, pages 240–247, April 1991.
340. M. van Emden and R. Kowalski. The Semantics of Predicate Logic as a Programming Language. *J.ACM*, 23(4):733–742, 1976.
341. A. Van Gelder. Negation as failure using tight derivations for general logic programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 149–176. Morgan Kaufmann, 1988.
342. M. Vardi. The complexity of relational query languages. In *Proc. of the 14th ACM Symposium on Theory of Computing*, pages 137–146, May 1982.
343. Y. Vassiliou. Null values in data base management: A denotational semantics approach. In *Proceedings of the ACM SIGMOD International Symposium on Management of Data*, pages 162–169, Boston, MA, 1979. ACM, New York.
344. L. Vieille. Recursive axioms in deductive databases: the Query/SubQuery approach. In *Proc. 1st. Int. Conf. on Expert Database Systems*, April 1986.
345. L. Vieille. Database-complete proof procedures based on SLD-resolutions. In *Proc. 4th Int. Conf. on Logic Programming*, May 1987.
346. L. Vieille. Recursive query processing: The power of logic. *Theoretical Computer Science*, 69, 1989.
347. L. Vieille, P. Bayer, V. Kuechenhoff, and A. Lefebvre. EKS-V1, a short overview. *AAAI'90 Workshop on Knowledge Base Management Systems*, July 1990.
348. L. Vielle, P. Bayer, and V. Kuechenhoff. Integrity checking and materialized view handling by update propagation in the EKS-V1 system. Technical report, ECRC, 1991. to appear as a chapter in the book "Materialized Views", A. Gupta and I. Mumick (eds), MIT Press, Cambridge, MA, USA, 1996.
349. A. Yahya and L. Henschen. Deduction in Non-Horn Databases. *J. Automated Reasoning*, 1(2):141–160, 1985.
350. J.-H. You and L. Yuan. Three-valued formalization of logic programming. In *Proceedings of the 9th ACM PODS*, pages 172–182, 1990.
351. J.-H. You and L. Yuan. A three-valued semantics for deductive databases and logic programs. *Journal of Computer and System Sciences*, 49:334–361, 1994.

352. J.-H. You and L. Yuan. On the equivalence of semantics for normal logic programs. *Journal of Logic Programming*, pages 211–222, 1995.
353. L. Yuan and J.-H. You. Autoepistemic circumscription and logic programming. *Journal of Automated Reasoning*, 10:143–160, 1993.
354. L. Y. Yuan and D.-A. Chiang. A sound and complete query evaluation algorithm for relational databases with disjunctive information. In *Proceedings of the Eighth Symposium on Principles of Database Systems*, pages 66–74. ACM Press, Mar. 1989.
355. C. Zaniolo. Database relations with null values. *JCSS*, 28:142–166, 1984.
356. C. Zaniolo. A unified semantics for active and deductive databases. In *Proceedings of 1st international workshop on rules in database systems*, pages 271–287. Springer-Verlag, 1993.
357. C. Zaniolo. Active database rules with transaction-conscious stable models semantics. In *Proceedings of DOOD 1996*, pages 55–72, 1996.